Modeling a micro-mirror array and contribution to the development of a simulator of micro-system arrays

Duy Duc Nguyen

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE FRANCHE-COMTÉ

Modélisation d'une matrice de micro-miroirs et contribution au développement d'un simulateur de matrices de micro-systèmes

DUY DUC NGUYEN

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE FRANCHE-COMTÉ

N° _ _ _

THÈSE présentée par

DUY DUC NGUYEN

pour obtenir le

Grade de Docteur de l'Université de Franche-Comté

Spécialité : **Mécanique**

Modélisation d'une matrice de micro-miroirs et contribution au développement d'un simulateur de matrices de micro-systèmes

Unité de Recherche :
Institute FEMTO-ST, Besançon, France

Soutenue publiquement le 10 Mars 2017 devant le Jury composé de :

PIERRE-ETIENNE MOREAU	Examinateur	Professeur à l'Université de Lorraine
SKANDAR BASROUR	Examinateur	Professeur à l'Université de Grenoble Alpes
ANTONIO GAUDIELLO	Rapporteur	Maître de Conférences à l'Université de Cassino et Lazio du sud (UPHE), Italie
TEMUR KUTSIA	Rapporteur	Maître de Conférences à l'Université de Linz, Autriche
MICHEL LENCZNER	Directeur de thèse	Professeur à l'Université de Technologie Belfort-Montbéliard (UTBM)
FRÉDÉRIC ZAMKOTSIAN	Co-Directeur	Chargé de Recherche CNRS au Laboratoire d'Astrophysique de Marseille (LAM)
NICOLAS RATIER	Co-Directeur	Maître de Conférences à l'Ecole Nationale Supérieure de Mécanique et de Microtechniques (ENSMM)
HORATIU CIRSTEA	Co-Directeur	Professeur à l'Université de Lorraine

UNIVERSITÉ
DE FRANCHE-COMTÉ

Département Temps-Fréquence
UFR-ST Sciences et Techniques

École doctorale SPIM Besançon

Modeling a micro-mirror array and contribution to the development of a simulator of micro-system arrays

Dissertation

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Ph.D. in Mecanics

10 March 2017

Doctoral School of University of Franche-Comté

By

Duy Duc NGUYEN

Doctoral Committee:

President :	Professor Pierre-Etienne MOREAU
Reviewers :	Associate Professor Antonio GAUDIELLO
	Associate Professor Temur KUTSIA
Examiners :	Professor Michel LENCZNER
	CNRS Research Scientist Frédéric ZAMKOTSIAN
	Associate Professor Nicolas RATIER
	Professor Horatiu CIRSTEA
	Professor Skandar BASROUR

ACKNOWLEDGEMENTS

First of all, I would like to deeply thank Prof. Michel LENCZNER for his support, guidance, and supervision. His knowledge, dedication, enthusiasm, patience, encouragement, and personality have made my graduate experience at University of Franche-Comté rewarding and indeed unforgettable. I would like to sincerely thank my co-advisors, Prof. Frédéric ZAMKOTSIAN, Prof. Nicolas RATIER and Prof. Horatiu CIRSTEA for their invaluable comments, suggestions of my work. And I would like to extend my special thanks to Dr. Walid BELKHIR for his great supports.

I would like to send my appreciation to all the members of Jury, Prof. Antonio GAUDIELLO, Prof. Temur KUTSIA for reviewing my thesis.

I would like to thank my colleagues: Dr. Thi Trang NGUYEN, Dr. Do Huu QUAN, Dr. Bin YANG, Dr. Mohamed ABAIDI and doctoral student Nguyen Nhat Binh TRINH for their helpful discussions with mathematical problems, symbolic computation and simulation issues that I encountered during my work.

I hereby thank all staff members of the Institute FEMTO-ST and of the University of Franche-Comté for providing me a lot of helps. In particular, I would like to sincerely thank the Director of the PhD graduate school of Engineering science and Microsystems: Dr. Philippe Lutz; the secretaries from Time and Frequency Department: Mrs. Fabienne CORNU and Mrs. Sarah DJAOUTI; the secretaries of the team COSYMA: Ms. Isabelle GABET and Ms. Sandrine FRANCHI for helping me in work contract issues. I am also thankful for the secretaries of doctoral school of University of Franche-Comté for helping me with registration and preparation of my defense.

I also express my gratitude to Prof. Minh Duc DUONG and Prof. Denis GREBENKOV for teaching me, developing my background in mathematics and research experiences for the past several years.

Last but not least, I want to give my deepest grateful to my parents, my younger brother, my younger sister and my grand mother, who always encourage me to get through all these years and for their endless love. And a special thank to my wife Thi Thanh Ha PHAN who has always been by my side since I was a student in the University of Sciences, in HCM city, Vietnam.

To my parents
my younger brother
my younger sister
my grand mother
and my wife

CONTENTS

INTRODUCTION

INTRODUCTION

To understand the formation of galaxies and to study the expansion of the Universe, astronomers observe faint galaxies. In order to study and collect their spectra, the telescope has to be equipped with a spectrograph having Multi-Object Spectroscopy (MOS) capabilities. MOS is an astronomical technique that allows the simultaneous collection of the visible and infrared (IR) spectra of faint galaxies and stars, thereby providing information on the origin of the Universe and the formation of galaxies. Moreover, to remove spoiling sources such as bright stars and the stellar background and to avoid the overlapping of spectra between the measured objects, the modern MOS has a field selector placed at the focal plane of the telescope which allows the individual selection of the objects. Motivated by potential improvement of microtechnology, micro-electro-mechanical systems (MEMS)-based field selectors are developed which can achieve higher performances, higher number of targets, faster reprogrammable time and cryogenic operation compatibility. However, the performance of these spectrometers was mainly limited by their minimum operating temperature $-40°$C. Indeed, for IR observations when the instrument is not at cryogenic temperature, the measurements are spoiled by the thermal emission of the instrument itself. Therefore, to cope with these limitations, a group led by the Laboratoire d'Astrophysique de Marseille (LAM), has worked for many years to develop a european micro-mirror array (MMA) with electrostatically actuated tilting mono-crystalline silicon micro-mirrors Figure 1. It is designed as a field selector for MOS [Waldis, 2010], [Canonica, 2012], with stringent requirements summarized in Table 1.

Features	**Constraints**
Mirror size	$200 \times 100 \mu m$
Tilt angle	$\geq 20°$
Variation of tilt angle across the array	$\leq 0.6°$
Filling factor	$\geq 80\%$
Contrast ratio	≥ 1000
Wavelength band	from visible to IR ($\lambda = 1 \mu m$)
Mirror Peak-to-Valley (PTV) deformation	$\leq \lambda/20$
Electrostatic actuation voltage	$< 100V$
Operating temperature	from RT to cryogenic temperature ($< 100K$)

Table 1: requirements on a cell of the array from [Canonica, 2012].

Moreover, the size of the array must cover the field of view of a telescope.

A design fulfilling these conditions was built and then used for fabrication. A Finite Element Method model was made in the software package Coventor 2010. Various configurations of the micro-mirror arrays were simulated.

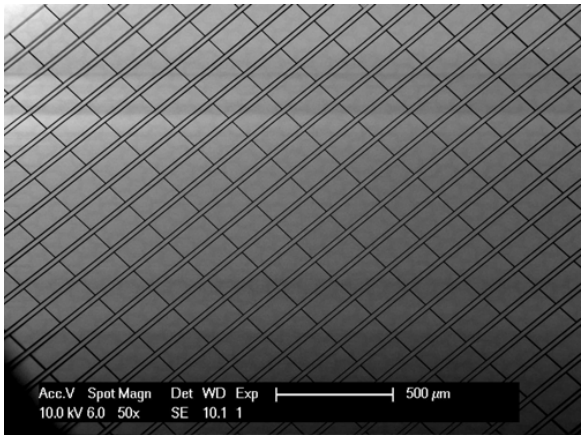


Figure 1: Real Micro-Mirror Array

The purpose of the FEM simulations was to determine the static electromechanical behavior of the micro-mirror, the maximum Von Mises stress in its suspending beams, the resonance modes of the micro-mirror and the contact locations between parts. Since geometry uncertainties due to the fabrication process are unavoidable, several models were created with different sets of parameters to analyze their effect on some aspect of the electromechanical behavior as for instance the tilt angle and voltage hysteresis.

A first requirement of my thesis was to simulate and optimize the same cell using Comsol Multi-physics, another popular FEM software package. It allows for modeling various physical and engineering applications and especially coupled phenomena thanks to specialized toolboxes. Another manner to use Comsol is by setting a system of partial differential equations (PDEs). This allows to investigate taylor-made models not available in the specialized toolboxes. Programing Comsol is made through Matlab scripts thanks to a link between the two packages. This provides to Comsol the full power of Matlab with its toolboxes for preprocessing, model manipulation, and postprocessing. This is thanks to this connection that SIMBAD, an in-house optimization software package based on Matlab functions, operates on Comsol models.

The mathematical description of the equations involved and the design of the micro-mirrors is recalled in Chapter 1. The mechanical deformations, the electrical field, the thermal effects are taken into account. The equations are written under their strong and weak forms. The geometry, the choice of materials and their coefficients are detailed.

The simulation and optimization results are reported in Chapter 2. The implemented model takes into account the quasi-static nonlinear elasticity coupled with the electrostatic field, but not the contact condition. This coupling is implemented in the complete geometry of a cell. The parameters used in simulation are precisely reported. In terms of analysis and optimization, the focus is made on the pull-in voltage, that is the determination of the limit of the actuation voltage before instability. I have shown that the pull-in voltage strongly depends on two main parameters: the length and thickness of the suspending beam. Unfortunately, the ratio ”beam length/beam thickness” is extremely large and is very critical in the simulation. Actually, it is a source of strong instability problems in the nonlinear solver. Other specific difficulties were met, in particular the strong nonlinearity of the problem requires very small computation steps and the relatively complex geometry, including small gaps, requires a very large mesh. In total, and after a number

of improvements, the simulation of the pulling voltage is successful for all the required sizes of the suspending beams, but still with long simulation time i.e. up to a dozen of hours. Finally, the optimization of the pull-in voltage in a cell has been achieved based on a polynomial metamodel built from some twenty pull-in voltage computations.

The second question addressed during my thesis is the modeling, and if possible the simulation, of an array of micro-mirrors. Evidently, direct simulations of the thousands of coupled micro-mirrors is outside of our scope. Indeed we expect to build models that can run in a reasonable time on a personal computer. The goal of this work was to adapt to this problem a periodic homogenization method based on an asymptotic approach. This choice was made to contribute to MEMSALab, a software package dedicated to asymptotic methods, which is at the very heart of Chapter 4.

Periodic homogenization has been developed for a long time and several methods have been emerging over the years. In this thesis, we focus on the two-scale convergence. In 1989, Nguetseng have introduced the notion of two-scale convergence in [Nguetseng, 1989] and this idea was further developed in [Allaire, 1992]. Independently, in 1990, Arbogast et al. [Arbogast et al., 1990] introduced a dilation operation to study homogenization for a periodic medium with double porosity. This technique was used again in Bourgeat [Bourgeat et al., 1996], Allaire [Allaire et al., 1998] and Lukkassen [Lukkassen et al., 2002]. M. Lenczner with his co-workers in [Lenczner, 1997], [Lenczner, 2007], [Lenczner et al., 2007], [Lenczner, 2006], [Lenczner et al., 1999], used the same idea to develop a complete framework yielding similar results as for periodic homogenization as the two-scale convergence method. They first introduced this new technique to address homogenization of spatially periodic analog electronic circuits in view of their application in arrays of MEMS. The advantage of this technique compared to the previous ones is more modularity, simpler calculations, less technicality in the proofs, and a wider range of applications e.g. it is easily applicable to manifolds as electrical networks. Then, J. Casado Diaz et al. [Casado-Diaz, 2000], [Casado-Diaz et al., 2004], [Casado-Díaz et al., 2001] combined it with the two-scale convergence to study perforated domains and thin structures. Then, the same concept was renamed the "*periodic unfolding method*" and popularized by D. Cioranescu, A. Damlamian and G. Griso who have developed a number of their properties, including error estimates, in [Casado-Díaz et al., 2001], [Cioranescu et al., 2008], [Griso, 2002], [Griso, 2004] and [Griso, 2006]. This technique has been extensively developed by many other authors in a variety of applications. In particular, it has been applied to find models of complex structures combining other asymptotic features, as thin structure or strong heterogeneity of coefficients, with the periodic homogenization, see among others [Blanchard et al., 2007], [Blanchard et al., 2007], [Blanchard et al., 2008], [Lenczner et al., 2007], or [Lenczner, 2007]. We notice that in [Lenczner et al., 2007], an attention has been paid to formulate the proofs of model derivation as a sequence of calculation without relying on any abstract arguments. The model derivation adopted in this thesis relates to the techniques developed in [Lenczner et al., 2007].

Ultimately, an homogenized model of the micro-mirror array should take into account the electro-thermo-mechanical coupling. It is worth recalling that the two-scale transformation consists in transforming a physical domain occupied by a periodic structure into a pair of a macroscopic domain that ignore the details in the cells and a microscopic domain which is a scaling of one of the cells. Mathematically, the periodic homogenization method consists in considering that the size of the array is fixed and that the number of cells is very large. The asymptotic model is found by passing to the limit when the number of cells

goes to infinity. To do this without loosing the details of the cells, the asymptotic analysis is carried out after application of the two-scale transformation.

To start the modeling it has been decided to address first the electrical problem. Since in each cell an electrical field is created in the vertical direction by imposing a difference of potential between the electrodes, it results that the leading part of the electrical potential of the asymptotic model is an electric potential having variations in both the macroscopic and the microscopic domains. However, in the expected applications of the micro-mirror array, the set of cells is split into two connected subsets, the sets of those switched ”on” and the set of those switched ”off”. It results that only two values of the imposed voltage have to be distinguished in the array. Since the imposed voltage is constant over all cells of a connected part of the array, the leading electrical potential varies only about the coordinates of the microscopic domain. It is a solution of the electrostatic equation over a single cell with periodic boundary conditions. Therefore, it suffers from a mismatch between both sides of the interface of the two connected parts of the array. To correct this defect, a boundary layer is introduced. Moreover, the periodic boundary conditions satisfied by the leading electrical potential are also incompatible with the boundary condition at the outer boundary of the array, as for instance a zero normal flux condition. Then, the model takes also into account a boundary layer effect on the external boundary of the array. As usual, a boundary layer is a part of solution that decreases very fast. Due to a lack of time, the boundary layer analysis and, for the sake of consistency, the complete models are presented for a one-dimensional array only. It can be extended to two-dimensional array to the price of an extra-work for the boundary layer on the interface between the two connected areas. It is also worthwhile to mention that the mathematical techniques for boundary layer derivation are taken from [Nguyen, 2014].

The third point where a contribution of my thesis was expected is related to the software package MEMSALab (for MEMS Array Lab). The concept underlying MEMSALab originates in the observation that although asymptotic analysis is a well established and good way to simplify models, it requires case-by-case derivation. That is the full process of model derivation is redone from the scratch whenever a new problem is met, eventhough it may share many features with an already solved problem. Consequently, given the variety of possible physics and geometrical configurations it appears insurmountable to adopt asymptotic modeling in general simulation software. However, they are implemented in laboratory software or in specialized software tools such as `Helius, MAC/GMC, CZone, DIGIMAT`. Evidently, the number of models that these software packages can cover is limited. The MEMSALab project aims to going beyond the principle of case-by-case model production and implementation.

Its leading principle is inspired by the human behaviour for the resolution of a new problem. More specifically, it copies the behaviour of a mathematician, since the approach is developed for the derivation of models which basically are mathematical proofs. An usual method for building a new mathematical model starts from a known theory that works in simpler cases, and by identifyng the new features to be taken into account. Then, an efficient principle is to find the way to transform a known proof into other ones, each taking into account a new features. This is the step of extension. Then the combination of these transformations yields the proof for the full problem. This approach is called the by-extension-combination method [Belkhir et al., 2015] that we sketch more precisely in the context of asymptotic model derivation.

An asymptotic model derivation starts with an input PDE coming from any scientific field to which a derivation, also said proof, is applied ending to the expected model. This scheme is build for a reference case, which is the simplest that we can consider, so we call it the *reference scheme*. Then, it is complexified, we say extended, in several manners to take into account new features yielding new schemes. The input PDEs still arise from an application area but with 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 asymptotic models. Finally, a new scheme for an input PDE covering a group of new features is built by *combination*. Precisely, its input PDE is still issued from a practical problem. Its proof is obtained by applying a combination of two or more extensions, built in the previous step, to the reference proof. Finally, applying the resulting proof to the input PDE yields an asymptotic model enjoying the groups of features. In summary, combining extensions related to new elementary features allows for building new proofs and therefore new asymptotic models in an incremental manner.

Figure 2: MEMSALab-Simplified-Flow

From a very global viewpoint, MEMSALab is designed to be in a chain of operations as represented on Figure 2: the designer states the nominal model (geometry and equations) in FEM software, the model is sent along with non-technical descriptions for the choice of asymptotic methods in MEMSALab which transforms it into a multi-scale model, the latter is appropriately implemented in the FEM, then the designer takes the reins and can apply its favorite operations for simulation and result visualization.

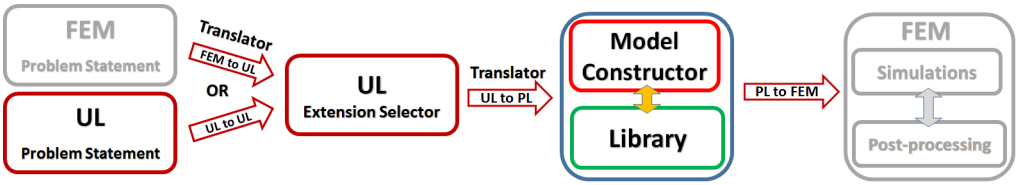


Figure 3: MEMSALab-Simplified-Flow

Figure 3 shows more details of the flow of operation of MEMSALab. The main components are the Library of (elementary) Extensions and Combinations (E-C Library) and the software Core consisting of SymbTrans, an engine of rewriting strategies, and SymbComb, an engine of extensions and combinations. The Core operates on expressions written in the Processing Language (PL). The users specify their problems using the User Language (UL), a language close to the usual mathematical language, and there exist translators from the PL to the UL and vice-versa. All operations are coordinated by the Control Manager written in MATLAB while the Core and the Translators are implemented in OCaml. The

The operations 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 specification of the asymptotic reduction is specified using the Extension Selector part of the UL. The necessary elementary Extensions being assumed to be pre-defined in the EC-Library, they are combined thanks to SymbComb, the engine of combinations. The result of this combination is a complex Extension which is applied to the Reference Proof through SymbTrans, so that to generate the Extended Proof. The latter is applied to the Input Model, through SymbTrans also, yielding the final Asymptotic Model which can be output in UL format and subsequently sent to FEM thanks to the UL to FEM Language translator.

Before the start of my thesis, several concepts and their implementation were already presented in the PhD thesis of Bin Yang [Yang et al., 2014, Yang, 2014]. SymbTrans was implemented as a Maple package and the reference proof in the style of [Lenczner et al., 2007] was implemented together with three extensions. The mathematical properties, lemmas and theorems were represented as rewriting rules, and the proofs as rewriting strategies. The extension mechanisms for the multi-scale model derivations was well established. Theoretical work for extension and combination were approached by both positional computation and strategies [Belkhir et al., 2015]. More precisely, an extension is an application of a transformation to the reference proof while a combination is a combining of the already implemented extensions. Through combinations, a complex models will be generated in the benefit of reusing the proofs and tools built for the generation of simpler models. Let us consider a reference proof, for example, having an extension E1 (viewed as a transformation) to some general setting (e.g. multi-dimensional setting) and another extension E2 to another general setting (e.g. thinness setting). The two extensions combined yields a new extension E' that covers both settings when applying to the reference proof.

Although a small class of usual rewriting strategies as `OuterMost` and `BottomUp` were considered for combinations, the question whether this class, or possibly a wider class, is closed under combination was left open, as well as the question of the correctness and soundness of the combination formula. Addressing these properties are one of the main contribution of this thesis to the kernel of MEMSALab, Chapter 4. Precisely, the idea of combination is kept the same but the tools and the techniques are different. We identify an operation of combination over a class of extensions named HCE-strategies, expressed as rewriting strategies that navigate along trees and insert contexts. We prove that this class is closed by combination after establishing an explicit formula of combinations. We shown that usual traversal strategies as `TopDown` or `BottomUp` belong to the class of HCE-strategies. Several nice algebraic properties of the HCE-strategies are also proved. Besides, in this thesis, we present the design and implementation of a user language for the specification of rewriting strategies based proofs and extensions. The user can use it to express PDEs, proof and extensions manually. A manager program built with Matlab for compilation, application, combination is also introduced. The reference proof mentioned above is completely implemented using the User Language.

1

INTRODUCTION TO MICROMIRROR AND MODELS

1.1/ INTRODUCTION

The LAM's micro-mirror array consists in several thousand micro-mirrors cells arranged in a rectangular shape. From a general point of view, the micromirror cell has two parts that are assembled, Figure 1.1, [Canonica, 2012]. The mirror part is composed of a mirror, a system of beams and a frame. The mirror is attached to the frame by the suspended beams. Two landing beams are placed on the tips of the suspended beams to prevent the mirror from a short-circuit generation in contact with the electrode during actuation. And, the stopper beam placed under the frame provides a precise tilt angle after actuation. The electrode part is composed of an electrode, two landing pads and two pillars. The electrode allows to apply an electrostatic force attracting the micromirror. The two landing pads define the landing regions of the landing beams and the pillars determine a precise electrostatic gap and a stiff link between the mirror and the electrode.

The concept of actuation of the micromirror cell is based on the double plate electrostatic actuator. At rest, when no voltage is applied, the micromirror is held in a flat position by the suspended beams. When a voltage difference is applied between the micromirror and the electrode, an electrostatic force is generated, resulting in the attraction of the micromirror toward the electrode. Before the pull-in voltage, the micromirror moves a little allowing the angle to be set to a few degrees. At the pull-in voltage, the force increases and the micromirror snaps toward the electrode. During this motion, it touches its landing pads first and touches its stopper beam latter, Figure 1.2(a), or it touches its stopper beam and lands on its landing pads, Figure 1.2(b). After pull-in, the micromirror is fixed at a precise tilt angle. When the voltage is reduced, the micromirror angle remains constant until the mirror detaches from its stopper beam and increased its tilt angle. 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 concepts for addressing individual micromirror cells are based on a line-column algorithm using the property of the tilt angle/voltage hysteresis. The electrode under the micromirrors is placed in a direction perpendicular to the frame, Figure 1.3. Each line of micromirrors and each electrode is connected to an individual voltage sources. To actuate individually one micromirror of the array, two different voltages are set on its line and column, while to address individually several micromirrors, the property of the tilt angle/voltage hysteresis is used, Figure 1.4. Since the electrostatic force is inversely

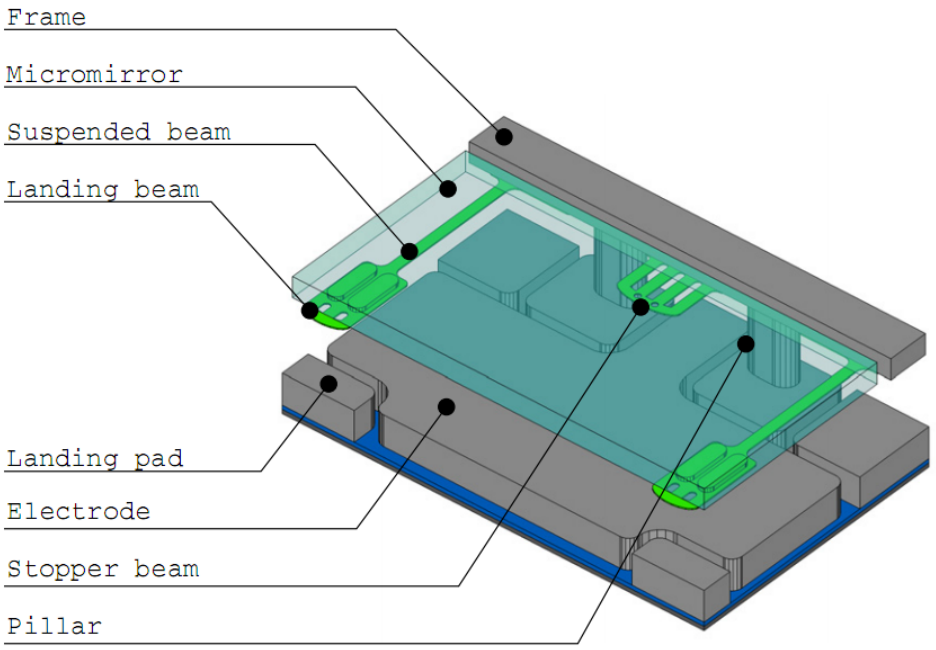


Figure 1.1: The geometry of a micromirror cell [Canonica, 2012]. For clarity, the micromirror is drawn as transparent.

proportional to the distance between the two conductors, for any voltage lower than the pull-in voltage, the micromirror has two different tilt angles; one occurs before and another one occurs after the collapse angle. And accordingly, it only require a lower voltage than the pull-in voltage to keep titling angle constant.

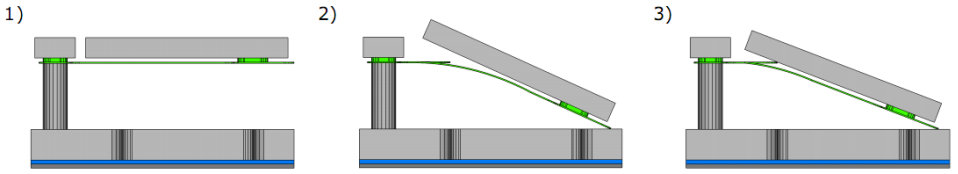
With this property, it allows to avoid the full actuation of neighboring micromirrors when one micromirrors is being addressed and also to actuate the neighboring elements without modifying the tilt angle of the full tilted mirror.

The aim of this chapter is the statement of the mathematical models describing the physical phenomena occurring during the actuation of the micro-mirror. Due to the large deformation of the suspending beams, the structural deformations are governed by the system of nonlinear elasticity. The electric effect are governed by the electrostatic equation that is the cause of the electrostatic force operating on all mechanical parts. Finally, the heat transfer are modeled by the heat equation which is coupled to the system of elasticity. The strong forms of these equations are recalled as well as their weak forms.

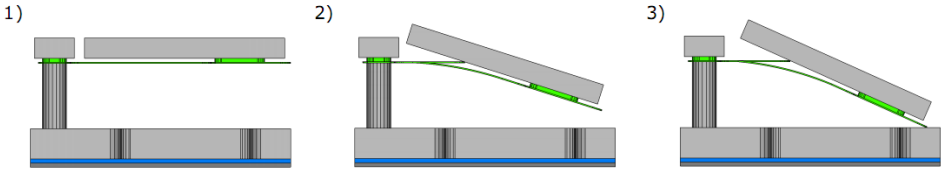
1.2/ PHYSICAL PHENOMENA OCCURRING IN A MICROMIRROR CELL

The deformations of the beams and the mirror are because of electrostatic forces caused by different applied potentials. A Computational analysis of this phenomenon requires the coupling of an electrostatic analysis and a mechanical analysis.

Electrostatics problem: To start electrostatics analysis, consider a micromirror cell as shown in Figure 1.5, Ω_{Vac} denotes the domain occupied by vacuum between the mirror



(a) First sequence: at pull-in voltage the micromirror is snapped towards its electrode. During this motion, the micromirror first touches its landing pads (2) and then its stopper beam(3).



(b) Second sequence: at pull-in voltage the micromirror is snapped towards its electrode. During this motion, the micromirror first touches its stopper beam(2) and then its landing pads (3).

Figure 1.2: Two actuation sequences are observed depending on the dimensions of the beams. After pull-in, the micromirror has a precise tilt angle due to its contact with its stopper beam and landing pads [Canonica, 2012].

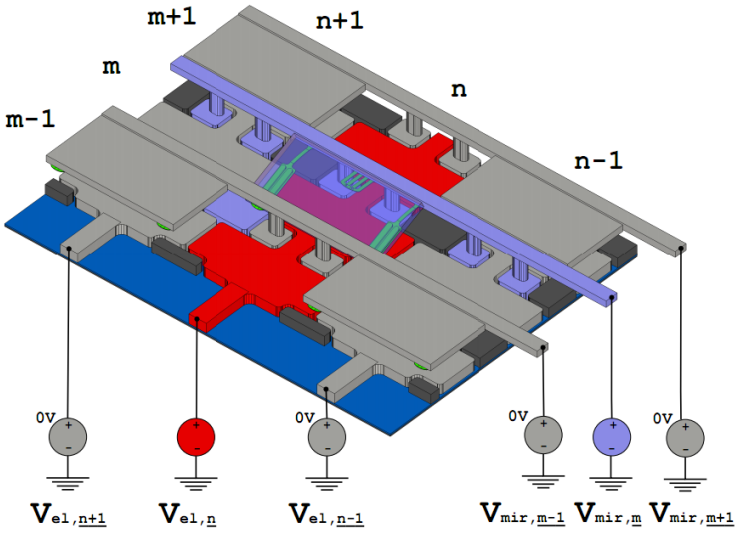


Figure 1.3: Individually addressing using line-column method. Each line of micromirrors and each electrode is controlled by an individual voltage source. For example, the mirror (m,n) is addressed by the voltage of its line (violet part) and the voltage of its electrode (red part). Some micromirrors are removed for better clarity.

and the electrode, ϕ denotes the electric potential, V_1 and V_2 denote two different potentials imposed on the electrode part and the mirror part, i.e. V_1 is applied to the surface of the electrode while V_2 is applied to the surfaces of the frame, of the two pillars and of the mirror through the beams, of the landing beams and of the golden parts. The notations $\Gamma^e_{0,1}$, $\Gamma^e_{0,2}$ denote the boundaries of conductors where $\phi = V_1$ and $\phi = V_2$ and Γ^e_1 denotes

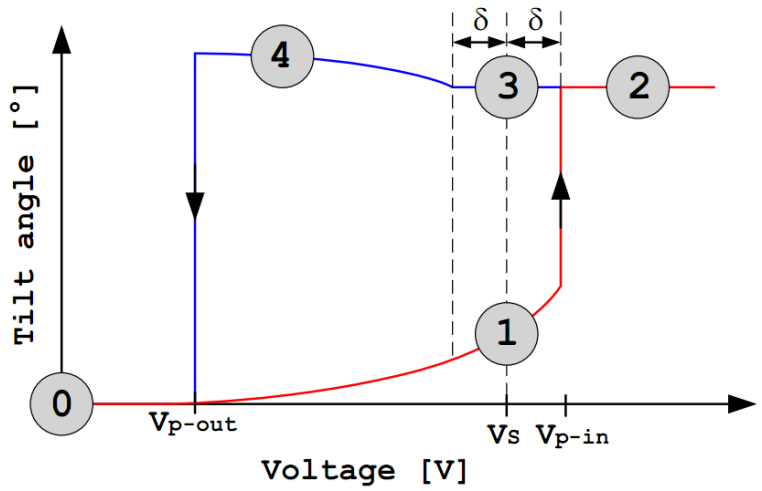


Figure 1.4: Tilt angle/voltage hysteresis of the micromirror. For some voltage Vs the micromirror can be in either position 1 or 3. V_{p-in} is the pull-in voltage; at this voltage the mirror snaps toward the electrode. V_{p-out} is the pull-out voltage at this voltage the mirror returns to its rest position. The voltage VT is defined by $VS - \delta < VT < VS$. For individual addressing using a line-column algorithm, the tilt angles in positions 2 and 3 have to be the same, and the range of voltage that constitutes position 3 has to be as large as possible, to compensate for fabrication variations [Canonica, 2012]

the lateral boundaries of Ω_{Vac}, it can have Neumann condition for simplification or periodic condition in voltage for the case of simulating a cell of a two dimension array.

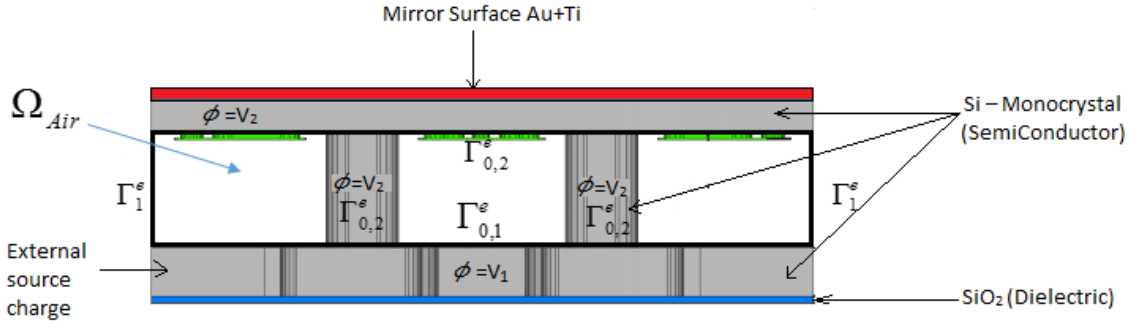


Figure 1.5: The function ϕ has the space between the mirror and the electrode Ω_{Vac} as domain. The Dirichlet boundary conditions are defined such as $\phi = V_1$ on $\Gamma^e_{0,1}$ which is the surface of the electrode and $\phi = V_2$ on $\Gamma^e_{0,2}$ which is a combination of the surfaces of micromirror, of the pillars, of the beams, of the frame and of the golden parts. The lateral boundaries Γ^e_1 can have the Neumann condition for individual simulation or periodic conditions for simulation of an array.

Mechanical problem: Since different voltages are applied, there is an electrostatic force, denoted by $\mathbf{f}^{elec}$, prescribed on the surfaces of all components of the mirror part and the electrode as shown in Figure 1.6 and 1.7. Let Ω^m denote the domains of all components

which accompany the mechanical action, i.e.

$$\Omega^m = \Omega_{Frame} \cup \Omega_{Beams} \cup \Omega_{Mirror},$$

Γ_0^m denote the boundary where clamping conditions are applied, and Γ_1^m denote the boundary where the electric forces are applied.

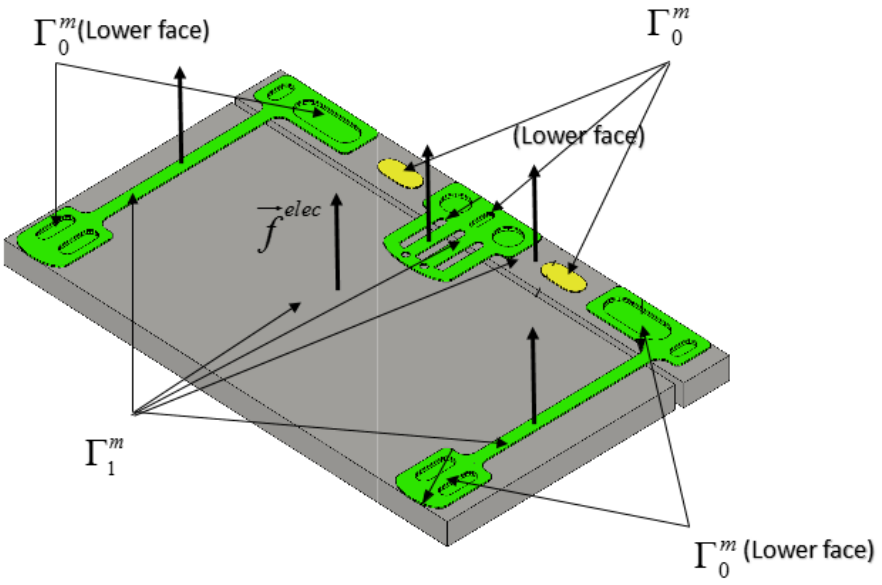


Figure 1.6: Electrostatic force $\mathbf{f}^{elec}$ applied on the mirror part such as on the surfaces of the frame, of the beams and of the mirror.

Time dependent thermoelastic problems: In addition, we consider linear time dependent thermoelastic problems as shown in Figure 1.8. Let Ω^{ther} be the domain which contains all the components of the micromirror except Ω_{Vac}, $r(t, \mathbf{x})$ be total external volume heat source given by

$$r(t, \mathbf{x}) = r_{radiation} \tag{1.1}$$

where $r_{radiation}$ is the external source. The notation θ denotes the unknown temperature while θ_0 denotes the constant reference temperature, $\hat{\theta}$ denotes the difference of the temperature given by $\hat{\theta} = \theta - \theta_0$. The notation Γ_0^{ther} denote the boundary where the reference temperature θ_0 is imposed and Γ_1^{ther} denote the boundary where the outer heat source is applied. The micromirror is kept in vacuum condition, so that there is no heat exchange between the body and the outside environment.

1.3/ GOVERNING EQUATIONS

We provide the models corresponding to the stated problems in the previous section.

Electrostatic problem: The governing equation for electrostatic analysis is given by

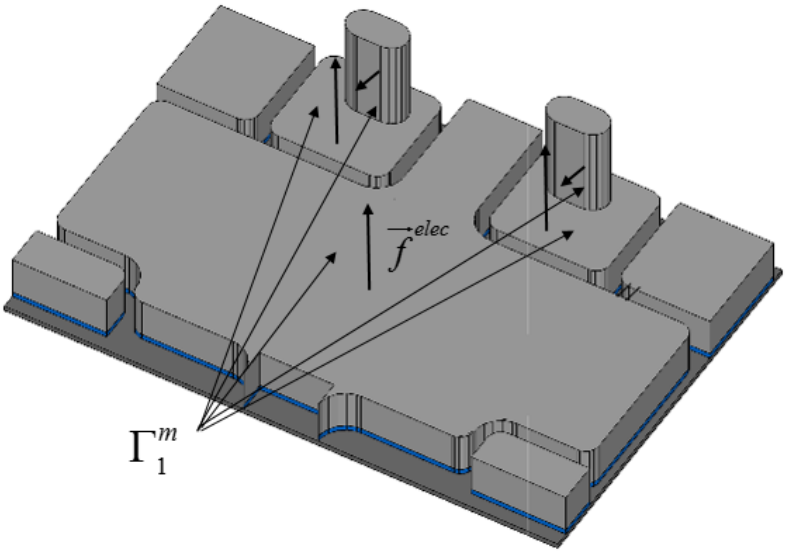


Figure 1.7: Electrostatic force $\mathbf{f}^{elec}$ applied on the surface of the electrode and the pillar.

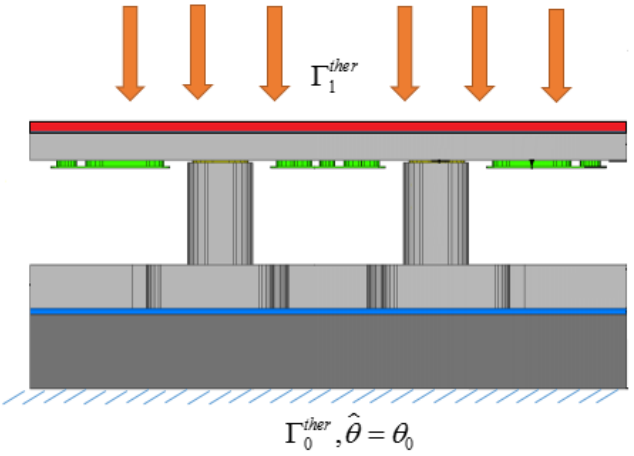


Figure 1.8: Ω^{ther} and its thermal boundaries condition.

[Griffiths et al., 1999] (page 69)

$$\begin{cases} \mathrm{div}_x\left(\nabla_x \phi\right) = 0 & \text{in} \quad \Omega_{Vac} \\ \phi = V_1 & \text{on} \quad \Gamma_{0,1}^e \\ \phi = V_2 & \text{on} \quad \Gamma_{0,2}^e \\ \nabla_{\mathbf{x}} \phi \cdot \mathbf{n} = 0 & \text{on} \quad \Gamma_1^e \end{cases}, \tag{1.2}$$

where $\mathbf{n}$ is the unit outward normal vector on the surface of conductors. In addition, electrostatic force $\mathbf{f}^{elec}$ per unit area applied on surface of conductors is given by

[Kovetz, 2000] (page 225),

$$\mathbf{f}^{elec} = -\left(\frac{1}{2}\mathbf{E}^{elec}\cdot\mathbf{D}\right)\mathbf{n} + \left(\mathbf{n}\cdot\mathbf{E}^{elec}\right)\mathbf{D}^{T}, \tag{1.3}$$

where the electrical field $\mathbf{E}^{elec}$ is given by

$$\mathbf{E}^{elec} = -\nabla_{\mathbf{x}}\phi, \tag{1.4}$$

and $\mathbf{D}$ is the charge potential given by

$$\mathbf{D} = \epsilon_0\mathbf{E}^{elec}, \tag{1.5}$$

with ϵ_0 the permittivity in vacuum.

Mechanical problem: We first introduce the linearized static homogeneous isotropic elasticity problem coupled with the electrostatic problem. The governing equation using the Euler description of the mechanical analysis is given as [Mase et al., 1970]

$$\begin{cases} -\nabla_{\mathbf{x}}\cdot\boldsymbol{\Sigma} = \mathbf{0} & \text{in} \quad \Omega^m \\ \mathbf{u} = \mathbf{0} & \text{on} \quad \Gamma_0^m \\ \boldsymbol{\Sigma}\mathbf{n} = \mathbf{f}^{elec} & \text{on} \quad \Gamma_1^m \end{cases}, \tag{1.6}$$

where the body force per unit volume in the equilibrium equation is assumed to be negligible, $\mathbf{f}^{elec}$ is given by (1.3), $\mathbf{u} = (u_i)$ is the mechanical displacement vector, $\boldsymbol{\Sigma} = (\sigma_{ij})$ is the Cauchy stress tensor given by the Hooke's law

$$\boldsymbol{\Sigma} = \lambda\mathbf{I}_{\epsilon} + 2\mu\mathbf{E}, \tag{1.7}$$

where λ and μ are Lamé constants of a homogeneous isotropic elastic material, $\mathbf{E} = (\epsilon_{ij})$ is the infinitesimal strain tensor given by the linearized strain-displacement relation

$$\mathbf{E} = \frac{1}{2}\left(\nabla_{\mathbf{x}}\mathbf{u} + \nabla_{\mathbf{x}}\mathbf{u}^{T}\right), \tag{1.8}$$

where $\mathbf{I}_{\epsilon} = tr(\mathbf{E})\mathbf{I} = \epsilon_{kk}\mathbf{I}$ and $\nabla_{\mathbf{x}}\mathbf{u}$ is Jacobian matrix of $\mathbf{u}$.

Secondly, we introduce the static nonlinear-elasticity system coupled with electrostatics. We observe that the software suite COMSOL and general commercial FEM packages using the Finite Element Method (FEM) or Boundary Element Method (BEM) based methods for the analysis of MEMS, such as MEMCAD and FASTCAP, perform a mechanical analysis on the undeformed geometry of the device using a Lagrangian approach and the electrostatic analysis is performed on the deformed geometry. This semi-Lagrangian scheme is normally preferred to the full-Eulerian schemes in nonlinear analysis, since it allows to avoid the need to update the geometry of the conductors, therefore avoids to remesh the surfaces and recompute interpolation functions whenever the geometry changes. The governing equations for nonlinear mechanical and electrostatic analysis using semi-Lagrangian description for a micromirror cell is given by [Ciarlet, 1993], [Fu et al., 2001] and [Li et al., 2003]

$$\begin{cases} -\text{div}\,(\mathbf{FS}) = \mathbf{0} & \text{in} \quad \Omega^m \\ \mathbf{u} = \mathbf{0} & \text{on} \quad \Gamma_0^m \\ \mathbf{P}\cdot\mathbf{N} = \mathbf{f}^{elec} & \text{on} \quad \Gamma_1^m \end{cases} \tag{1.9}$$

where $\mathbf{N}$ is the unit outward normal vector in the initial configuration, $\mathbf{F}$ is the deformation gradient given by

$$\mathbf{F} = \mathbf{I} + \nabla \mathbf{u}, \tag{1.10}$$

in which $\mathbf{I}$ is identity tensor, $\mathbf{S}$ is the second Piola-Kirchhoff stress given by

$$\mathbf{S} = \mathbf{C}\mathbf{E}, \tag{1.11}$$

where $\mathbf{C}$ is the material tensor and $\mathbf{E}$ is the Green-Lagrangian strain tensor given by

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right), \tag{1.12}$$

and $\mathbf{P}$ is the fist Piola-Kirchhoff stress tensor given by

$$\mathbf{P} = \mathbf{F}\mathbf{S}. \tag{1.13}$$

Finally, the dynamic model of the coupled electrical-mechanical nature of micromirror cell can be established according to [De et al., 2004].

Time dependent thermoelastic problem: The governing equation for coupled thermal-elastic problem is given by

$$\begin{cases} C_E \partial_t \hat{\theta} - \theta_0 \mathbf{M} : \partial_t \mathbf{E} + \nabla_{\mathbf{x}} \cdot \mathbf{q} = r & \text{in} \quad \Omega^{ther} \\ \mathbf{q} \cdot \mathbf{n} = \mathbf{g}_s & \text{on} \quad \partial\Omega^{ther} \backslash \Gamma_1^{ther} \\ \hat{\theta} = 0 & \text{on} \quad \Gamma_0^{ther} \end{cases} \tag{1.14}$$

where $\mathbf{M}$ is the stress-temperature tensor, C_E is the specific heat at zero, $\mathbf{q}$ is the heat flux given by the Fourier's law

$$\mathbf{q} = -\mathbf{K}\nabla_{\mathbf{x}}\hat{\theta}, \tag{1.15}$$

and $\mathbf{K}$ is the thermal conductivity.

1.4/ VARIATIONAL FORMULATIONS

Electrostatic problem: We define a function $\overline{\phi} \in H^1(\Omega_{Vac})$ that gets the values of ϕ on boundaries $\Gamma^e_{0,1}, \Gamma^e_{0,2}$, i.e.

$$\overline{\phi} = \begin{cases} V_1 & \text{on} \quad \Gamma^e_{0,1} \\ V_2 & \text{on} \quad \Gamma^e_{0,2} \end{cases}.$$

Assume that $\phi = \widetilde{\phi} + \overline{\phi}$, the equation (1.2) can be rewritten in indicial form

$$\begin{cases} -\partial_{x_i}(\partial_{x_i}\widetilde{\phi} + \partial_{x_i}\overline{\phi}) = 0 & \text{in} \quad \Omega_{Vac} \\ \widetilde{\phi} = 0 & \text{on} \quad \Gamma^e_{0,1} \cup \Gamma^e_{0,2} \\ \partial_{x_i}\widetilde{\phi} n_i = -\partial_{x_i}\overline{\phi} n_i & \text{on} \quad \Gamma^e_1 \end{cases}. \tag{1.16}$$

We present an appropriate space for the function $\widetilde{\phi}$, i.e.

$$\widetilde{\phi} \in H^1_{\Gamma^e_{0,1} \cup \Gamma^e_{0,2}}(\Omega_{Vac}) = \{v \in H^1(\Omega_{Vac}) \mid v = 0 \text{ on } \Gamma^e_{0,1} \cup \Gamma^e_{0,2}\}.$$

Multiplying a test function $v \in H^1_{\Gamma^e_{0,1} \cup \Gamma^e_{0,2}}(\Omega_{Vac})$ to (1.16), and integrating it over the domain Ω_{Air}, we obtain

$$-\int_{\Omega_{Vac}} \left(\frac{\partial^2 \widetilde{\phi}}{\partial x_i^2} + \frac{\partial^2 \overline{\phi}}{\partial x_i^2} \right) v \, \mathrm{d}x = 0.$$

Applying the Green's first identity with $\partial\Omega_{Vac} = \Gamma^e_{0,1} \cup \Gamma^e_{0,2} \cup \Gamma^e_1$, we get

$$\int_{\Omega_{Vac}} \frac{\partial\widetilde{\phi}}{\partial x_i}\frac{\partial v}{\partial x_i} + \frac{\partial\overline{\phi}}{\partial x_i}\frac{\partial v}{\partial x_i}\,\mathrm{d}x = \int_{\Gamma^e_{0,1}\cup\Gamma^e_{0,2}\cup\Gamma^e_1} \left(tr\left(\frac{\partial\widetilde{\phi}}{\partial x_i}\right) + tr\left(\frac{\partial\overline{\phi}}{\partial x_i}\right)\right) tr\,(v)\, n_i\,\mathrm{d}s\,(x)\,,$$

and from $v = 0$ on $\Gamma^e_{0,1} \cup \Gamma^e_{0,2}$ and $\left(\partial_{x_i}\widetilde{\phi} + \partial_{x_i}\overline{\phi}\right) n_i = 0$ on Γ^e_1, the weak form of the problem is stated as finding $\widetilde{\phi} \in H^1_{\Gamma^e_{0,1}\cup\Gamma^e_{0,2}}(\Omega_{Vac})$ such that

$$\int_{\Omega_{Air}} \frac{\partial\widetilde{\phi}}{\partial x_i}\frac{\partial v}{\partial x_i} + \frac{\partial\overline{\phi}}{\partial x_i}\frac{\partial v}{\partial x_i}\,\mathrm{d}x = 0 \text{ for all } v \in H^1_{\Gamma^e_{0,1}\cup\Gamma^e_{0,2}}(\Omega_{Vac}). \tag{1.17}$$

Mechanical problem: The indicial form of the governing equation of the linear static elasticity problem (1.6) is given by

$$\begin{cases} -\partial_{x_j}\sigma_{ij} = 0 & \text{in} \quad \Omega^m \\ u_i = 0 & \text{on} \quad \Gamma^m_0 \\ \sigma_{ij}n_j = f^{elec}_i & \text{on} \quad \Gamma^m_1 \end{cases}, \tag{1.18}$$

while the indicial form of Hooke's law (1.7) and strain-displacement relation (1.8) are given as

$$\sigma_{ij} = \lambda\delta_{ij}\epsilon_{kk}\,(\mathbf{u}) + 2\mu\epsilon_{ij}\,(\mathbf{u})\,, \tag{1.19}$$

$$\epsilon_{ij}\,(\mathbf{u}) = \frac{1}{2}\left(\partial_{x_j}u_i + \partial_{x_i}u_j\right). \tag{1.20}$$

The appropriate space for the function u_i is

$$u_i \in H^1_{\Gamma^m_0}(\Omega^m) = \{w \in H^1(\Omega^m) : w = 0 \text{ on } \Gamma^m_0\}. \tag{1.21}$$

Multiplying to the equilibrium equation (1.18) by a test function $w_i \in H^1_{\Gamma^m_0}(\Omega^m)$ and integrating it over the domain Ω^m, we get

$$-\int_{\Omega^m} \frac{\partial\sigma_{ij}}{\partial x_j} w_i\,\mathrm{d}x = 0.$$

Applying the chain rule, i.e. $\partial_{x_j}\left(\sigma_{ij}w_i\right) = w_i\partial_{x_j}\sigma_{ij} + \sigma_{ij}\partial_{x_j}w_i$, we obtain

$$-\int_{\Omega^m} \frac{\partial\left(\sigma_{ij}w_i\right)}{\partial x_j} - \sigma_{ij}\frac{\partial w_i}{\partial x_j}\,\mathrm{d}x = 0,$$

or by changing the side, it reads

$$\int_{\Omega^m} \sigma_{ij}\frac{\partial w_i}{\partial x_j}\,\mathrm{d}x = \int_{\Omega^m} \frac{\partial\left(\sigma_{ij}w_i\right)}{\partial x_j}\,\mathrm{d}x.$$

Applying divergence theorem to the right hand side with $\partial\Omega^m = \Gamma^m_0 \cup \Gamma^m_1$ and $w = 0$ on Γ^m_0, $\sigma_{ij}n_j = f^{elec}_i$, we get

$$\int_{\Omega^m} \sigma_{ij}\frac{\partial w_i}{\partial x_j}\,\mathrm{d}x = \int_{\Gamma^m_1} \sigma_{ij}w_i n_j\,\mathrm{d}s\,(x) = \int_{\Gamma^m_1} f^{elec}_i w_i\,\mathrm{d}s\,(x)\,.$$

Applying the Hooke's law (1.19), the weak formulation states as,

$$\int_{\Omega^m} \left(\lambda \delta_{ij}\epsilon_{kk}(\mathbf{u}) + 2\mu\epsilon_{ij}(\mathbf{u})\right) \frac{\partial w_i}{\partial x_j}\,\mathrm{d}x = \int_{\Gamma_1^m} f_i^{elec} w_i\,\mathrm{d}s(x),$$

in which, from definition of $\epsilon_{ij}(\mathbf{w})$, one can prove that $\epsilon_{ij}(\mathbf{w}) = \epsilon_{ji}(\mathbf{w}), \partial_{x_j} w_i + \partial_{x_i} w_j = 2\epsilon_{ij}(\mathbf{w}) = \epsilon_{ij}(\mathbf{w}) + \epsilon_{ji}(\mathbf{w})$, which lead to

$$\epsilon_{ij}(\mathbf{u})\frac{\partial w_i}{\partial x_j} = \epsilon_{ij}(\mathbf{u})\,\epsilon_{ij}(\mathbf{w}) \text{ and } \delta_{ij}\epsilon_{kk}(\mathbf{u})\frac{\partial w_i}{\partial x_j} = \epsilon_{kk}(\mathbf{u})\,\epsilon_{qq}(\mathbf{w}).$$

Applying these properties, the weak form of the problem, ([Ciarlet, 1997]), is stated as finding $u_i \in H^1_{\Gamma_0^m}(\Omega_{Mir})$ such that

$$\int_{\Omega^m} \left[\lambda\epsilon_{kk}(\mathbf{u})\epsilon_{qq}(\mathbf{w}) + 2\mu\epsilon_{ij}(\mathbf{u})\epsilon_{ij}(\mathbf{w})\right]\mathrm{d}x = \int_{\Gamma_1^m} f^{elec} w_i\,\mathrm{d}s(x) \text{ for all } w_i \in H^1_{\Gamma_0^m}(\Omega_{Mir}). \quad (1.22)$$

Time dependent thermoelastic problem: The indicial form of the governing equation (1.14) is given by

$$\begin{cases} C_E\partial_t\widehat{\theta} - \sum_{i,j}\theta_0 M_{ij}\partial_t\epsilon_{ij} + \partial_{x_i}q_i = r & \text{in} \quad \Omega^{ther} \\ q_i n_i = g_i^s & \text{on} \quad \partial\Omega^{ther} \text{ except } \Gamma_1^{ther} \\ \widehat{\theta} = 0 & \text{on} \quad \Gamma_0^{ther} \end{cases}, \quad (1.23)$$

while the indicial form of Fourier's law (1.15) is given by

$$q_i = -k_{ij}\frac{\partial\widehat{\theta}}{\partial x_i}. \quad (1.24)$$

We present an appropriate space for the function $\widehat{\theta}$, i.e

$$\widehat{\theta} \in H^1_{\Gamma_0^{ther}}(\Omega^{ther}) = \left\{\tilde{\theta} \in H^1(\Omega^{ther}) \mid \tilde{\theta} = 0 \text{ on } \Gamma_0^{ther}\right\}. \quad (1.25)$$

Multiplying a test function $\tilde{\theta} \in H^1_{\partial\Omega^{ther}}(\Omega^{ther})$ to (1.14), integrating it over the domain Ω^{ther} and using Fourier's law, it becomes

$$\int_{\Omega^{ther}} \left[C_E\frac{\partial\widehat{\theta}}{\partial t}\tilde{\theta} - \left(\theta_0 M_{ij}\frac{\partial\epsilon_{ij}(\mathbf{u})}{\partial t}\right)\tilde{\theta}\right]\mathrm{d}x - \int_{\Omega^{ther}} \frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial\widehat{\theta}}{\partial x_i}\right)\tilde{\theta}\,\mathrm{d}x = \int_{\Omega^{ther}} r\tilde{\theta}\,\mathrm{d}x.$$

Applying the Green formula with $\partial\Omega^{ther} = \partial\Omega^{ther}\backslash\Gamma_0^{ther} \cup \Gamma_0^{ther}$, $q_i n_i = 0$ on $\partial\Omega^{ther}\backslash\Gamma_0^{ther}$ and $\tilde{\theta} = 0$ on Γ_0^{ther}. The weak form of the problem stated as finding $\widehat{\theta} \in H^1_{\Gamma_0^{ther}}(\Omega^{ther})$ such that

$$\int_{\Omega^{ther}} \left[C_E\frac{\partial\widehat{\theta}}{\partial t}\widehat{\theta}\tilde{\theta} - \left(\theta_0 M_{ij}\frac{\partial\epsilon_{ij}(\mathbf{u})}{\partial t}\right)\tilde{\theta}\right]\mathrm{d}x + \int_{\Omega^{ther}} k_{ij}\frac{\partial\widehat{\theta}}{\partial x_i}\frac{\partial\tilde{\theta}}{\partial x_i}\,\mathrm{d}x = \int_{\Omega^{ther}} r\tilde{\theta}\,\mathrm{d}x, \quad (1.26)$$

$$\text{for all } \tilde{\theta} \in H^1_{\Gamma_0^{ther}}(\Omega^{ther}).$$

2

MICROMIRROR DESIGN SIMULATION RESULT

2.1/ INTRODUCTION

In this chapter, we present simulation results of a line-column-addressed-with-two-landing-beams (LC2) micro-mirror cell carried out with COMSOL Multiphysics. The configuration and the parameters of the cell are taken in [Canonica, 2012]. Thanks to the electromechanics interface, the simulation can take into account the non-linear deformations of the structure and the electrostatic forces generated by the voltage difference between the mirror and the base. Then, it is used to perform the pull-in analysis, that is to predict the point at which the biased system becomes unstable. Due to the highly nonlinear nature of this inverse problem, it must be done with very much care. In particular choosing carefully the initial conditions of the nonlinear solver is mandatory to avoid divergence. The load ramping technique, that is the choice of a sequence of initial displacements with small increments, guaranties the convergence of the pull-in voltage analysis. The displacement at the pulling voltage in about a third of the gap between the two conductors. Besides, all variables in the model are scaled which is mandatory for a better convergence. The swept meshing technique has been used to reduce the size of the mesh in the very thin parts as the suspending beams. It corresponds to create cylindrical elements with triangular bases that are further divided so that all their faces are triangle. However, the total number of mesh elements is still large, up to 30,000 elements. This is due to the presence of several small regions requiring fine meshes: the gaps between the electrodes, the suspending beams, and the silicon dioxyde layer. As a result, a complete pull-in analysis from an hour to a dozen of hours.

Once simulation is available, many optimization can be envisioned, and we propose a list of some of them that have an interest from the designer point of view as: to minimize the restoring force of the beams, to minimize the pull-in voltage, to reach the correct tilt angle of the mirror or to minimize the speed of the mirror leaving the landing pad during the pull-out process. Here, we report results on the minimization of the pull-in voltage depending on the two most influencial parameters, namely the suspending beam thickness and length. Due to the model simulation time, the optimization is conducted on a metamodel based on a sample of 25 simulations.

Organization of the Chapter: The Chapter is structured as follows. In Section 2.2, we introduce the geometrical parameters of all the cell parts and the material coefficients. In Section 2.3, we present results of the simulation including the mechanical deformation of

the beams and the mirror as well as the electric field. In Section 2.4, the principle of the pull-in analysis is detailed. Finally, in Section 2.4.3, the optimization objectives are listed together with the corresponding design variables, constraints and trade-offs. The chapter ends with the optimization result for the pull-in voltage.

2.2/ PARAMETERS

The mirror, the frame, the stopper beam and the electrode are made from single crystal isotropic silicon (Si). The suspended beam, the anchor of the stopper beam, the anchor of the suspended beams and the landing beam are made from polycrystalline silicon. And the assembly is made from *gold*. They are described in Figures 2.1, 2.2, 2.3, 2.4, 2.5, 2.6 and in table 2.1.

In the table, the parameter $mpover$ is used to oversize the photolithography mask of the polysilicon layer to compensate the over-etching occuring during the patterning of the beams; if the beam is composed by n sub-beams, the width of the beam is calculated by

$$mbeamy = n \times msuy + (n-1) \times (0.5 \times msuy + 2 \times mpover)\,.$$

The width of the stopper beam, $msty$, is given as $msty = mbeamy\,(4) + msuy = 5 \times msuy + 3 \times (0.5 \times msuy + 2 \times mpover)$ the length of the stopper beam is given as $mstx = 25\mu m$. The gap between the sub-beams is set to $gap = 0.5 \times msuy + 2 \times mpover$. The width of the anchor, $x1$, the length of the anchor $x2$ and the parameter $x4$ are given as $x1 = mf - 2 \times (g1 + g3)\,, x2 = msty - 2 * g3$ and $x4 = msux - g3$. The parameters $x5$ is given as

$$\begin{aligned} x5 &= msuy + mstsy + mstsy + 4 \times msuy + mpover \\ &= 5 \times msuy + 2 \times mstsy + mpover. \end{aligned}$$

The parameter $matby$ is given as $matby = (mlay - msuy)\,/2$. The parameter $mlay$ and the distance between the electrode and the tilted edge of the micromirror after actuation are given as

$$mlay = mbeamy\,(3) = 3 \times msuy + 2 \times (0.5 \times msuy + 2 \times mpover)\,,$$

$$d_{mee} = d_1 + d_2 = \frac{ox + poly}{\cos(\alpha)} + (mlax - metax)\,(\sin(\alpha))\,.$$

The length and the width of the gold pad are given as $max = x3 - g1$ and $may = epix - g1$. The parameter $x3$ and $epiz$ of the pillars are given as

$$x3 = my/2 - metay - msbay - g2 - g2 - msty/2 \text{ and } epiz = 35\mu m.$$

The parameters $XPitch = mx + mg + mf + egav$ and $x6 = XPitch - 4 \times egav - ex$.

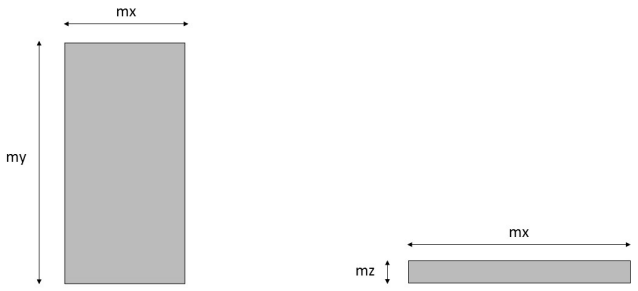


Figure 2.1: Description of the parameters used to model the mirror of the micro-mirror cell.

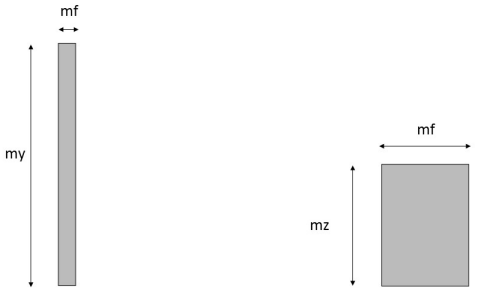


Figure 2.2: Description of the parameters used to model the frame of the micro-mirror cell.

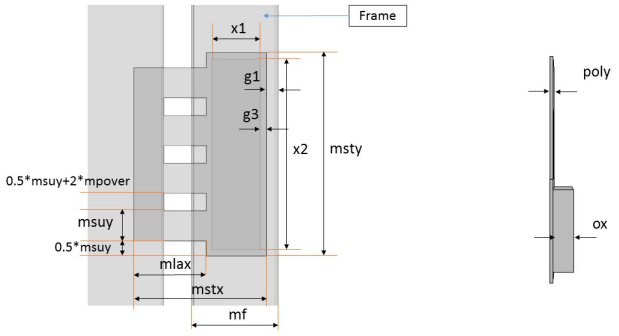


Figure 2.3: Description of the parameters used to model the stopper beam of the micro-mirror cell.

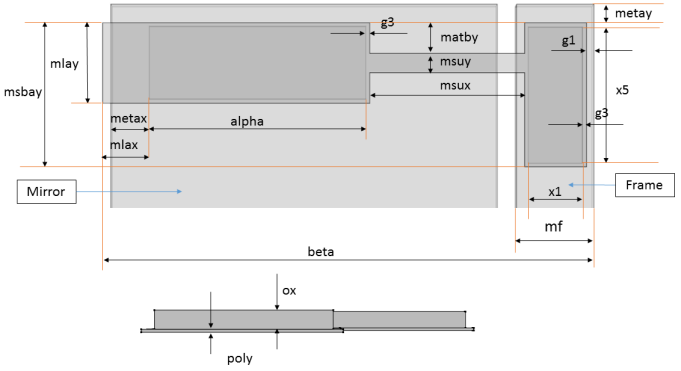


Figure 2.4: Description of the parameters used to model the beam of the micro-mirror cell.

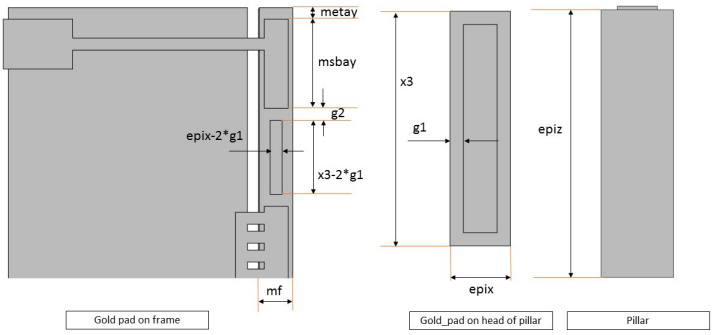


Figure 2.5: Description of the parameters used to model the golden pad and the pillar of the micro-mirror cell.

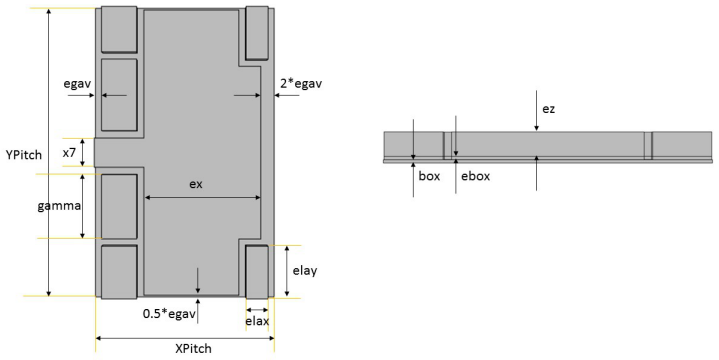


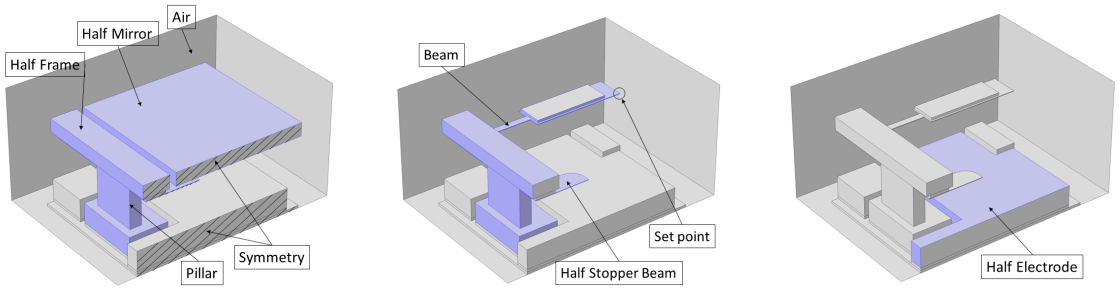
Figure 2.6: Description of the parameters used to model the electrode of the micro-mirror cell.

2.3/ Stationary Simulation

2.3.1/ Model Definition

Because the geometry is symmetric only half of the micro-mirror cell needs to modeled. The micro-mirror is surrounded by a vacuum domain that is an electrical insulator. The cell is fixed at its bottom surface to the substrate. The voltages of the upper and lower electrodes are imposed at VM and VE respectively. As a result, an electrostatic force is generated which bends the mirror toward the electrode. As the beams bends, the geometry of the air gap changes continuously, resulting in a change in the electric field between the electrodes, and therefore, strengthening the electrostatic force. This coupled physics is handled thanks to an interface called *Electromechanics* in COMSOL. The geometry and physics settings are in Figure 2.7 and a simulation result for $VM = 60$ V, $VE = -30$

V is shown in Figure 2.8. The instructions for generating this simulation are in Appendix A.1.



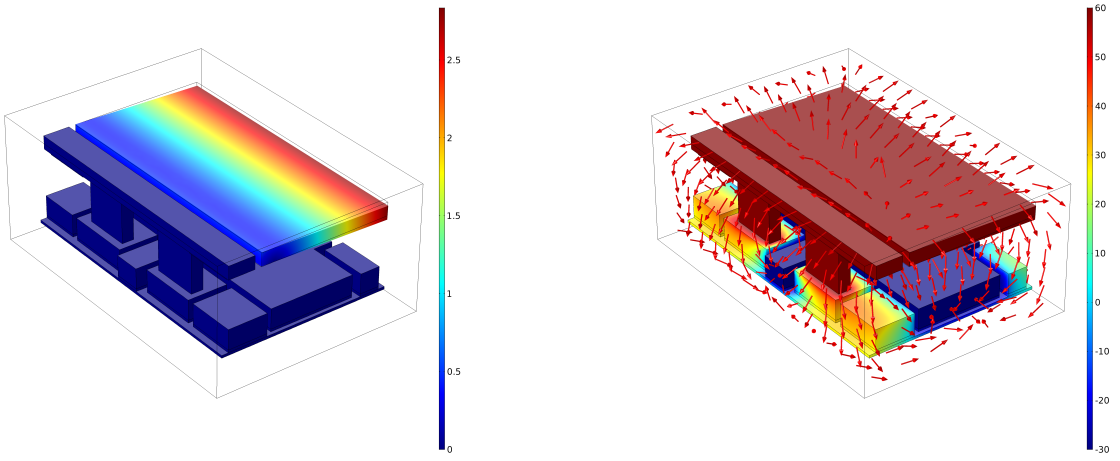
(a) The potential VM is applied on the boundaries of the mirror, the frame, the golden pad and the pillar.

(b) The potential VM is applied on the boundaries of the beam and the stopper beam.

(c) The potential VE is applied on the boundaries of the electrode.

Figure 2.7: Model geometry of half micro-mirror cell. The model uses symmetry on the zx-plane. The first and the second figures depict the applied potential VM on the upper part while the third figure shows the applied potential VM on the lower part.

2.3.2/ RESULTS



(a) The surface displacement field, Z component, caused by $VM = 60$ V and $VE = -30$ V.

(b) The electric potential (color) and the electric field (arrows) at various cross sections through the cell.

Figure 2.8: The displacement field and electric field caused by the voltages $VM = 60$ V and $VE = -30$ V.

2.4/ PULL-IN ANALYSIS

2.4.1/ MODEL DEFINITION

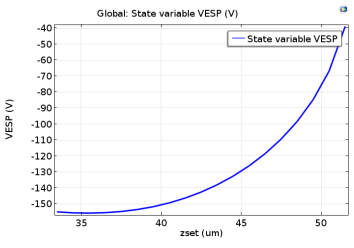
Computing the pull-in voltage is done by solving an inverse problem. The pull-in voltage is found by imposing the mirror end at successive predefined positions starting from the rest position and ending to a displacement equal to the third of the gap. For each position, the voltage is adjusted to cancel the force exerted by imposing the displacement. The choice of an initial displacement close to zero and of a sufficiently small displacement step are to guaranty correct convergence of the nonlinear solver, provided that each new computation starts with an initial condition built from the previous result.

The higher position requires the largest voltage, but in turn, the lower position increases the electrostatic force and thereby requires a lower voltage. As a result, the plot of the voltages versus the positions is a parabola in which the maximum corresponds to the pull-in voltage.

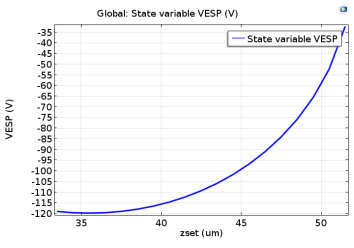
Let call *set point* the point at the end of the landing beam represented in Figure 2.7 (b). The hight of the set-point is calculated by taking **the integral of** z **at itself** which is given by integration operator $(intop1\,(z))$ in COMSOL. The potential VM is set to 1 instead of 0 to avoid an overflow problem. **In the following, the potential** VE **is denoted** $VESP$. The nominal gap between the conductors being $g_0 = 33\ \mu m$, the list of positions is set to $zset = [51.5 : -1 : 33]$ $[\mu m]$. The problem is stated as finding $VESP$ such that $intop1\,(z) - zset = 0$. Then, the pull-in voltage is $V_p = max(|VESP - VM|)$. The large difference in scale between $zset$ and $VESP$ means that care must be taken with the dependent variable scaling in the solver settings. In this model, the geometry and the displacement are scaled by 10^{-5} while the potential and $VESP$ are scaled by 100.

As an example, Figure 2.9 shows the voltage-displacement curves for the micro-mirror at equilibrium and Table 2.2 summarizes the pull-in voltages as a function of the suspending beam length and thickness. The model instruction is in Appendix A.3.

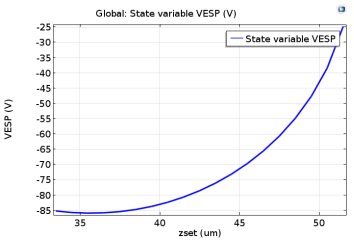
2.4.2/ RESULT



(a) Plot of $VESP$ versus $zset$ in the case of $poly = 600\ nm$, $msux = 40\ \mu m$ and $V_p = 156\ V$.



(b) Plot of $VESP$ versus $zset$ in the case of $poly = 500\ nm$, $msux = 40\ \mu m$ and $V_p = 120\ V$.



(c) Plot of $VESP$ versus $zset$ in the case of $poly = 400\ nm$, $msux = 40\ \mu m$ and $V_p = 87\ V$.

Figure 2.9: Voltage required to achieve a set of displacements versus the target displacement in the case of $poly = 400, 500, 600\ nm$, $msux = 40\ \mu m$.

2.4.3/ DESIGN VARIABLES, OBJECTIVES, CONSTRAINTS AND TRADE-OFF OF OPTIMIZATION PROBLEMS

Design variables and objectives: The restoring force of the beam $FRes$ and the pull-in voltage V_{PI} depend both on the thickness $poly$ and the length $msux$ of the beam. If the beam is too thin or the length of the suspended beam is too long, the restoring force is weak and may not be able to pull the mirror back to its original position. The pull-in voltage is sensitive to the same variables. If the beam is too thick or the length of the suspended beam is too small, it requires a larger voltage to actuate the mirror. The maximum tilt angle of the mirror is sensitive to the height $epiz$ of the pillar. However, the thickness and the length $mstx$ of the stopper beam also determine if the mirror can reach its maximum movement or not. The speed of the mirror during the pull-in and pull-out process Oos relies mainly on the weight of the mirror which depends on its thickness mz.

Constraints: To minimize the cost of actuation, the pull-in voltage of each cell should not be over $140\ V$. For observing faint objects, the micro-mirror array has to achieve the highest contrast, therefore, each cell should have a tilt angle close to $20°$. To avoid breaking the system the on-off speed of each cell has to be lower than $2\ KHz$.

Parameters: For infrared application, the micromirror array is tested in a cryogenic chamber at a temperature of $162.15\ K$ and in vacuum environment. This environmental temperature T influences not only on mechanical body through thermal expansion but also on the electrical resistance of the beam also depending on its doping level. Although the voltage cross-talk between neighbooring cells is not strong enough to fully actuate a mirror, however it has a non-negligible effect on the pull-in voltage. The design variables are summarized in Table 2.3, the list of parameters are given in Table 2.4 and the objectives are given in Table 2.5.

Trade-off: Firstly, there is a trade-off between $FRes$ and V_{PI}. The beam should be thick enough to be able to pull up the mirror. However, the increase of $poly$ leads to the increase of the required voltage and therefore leads to the increase of pull-in voltage. Secondly, there is a trade-off between Ang and V_{PI}. The stopper beam should be thick enough to prevent the variation of the tilt angle. However, the beam has the same thickness as the stopper beam. Like the first trade-off, the increasement of $poly$ will lead to the increase of the pull-in voltage.

Technical points: The **on-off** speed is the necessary voltage square step duration that makes possible the tilt. This should be computed with structural dynamics. The control of MMA is through sequential line/column addressing: the time needed for establishing a full pattern in the array is the number of cells times the “On-Off speed”. The **damping effect** is due to air and therefore is not present in vacuum. In air it will increase the “On-Off speed”. The **“bumping effect”** is the fact that the end of beam has a bounce when it touches the bottom. In air, the damping is limiting this effect to one (or maximum two) bounce(s). In vacuum, it takes time to stabilize the mirror. This effect increases significantly the overall “On-Off speed”. The **doping level** in the **polysilicon beam** is not well controlled. The thin layer is deposited by LPCVD (low pressure chemical vapor deposition). We would have expected a perfect conductor (10^{18} doping level) but due to limitation in the fabrication techniques, it may have a doping level of 10^{16} only resulting in a resistance and therefore a variation of the voltage along the beam, reducing the voltage in the mirror. Then the resistance depends on temperature. Regarding the resistance law, we use bulk values from literature, since we consider that 400 nm is already thick

enough to consider that the boundary effects are not dominating.

Robustness: The optimized objectives must be analyzed versus the uncertainties present in the variables

Optimization of Vpi: the optimization of Vpi consists of a single objective, the two variables $poly$ and $msux$ and one constraint. Despite its apparent simplicity, its direct determination is impractical because of the difficulties discussed in Chapters 1 and 2. An approximated model is built, usually referenced to meta-model, by sampling the two variables $poly$ and $msux$. The meta-model is built with 25 samples corresponding to 5 values of $poly = \{400, 450, 500, 550, 600\}\ nm$ and 5 values of $msux = \{40, 50, 60, 70, 80\}\ \mu m$. It is a fourth order polynomial interpolation. The means square error is 1.8%, Figure 2.10. The graph of the meta-model is shown in Figure 2.11.

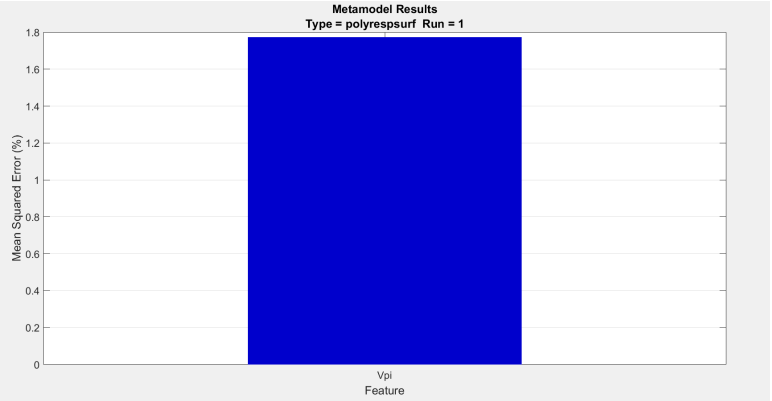


Figure 2.10: Mean squared error of the Meta-model.

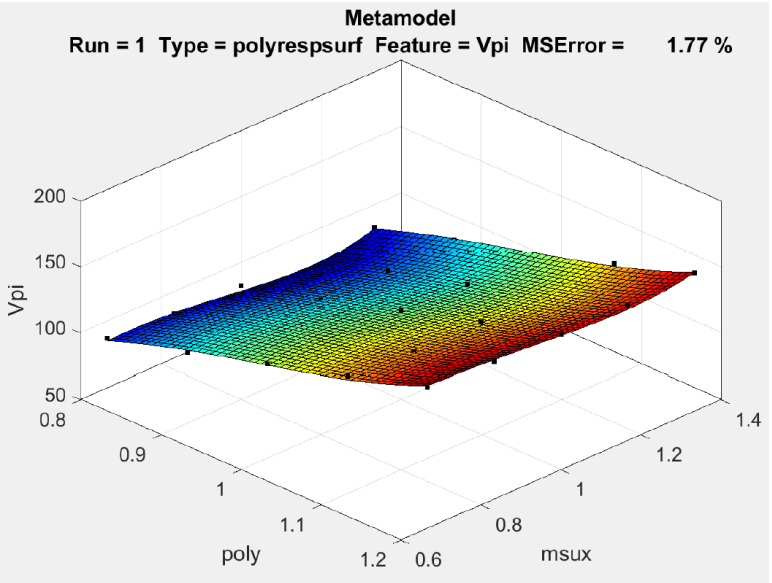


Figure 2.11: The graph of the Meta-model for the feature V_{pi} and its sampling points are shown by black dots.

Result: The minimization of the pull-in voltage V_{pi} is performed using the meta-model. The minimum is reach is for $poly = 400\ nm$ and $msux = 70.16\ \mu m$, see Table 2.6.

2.4.4/ BUMPING EFFECT

The simulations of the mirror bounces are done using the electromechanical interface of COMSOL in the dynamic regime for a two-dimensional geometry. The contact between the landing beams and the landing pads together with the contact between the mirror and the stopper beam are handled by an approximate penalty or barrier method, as described in [El-Zafrany, 1997]. Precisely, nonlinear spring forces F_c are used for modeling the elastic contact between the landing pads and a part of the mirror surface and the stopper beam, see Figure 2.12. When these surfaces are moved away from each other, the springs have a low stiffness and consequently a negligible influence on the deformation of the beam and the mirror. As the gap is reduced the springs become stiffer and resist to the gap closure, see Figure 2.13.

The results of Figure 2.14 show that there is almost no bounce when the beam thickness is lower than $1\mu m$, while the results of Figure 2.15 show a few bounces when the beam thickness is greater than $1\mu m$. This difference of behavior is due to a weaker spring force, indeed the restoring force of a thin $(0.7\mu m)$ beam is weak, leading to a fast tilt actuation and fast stabilization. In the opposite, when the beam thickness is larger $(1\mu m)$, the time scale of the restoring force is longer, and the tilting time is also longer with many bounces before stabilization.

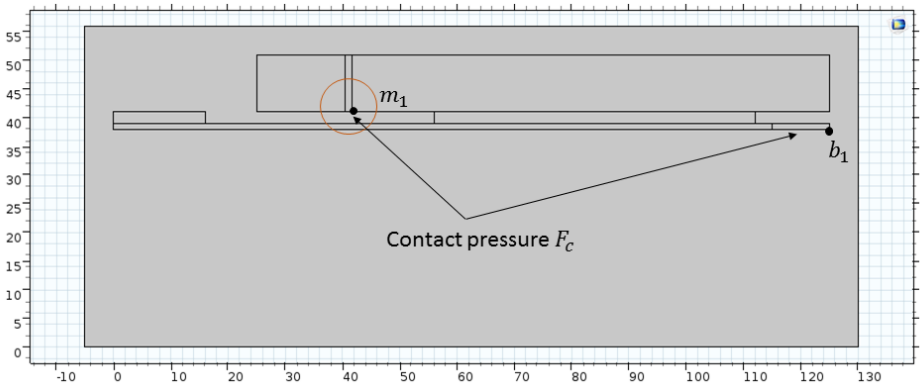


Figure 2.12: A two-dimensional model of Micro-Mirror. A point b_1 is placed at the head of landing beam and a point m_1 is placed on the lower surface of the mirror.

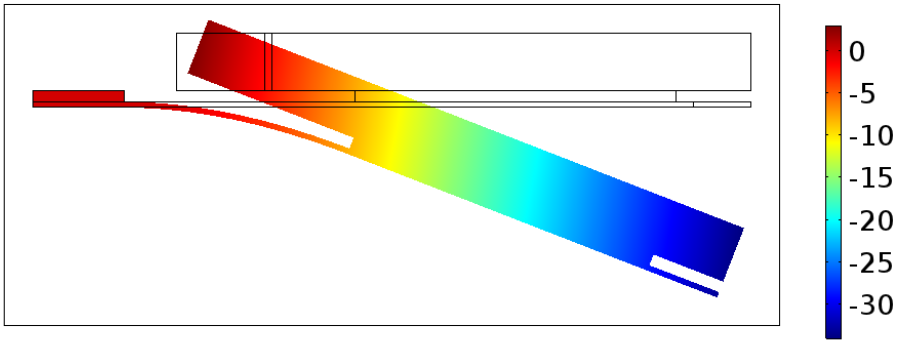


Figure 2.13: Position of the mirror in its maximal displacement for a voltage exceeding the pull-in voltage.

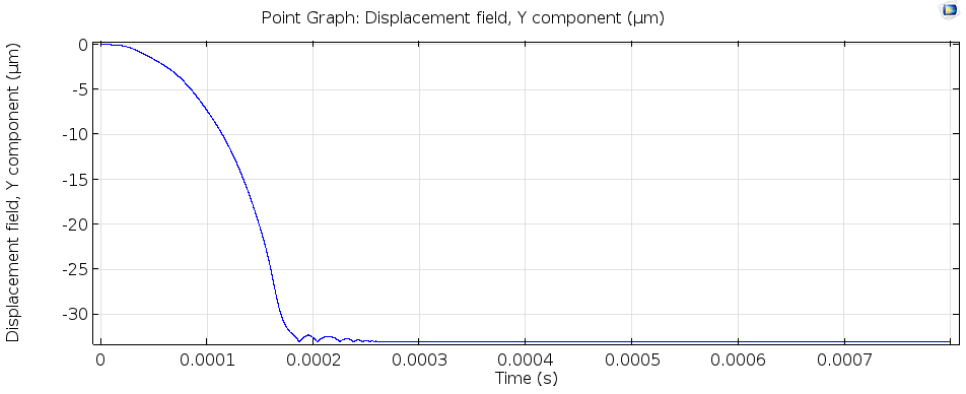


Figure 2.14: Bounces of the mirror materialized by the trajectory of b_1 in the case of a $0.7\mu m$-thick suspended beam.

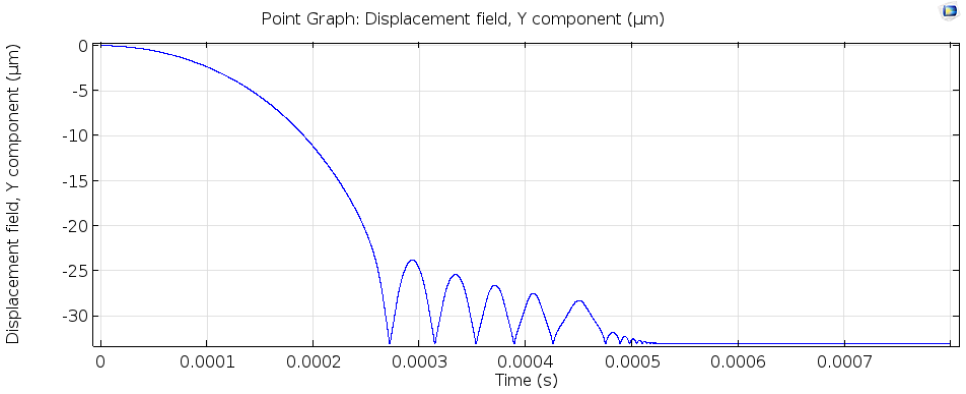


Figure 2.15: Bounces of the mirror materialized by the trajectory of b_1 in the case of a $1\mu m$-thick suspended beam.

	Parameters	Value	Description
1	mx	100 [μm]	Width of the micromirror
2	my	200 [μm]	Length of the micromirror
3	mz	10 [μm]	Thickness of the micromirror
4	mf	20 [μm]	Width of the frame
5	mg	5 [μm]	Width of the gap around the micromirror
6	$msty$		Width of the stopper beam
7	$mstx$	$25\mu m$	Length of the stopper beam
8	$mlax$	$12\mu m$	Length of the landing beam
9	$msuy$	$[3:1:5]\,\mu m$	Width of the Suspended beams
10	$mpover$		
11	$poly$	$[600, 700]\,nm$,	Thickness of the beam
12	$x1$		Width of the anchor of the suspended beam
13	$x2$		Length of the anchor
14	ox	$[1, 2]\,\mu m$	Thickness of the sacrificial layer under the SB
15	$g1$	$2\mu m$	
16	$x4$		devide sub-anchors by $x4$
17	$msux$	$40\mu m$	Length of the Suspended beams
18	$msuy$	$[3, 1, 5]\,\mu m$	Width of the Suspended beams
19	$metay$		Distance from the edge of the frame to the anchor of the SB
20	$matby$		Distance from the anchor to the edge of the SB
21	$mlay$	$mbeamy\,(3)$	landing beams
22	$gold$	$600nm$	Thickness of the gold layer
23	$epix$	$9\mu m$	Width of the pillars (base on mf)
24	may		The length of the gold pad
25	max		The width of the gold pad
26	$x3$		Length of the pillars
27	$epiz$	$35\mu m$	Height of the pillar
28	ex	$80\mu m$	Length of the Electrode
29	ez	$15\mu m$	Thickness of the Electrode
30	$egav$	$5\mu m$	Width of the gap between two parts at different voltage
31	$elax$	$15\mu m$	Width of the landing pads
32	$elay$	$36\mu m$	Length of the landing pads
33	$ebox$	$2\mu m$	Thickness of the BOX of the electrode wafer
34	$x6$		Width of the step sustaining the pillars
35	$x7$	$20\mu m$	Width of the connection between two electrodes
36	$XPitch$		Width of the micromirror cell
37	$YPitch$	my	Length of the micromirror cell
38	box	$2\mu m$	
39	ey	$YPitch - egav$	

Table 2.1: parameters for the mirror, the frame, the golden pad, the beams, the pillar and the electrode.

$Poly/msux$	40 μm	60 μm	80 μm
400 nm	87 V	81 V	82 V
500 nm	120 V	115 V	115 V
600 nm	157 V	150 V	154 V

Table 2.2: the pull-in voltage as a function of the length and width of the suspended beams

Name	**Description**	**Range**	**Uncertainty**
$poly$	Thickness of the beam and the stopper beam	$[400, 500, 600]$ nm	$+-100nm$
$msux$	Length of the suspended beam	$[40, 60, 80]$ μm	$+-1\mu m$
$mstx$	Length of the stopper beam	$[15, 25, 35]$ μm	$+-1\mu m$
$epiz$	Height of the pillar	$[33, 35, 37]$ μm	$+-0.5\mu m$
mz	The thickness of the frame and the mirror	10 μm (fixed)	0 μm (fixed)

Table 2.3: Table of optimization variables.

Name	**Description**	**Range**
T	Temperature	$[30, 77, 150, 300]$ K
V_{nei}	Voltage form neighboor cells	$[0, 150]$ V
Dl	Doping level	

Table 2.4: Table of optimization parameters.

Name	**Description**	**Objective**	**Constraint**	**Sensitive variable**
$FRes$	Restoring force of the beam	$FRes$ should be large enough to be able to recover the origin position of the mirror.		$poly, msux$
V_{PI}	Pull-in voltage	Minimize V_{PI}	$V_{PI} \leq 140V$	$poly, msux$
Ang	Tilt angle of the mirror	Ang should be close to 20^0.	$Ang \geq 18^0$	$poly, mstx, epiz$
Oos	On-Off speed	Minimize Oos	$Oos \leq 2KHz$	mz

Table 2.5: Table of objectives.

Design Variable	**Initial Value**	**Optimal value**
$poly$	500 nm	400 nm
$msux$	60 μm	70.16 μm
Feature		
V_{PI}	115.2 V	74.4 V

Table 2.6: The initial values of $poly$, $msux$ given to optimization procedure and the corresponding V_{PI} compared to their optimal results.

3

MODEL DERIVATION

3.1/ INTRODUCTION

This chapter is devoted to modeling of a micro-mirror array. A full model should include the same physical phenomena as those taken into account in the previous chapters. However, the reported work covers only a part of them. For a first approach, modeling the electrostatic field presents more interest than modeling the elastic deformations. Actually, cross-talk between cells is significantly more present through the electrical field than through the elastical deformations. Given the thiness of the supporting beam and the stiffness of the pillars, the mechanical influence on the neighbouring cells of a mirror tilt is significantly less than the electrical influence.

Thus, in this chapter, we introduce a two-scale models for the electrical field occurring in one- and two-dimensional arrays with a method of proof that follows as much as possible the reference proof implemented in MEMSALab so that it can be expressed as extensions and their combination.

Here, the special feature of the model to be taken into account as an extension is that the electrical potential ϕ is imposed at several (two in this case) positions in each cell. The difference of the imposed electrical potentials is assumed in the range of 1 compared to the ratio ε of the cell length and the array length. Therefore, an *a priori* estimate shows that ϕ and the scaled electrical field $\varepsilon E = -\varepsilon\nabla\phi$ are uniformly bounded in ε. Similar estimates were encountered in the homogenization of the high frequency part of a spectral problem and of the wave equation analyzed by Nguyen Thi Trang in her thesis [Nguyen, 2014]. The analysis were carried out using a very weak formulation instead of a weak formulation. The requirement of this approach is only the weak convergence of ϕ when ε goes to zero. It turns out that the two-scale limit ϕ^0 of ϕ is depending on both the macroscopic and the microscopic variables and satisfies periodic boundary conditions.

In the operational regime, the actuation voltage source is piecewise constant. Precisely, the illuminated parts are actuated with a constant voltage when the others are not. Therefore, only two cell configurations are useful and then only two cell solutions, each depending only on the microscopic variable. Due to the periodicity conditions, the electrical potential is continuous at the interface between cells having the same mirror actuation. However, it is discontinuous at the interface between regions with different actuation. This model weakness is fixed by introducing boundary layer terms. The same solution is brought to insure that the nominal boundary condition are satisfied at the external lateral boundary of the array. Despite the shortage of time, the boundary layer models are not

given in the two-dimensional array case.

As already said, the proof is carried out with the same tools of two-scale convergence as the reference proof in MEMSALab. However, since the convergence of the gradient is not necessary when using the very weak formulation, the proof length is significantly reduced. Precisely, the first part of our proof, before introduction of the boundary layer, is similar to the lemmas that built the two-scale weak formulation and its interpretation. An effort has been made to derive the boundary layer problem in a similar way.

We expect that this approach based on the very weak formulation and boundary layers can be extended to take into account the elasticity and thermal fields. Indeed, for these two problems the fields have imposed values in each cell: the structure is clamped to the base and the base is a thermal sink. It is also worthwhile to mention that the non linearities due to the suspending beams and the electrostatic forces are local to the cell and we do not expect that they have a significant influence on the model structure.

3.2/ One and Two-dimensional arrays

3.2.1/ The governing equations

This subsection is dedicated to the derivation of a two-scale model of the electrostatic field in the two-dimensional Micro Mirror Array (MMA). The description of the physical phenomena was done in Section 1.2. Our framework uses the same approach as in [Yang, 2014] and [Lenczner, 2007], in order to facilitate its implementation as an extension of proof used in `MEMSALab`.

Let $\Omega = \Omega^{mec} \cup \Omega^{vac}$ be the domain occupied by the mechanical body and the surrounding vacuum. It is divided into $n_1 \times n_2$ cells, where $n_1, n_2 \in \mathbb{N}^*$. Each cell is denoted by Ω_c where $c = (c_1, c_2)$ is a multi-indices with $c_1 \in \{1, ..., n_1\}$ and $c_2 \in \{1, ..., n_2\}$. In a cell Ω_c, we denote the upper conductor, the lower conductor and the vacuum domain with the notations $\Omega^{cond}_{1,c}$, $\Omega^{cond}_{2,c}$ and Ω^{vac}_c respectively. Their unions over all cells are denoted by $\Omega^{cond}_1 := \cup_c \Omega^{cond}_{1,c}$, $\Omega^{cond}_2 = \cup_c \Omega^{cond}_{2,c}$ and $\Omega^{vac} = \cup_c \Omega^{vac}_c$. The internal boundary of Ω^{vac} is defined as the union of all the boundaries of the conductors $\Gamma^{vac}_{int} = \partial\Omega^{vac}_{int} = \partial\Omega^{cond}_1 \cup \partial\Omega^{cond}_2$, while the external boundary is the union of the lateral boundaries, the upper surfaces and the lower surfaces $\Gamma^{vac}_{ext} = \Gamma^{vac}_{lat} \cup \Gamma^{vac}_{+\cup-}$. The array is shown in Figure 3.1. The domain Ω^{vac} is also divided into two parts $\Omega^{vac,1}$ and $\Omega^{vac,2}$ that refer to two different imposed voltages V^1 and V^2. Each voltage field V^α, $\alpha \in \{1, 2\}$, takes constant values V^α_1 and V^α_2 on $\partial\Omega^{cond}_1$ and $\partial\Omega^{cond}_2$. The interface between these two parts is denoted by Γ^{vac}_{interf}. The governing equations of the electric potential ϕ in the vacuum domain Ω^{vac} is given as

$$\begin{cases} -\partial_{x_i}(\epsilon\partial_{x_i}\phi) = 0 & \text{in} \quad \Omega^{vac} \\ \phi = V & \text{on} \quad \Gamma^{vac}_{int} \\ \partial_{x_i}\phi n_i = 0 & \text{on} \quad \Gamma^{vac}_{ext} \end{cases} . \tag{3.1}$$

Moreover, on the interface

$$\phi|_{\Omega^{vac,1}} = \phi|_{\Omega^{vac,2}} \text{ and } \partial_{x_i}\phi|_{\Omega^{vac,1}} n^1_i = -\partial_{x_i}\phi|_{\Omega^{vac,2}} n^2_i \quad \text{on} \quad \Gamma^{vac}_{interf}$$

where ϵ is the electrical permittivity in vacuum, V is a single notation to represents the two fields V^α and n^α_i are unit outward normal vectors of $\Omega^{vac,\alpha}$. The weak formulation is

written with the functional spaces,

$$H^1_{\Gamma^{vac}_{int},V}\left(\Omega^{vac}\right) = \left\{v \in H^1\left(\Omega^{vac}\right) \mid v = V \text{ on } \Gamma^{vac}_{int}\right\}, \text{ and} \tag{3.2}$$

$$H^2_{\Gamma^{vac}_{int},\Gamma^{vac}_{ext}}\left(\Omega^{vac}\right) = \left\{v \in H^1\left(\Omega^{vac}\right) \mid v = 0 \text{ on } \Gamma^{vac}_{int},\ \nabla v \cdot \mathbf{n} = 0 \text{ on } \Gamma^{vac}_{ext}\right\}. \tag{3.3}$$

Multiplying the internal equation of (3.1) by the test function $v \in H^1_{\Gamma^{vac}_{int},0}\left(\Omega^{vac}\right)$, integrating over the domain Ω^{vac} and applying the Green formula, we get the weak formulation

$$\int_{\Omega^{vac}} \frac{\partial \phi}{\partial x_i}\frac{\partial v}{\partial x_i}\,\mathrm{d}x = 0 \tag{3.4}$$

which has a unique solution $\phi \in H^1_{\Gamma^{vac}_{int},V}\left(\Omega^{vac}\right)$. The very weak formulation is derived by choosing $v \in H^2_{\Gamma^{vac}_{int},\Gamma^{vac}_{ext}}\left(\Omega^{vac}\right)$. Applying the Green formula,

$$\int_{\Omega^{vac}} \phi\frac{\partial}{\partial x_i}\frac{\partial v}{\partial x_i}\,\mathrm{d}x = \int_{\Gamma^{vac}_{int}} V\frac{\partial v}{\partial x_i}n_i\,\mathrm{d}s\left(x\right). \tag{3.5}$$

We do not further discuss the existence and uniqueness of its solution ϕ in $L^2(\Omega^{vac})$, since to we consider the weak solution is sufficient for our purpose.

3.2.2/ GLOBAL SCALING

We denote by L_1, L_2 and L_3 the width, the length and the thickness of Ω while ℓ_1, ℓ_2 and ℓ_3 represent the corresponding sizes of Ω_c. The small parameter ε is the inverse of the largest number of cells in the two directions x_1 and x_2. In other words, it is the smallest ratio $\min_{\alpha\in\{1,2\}} \ell_\alpha/L_\alpha$. The asymptotic model is obtained by passing to the limit when $\varepsilon \to 0$.

We use the usual big O notation, $f\left(\varepsilon\right) = O\left(g\left(\varepsilon\right)\right)$ for $\varepsilon \in S$ iff there exists a constant $c > 0$ such that $|f\left(\varepsilon\right)| \leq c|g\left(\varepsilon\right)|$ for all $\varepsilon \in S$. Writing $f\left(\varepsilon\right) = O\left(g\left(\varepsilon\right)\right)$ when $\varepsilon \to \varepsilon_0$ is meaning that there exists a constant $\delta > 0$ such that $f\left(\varepsilon\right) = O\left(g\left(\varepsilon\right)\right)$ for all ε such that $|\varepsilon - \varepsilon_0| \leq \delta$. For instance,

$$\frac{\ell_i}{L_\alpha} \sim \varepsilon\ \forall i \in \{1,2,3\} \text{ and } \alpha \in \{1,2\}. \tag{3.6}$$

Since the lengths L_α are very small, and generally even smaller than ε, it is convenient to scale the domains Ω and Ω_c by their order of magnitude L (here $L = 10^{-5}$) yielding the scaled domains $\widehat{\Omega}^\varepsilon$ and $\widehat{\Omega}^\varepsilon_c$. We introduce ε in the notation of the scaled geometries to make explicit their dependency on the order of magnitude, for instance, the thickness of the array are in the range of ε. The scaled sizes and variables is denoted

$$\widehat{L_i} = \frac{L_i}{L},\ \widehat{\ell_i} = \frac{\ell_i}{L} \text{ and } \widehat{x_i^\varepsilon} = \frac{1}{L}x_i \in \widehat{\Omega}^\varepsilon\ \forall i \in \{1,2,3\} \tag{3.7}$$

where $x \in \Omega$. Evidently, the ratio ε is conserved through this transformation, namely $\varepsilon = \min_{\alpha\in\{1,2\}} \widehat{\ell_\alpha}/\widehat{L_\alpha}$. For the sake of simplicity of the presentation, we assume[1] that $\widehat{\ell_i} = \varepsilon$, $\forall i$. Then, the size of the scaled array and cell are

$$|\widehat{\Omega}^\varepsilon| = \prod_i \widehat{L_i} = \varepsilon \text{ and } |\widehat{\Omega}^\varepsilon_c| = \prod_i \widehat{\ell_i} = \varepsilon^3. \tag{3.8}$$

[1] The derivation in the general case follows the same principle.

The vacuum permittivity ($\epsilon_0 = 0.885 \times 10^{-11}\ Fm^{-1}$) is introduced as a scaling of the permittivity and of the potentials ϕ and V,

$$\widehat{\epsilon} = \frac{\epsilon}{\epsilon_0},\ \widehat{\phi^\varepsilon} = \frac{\sqrt{\epsilon_0}}{L}\phi \text{ and } \widehat{V^\varepsilon} = \frac{\sqrt{\epsilon_0}}{L}V \tag{3.9}$$

leading to the scaled electric field (1.4), $\widehat{\mathbf{E}}^{elec,\varepsilon} = \sqrt{\epsilon_0}\mathbf{E}^{elec}$. The electrostatic force (1.3), $\widehat{\mathbf{f}}^{elec,\varepsilon} = \widehat{\mathbf{f}}^{elec}$ is left unchanged. Therefore, the weak formulation of the elasticity system is also left unaffected

$$\int_{\widehat{\Omega}^{\varepsilon,m}} \left[\lambda \epsilon_{kk}\left(\widehat{\mathbf{u}}^\varepsilon\right)\epsilon_{qq}\left(\widehat{\mathbf{w}}^\varepsilon\right) + 2\mu\epsilon_{ij}\left(\widehat{\mathbf{u}}^\varepsilon\right)\epsilon_{ij}\left(\widehat{\mathbf{w}}^\varepsilon\right)\right] \mathrm{d}\widehat{x^\varepsilon} = \int_{\widehat{\Gamma}_1^{\varepsilon,m}} \widehat{f}_i^{elec,\varepsilon}\widehat{w}_i^\varepsilon \,\mathrm{d}s\left(\widehat{x^\varepsilon}\right), \forall \widehat{w}_i^\varepsilon \in H^1_{\widehat{\Gamma}_0^{\varepsilon,m}}\left(\widehat{\Omega}^\varepsilon_{Mir}\right), \tag{3.10}$$

where $\widehat{u}_i^\varepsilon = u_i/L$ and $\widehat{w}_i^\varepsilon = w_i/L$. The scaled system (3.1) is thus

$$\begin{cases} -\partial_{\widehat{x}_i^\varepsilon}\left(\widehat{\epsilon}\partial_{\widehat{x}_i^\varepsilon}\widehat{\phi}^\varepsilon\right) = 0 & \text{in} \quad \widehat{\Omega}^{\varepsilon,vac} \\ \widehat{\phi}^\varepsilon = \widehat{V}^\varepsilon & \text{on} \quad \widehat{\Gamma}^{\varepsilon,vac}_{int} \\ \partial_{\widehat{x}_i^\varepsilon}\widehat{\phi}^\varepsilon n_i^\varepsilon = 0 & \text{on} \quad \widehat{\Gamma}^{\varepsilon,vac}_{ext} \end{cases}, \tag{3.11}$$

when the scaled weak form (3.4) and the very weak form (3.5) are

$$\int_{\widehat{\Omega}^{\varepsilon,vac}} \frac{\partial\widehat{\phi}^\varepsilon}{\partial\widehat{x}_i^\varepsilon}\frac{\partial\widehat{v^\varepsilon}}{\partial\widehat{x}_i^\varepsilon}\,\mathrm{d}\widehat{x^\varepsilon} = 0, \tag{3.12}$$

where $\widehat{v^\varepsilon} \in H^1_{\widehat{\Gamma}^{\varepsilon,vac}_{int},0}\left(\widehat{\Omega}^{\varepsilon,vac}\right)$, and

$$\int_{\widehat{\Omega}^{\varepsilon,vac}} \widehat{\phi}^\varepsilon \frac{\partial}{\partial\widehat{x}_i^\varepsilon}\frac{\partial\widehat{v^\varepsilon}}{\partial\widehat{x}_i^\varepsilon}\,\mathrm{d}\widehat{x^\varepsilon} = \int_{\widehat{\Gamma}^{\varepsilon,vac}_{int}} \widehat{V}^\varepsilon \frac{\partial\widehat{v^\varepsilon}}{\partial\widehat{x}_i^\varepsilon} n_i^\varepsilon \,\mathrm{d}s\left(\widehat{x^\varepsilon}\right) \tag{3.13}$$

where $\widehat{v^\varepsilon} \in H^2_{\widehat{\Gamma}^{\varepsilon,vac}_{int},\widehat{\Gamma}^{\varepsilon,vac}_{ext}}\left(\widehat{\Omega}^{\varepsilon,vac}\right)$.

3.2.3/ Two-Scale Transform for a Thin Region

In this section, we recall the two-scale transform operators or unfolding operators for internal and boundary layer approximations. The definitions are taken and adapted from [Lenczner et al., 2007], [Yang, 2014] and [Nguyen, 2014]. It is illustrated in Figure 3.2. The two-scale model of the micro-mirror array will be derived from the scaled system presented in Section 3.2.2. Besides, for simplification purpose, the hat on each scaled notation will be removed, for examples, $\widehat{\Omega}^\varepsilon$ will be replaced by Ω^ε, $\widehat{x^\varepsilon}$ by x^ε, etc. For the asymptotic model derivation, we assume (without proof) the uniform estimate on the solution

$$\frac{1}{|\Omega^{\varepsilon,vac}|}\|\phi^\varepsilon\|_{L^2(\Omega^{\varepsilon,vac})} \leq C \tag{3.14}$$

where C is a constant independent of ε. Hereafter, we distinguish between the cases of a two-dimensional array and a one-dimensional array.

Two-dimensional array : Considering an arbitrary cell Ω_c^ε of the two-dimensional array with $c = (c_1, c_2)$, any $x^\varepsilon \in \Omega_c^\varepsilon$ satisfies $x_\alpha^\varepsilon \in ((c_\alpha - 1)\varepsilon, c_\alpha\varepsilon)$ for $\alpha \in \{1,2\}$ and $x_3^\varepsilon \in (0,\varepsilon)$. Let $x^{\varepsilon,c}$ be the center of Ω_c^ε, i.e. $x_\alpha^{\varepsilon,c} = (c_\alpha - 1)\varepsilon + \varepsilon/2$ and $x_3^{\varepsilon,c} = \varepsilon/2$. We introduce the expanded and shifted cell Ω^1 defined by $x^1 := \varepsilon^{-1}\left(x^\varepsilon - x^{\varepsilon,c}\right) \in \Omega^1$ for $x^\varepsilon \in \Omega_c^\varepsilon$. The domain

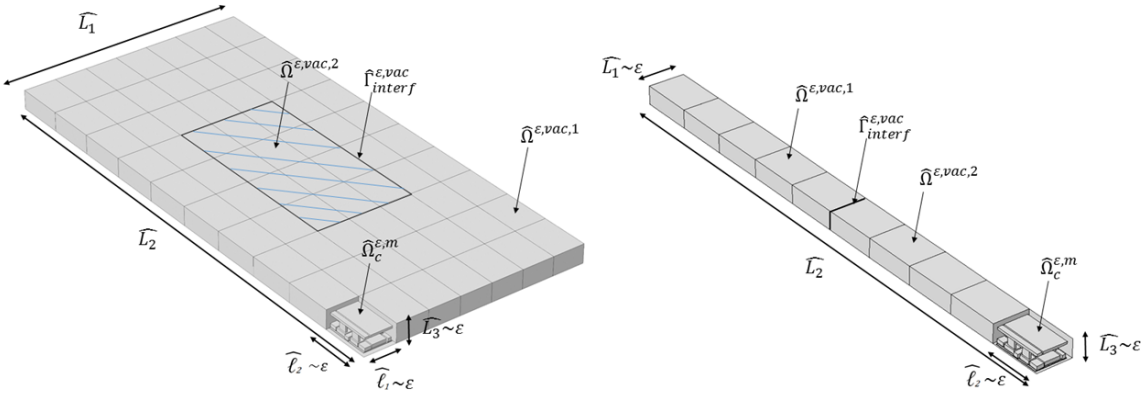


Figure 3.1: A two-dimensional and a one-dimensional micro-mirror arrays with their surrounding vacuum domain, $\widehat{\Omega}^{\varepsilon} = \widehat{\Omega}^{\varepsilon,vac} \cup \widehat{\Omega}^{m,\varepsilon}$. The domain $\widehat{\Omega}^{\varepsilon}$ is divided into two parts $\widehat{\Omega}^{\varepsilon,vac,1}$ and $\widehat{\Omega}^{\varepsilon,vac,2}$ corresponding to the two different voltages. They are separated by the interface $\widehat{\Gamma}^{\varepsilon,vac}_{interf}$. The scaling assumptions on the size of each cell and of the array are also represented.

Ω^1 is called the microscopic domain and is free from ε : $\Omega^1 =]-1/2, 1/2[^3$. Its boundary is split into the external boundary and the internal boundary, i.e. $\partial\Omega^1 = \Gamma^1_{ext} \cup \Gamma^1_{int}$. The external boundary includes the lateral boundary and the upper and the lower boundaries, i.e. $\Gamma^1_{ext} = \Gamma^1_{per} \cup \Gamma^1_{+\cup-}$. The internal boundary includes the boundaries of the two scaled conductors, i.e. $\Gamma^1_{int} = \partial\Omega^{1,cond}_1 \cup \partial\Omega^{1,cond}_2$. Then, we introduce the macroscopic domain $\Omega^\sharp = (0, L_1) \times (0, L_2)$ that can be divided into $n_1 \times n_2$ cells as Ω^ε, $\Omega^\# = \cup_c \Omega^\#_c$. The set of variable indices in Ω^ε and $\Omega^\sharp$ are denoted $I = \{1, 2, 3\}$ and $I^\sharp = \{1, 2\}$. The physical domain, the macroscopic domain and the microscopic domain are represented in Figure 3.2 and their sizes are computed as

$$|\Omega^\#| = 1,\ |\Omega^\#_c| = \varepsilon^2 \text{ and } |\Omega^1| = 1. \tag{3.15}$$

One-dimensional array : The domain Ω^ε is divided into $n \in \mathbb{N}^*$ cells Ω^ε_c for $c \in \{1, ..., n\}$. The coordinates x^ε in a cell Ω^ε_c satisfy $x^\varepsilon_1 \in ((c-1)\,\varepsilon, c\varepsilon)$ while x^ε_2 and $x^\varepsilon_3 \in (0, \varepsilon)$. Therefore, the coordinates of the center $x^{\varepsilon,c}$ of a cell are $x^{\varepsilon,c}_1 = (c-1)\,\varepsilon + \varepsilon/2$ and $x^\varepsilon_2 = x^\varepsilon_3 = \varepsilon/2$. The microscopic domain Ω^1 is defined by $x^1 := \varepsilon^{-1}(x^\varepsilon - x^{\varepsilon,c}) \in \Omega^1, \forall x^\varepsilon \in \Omega^\varepsilon_c$. Consequently, $-1/2 < x^1_i < 1/2, \forall i \in \{1, 2, 3\}$. The domain Ω^1 has its boundary $\partial\Omega^{1,vac}$ split into $\Gamma^1_{int} \cup \Gamma^{1,vac}_{+\cup-} \cup \Gamma^{1,vac}_{per} \cup \Gamma^{1,vac}_{lat}$ where $\Gamma^1_{int} = \Gamma^{1,cond}_{1\cup2}$ and $\Gamma^{1,vac}_{lat}$ is the remaining part. The macroscopic domain is $\Omega^\sharp = (0, L_1)$, the projection of Ω^ε on the line generated by the first unit vector e_1 of the frame. Then, the indices of the coordinates in Ω^ε and $\Omega^\sharp$ vary in $I^\sharp = \{1\}$ and $I = \{1, 2, 3\}$. Figure 3.3 summarizes these domains which order of magnitude in powers of ε are

$$|\Omega^\varepsilon| = \varepsilon^2,\ |\Omega^\varepsilon_c| = \varepsilon^3,\ |\Omega^\sharp| = 1,\ |\Omega^\sharp_c| = \varepsilon \text{ and } |\Omega^1| = 1. \tag{3.16}$$

Boundary Layers: The boundary layer models, which are derived for the one-dimensional array only, require further notations. The two ends of Ω^ε are denoted $\Gamma^\varepsilon_{end,0}$ and $\Gamma^\varepsilon_{end,L_1}$. The boundary layer cell is introduced as $\Omega^{1,+\infty}$ which is defined by shifting and reproducing Ω^1 in the direction of positive x^1_1 by a vector $(p + 1/2, 1/2, 1/2)^T$ for $p = 1, ..., \infty$. Precisely, $\Omega^{1,+\infty} = \cup^{+\infty}_{p=0} \Omega^{1,+\infty}_p$ where $\Omega^{1,+\infty}_p = \Omega^1 + (p + 1/2, 1/2, 1/2)^T$. The domain $\Omega^{1,+\infty}$

with detailed notations regarding its boundary can be found in Figure 3.4. This boundary layer cell is later used in Definition 2 to define both T_b^0 and $T_b^{L_1}$ because of the symmetry in the desgin of each cell.

One-dimensional array with interface : We recall that the domain Ω^ε is divided into two parts $\Omega^{\varepsilon,1}$ and $\Omega^{\varepsilon,2}$ distinguished by their imposed voltages V^α and that their interface is $\Gamma_{interf}^{\varepsilon,vac}$ at L_1^m. The domain and its related notations are shown in Figure 3.3. In order determine the contribution of the boundary layer at the interface, another boundary layer cell $\Omega^{1,\infty}$ is defined as the repetition of Ω^1 which is moved by $(1/2, 1/2, 1/2)$ in the two directions of negative and positive x_1^1. Precisely, $\Omega^{1,\infty} = \cup_{q=-\infty}^{+\infty}\Omega_q^{1,\infty}$ where $\Omega_q^{1,\infty} = \Omega^1 + (q + 1/2, 1/2, 1/2)$. The domain is represented in Figure 3.5.

The two-scale transform : The two-scale transform operator T or unfolding operator is defined as in Definition 1 and shown in Figures 3.2, and 3.3.

Definition 1: Definition of T

The two-scale operator $T : L^2(\Omega^\varepsilon) \to L^2\left(\Omega^\sharp \times \Omega^1\right)$ is defined for any $u \in L^2(\Omega^\varepsilon)$ by

$$Tu(x^\sharp, x^1) = \sum_c \chi_{\Omega_c^\varepsilon}(x^\sharp) u(x^{\varepsilon,c} + \varepsilon x^1) \text{ for all } x^\sharp \in \Omega^\sharp \text{ and } x^1 \in \Omega^1, \tag{3.17}$$

where χ is the characteristic function.

The two-scale transform operator T_b^ϑ for boundary layers at the two ends and T_b^{interf} for the interface of the one-dimensional array having two parts of different applied voltages are defined in Definition 2 and shown in Figures 3.4 and 3.5.

Definition 2: Definition of T_b^ϑ and T_b^{interf}

Let $\vartheta \in \{0, L_1\}$, the boundary layer two-scale transform operators $T_b^\vartheta : L^2(\Omega^\varepsilon) \to L^2\left(\Omega^{1,+\infty}\right)$ are defined $\forall u \in L^2(\Omega^\varepsilon), \forall x^1 \in \Omega^{1,+\infty}$ by

$$T_b^0 u\left(x^1\right) = u\left(\varepsilon x^1\right)\chi_{(0,L_1^m/\varepsilon)}\left(x_1^1\right) \text{ and } T_b^{L_1} u\left(x^1\right) = u\left(L_1 - \varepsilon x_1^1, \varepsilon x_2^1, \varepsilon x_3^1\right)\chi_{(0,(L_1-L_1^m)/\varepsilon)}\left(x_1^1\right). \tag{3.18}$$

The interface two scale transform operator $T_b^{interf} : L^2(\Omega^\varepsilon) \to L^2\left(\Omega^{1,\infty}\right)$ is defined $\forall u \in L^2(\Omega^{\varepsilon,vac}), \forall x^1 \in \Omega^{1,\infty}$ by

$$T_b^{interf} u\left(x^1\right) = u\left(L_1^m + \varepsilon x_1^1, \varepsilon x_2^1, \varepsilon x_3^1\right)\chi_{(-L_1^m/\varepsilon,(L_1-L_1^m)/\varepsilon)}\left(x_1^1\right). \tag{3.19}$$

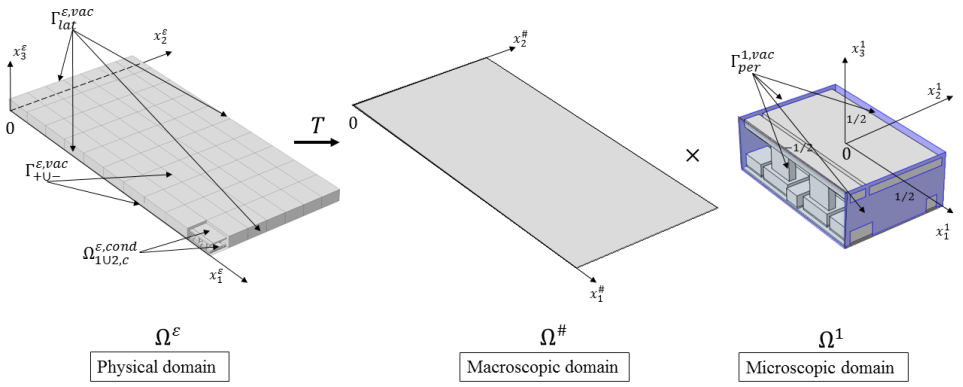


Figure 3.2: The mechanical body of a two-dimensional array and its surrounding vacuum Ω^{ε}, the macroscopic domain $\Omega^{\sharp} = (0, L_1) \times (0, L_2)$ and the microscopic domain $\Omega^1 = \varepsilon^{-1}(\Omega^{\varepsilon}_c - x^{\varepsilon}_c)$. The boundary $\Gamma^{vac}_{+\cup-}$ are the upper and lower surface of $\Omega^{\varepsilon,vac}$. The boundary Γ^{vac}_{lat} is the union of all lateral boundaries of $\Omega^{\varepsilon,vac}$. The boundary $\Gamma^{1,vac}_{per}$ on which the periodic boundary condition will be imposed is shown in blue. In this case, it is all lateral boundaries of Ω^1. The boundary $\Gamma^{1,vac}_{+\cup-}$ is the union of the upper and lower surfaces of $\Omega^{1,vac}$. The operator T transforms a function defined in Ω^{ε} into a function defined in the two-scale domain $\Omega^{\sharp} \times \Omega^1$.

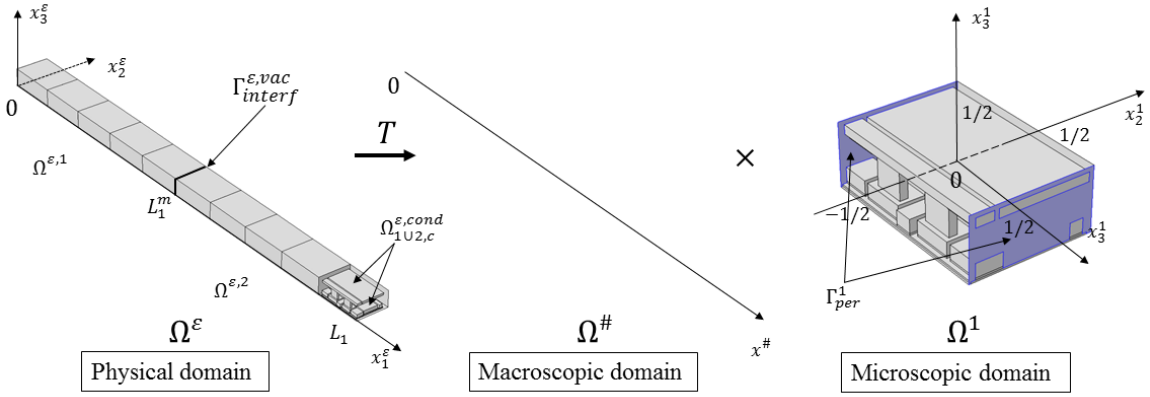


Figure 3.3: The physical domain Ω^{ε} is divided into $\Omega^{\varepsilon,1}$ and $\Omega^{\varepsilon,2}$ by the interface $\Gamma^{\varepsilon,vac}_{interf}$ at L^m_1, the macroscopic domain $\Omega^{\sharp} = (0, L_1)$ and the microscopic domain $\Omega^1 = \varepsilon^{-1}(\Omega^{\varepsilon}_c - x^{\varepsilon}_c)$. The operator T transfers a function defined in Ω^{ε} into a function defined in the two scale domain $\Omega^{\sharp} \times \Omega^1$.

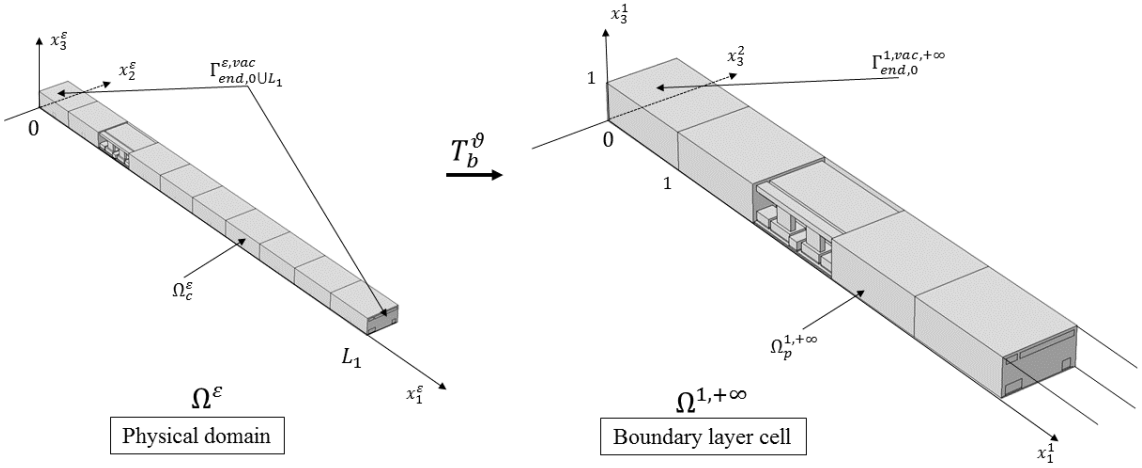


Figure 3.4: The physical domain Ω^ε and the boundary layer cell $\Omega^{1,+\infty} = \cup_{p=0}^{+\infty}\Omega_p^{1,+\infty}$. The notation $\Gamma_{end,0\cup L_1}^{\varepsilon,vac}$ represent the boundaries at the two ends of $\Omega^{\varepsilon,vac}$. The boundary of the domain $\Omega^{1,vac,+\infty}$ is defined similarly as this of $\Omega^{\varepsilon,vac}$. The internal boundaries where Dirichlet conditions are imposed are denoted by $\Gamma_{int}^{1,vac,+\infty}$. It is the union of all internal boundaries of each cell $\Gamma_{int,p}^{1,vac,+\infty}$. The external boundary $\Gamma_{ext}^{1,vac,+\infty}$ is the union of the upper and lower surfaces $\Gamma_{-\cup+}^{1,vac,+\infty}$, of the boundary at the ends $\Gamma_{end,0}^{1,vac,+\infty}$ and $\Gamma_{end,L_1}^{1,vac,+\infty}$ and of the lateral boundaries $\Gamma_{lat}^{1,vac,+\infty}$. The operator T_b^ϑ transforms a function defined in Ω^ε into a function defined in $\Omega^{1,+\infty}$.

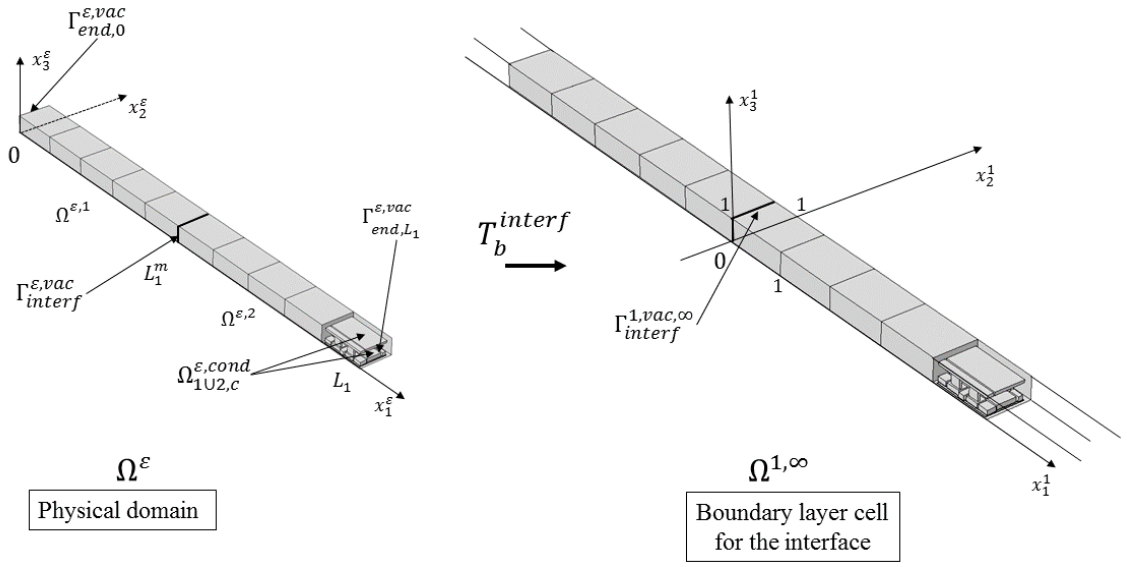


Figure 3.5: The physical domain Ω^ε has two different parts $\Omega^{\varepsilon,1}$ and $\Omega^{\varepsilon,2}$ and the boundary layer cells $\Omega^{1,\infty} = \cup_{q=-\infty}^{+\infty}\Omega_q^{1,\infty}$. The notation $\Gamma^{\varepsilon,vac}_{interf}$ represents the interface at L_1^m, and it becomes $\Gamma^{1,vac,\infty}_{interf}$ after scaling. The internal boundary $\Gamma^{1,vac,\infty}_{int}$ is the union of all internal boundaries of all cells $\Gamma^{1,vac,\infty}_{int,q}$. The external boundary is the combination of the upper and lower surface $\Gamma^{1,vac,\infty}_{+\cup-}$ and the lateral boundary $\Gamma^{1,vac,\infty}_{lat}$. The operator T_b^{interf} transforms a function defined in Ω^ε into a function defined in $\Omega^{1,\infty}$.

The functions ϕ^ε and V^ε satisfy the following assumptions.

Assumption 1: Two-scale weak convergence of ϕ^ε

We assume that $\exists \phi^0 \in L^2\left(\Omega^\sharp \times \Omega^1\right)$ such as

$$T\phi^\varepsilon \rightharpoonup \phi^0 \text{ in } L^2\left(\Omega^\sharp \times \Omega^1\right). \tag{3.20}$$

Assumption 2: Two-scale weak convergence of V^ε

We assume that $\exists V^0 \in L^2\left(\Omega^\sharp \times \Gamma^1_{int}\right)$ piecewise constant such as

$$TV^\varepsilon \rightharpoonup V^0 \text{ in } L^2\left(\Omega^\sharp \times \Gamma^1_{int}\right). \tag{3.21}$$

Definition 3: Boundary layer terms

Let ϕ_b^ε and $V_b^\varepsilon \in L^2\left(\Omega^\varepsilon\right)$ be the boundary layer terms of ϕ^ε and of V^ε such as

$$\phi_b^\varepsilon\left(x^\varepsilon\right) = \phi^\varepsilon\left(x^\varepsilon\right) - B\phi^0\left(x^\varepsilon\right) \text{ and } V_b^\varepsilon\left(x^\varepsilon\right) = V^\varepsilon\left(x^\varepsilon\right) - BV^0\left(x^\varepsilon\right) \tag{3.22}$$

where ϕ^0 and V^0 are weak limits of $T\phi^\varepsilon$ and TV^ε in $L^2\left(\Omega^\sharp \times \Omega^1\right)$ presented in Assumptions 1, 2 and B is defined in Definition 5.

Assumption 3: Boundary layer two-scale convergence assumption on ϕ_b^ε

We assume that $\forall \vartheta \in \{0, L_1\}$, $\exists \phi_b^\vartheta \in L^2\left(\Omega^{1,+\infty}\right)$ and $\exists \phi_b^{interf} \in L^2\left(\Omega^{1,\infty}\right)$ such that

$$T_b^\vartheta \phi_b^\varepsilon \rightharpoonup \phi_b^\vartheta \text{ in } L^2\left(\Omega^{1,+\infty}\right) \text{ and} \tag{3.23}$$

$$T_b^{interf} \phi_b^\varepsilon \rightharpoonup \phi_b^{interf} \text{ in } L^2\left(\Omega^{1,\infty}\right). \tag{3.24}$$

Assumption 4: Boundary layer two-scale convergence assumption on V_b^ε

We assume that $\forall \vartheta \in \{0, L_1\}$, $\exists V_b^\vartheta \in L^2\left(\Gamma_{int}^{1,+\infty}\right)$ and $\exists V_b^{interf} \in L^2\left(\Gamma_{int}^{1,\infty}\right)$ such that

$$T_b^\vartheta V_b^\varepsilon \rightharpoonup V_b^\vartheta \text{ in } L^2\left(\Gamma_{int}^{1,+\infty}\right) \text{ and} \tag{3.25}$$

$$T_b^{interf} V_b^\varepsilon \rightharpoonup V_b^{interf} \text{ in } L^2\left(\Gamma_{int}^{1,\infty}\right). \tag{3.26}$$

3.2.4/ Statement of the Two-Scale Model

Let ϕ^ε be the solution of the very weak formulation (3.5) satisfying the uniform bound (3.14) and Assumption 1, V^ε satisfying Assumption 2, ϕ_b^ϑ, $V_b^\varepsilon(x^\varepsilon)$ the boundary layer terms defined in Definition 3 satisfying Assumptions 3, 4, and ϕ_b^{interf} introduced as in Assumption 3 which corrects the interface of the array. The asymptotic models are stated as follows. For one and two-dimensional arrays, the models are $\forall x^\sharp \in \Omega^\sharp$,

$$\begin{cases} -\mathrm{div}_{x^1}\left(\nabla_{x^1}\phi^0\right) = 0 & \text{in} \quad \Omega^{1,vac} \\ \phi^0 = V^0 & \text{on} \quad \Gamma_{int}^{1,vac} \\ \nabla_{x^1}\phi^0 \cdot \mathbf{n}^1 = 0 & \text{on} \quad \Gamma_{+\cup-}^{1,vac} \cup \Gamma_{lat}^{1,vac} \\ \nabla_{x^1}\phi^0 \cdot \mathbf{n}^1 & \text{is} \quad \Gamma_{per}^{1,vac}\text{-anti-periodic} \\ \phi^0 & \text{is} \quad \Gamma_{per}^{1,vac}\text{-periodic} \end{cases}, \tag{3.27}$$

where $\Gamma_{lat}^{1,vac} = \emptyset$ in the case of two-dimensional array. For boundary layers at the two ends of the one-dimensional array, the models are $\forall \vartheta \in \{0, L_1\}$

$$\begin{cases} -\mathrm{div}_{x^1}\left(\nabla_{x^1}\phi_b^\vartheta\right) = 0 & \text{in} \quad \Omega^{1,vac,+\infty} \\ \phi_b^\vartheta = V_b^\vartheta & \text{on} \quad \Gamma_{int}^{1,vac,+\infty} \\ \nabla_{x^1}\phi_b^\vartheta \cdot \mathbf{n}^{+\infty} = 0 & \text{on} \quad \Gamma_{lat}^{1,vac,+\infty} \cup \Gamma_{+\cup-}^{1,vac,+\infty} \\ \nabla_{x^1}\phi_b^\vartheta \cdot \mathbf{n}^{+\infty} = -\nabla_{x^1}\phi^0\left(\vartheta, x^1\right) \cdot \mathbf{n}^{+\infty} & \text{on} \quad \Gamma_{end,\vartheta}^{1,vac,+\infty} \end{cases}. \tag{3.28}$$

Finally, for the boundary layer at the interface of the one-dimensional array, the model is

$$\begin{cases} -\mathrm{div}_{x^1}\left(\nabla_{x^1}\phi_b^{interf}\right) = 0 & \text{in} \quad \Omega^{1,vac,\infty} \\ \phi_b^{interf} = V_b^{interf} & \text{on} \quad \Gamma_{int}^{1,vac,\infty} \\ \nabla_{x^1}\phi_b^{interf} \cdot \mathbf{n}^{\infty} = 0 & \text{on} \quad \Gamma_{lat}^{1,vac,\infty} \cup \Gamma_{+\cup-}^{1,vac,\infty} \\ [\![\partial_{x_1^1}\left(\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\right)]\!] = 0 & \text{on} \quad \Gamma_{interf}^{1,vac,\infty} \\ [\![\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}]\!] = 0 & \text{on} \quad \Gamma_{interf}^{1,vac,\infty} \end{cases}, \tag{3.29}$$

where $[\![f(x)]\!]$ is the "jump" of $f(x)$ at the interface. The proofs are presented latter in this section after all properties regarding to the two-scale transform method have been stated.

3.2.5/ PROPERTIES OF THE TWO-SCALE TRANSFORM

We introduce the average over a domain A, which means $|A|$ by $\fint_A f(x)\,\mathrm{d}x = |A|^{-1}\int_A f(x)\,\mathrm{d}x$ as well as the related L^2 norm $|||f|||^2_{L^2(A)} = \fint_A |f(x)|^2\,\mathrm{d}x$. The index set $I^\sharp$ used in the following work can be $\{1,2\}$ or $\{1\}$ and depends on either the two-dimensional array or one-dimensional array is being considered. We recall some well known properties of T $\forall u, v \in L^2(\Omega^\varepsilon)$

$$\forall i \in I : T\left(\partial_{x_i^\varepsilon} u\right) = \varepsilon^{-1}\left(\partial_{x_i^1} Tu\right),\ |||Tu|||^2_{L^2(\Omega^\sharp\times\Omega^1)} = |||u|||^2_{L^2(\Omega^\varepsilon)},$$

$$\fint_{\Omega^\sharp\times\Omega^1} Tu\left(x^\sharp, x^1\right)\mathrm{d}x^\sharp\mathrm{d}x^1 = \fint_{\Omega^\varepsilon} u\left(x^\varepsilon\right)\mathrm{d}x^\varepsilon, \tag{3.30}$$

$$TuTv = T(uv) \text{ and } T(u+v) = Tu + Tv. \tag{3.31}$$

As T is a bounded linear operator from $L^2(\Omega)$ to $L^2\left(\Omega^\sharp\times\Omega^1\right)$, its adjoint is a bounded linear operator defined in the following Definition.

Definition 4: Adjoint of T

The adjoint $T^* : L^2\left(\Omega^\sharp\times\Omega^1\right) \to L^2(\Omega^\varepsilon)$ is defined $\forall u \in L^2(\Omega^\varepsilon), \forall v \in L^2\left(\Omega^\sharp\times\Omega^1\right)$ and $\forall w \in L^2\left(\Omega^\sharp\times\Gamma^1_{int}\right)$ by

$$\fint_{\Omega^\varepsilon} T^*(v)\left(x^\varepsilon\right)u\left(x^\varepsilon\right)\mathrm{d}x^\varepsilon = \fint_{\Omega^\sharp\times\Omega^1} v\left(x^\sharp, x^1\right)Tu\left(x^\sharp, x^1\right)\mathrm{d}x^\sharp\mathrm{d}x^1 \text{ and} \tag{3.32}$$

$$\fint_{\Gamma^\varepsilon_{int}} T^* w_i\left(x^\varepsilon\right)u\left(x^\varepsilon\right)n_i^\varepsilon\,\mathrm{d}s\left(x^\varepsilon\right) = \fint_{\Omega^\sharp\times\Gamma^\varepsilon_{int}} w_i\left(x^\sharp, x^1\right)Tu\left(x^\sharp, x^1\right)n_i^1\,\mathrm{d}x^\sharp\mathrm{d}s\left(x^\varepsilon\right).$$

The notation T^* is used when the adjoint operators are operating on v and w which are defined on different domains because they have the same properties in the two cases. For the sake of simplicity, the same thing will be applied to the operators B, B_b^ϑ, B_b^{infa}, T_b^ϑ and T_b^{infa} which are defined latter. The operator T^* satisfies $\forall u \in L(\Omega^\varepsilon), \forall v \in L^2\left(\Omega^\sharp\times\Omega^1\right)$: $T^*(Tu)(x^\varepsilon) = u(x^\varepsilon)$ and

$$T^* v\left(x^\varepsilon\right) = \frac{1}{|\Omega_c^\sharp|}\int_{\Omega_c^\sharp} v\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right)\mathrm{d}x^\sharp. \tag{3.33}$$

The image T^*v is not a regular functions. The operator B defined here after yields regular functions Bv.

Definition 5: Operator B

The operator $B : L^2\left(\Omega^\sharp\times\Omega^1\right) \to L^2(\Omega^\varepsilon)$, is defined by $\forall v \in L^2\left(\Omega^\sharp\times\Omega^1\right)$

$$Bv(x^\varepsilon) = v\left(P^\sharp\left(x^\varepsilon\right), \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right), \tag{3.34}$$

where $P^\sharp(x^\varepsilon)$ is the projection of Ω^ε onto $\Omega^\sharp$.

If v is Ω^1-periodic in the x_1-direction which means $v\left(x^\sharp, x_1^1 + z, x_2^1, x_3^1\right) = v\left(x^\sharp, x^1\right)\ \forall z \in \mathbb{Z}$, we have $\varepsilon^{-1} x^{\varepsilon,c} = \varepsilon^{-1}\left((c-1)\varepsilon + \varepsilon/2, \varepsilon/2, \varepsilon/2\right) = (c - 1/2, 1/2, 1/2)$ and

$$Bv\left(x^\varepsilon\right) = v\left(P^\sharp\left(x^\varepsilon\right), \frac{x_1^\varepsilon}{\varepsilon} - (c-1) - \frac{1}{2}, \frac{x_2^\varepsilon}{\varepsilon} - \frac{1}{2}, \frac{x_3^\varepsilon}{\varepsilon} - \frac{1}{2}\right) = v\left(P^\sharp\left(x^\varepsilon\right), \frac{x_1^\varepsilon}{\varepsilon} - \frac{1}{2}, \frac{x_2^\varepsilon}{\varepsilon} - \frac{1}{2}, \frac{x_3^\varepsilon}{\varepsilon} - \frac{1}{2}\right)$$
$$= v\left(P^\sharp\left(x^\varepsilon\right), \frac{x^\varepsilon}{\varepsilon} - \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)\right).$$

For simplicity, we denote $P^\sharp\left(x^\varepsilon\right) = x^\sharp$ in the following work.

Proposition 1: Properties of B

The operator B satisfies

(i) $\forall v \in L^2\left(\Omega^\sharp \times \Omega^1\right), \forall i \in I = \{1, 2, 3\}$

$$\frac{\partial Bv}{\partial x_i^\varepsilon} = \chi_{I^\sharp}(i)\, B\left(\frac{\partial v}{\partial x_i^\sharp}\right) + \frac{1}{\varepsilon} B\left(\frac{\partial v}{\partial x_i^1}\right). \tag{3.35}$$

(ii) $\forall u \in L^2\left(\Omega^\sharp \times \Omega^1\right) : T(Bu)\left(x^\sharp, x^1\right) = u\left(x^\sharp, x^1\right)$.

Proof. We only prove the case of two-dimensional arrays. The proof of one dimensional case is similar. The first point is proven by applying the chain rule to (3.34). The second point is proven by applying the Definitions 1, 5, we obtain

$$T(Bv) = \sum_c \chi_{\Omega_c^\varepsilon}(x^\sharp) Bv(x^{\varepsilon,c} + \varepsilon x^1) = \sum_c \chi_{\Omega_c^\varepsilon}(x^\sharp) v\left(x^{\varepsilon,c} + \varepsilon x^1, \frac{x^{\varepsilon,c} + \varepsilon x^1 - x^{\varepsilon,c}}{\varepsilon}\right)$$
$$= \sum_c \chi_{\Omega_c^\varepsilon}(x^\sharp) v\left(x^\varepsilon, x^1\right) = v\left(x^\varepsilon, x^1\right).$$

□

Proposition 2: Approximation between T^* and B

If the function $v\left(x^\sharp, x^1\right)$ is continuous, continuously differentiable in $x^\sharp$ and Ω^1-periodic in x^1 then the first order approximation of B and T^* is given as

$$T^*(v) = B\left(v - \varepsilon \sum_{i \in I^\sharp} x_i^1 \frac{\partial v}{\partial x_i^\sharp}\right) + \varepsilon O(\varepsilon) \text{ or } B(v) = T^*\left(v + \varepsilon \sum_{i \in I^\sharp} x_i^1 \frac{\partial v}{\partial x_i^\sharp}\right) + \varepsilon O(\varepsilon), \tag{3.36}$$

where $O(\varepsilon)$ tends to 0 in the strong sense.

Proof. We recall the property of T^*

$$T^* v\left(x^\varepsilon\right) = \frac{1}{\varepsilon^2} \int_{\Omega_c^\sharp} v\left(z^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right) \mathrm{d}z^\sharp.$$

Applying Taylor expansion for the first variable of v, the above equation becomes

$$= \frac{1}{\varepsilon^2}\int_{\Omega_c^\sharp}\left[v\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right) + \sum_\alpha \left(z_\alpha^\sharp - x_\alpha^\sharp\right)\frac{\partial v}{\partial x_\alpha^\sharp}\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right) + \varepsilon O(\varepsilon)\right] \mathrm{d}z^\sharp$$

$$= \frac{1}{\varepsilon^2}\int_{\Omega_c^\sharp} v\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right)\mathrm{d}z^\sharp + \frac{1}{\varepsilon^2}\sum_\alpha \frac{\partial v}{\partial x_\alpha^\sharp}\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right)\int_{\Omega_c^\sharp}\left(z_\alpha^\sharp - x_\alpha^\sharp\right)\mathrm{d}z^\sharp + \frac{1}{\varepsilon^2}\int_{\Omega_c^\sharp}\varepsilon O(\varepsilon)\,\mathrm{d}z^\sharp.$$

Since

$$\int_{\Omega_c^\sharp} z_\alpha^\sharp - x_\alpha^{\sharp,c}\,\mathrm{d}z^\sharp = \int_{\Omega_{(1,1)}^\sharp} z_\alpha^\sharp - x_\alpha^{\sharp,c}\,\mathrm{d}z^\sharp = 0, \text{ and } x_\alpha^\sharp - x_\alpha^{\sharp,c} = \varepsilon\frac{x_\alpha^\sharp - x_\alpha^{\sharp,c}}{\varepsilon} = \varepsilon x_\alpha^1,$$

and the second integral is calculated as

$$\int_{\Omega_c^\sharp}\left(z_\alpha^\sharp - x_\alpha^\sharp\right)\mathrm{d}z^\sharp = \int_{\Omega_c^\sharp}\left(z_\alpha^\sharp - x_\alpha^{\sharp,c} + x_\alpha^{\sharp,c} - x_\alpha^\sharp\right)\mathrm{d}z^\sharp = \int_{\Omega_c^\sharp} z_\alpha^\sharp - x_\alpha^{\sharp,c}\,\mathrm{d}z^\sharp - \int_{\Omega_c^\sharp} x_\alpha^\sharp - x_\alpha^{\sharp,c}\,\mathrm{d}z^\sharp$$
$$= -\varepsilon^2\left(x_\alpha^\sharp - x_\alpha^{\sharp,c}\right) = -\varepsilon^3 x_\alpha^1,$$

Then

$$T^* v(x^\varepsilon) = v\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right) - \varepsilon\sum_\alpha x_\alpha^1\frac{\partial v}{\partial x_\alpha^\sharp}\left(x^\sharp, \frac{x^\varepsilon - x^{\varepsilon,c}}{\varepsilon}\right) + \varepsilon O(\varepsilon), \text{ or}$$

$$T^* v(x^\varepsilon) = Bv - \varepsilon\sum_\alpha x_\alpha^1 B\frac{\partial v}{\partial x_\alpha^\sharp} + \varepsilon O(\varepsilon) = B\left(v - \varepsilon\sum_\alpha x_\alpha^1\frac{\partial v}{\partial x_\alpha^\sharp}\right) + \varepsilon O(\varepsilon).$$

Conversely, we have

$$Bv = T^* v + \varepsilon\sum_\alpha x_\alpha^1 B\frac{\partial v}{\partial x_\alpha^\sharp} + \varepsilon O(\varepsilon) = T^* v + \varepsilon\sum_\alpha x_\alpha^1 T^*\left(\frac{\partial v}{\partial x_\alpha^\sharp}\right) + \varepsilon O(\varepsilon)$$
$$= T^*\left(v + \varepsilon\sum_\alpha x_\alpha^1\frac{\partial v}{\partial x_\alpha^\sharp}\right) + \varepsilon O(\varepsilon).$$

□

Now, let us recall elementary properties of boundary layer two-scale transform at the ends $T_b^\vartheta : \forall u, v \in L^2(\Omega^\varepsilon)$,

$$T_b^\vartheta(uv) = T_b^\vartheta(u)\,T_b^\vartheta(v) \text{ and } \frac{\varepsilon^2}{|\Omega^\varepsilon|}\int_{\Omega^{1,+\infty}} T_b^\vartheta u\left(x^1\right)\mathrm{d}x^1 = \frac{1}{\varepsilon}\oint_{\Omega^\varepsilon} u(x^\varepsilon)\,\mathrm{d}x^\varepsilon.$$

The boundary layer two-scale transform operator at the interface T_b^{interf} satisfies: $\forall u \in L^2(\Omega^\varepsilon)$,

$$\frac{\varepsilon^2}{|\Omega^\varepsilon|}\int_{\Omega^{1,\infty}} T_b^{interf} u\left(x^1\right)\mathrm{d}x^1 = \frac{1}{\varepsilon}\oint_{\Omega^\varepsilon} u(x^\varepsilon)\,\mathrm{d}x^\varepsilon. \tag{3.37}$$

Their adjoint operators are defined in the following Definition.

Definition 6: Adjoint operators of T_b^{ϑ} and T_b^{interf}

The adjoint operator $T_b^{\vartheta *} : \forall \vartheta \in \{0, L_1\}$, the adjoint operators $T_b^{\vartheta *} : L^2\left(\Omega^{1,+\infty}\right) \rightarrow L^2\left(\Omega^{\varepsilon}\right)$ are defined $\forall u \in L^2\left(\Omega^{\varepsilon}\right)$, $\forall v \in L^2\left(\Omega^{1,+\infty}\right)$ and $w \in L^2\left(\Gamma_{int}^{1,+\infty}\right)$ by

$$\frac{1}{\varepsilon}\oint_{\Omega^{\varepsilon}} u\left(x^{\varepsilon}\right) T_b^{\vartheta *} v\left(x^{\varepsilon}\right) \mathrm{d}x^{\varepsilon} = \frac{\varepsilon^2}{|\Omega^{\varepsilon}|}\int_{\Omega^{1,+\infty}} T_b^{\vartheta} u\left(x^1\right) v\left(x^1\right) \mathrm{d}x^1 \text{ and} \tag{3.38}$$

$$\frac{1}{\varepsilon}\oint_{\Gamma_{int}^{\varepsilon}} u\left(x^{\varepsilon}\right) T_b^{\vartheta *} w_i\left(x^{\varepsilon}\right) n_i^{\varepsilon} \mathrm{d}s\left(x^{\varepsilon}\right) = \frac{\varepsilon}{|\Gamma_{int}^{\varepsilon}|}\int_{\Gamma_{int}^{1,+\infty}} T_b^{\vartheta} u\left(x^1\right) w_i\left(x^1\right) n_i^{+\infty} \mathrm{d}s\left(x^1\right), \tag{3.39}$$

The adjoint operator $T_b^{interf *} : L^2\left(\Omega^{1,\infty}\right) \rightarrow L^2\left(\Omega^{\varepsilon}\right)$ is defined such as $\forall u \in L^2\left(\Omega^{\varepsilon}\right)$, $\forall v \in L^2\left(\Omega^{1,\infty}\right)$ and $w \in L^2\left(\Gamma_{int}^{1,\infty}\right)$ by

$$\frac{1}{\varepsilon}\oint_{\Omega^{\varepsilon}} u\left(x^{\varepsilon}\right) T_b^{interf *} v\left(x^{\varepsilon}\right) \mathrm{d}x^{\varepsilon} = \frac{\varepsilon^2}{|\Omega^{\varepsilon}|}\int_{\Omega^{1,\infty}} T_b^{interf} u\left(x^1\right) v\left(x^1\right) \mathrm{d}x^1 \text{ and} \tag{3.40}$$

$$\frac{1}{\varepsilon}\oint_{\Gamma_{int}^{\varepsilon}} u\left(x^{\varepsilon}\right) T_b^{interf *} w_i\left(x^{\varepsilon}\right) n_i^{\varepsilon} \mathrm{d}s\left(x^{\varepsilon}\right) = \frac{\varepsilon}{|\Gamma_{int}^{\varepsilon}|}\int_{\Gamma_{int}^{1,\infty}} T_b^{interf} u\left(x^1\right) w_i\left(x^1\right) n_i^{\infty} \mathrm{d}s\left(x^1\right). \tag{3.41}$$

Definition 7: Operators B_b^{ϑ} and B_b^{interf}

Let $\vartheta \in \{0, L_1\}$, the linear operator $B_b^{\vartheta} : L^2\left(\Omega^{1,+\infty}\right) \rightarrow L^2\left(\Omega^{\varepsilon}\right)$ are defined $\forall v \in L^2\left(\Omega^{1,+\infty}\right)$ such as

$$B_b^0(v)\left(x^{\varepsilon}\right) = v\left(\frac{x^{\varepsilon}}{\varepsilon}\right) \text{ and } B_b^{L_1}(v)\left(x^{\varepsilon}\right) = v\left(\frac{L_1 - x_1^{\varepsilon}}{\varepsilon}, \frac{x_2^{\varepsilon}}{\varepsilon}, \frac{x_3^{\varepsilon}}{\varepsilon}\right). \tag{3.42}$$

The linear operator $B_b^{interf} : L^2\left(\Omega^{1,\infty}\right) \rightarrow L^2\left(\Omega^{\varepsilon}\right)$ is defined $\forall v \in L^2\left(\Omega^{1,\infty}\right)$ such as

$$B_b^{interf} v\left(x^{\varepsilon}\right) = v\left(\frac{x_1^{\varepsilon} - L_1^m}{\varepsilon}, \frac{x_2^{\varepsilon}}{\varepsilon}, \frac{x_3^{\varepsilon}}{\varepsilon}\right). \tag{3.43}$$

These operators have the same properties as the operators T, T^* and B presented above. However, to avoid confusions caused by new notations, we precise some important properties of B_b^{ϑ} such as for all $v \in L^2\left(\Omega^{1,+\infty}\right)$ and all $i \in \{1,2,3\}$ we have

$$T_b^{\vartheta *} v\left(x^{\varepsilon}\right) = B_b^{\vartheta} v\left(x^{\varepsilon}\right) \chi_{\Omega^{\varepsilon, s(\vartheta)}}\left(x_1^{\varepsilon}\right), \text{ where } s(0) = 1 \text{ and } s(L_1) = 2 \text{ and} \tag{3.44}$$

$$T_b^{\vartheta *} v_i\left(x^{\varepsilon}\right) = (-1)^{\chi_{\{L_1\}}(\vartheta)} B_b^{\vartheta} v_i\left(x^{\varepsilon}\right) \chi_{\Omega^{\varepsilon, s(\vartheta)}}\left(x_1^{\varepsilon}\right). \tag{3.45}$$

$$\frac{\partial B_b^0 v}{\partial x_i^{\varepsilon}}\left(x^{\varepsilon}\right) = \frac{1}{\varepsilon} B_b^0\left(\frac{\partial v}{\partial x_i^1}\right)\left(x^{\varepsilon}\right) \text{ and } \frac{\partial B_b^{L_1} v}{\partial x_i^{\varepsilon}}\left(x^{\varepsilon}\right) = (-1)^{\chi_{\{1\}}(i)} \frac{1}{\varepsilon} B_b^{L_1}\left(\frac{\partial v}{\partial x_i^1}\right)\left(x^{\varepsilon}\right) \text{ or in a compact form}$$

$$\frac{\partial B_b^{\vartheta} v}{\partial x_i^{\varepsilon}} = C(\vartheta, i) \frac{1}{\varepsilon} B_b^{\vartheta}\left(\frac{\partial v}{\partial x_i^1}\right)\left(x^{\varepsilon}\right) \text{ with } C(\vartheta, i) = (-1)^{\chi_{\{L_1\}}(\vartheta)\chi_{\{1\}}(i)}. \tag{3.46}$$

The operator B_b^{interf} satisfies $\forall v \in L^2\left(\Omega^{1,\infty}\right)$,

$$T_b^{interf *} v\left(x^{\varepsilon}\right) = B_b^{interf} v\left(x^{\varepsilon}\right) \text{ and}$$

$$\frac{\partial B_b^{interf} v}{\partial x_i^\varepsilon}\left(x^\varepsilon\right) = \frac{1}{\varepsilon} B_b^{interf}\left(\frac{\partial v}{\partial x_i^1}\right)\left(x^\varepsilon\right). \tag{3.47}$$

3.2.6/ PROPERTIES USED IN MODEL DERIAVATIONS

Proposition 3:

Assume that ϕ^0 and V^0 introduced in Assumptions 1, 2 are continuous in $\bar{\Omega}^1 = \Omega^1 \cup \partial\Omega^1$ and x_1^1–periodic, $\forall \vartheta \in \{0, L_1\}$ we have

$$T_b^\vartheta B\phi^0\left(x^1\right) = \phi^0\left(\vartheta, x^1\right) + O\left(\varepsilon\right), \tag{3.48}$$

$$T_b^{interf} B\phi^0\left(x^1\right) = \phi^0\left(L_1^m, x^1\right) + O\left(\varepsilon\right), \tag{3.49}$$

$$T_b^\vartheta BV^0\left(x^1\right) = V^0\left(\vartheta, x^1\right) + O\left(\varepsilon\right), \tag{3.50}$$

$$T_b^{interf} BV^0\left(x^1\right) = V^0\left(L_1^m, x^1\right) + O\left(\varepsilon\right), \tag{3.51}$$

where B is the operator defined in Definition 5, T_b^ϑ are defined in Definition 2 and $O\left(\varepsilon\right)$ strongly converges to 0 when ε goes to 0.

Proof. Consider the one-dimensional array, we know that each cell is denoted by the index $c \in \{1, ..., n\}$, the center $x^{\varepsilon,c}$ of each cell is related to the center $x^{1,c} = (1/2 + c - 1, 1/2, 1/2)$ of $\Omega_c^{1,+\infty}$ through the relation $x^{\varepsilon,c} = \varepsilon x^{1,c}$. In the case $\vartheta = 0$, from the definition of T_b^0 and B, we get

$$T_b^0 B\phi^0\left(x^1\right) = B\phi^0\left(\varepsilon x^1\right)\chi_{(0,L_1/\varepsilon)}\left(x_1^1\right) = \phi^0\left(\varepsilon x_1^1, \frac{\varepsilon x^1 - \varepsilon x^{1,c}}{\varepsilon}\right)\chi_{(0,L_1/\varepsilon)}\left(x_1^1\right)$$

$$= \phi^0\left(\varepsilon x_1^1, x^1 - x^{1,c}\right)\chi_{(0,L_1/\varepsilon)}\left(x_1^1\right) = \sum_{k=0}^{n-1} \phi^0\left(\varepsilon x_1^1, x^1 - x^{1,c}\right)\chi_{(k-1/2,k+1/2)}\left(x_1^1 - x_1^{1,c}\right).$$

For each $x_1^1 \in (0, L_1/\varepsilon)$, there exists k such that $x_1^1 - x_1^{1,c} \in (k - 1/2, k + 1/2)$, i.e. $x_1^1 - x_1^{1,c} - k \in (-1/2, 1/2)$. Thus,

$$T_b^0 B\phi^0\left(x^1\right) = \phi^0\left(\varepsilon x_1^1, x^1 - x^{1,c}\right) = \phi^0\left(\varepsilon x_1^1, x_1^1 - x_1^{1,c} - k + k, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right).$$

Applying the assumptions of ϕ^0, it becomes

$$T_b^0 B\phi^0\left(x^1\right) = \phi^0\left(\varepsilon x_1^1, x^1\right) = \phi^0\left(0, x^1\right) + O\left(\varepsilon\right).$$

Applying the same approach for the case $\vartheta = L_1$, we get

$$T_b^{L_1} B\phi^0\left(x^1\right) = B\phi^0\left(L_1 - \varepsilon x_1^1, \varepsilon x_2^1, \varepsilon x_3^1\right)\chi_{(0,L_1/\varepsilon)}\left(x_1^1\right)$$

$$= \phi^0\left(L_1 - \varepsilon x_1^1, \frac{L_1 - \varepsilon x_1^1 - \varepsilon x_1^{1,c}}{\varepsilon}, \frac{\varepsilon x_2^1 - \varepsilon x_2^{1,c}}{\varepsilon}, \frac{\varepsilon x_3^1 - \varepsilon x_3^{1,c}}{\varepsilon}\right)\chi_{(0,L_1/\varepsilon)}\left(x_1^1\right)$$

$$= \phi^0\left(L_1 - \varepsilon x_1^1, n - x_1^1 - x_1^{1,c}, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right)\chi_{(0,n)}\left(x_1^1\right).$$

Because of $n - x_1^1 - x_1^{1,c} \in (-1/2, n+1/2) = \cup_{k=0,\ldots,n-1}(k-1/2, k+1/2] \cup [n-1, 1/2)$, there exists k such that $n - x_1^1 - x_1^{1,c} \in (k-1/2, k+1/2)$ which leads to $n - x_1^1 - x_1^{1,c} - k \in (-1/2, 1/2)$. Thus

$$\begin{aligned} T_b^{L_1} B\phi^0\left(x^1\right) &= \phi^0\left(L_1 - \varepsilon x_1^1, n - x_1^1 - x_1^{1,c} - k + k, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right) \\ &= \phi^0\left(L_1 - \varepsilon x_1^1, n - x_1^1 - x_1^{1,c} - k, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right) \\ &= \phi^0\left(L_1 - \varepsilon x_1^1, x^1\right) = \phi^0\left(L_1, x^1\right) + O(\varepsilon). \end{aligned}$$

From the Defintion 2 and Definition 5 , we get

$$\begin{aligned} T_b^{interf} B\phi^0\left(x^1\right) &= B\phi^0\left(L_1^m + \varepsilon x_1^1, \varepsilon x_2^1, \varepsilon x_3^1\right)\chi_{\left(-L_1^m/\varepsilon, (L_1 - L_1^m)/\varepsilon\right)}\left(x_1^1\right) \\ = \phi^0\left(L_1^m + \varepsilon x_1^1, \frac{L_1^m + \varepsilon x_1^1 - \varepsilon x_1^{1,c}}{\varepsilon}, \frac{\varepsilon x_2^1 - \varepsilon x_2^{1,c}}{\varepsilon}, \frac{\varepsilon x_3^1 - \varepsilon x_3^{1,c}}{\varepsilon}\right)&\chi_{\left(-L_1^m/\varepsilon, (L_1 - L_1^m)/\varepsilon\right)}\left(x_1^1\right) \\ = \phi^0\left(L_1^m + \varepsilon x_1^1, n^m + x_1^1 - x_1^{1,c}, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right)&\chi_{\left(-L_1^m/\varepsilon, (L_1 - L_1^m)/\varepsilon\right)}\left(x_1^1\right) \end{aligned}$$

where $L_1^m/\varepsilon = n^m$ is the number of cells of the left part of the array. Considering $x_1^1 \in \left(-L_1^m/\varepsilon, \left(L_1 - L_1^m\right)/\varepsilon\right) = (-n^m, n - n^m)$, so that $n^m + x_1^1 - x_1^{1,c} \in (-1/2, n-1/2) = \cup_{k=0,\ldots,n-2}(k - 1/2, k+1/2] \cup (n-1-1/2, n-1+1/2)$. Therefore, $\exists k_0 \in \{0, ..., n-2\}$ such that $n^m + x_1^1 - x_1^{1,c} \in (k_0 - 1/2, k_0 + 1/2)$ which leads to $n^m + x_1^1 - x_1^{1,c} - k_0 \in (-1/2, 1/2)$ and

$$T_b^{interf} B\phi^0\left(x^1\right) = \phi^0\left(L_1^m + \varepsilon x_1^1, n^m + x_1^1 - x_1^{1,c} - k_0 + k_0, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right).$$

Applying the assumptions of ϕ^0, we have

$$\begin{aligned} T_b^{interf} B\phi^0\left(x^1\right) &= \phi^0\left(L_1^m + \varepsilon x_1^1, n^m + x_1^1 - x_1^{1,c} - k_0, x_2^1 - x_2^{1,c}, x_3^1 - x_3^{1,c}\right) \\ &= \phi^0\left(L_1^m + \varepsilon x_1^1, x^1\right) = \phi^0\left(L_1^m, x^1\right) + O(\varepsilon). \end{aligned}$$

The remain points can be proved by following the same approach. □

Applying the operators T_b^ϑ and T_b^{interf} to boundary layer terms Definition 3, we get

$$T_b^\vartheta \phi^\varepsilon\left(x^1\right) = T_b^\vartheta B\phi^0\left(x^1\right) + T_b^\vartheta \phi_b^\varepsilon\left(x^1\right), \tag{3.52}$$

$$T_b^{interf} \phi^\varepsilon\left(x^1\right) = T_b^{interf} B\phi^0\left(x^1\right) + T_b^{interf} \phi_b^\varepsilon\left(x^1\right), \tag{3.53}$$

$$T_b^\vartheta V^\varepsilon\left(x^1\right) = T_b^\vartheta BV^0\left(x^1\right) + T_b^\vartheta V_b^\varepsilon\left(x^1\right), \tag{3.54}$$

$$T_b^{interf} V^\varepsilon\left(x^1\right) = T_b^{interf} BV^0\left(x^1\right) + T_b^{interf} V_b^\varepsilon\left(x^1\right). \tag{3.55}$$

Applying Proposition 3, the above equations become

$$T_b^\vartheta \phi^\varepsilon\left(x^1\right) = \phi^0\left(\vartheta, x^1\right) + \phi_b^\vartheta\left(x^1\right) + O(\varepsilon), \tag{3.56}$$

$$T_b^{interf} \phi^\varepsilon\left(x^1\right) = \phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\left(x^1\right) + O(\varepsilon), \tag{3.57}$$

$$T_b^\vartheta V^\varepsilon\left(x^1\right) = V^0\left(\vartheta, x^1\right) + V_b^\vartheta\left(x^1\right) + O(\varepsilon), \tag{3.58}$$

$$T_b^{interf} V^\varepsilon\left(x^1\right) = V^0\left(L_1^m, x^1\right) + V_b^{interf}\left(x^1\right) + O(\varepsilon). \tag{3.59}$$

3.2.7/ Model Derivations

Recall the scaled very weak formulation (3.13) with the simplified notations

$$\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon} \frac{\partial}{\partial x_i^{\varepsilon}} \frac{\partial v^{\varepsilon}}{\partial x_i^{\varepsilon}} \mathrm{d}x^{\varepsilon} = \frac{1}{|\Omega^{\varepsilon,vac}|} \int_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} \frac{\partial v^{\varepsilon}}{\partial x_i^{\varepsilon}} n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right), \tag{3.60}$$

where $v^{\varepsilon} \in H^1_{\Gamma_{int}^{\varepsilon,vac},0}\left(\Omega^{\varepsilon,vac}\right) \cap H^2_{\Gamma_{int}^{\varepsilon,vac},\Gamma_{ext}^{\varepsilon,vac}}\left(\Omega^{\varepsilon,vac}\right)$.

Two-scale model derivation for a two-dimensional array : Replacing the test function v^{ε} in (3.60) by Bw in which $w = w^0\left(x^{\sharp}\right) w^1\left(x^1\right)$, $w^0 = \partial_{x_i^{\sharp}} w^0 = 0$ on $\partial\Omega^{\sharp,vac}$ and w^1 is Ω^1-periodic, we get

$$\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon} \frac{\partial}{\partial x_i^{\varepsilon}} \frac{\partial Bw}{\partial x_i^{\varepsilon}} \mathrm{d}x^{\varepsilon} = \frac{1}{|\Omega^{\varepsilon,vac}|} \int_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} \frac{\partial Bw}{\partial x_i^{\varepsilon}} n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right). \tag{3.61}$$

The first order and the second order partial derivative of Bw is obtained by applying Proposition 1

$$\frac{\partial}{\partial x_i^{\varepsilon}} \frac{\partial Bw}{\partial x_i^{\varepsilon}} = B\left(\chi_{I^{\sharp}}(i) \frac{\partial}{\partial x_i^{\sharp}} \frac{\partial w}{\partial x_i^{\sharp}} + \chi_{I^{\sharp}}(i) \frac{2}{\varepsilon} \frac{\partial}{\partial x_i^{\sharp}} \frac{\partial w}{\partial x_i^1} + \frac{1}{\varepsilon^2} \frac{\partial}{\partial x_i^1} \frac{\partial w}{\partial x_i^1}\right),$$

$$\frac{\partial Bw}{\partial x_i^{\varepsilon}} = B\left(\chi_{I^{\sharp}}(i) \frac{\partial w}{\partial x_i^{\sharp}} + \frac{1}{\varepsilon} \frac{\partial w}{\partial x_i^1}\right).$$

Substituting these results in (3.61), we obtain

$$\begin{aligned} &\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon} B\left(\chi_{I^{\sharp}}(i) \frac{\partial}{\partial x_i^{\sharp}} \frac{\partial w}{\partial x_i^{\sharp}} + \chi_{I^{\sharp}}(i) \frac{2}{\varepsilon} \frac{\partial}{\partial x_i^{\sharp}} \frac{\partial w}{\partial x_i^1} + \frac{1}{\varepsilon^2} \frac{\partial}{\partial x_i^1} \frac{\partial w}{\partial x_i^1}\right) \mathrm{d}x^{\varepsilon} \\ &= \frac{1}{|\Omega^{\varepsilon,vac}|} \int_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} B\left(\chi_{I^{\sharp}}(i) \frac{\partial w}{\partial x_i^{\sharp}} + \frac{1}{\varepsilon} \frac{\partial w}{\partial x_i^1}\right) n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right). \end{aligned} \tag{3.62}$$

Multiplying ε^2 both sides, it becomes

$$\begin{aligned} \oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon} B\left(\frac{\partial}{\partial x_i^1} \frac{\partial w}{\partial x_i^1}\right) \mathrm{d}x^{\varepsilon} &= \frac{\varepsilon|\Gamma^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|} \oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} B\left(\frac{\partial w}{\partial x_i^1}\right) n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right) + O(\varepsilon) \\ &= C \oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} B\left(\frac{\partial w}{\partial x_i^1}\right) n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right) + O(\varepsilon), \end{aligned}$$

where $C = \varepsilon|\Gamma^{\varepsilon,vac}|/|\Omega^{\varepsilon,vac}|$. Approximating B by T^* with Proposition 2, and applying the definition of T^*, Definition 4, the left hand side becomes

$$\begin{aligned} \oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon} B\left(\frac{\partial}{\partial x_i^1} \frac{\partial w}{\partial x_i^1}\right) \mathrm{d}x^{\varepsilon} &= \oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon} T^*\left(\frac{\partial}{\partial x_i^1} \frac{\partial w}{\partial x_i^1}\right) \mathrm{d}x^{\varepsilon} + O(\varepsilon) \\ &= \oint_{\Omega^{\sharp,vac} \times \Omega^{1,vac}} T\phi^{\varepsilon} \frac{\partial}{\partial x_i^1} \frac{\partial w}{\partial x_i^1} \mathrm{d}x^{\sharp} \mathrm{d}x^1 + O(\varepsilon), \end{aligned}$$

while the right hand side becomes

$$\begin{aligned} C \oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} B\left(\frac{\partial w}{\partial x_i^1}\right) n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right) + O(\varepsilon) &= C \oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon} T^*\left(\frac{\partial w}{\partial x_i^1}\right) n_i^{\varepsilon} \, \mathrm{d}s\left(x^{\varepsilon}\right) + O(\varepsilon) \\ &= C \oint_{\Omega^{\sharp,vac} \times \Gamma_{int}^{1,vac}} TV^{\varepsilon} \frac{\partial w}{\partial x_i^1} n_i^1 \, \mathrm{d}s\left(x^1\right) \mathrm{d}x^{\sharp} + O(\varepsilon), \end{aligned}$$

where n^1 is unit outward normal vector in Ω^1. The equation reads

$$\oint_{\Omega^{\sharp,vac}\times\Omega^{1,vac}} T\phi^{\varepsilon}\frac{\partial}{\partial x_i^1}\frac{\partial w}{\partial x_i^1}\,\mathrm{d}x^{\sharp}\mathrm{d}x^1 = C\oint_{\Omega^{\sharp,vac}\times\Gamma_{int}^{1,vac}} TV^{\varepsilon}\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right)\mathrm{d}x^{\sharp} + O(\varepsilon).$$

Passing ε to 0, Assumptions 1 and 2 imply

$$\oint_{\Omega^{\sharp,vac}\times\Omega^{1,vac}} \phi^0\frac{\partial}{\partial x_i^1}\frac{\partial w}{\partial x_i^1}\,\mathrm{d}x^{\sharp}\mathrm{d}x^1 = C\oint_{\Omega^{\sharp,vac}\times\Gamma_{int}^{1,vac}} V^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right)\mathrm{d}x^{\sharp}.$$

Applying Green formula twice, we get

$$\begin{aligned}&\oint_{\Omega^{\sharp,vac}\times\Omega^{1,vac}} \frac{\partial}{\partial x_i^1}\frac{\partial\phi^0}{\partial x_i^1}w\,\mathrm{d}x^{\sharp}\mathrm{d}x^1 - \frac{1}{|\Omega^{\sharp,vac}\times\Omega^{1,vac}|}\int_{\Omega^{\sharp}\times\partial\Omega^{1,vac}}\frac{\partial\phi^0}{\partial x_i^1}wn_i^1\,\mathrm{d}s\left(x^1\right)\mathrm{d}x^{\sharp} \\ &+\frac{1}{|\Omega^{\sharp,vac}\times\Omega^{1,vac}|}\int_{\Omega^{\sharp}\times\partial\Omega^{1,vac}}\phi^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right) - C\oint_{\Omega^{\sharp,vac}\times\Gamma_{int}^{1,vac}} V^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^{\varepsilon}\right)\mathrm{d}x^{\sharp} = 0.\end{aligned} \tag{3.63}$$

Remind that $w\left(x^{\sharp},x^1\right) = w^0\left(x^{\sharp}\right)w^1\left(x^1\right)$, and if $w^1 = \partial_{x^1}w^1n_i^1 = 0$ on $\partial\Omega^{1,vac}$, the equation (3.63) becomes

$$\oint_{\Omega^{\sharp,vac}}\oint_{\Omega^{1,vac}} w^0\left(\int_{\Omega^{1,vac}}\left(\frac{\partial}{\partial x_i^1}\frac{\partial\phi^0}{\partial x_i^1}w^1\right)\mathrm{d}x^1\right)\mathrm{d}x^{\sharp} = 0,$$

and thus $\forall x^{\sharp}\in\Omega^{\sharp}$,

$$\frac{\partial}{\partial x_i^1}\frac{\partial\phi^0}{\partial x_i^1} = 0 \text{ in } \Omega^{1,vac}. \tag{3.64}$$

Decomposing (3.63) with $\partial\Omega^{1,vac} = \Gamma_{int}^{1,vac}\cup\Gamma_{+\cup-}^{1,vac}\cup\Gamma_{per}^{1,vac}$ and substituting (3.64) we get

$$\begin{aligned}&-\frac{1}{|\Omega^{\sharp,vac}\times\Omega^{1,vac}|}\int_{\Omega^{\sharp}}\left(\int_{\Gamma_{int}^{1,vac}}\frac{\partial\phi^0}{\partial x_i^1}wn_i^1\,\mathrm{d}s\left(x^1\right) + \int_{\Gamma_{+\cup-}^{1,vac}}\frac{\partial\phi^0}{\partial x_i^1}wn_i^1\,\mathrm{d}s\left(x^1\right) + \int_{\Gamma_{per}^{1,vac}}\frac{\partial\phi^0}{\partial x_i^1}wn_i^1\,\mathrm{d}s\left(x^1\right)\right)\mathrm{d}x^{\sharp} \\ &+\frac{1}{|\Omega^{\sharp,vac}\times\Omega^{1,vac}|}\int_{\Omega^{\sharp}}\left(\int_{\Gamma_{int}^{1,vac}}\phi^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right) + \int_{\Gamma_{+\cup-}^{1,vac}}\phi^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right) + \int_{\Gamma_{per}^{1,vac}}\phi^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right)\right)\mathrm{d}x^{\sharp} \\ &-C\oint_{\Omega^{\sharp,vac}\times\Gamma_{int}^{1,vac}} V^0\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^{\varepsilon}\right)\mathrm{d}x^{\sharp} = 0.\end{aligned}$$

If $w = 0$ on $\Gamma_{int}^{1,vac}\cup\Gamma_{+\cup-}^{1,vac}$ and $\partial w/\partial x_i^1 n_i^1 = 0$ on $\Gamma_{int}^{1,vac}\cup\Gamma_{+\cup-}^{1,vac}$, it becomes

$$-\int_{\Omega^{\sharp}\times\Gamma_{per}^{1,vac}}\frac{\partial\phi^0}{\partial x_i^1}wn_i^1\,\mathrm{d}s\left(x^1\right)\mathrm{d}x^{\sharp} = 0,$$

applying the periodicity on $\Gamma_{per}^{1,vac}$ of w^1, we get $\forall x^{\sharp}\in\Omega^{\sharp}$,

$$\frac{\partial\phi^0}{\partial x_i^1}n_i^1 \text{ is } \Gamma_{per}^{1,vac} \text{ anti-periodic.} \tag{3.65}$$

Equation (3.63) with (3.64), (3.65) and if $w\in C^{\infty}\left(\Omega^{\sharp,vac}, C^{\infty}_{\partial\Omega^{1,vac}}\left(\Omega^{1,vac}\right)\right)$ satisfying $\partial_{x_i^1}wn_i^1 = 0$ on $\Gamma_{ext}^{1,vac}$, it becomes

$$|\Gamma_{int}^{\varepsilon,vac}|\oint_{\Omega^{\sharp}\times\Gamma_{int}^{1,vac}}\left(\phi^0 - V^0\right)\frac{\partial w}{\partial x_i^1}n_i^1\,\mathrm{d}s\left(x^1\right) = 0,$$

which leads to $\forall x^\sharp \in \Omega^\sharp$,

$$\phi^0 = V^0 \text{ on } \Gamma^{1,vac}_{int}. \tag{3.66}$$

Equation (3.63) with (3.64), (3.65), (3.66) with if w satisfies $\partial_{x_i^1} w n_i = 0$ on $\partial\Omega^{1,vac}$, $w = 0$ on $\Gamma^{1,vac}_{per} \cup \Gamma^{1,vac}_{int}$ becomes

$$\int_{\Omega^\sharp \times \Gamma^{1,vac}_{+\cup-}} w \frac{\partial \phi^0}{\partial x_i^1} n_i^1 \, \mathrm{d}s\left(x^1\right) \mathrm{d}x^\sharp = 0,$$

which leads to $\forall x^\sharp \in \Omega^\sharp$,

$$\frac{\partial \phi^0}{\partial x_i^1} n_i^1 = 0 \text{ on } \Gamma^{1,vac}_{+\cup-}. \tag{3.67}$$

Equation (3.63) with (3.64), (3.65), (3.66) and if $w = 0$ on $\Gamma^{1,vac}_{int}$, $\partial w / \partial x_i^1 n_i = 0$ on $\Gamma^{1,vac}_{int} \cup \Gamma^{1,vac}_{+\cup-}$, we get

$$\int_{\Omega^\sharp \times \Gamma^{1,vac}_{per}} \phi^0 \frac{\partial w}{\partial x_i^1} n_i^1 \, \mathrm{d}s\left(x^1\right) \mathrm{d}x^\sharp = 0,$$

applying the periodicity of w^1, we get $\forall x^\sharp \in \Omega^\sharp$,

$$\phi^0 \text{ is } \Gamma^{1,vac}_{per}\text{-periodic.} \tag{3.68}$$

Finally, from (3.64), (3.65), (3.66), (3.67), (3.68), the governing equation ϕ^0 in $\Omega^{1,vac}$ is given by $\forall x^\sharp \in \Omega^\sharp$,

$$\left\{\begin{array}{lll} -\mathrm{div}\left(\nabla_{x^1}\phi^0\right) = 0 & \text{in} & \Omega^{1,vac} \\ \phi^0 = V^0 & \text{on} & \Gamma^{1,vac}_{int} \\ \nabla_{x^1}\phi^0 \cdot \mathbf{n}^1 = 0 & \text{on} & \Gamma^{1,vac}_{+\cup-} \\ \nabla_{x^1}\phi^0 \cdot \mathbf{n}^1 & \text{is} & \Gamma^{1,vac}_{per} \text{ -anti-periodic} \\ \phi^0 & \text{is} & \Gamma^{1,vac}_{per} \text{ – periodic} \end{array}\right. . \tag{3.69}$$

Model derivation of boundary layers for the two ends and the interface : Replacing the test function in the very weak formulation (3.60) by $\left(B_b^\vartheta v_b^\vartheta + B_b^{interf} v_b^{interf}\right)(x^\varepsilon)$, $\vartheta \in \{0, L_1\}$, $v_b^\vartheta \in H^1_{\Gamma^{1,vac,+\infty}_{int},0}(\Omega^{1,vac,+\infty}) \cap H^2_{\Gamma^{1,vac,+\infty}_{int},\Gamma^{1,vac,+\infty}_{ext}}(\Omega^{1,vac,+\infty})$ and $v_b^{interf} \in H^1_{\Gamma^{1,vac,\infty}_{int},0}(\Omega^{1,vac,\infty}) \cap H^2_{\Gamma^{1,vac,\infty}_{int},\Gamma^{1,vac,\infty}_{ext}}(\Omega^{1,vac,\infty})$ we get

$$\oint_{\Omega^{\varepsilon,vac}} \phi^\varepsilon \frac{\partial}{\partial x_i^\varepsilon} \frac{\partial}{\partial x_i^\varepsilon} \left(B_b^\vartheta v_b^\vartheta + B_b^{interf} v_b^{interf}\right) \mathrm{d}x^\varepsilon = \frac{|\Gamma^{\varepsilon,vac}_{int}|}{|\Omega^{\varepsilon,vac}|} \oint_{\Gamma^{\varepsilon,vac}_{int}} V^\varepsilon \frac{\partial}{\partial x_i^\varepsilon} \left(B_b^\vartheta v_b^\vartheta + B_b^{interf} v_b^{interf}\right) n_i^\varepsilon \, \mathrm{d}s\left(x^\varepsilon\right).$$

Applying equations (3.46) and (3.47), the second order partial derivatives $\partial_{x_i^\varepsilon}\partial_{x_i^\varepsilon} B_b^\vartheta v_b^\vartheta$ and $\partial_{x_i^\varepsilon}\partial_{x_i^\varepsilon} B_b^{interf} v_b^{interf}$ can be computed as

$$\frac{\partial}{\partial x_i^\varepsilon} \frac{\partial B_b^\vartheta v_b^\vartheta}{\partial x_i^\varepsilon} = \frac{1}{\varepsilon^2} B_b^\vartheta \left(\frac{\partial}{\partial x_i^1} \frac{\partial v_b^\vartheta}{\partial x_i^1}\right) \text{ and}$$

$$\frac{\partial}{\partial x_i^\varepsilon} \frac{\partial B_b^{interf} v_b^{interf}}{\partial x_i^\varepsilon} = \frac{1}{\varepsilon^2} B_b^{interf} \left(\frac{\partial}{\partial x_i^1} \frac{\partial v_b^{interf}}{\partial x_i^1}\right).$$

The above equation becomes

$$\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon}\left(\frac{1}{\varepsilon^2}B_b^{\vartheta}\left(\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\right)+\frac{1}{\varepsilon^2}B_b^{interf}\left(\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{interf}}{\partial x_i^1}\right)\right)\mathrm{d}x^{\varepsilon}$$
$$=\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}\oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon}\left(C\left(\vartheta,i\right)\frac{1}{\varepsilon}B_b^{\vartheta}\left(\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\right)+\frac{1}{\varepsilon}B_b^{interf}\left(\frac{\partial v_b^{interf}}{\partial x_i^1}\right)\right)n_i^{\varepsilon}\,\mathrm{d}s\left(x^{\varepsilon}\right)\text{ or}$$

$$\frac{1}{\varepsilon}\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon}B_b^{\vartheta}\left(\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\right)\mathrm{d}x^{\varepsilon}+\frac{1}{\varepsilon}\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon}B_b^{interf}\left(\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{interf}}{\partial x_i^1}\right)\mathrm{d}x^{\varepsilon}$$
$$=\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}\oint_{\Gamma_{int}^{\varepsilon,vac}} C\left(\vartheta,i\right)V^{\varepsilon}B_b^{\vartheta}\left(\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\right)n_i^{\varepsilon}\,\mathrm{d}s\left(x^{\varepsilon}\right)+\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}\oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon}B_b^{interf}\left(\frac{\partial v_b^{interf}}{\partial x_i^1}\right)n_i^{\varepsilon}\,\mathrm{d}s\left(x^{\varepsilon}\right).$$

Applying the properties $B_b^{\vartheta}v = T_b^{\vartheta *}v\chi_{\Omega^{\varepsilon,s(\vartheta)}}\left(x_1^{\varepsilon}\right),\ \forall v \in L^2\left(\Omega^{1,+\infty}\right)$ and $B_b^{\vartheta}v_i = (-1)^{\chi_{\{L_1\}}(\vartheta)}T_b^{\vartheta *}v_i\chi_{\Omega^{\varepsilon,s(\vartheta)}}\left(x_1^{\varepsilon}\right),\ \forall v_i \in C^{\infty}\left(\Gamma_{int}^{\varepsilon}\right)$ and $B_b^{interf}v = T_b^{interf *}v, \forall v \in L^2\left(\Omega^{1,\infty}\right)$, it becomes

$$\frac{1}{\varepsilon^2}\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon}T_b^{\vartheta *}\left(\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\right)\mathrm{d}x^{\varepsilon}+\frac{1}{\varepsilon^2}\oint_{\Omega^{\varepsilon,vac}} \phi^{\varepsilon}T_b^{interf *}\left(\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{interf}}{\partial x_i^1}\right)\mathrm{d}x^{\varepsilon}$$
$$=\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}\frac{1}{\varepsilon}\oint_{\Gamma_{int}^{\varepsilon,vac}} C'\left(\vartheta,i\right)V^{\varepsilon}\left(-1\right)^{\chi_{\{L_1\}}(\vartheta)}T_b^{\vartheta *}\left(\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\right)n_i^{\varepsilon}\,\mathrm{d}s\left(x^{\varepsilon}\right)+\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}\frac{1}{\varepsilon}\oint_{\Gamma_{int}^{\varepsilon,vac}} V^{\varepsilon}T_b^{interf *}\left(\frac{\partial v_b^{interf}}{\partial x_i^1}\right)n_i^{\varepsilon}\,\mathrm{d}s\left(x^{\varepsilon}\right),$$

where $C'\left(\vartheta,i\right)=C\left(\vartheta,i\right)\left(-1\right)^{\chi_{\{L_1\}}(\vartheta)}$. Applying the Definition 6 of $T_b^{\vartheta *}$ and $T_b^{interf *}$, it becomes

$$\frac{1}{\varepsilon}\frac{\varepsilon^2}{|\Omega^{\varepsilon,vac}|}\int_{\Omega^{1,vac,+\infty}} T_b^{\vartheta}\phi^{\varepsilon}\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\,\mathrm{d}x^1+\frac{1}{\varepsilon}\frac{\varepsilon^2}{|\Omega^{\varepsilon,vac}|}\int_{\Omega^{1,vac,\infty}} T_b^{interf}\phi^{\varepsilon}\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{interf}}{\partial x_i^1}\,\mathrm{d}x^1$$
$$=\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}C'\left(\vartheta,i\right)\frac{\varepsilon}{|\Gamma_{int}^{\varepsilon,vac}|}\int_{\Gamma_{int}^{1,vac,+\infty}} T_b^{\vartheta}V^{\varepsilon}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)+\frac{|\Gamma_{int}^{\varepsilon,vac}|}{|\Omega^{\varepsilon,vac}|}\frac{\varepsilon}{|\Gamma_{int}^{\varepsilon,vac}|}\int_{\Gamma_{int}^{1,vac,\infty}} T_b^{interf}V^{\varepsilon}\frac{\partial v_b^{interf}}{\partial x_i^1}n_i^{\infty}\,\mathrm{d}s\left(x^1\right),\ \text{or}$$

$$\int_{\Omega^{1,vac,+\infty}} T_b^{\vartheta}\phi^{\varepsilon}\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\,\mathrm{d}x^1+\int_{\Omega^{1,vac,\infty}} T_b^{interf}\phi^{\varepsilon}\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{interf}}{\partial x_i^1}\,\mathrm{d}x^1$$
$$=C'\left(\vartheta,i\right)\int_{\Gamma_{int}^{1,vac,+\infty}} T_b^{\vartheta}V^{\varepsilon}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)+\int_{\Gamma_{int}^{1,vac,\infty}} T_b^{interf}V^{\varepsilon}\frac{\partial v_b^{interf}}{\partial x_i^1}n_i^{\infty}\,\mathrm{d}s\left(x^1\right),$$

where $n^{+\infty}$ and n^{∞} are unit outward normal vectors in $\Omega^{1,+\infty}$ and $\Omega^{1,\infty}$. Plugging the equations (3.56), (3.57), (3.58) and (3.59) into the above equation and passing ε to 0, we get

$$\int_{\Omega^{1,vac,+\infty}}\left(\phi^0\left(\vartheta,x^1\right)+\phi_b^{\vartheta}\right)\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}\,\mathrm{d}x^1+\int_{\Omega^{1,vac,\infty}}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)\frac{\partial}{\partial x_i^1}\frac{\partial v_b^{interf}}{\partial x_i^1}\,\mathrm{d}x^1$$
$$=C'\left(\vartheta,i\right)\int_{\Gamma_{int}^{1,vac,+\infty}}\left(V^0\left(\vartheta,x^1\right)+V_b^{\vartheta}\right)\frac{\partial v_b^{\vartheta}}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)+\int_{\Gamma_{int}^{1,vac,\infty}}\left(V^0\left(L_1^m,x^1\right)+V_b^{interf}\right)\frac{\partial v_b^{interf}}{\partial x_i^1}n_i^{\infty}\,\mathrm{d}s\left(x^1\right).$$

Applying the Green formula twice with $\partial\Omega^{1,vac,+\infty} = \Gamma^{1,vac,+\infty}_{int} \cup \Gamma^{1,vac,+\infty}_{end,\vartheta} \cup \Gamma^{1,vac,+\infty}_{lat} \cup \Gamma^{1,vac,+\infty}_{+\cup-}$, $\partial\Omega^{1,vac,\infty,\eta} = \Gamma^{1,vac,\infty,\eta}_{int} \cup \Gamma^{1,vac,\infty,\eta}_{interf} \cup \Gamma^{1,vac,\infty,\eta}_{lat} \cup \Gamma^{1,vac,\infty,\eta}_{+\cup-}$, $\eta \in \{1,2\}$ and writing them as $\partial\Omega^{1,vac,+\infty} = \Gamma^{1,vac,+\infty}_{int\cup end,\vartheta\cup lat\cup+\cup-}$ and $\Gamma^{1,vac,\infty,\eta}_{int\cup interf\cup lat\cup+\cup-}$ for simplicity, we get

$$
\begin{aligned}
&\int_{\Omega^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta,x^1\right)+\phi_b^\vartheta\right)v_b^\vartheta \,\mathrm{d}x^1 - \int_{\Gamma^{1,vac,+\infty}_{int\cup end,\vartheta\cup lat\cup+\cup-}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta,x^1\right)+\phi_b^\vartheta\right)v_b^\vartheta n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&+\int_{\Gamma^{1,vac,+\infty}_{int\cup end,\vartheta\cup lat\cup+\cup-}} \left(\phi^0\left(\vartheta,x^1\right)+\phi_b^\vartheta\right)\frac{\partial v_b^\vartheta}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_\eta \int_{\Omega^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)v_b^{interf,\eta}\,\mathrm{d}x^1\\
&-\sum_\eta \int_{\Gamma^{1,vac,\infty,\eta}_{int\cup interf\cup lat\cup+\cup-}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)v_b^{interf,\eta}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_\eta \int_{\Gamma^{1,vac,\infty,\eta}_{int\cup interf\cup lat\cup+\cup-}} \left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
=\;& C'(\vartheta,i)\int_{\Gamma^{1,vac,+\infty}_{int}} \left(V^0\left(\vartheta,x^1\right)+V_b^\vartheta\right)\frac{\partial v_b^\vartheta}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_\eta \int_{\Gamma^{1,vac,\infty,\eta}_{int}} \left(V^0\left(L_1^m,x^1\right)+V_b^{interf}\right)\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right).
\end{aligned}
$$

From assumption of v_b^ϑ such as $v_b^\vartheta = 0$ on $\Gamma^{1,vac,+\infty}_{int}$ and $\partial_{x_i^1} v_b^\vartheta n_i^{+\infty} = 0$ on $\Gamma^{1,vac,+\infty}_{end,\vartheta\cup lat\cup+\cup-}$, and of $v_b^{interf,\eta}$ such as $v_b^{interf,\eta} = 0$ on $\Gamma^{1,vac,\infty,\eta}_{int}$ and $\partial_{x_i^1} v_b^{interf,\eta} n_i^{\infty,\eta} = 0$ on $\Gamma^{1,vac,\infty,\eta}_{lat\cup+\cup-}$, it becomes

$$
\begin{aligned}
&\int_{\Omega^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta,x^1\right)+\phi_b^\vartheta\right)v_b^\vartheta \,\mathrm{d}x^1 - \int_{\Gamma^{1,vac,+\infty}_{end,\vartheta\cup lat\cup+\cup-}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta,x^1\right)+\phi_b^\vartheta\right)v_b^\vartheta n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&+\int_{\Gamma^{1,vac,+\infty}_{int}} \left(\phi^0\left(\vartheta,x^1\right)+\phi_b^\vartheta\right)\frac{\partial v_b^\vartheta}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_\eta \int_{\Omega^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)v_b^{interf,\eta}\,\mathrm{d}x^1\\
&-\sum_\eta \int_{\Gamma^{1,vac,\infty,\eta}_{interf\cup lat\cup+\cup-}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)v_b^{interf,\eta}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_\eta \int_{\Gamma^{1,vac,\infty,\eta}_{interf\cup int}} \left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
=\;& C'(\vartheta,i)\int_{\Gamma^{1,vac,+\infty}_{int}} \left(V^0\left(\vartheta,x^1\right)+V_b^\vartheta\right)\frac{\partial v_b^\vartheta}{\partial x_i^1}n_i^{+\infty,\vartheta}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_\eta \int_{\Gamma^{1,vac,\infty,\eta}_{int}} \left(V^0\left(L_1^m,x^1\right)+V_b^{interf}\right)\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right).
\end{aligned}
$$

The equation (3.69) implies $\partial_{x_i^1}\partial_{x_i^1}\phi^0\left(\vartheta,x^1\right) = \partial_{x_i^1}\partial_{x_i^1}\phi^0\left(L_1^m,x^1\right) = 0$, and $\phi^0\left(\vartheta,x^1\right) =$

$V^0\left(\vartheta, x^1\right)$ on $\Gamma_{int}^{1,vac,+\infty}$, $\phi^0\left(L_1^m, x^1\right) = V^0\left(L_1^m, x^1\right)$ on $\Gamma_{int}^{1,vac,\infty,\eta}$, the above equation becomes

$$\begin{aligned}
&\int_{\Omega^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\frac{\partial \phi_b^\vartheta}{\partial x_i^1} v_b^\vartheta \,\mathrm{d}x^1 - \int_{\Gamma_{end,\vartheta\cup lat\cup+\cup-}^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta, x^1\right) + \phi_b^\vartheta\right) v_b^\vartheta n_i^{+\infty}\,\mathrm{d}s\left(x^1\right) \qquad (3.70)\\
&+ \int_{\Gamma_{int}^{1,vac,+\infty}} \phi_b^\vartheta \frac{\partial v_b^\vartheta}{\partial x_i^1} n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&+ \sum_\eta \int_{\Omega^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\frac{\partial \phi_b^{interf}}{\partial x_i^1} v_b^{interf,\eta}\,\mathrm{d}x^1\\
&- \sum_\eta \int_{\Gamma_{interf\cup lat\cup+\cup-}^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\right) v_b^{interf,\eta} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right) \qquad (3.71)\\
&+ \sum_\eta \int_{\Gamma_{interf}^{1,vac,\infty,\eta}} \left(\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\right) \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
&+ \sum_\eta \int_{\Gamma_{int}^{1,vac,\infty,\eta}} \phi_b^{interf} \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right) \qquad (3.72)\\
=\;& C\left(\vartheta, i\right) \int_{\Gamma_{int}^{1,vac,+\infty}} V_b^\vartheta \frac{\partial v_b^\vartheta}{\partial x_i^1} n_i^{+\infty,\vartheta}\,\mathrm{d}s\left(x^1\right) + \sum_\eta \int_{\Gamma_{int}^{1,vac,\infty,\eta}} V_b^{interf} \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right).
\end{aligned}$$

If $v_b^\vartheta = 0$ on $\Gamma_{end,\vartheta\cup lat\cup+\cup-}^{1,vac,+\infty}$, $\partial_{x_i^1} v_b^\vartheta n_i^{+\infty} = 0$ on $\Gamma_{int}^{1,vac,+\infty}$, and $v_b^{interf,\eta} = 0$ on $\Gamma_{interf\cup lat\cup+\cup-}^{1,vac,\infty,\eta}$, $\partial_{x_i^1} v_b^{interf,\eta} n_i^{\infty,\eta} = 0$ on $\Gamma_{interf\cup int}^{1,vac,\infty,\eta}$, the above equation becomes

$$\int_{\Omega^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\frac{\partial \phi_b^\vartheta}{\partial x_i^1} v_b^\vartheta \,\mathrm{d}x^1 + \sum_\eta \int_{\Omega^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\frac{\partial \phi_b^{interf}}{\partial x_i^1} v_b^{interf,\eta}\,\mathrm{d}x^1 = 0.$$

Applying the interpretation of the weak equality, we get

$$\frac{\partial}{\partial x_i^1}\frac{\partial \phi_b^\vartheta}{\partial x_i^1} = 0 \text{ in } \Omega^{1,vac,+\infty} \text{ and } \frac{\partial}{\partial x_i^1}\frac{\partial \phi_b^{interf,\eta}}{\partial x_i^1} = 0 \text{ in } \Omega^{1,vac,\infty,\eta}. \qquad (3.73)$$

Applying (3.73) to (3.70), we get

$$\begin{aligned}
&- \int_{\Gamma_{end,\vartheta\cup lat\cup+\cup-}^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta, x^1\right) + \phi_b^\vartheta\right) v_b^\vartheta n_i^{+\infty}\,\mathrm{d}s\left(x^1\right) + \int_{\Gamma_{int}^{1,vac,+\infty}} \phi_b^\vartheta \frac{\partial v_b^\vartheta}{\partial x_i^1} n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&- \sum_\eta \int_{\Gamma_{interf\cup lat\cup+\cup-}^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\right) v_b^{interf,\eta} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
&+ \sum_\eta \int_{\Gamma_{interf}^{1,vac,\infty,\eta}} \left(\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\right) \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right) \qquad (3.74)\\
&+ \sum_\eta \int_{\Gamma_{int}^{1,vac,\infty,\eta}} \phi_b^{interf} \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right) \qquad (3.75)\\
=\;& C'\left(\vartheta, i\right) \int_{\Gamma_{int}^{1,vac,+\infty}} V_b^\vartheta \frac{\partial v_b^\vartheta}{\partial x_i^1} n_i^{+\infty}\,\mathrm{d}s\left(x^1\right) + \sum_\eta \int_{\Gamma_{int}^{1,vac,\infty,\eta}} V_b^{interf} \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right).
\end{aligned}$$

If $v_b^{\vartheta}=0$ on $\Gamma_{lat\cup end,\vartheta\cup+\cup-}^{1,vac,+\infty}$, $v_b^{interf,\eta}=0$ on $\Gamma_{interf\cup lat\cup+\cup-}^{1,vac,\infty,\eta}$ and $\partial_{x_i^1}v_b^{interf,\eta}n_i^{\infty,\eta}=0$ on $\Gamma_{interf}^{1,vac,\infty,\eta}$, it becomes

$$\begin{aligned}&\int_{\Gamma_{int}^{1,vac,+\infty}}\phi_b^{\vartheta}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)+\sum_{\eta}\int_{\Gamma_{int}^{1,vac,\infty,\eta}}\phi_b^{interf}\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\ =&\sum_{\vartheta,i}C'\left(\vartheta,i\right)\int_{\Gamma_{int}^{1,vac,+\infty}}V_b^{\vartheta}\frac{\partial v_b^{\vartheta}}{\partial x_i^1}n_i^{+\infty,\vartheta}\,\mathrm{d}s\left(x^1\right)+\sum_{\eta}\int_{\Gamma_{int}^{1,vac,\infty,\eta}}V_b^{interf}\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\end{aligned}\tag{3.76}$$

In the case $\vartheta=0$, the constant $C'\left(0,i\right)=(-1)^{\chi_{\{L_1\}}(0)\chi_{\{1\}}(i)}\left(-1\right)^{\chi_{\{L_1\}}(0)}=1,\forall i\in\{1,2,3\}$ and the equation above becomes

$$\int_{\Gamma_{int}^{1,vac,+\infty}}\left(\phi_b^0-V_b^0\right)\frac{\partial v_b^0}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)+\sum_{\eta}\int_{\Gamma_{int}^{1,vac,\infty,\eta}}\left(\phi_b^{interf}-V_b^{interf}\right)\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)=0.$$

Applying the interpretation of the weak equality, we have

$$\phi_b^0=V_b^0\text{ on }\Gamma_{int}^{1,vac,+\infty}\text{ and }\phi_b^{interf}=V_b^{interf}\text{ on }\Gamma_{int}^{1,vac,\infty,\eta}.\tag{3.77}$$

Applying (3.77) to the above equation and in the case $\vartheta=L_1$, the constant $C'\left(L_1,i\right)=(-1)^{\chi_{\{L_1\}}(L_1)\chi_{\{1\}}(i)}\left(-1\right)^{\chi_{\{L_1\}}(L_1)}=-\left(-1\right)^{\chi_{\{1\}}(i)}$ takes the value 1 if $i=1$ and -1 otherwise. We get

$$\int_{\Gamma_{int}^{1,vac,+\infty}}\phi_b^{L_1}\frac{\partial v_b^{L_1}}{\partial x_i^1}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)-\sum_i C'\left(L_1,i\right)\int_{\Gamma_{int}^{1,vac,+\infty}}V_b^{L_1}\frac{\partial v_b^{L_1}}{\partial x_i^1}n_i^{+\infty,L_1}\,\mathrm{d}s\left(x^1\right)=0\text{ or}$$

$$\begin{aligned}&\int_{\Gamma_{int}^{1,vac,+\infty}}\left(\phi_b^{L_1}-V_b^{L_1}\right)\frac{\partial v_b^{L_1}}{\partial x_1^1}n_1^{+\infty}\,\mathrm{d}s\left(x^1\right)+\int_{\Gamma_{int}^{1,vac,+\infty}}\left(\phi_b^{L_1}+V_b^{L_1}\right)\frac{\partial v_b^{L_1}}{\partial x_2^1}n_2^{+\infty}\,\mathrm{d}s\left(x^1\right)\\&+\int_{\Gamma_{int}^{1,vac,+\infty}}\left(\phi_b^{L_1}+V_b^{L_1}\right)\frac{\partial v_b^{L_1}}{\partial x_3^1}n_3^{+\infty}\,\mathrm{d}s\left(x^1\right)=0.\end{aligned}$$

Since $n^{+\infty}=(-1,0,0)$, it becomes

$$\int_{\Gamma_{int}^{1,vac,+\infty}}\left(\phi_b^{L_1}-V_b^{L_1}\right)\frac{\partial v_b^{L_1}}{\partial x_1^1}n_1^{+\infty}\,\mathrm{d}s\left(x^1\right)=0.$$

Applying the interpretation of the weak equality, we have

$$\phi_b^{L_1}=V_b^{L_1}\text{ on }\Gamma_{int}^{1,vac,+\infty}.\tag{3.78}$$

The equations (3.77) and (3.78) give us

$$\phi_b^{\vartheta}=V_b^{\vartheta}\text{ on }\Gamma_{int}^{1,vac,+\infty}.\tag{3.79}$$

Applying (3.73), (3.77) and (3.78) to (3.70), we get

$$\begin{aligned}&-\int_{\Gamma_{end,\vartheta\cup lat\cup+\cup-}^{1,vac,+\infty}}\frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta,x^1\right)+\phi_b^{\vartheta}\right)v_b^{\vartheta}n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\&-\sum_{\eta}\int_{\Gamma_{interf\cup lat\cup+\cup-}^{1,vac,\infty,\eta}}\frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)v_b^{interf,\eta}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\&+\sum_{\eta}\int_{\Gamma_{interf}^{1,vac,\infty,\eta}}\left(\phi^0\left(L_1^m,x^1\right)+\phi_b^{interf}\right)\frac{\partial v_b^{interf,\eta}}{\partial x_i^1}n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)=0.\end{aligned}$$

If $v_b^{\vartheta} = 0$ on $\Gamma_{end,\vartheta}^{1,vac,+\infty}$ and $v_b^{interf,\eta} = 0$ on $\Gamma_{interf}^{1,vac,\infty,\eta}$ and $\partial_{x_i^1} v_b^{interf,\eta} n_i^{\infty,\eta} = 0$ on $\Gamma_{interf}^{1,vac,\infty,\eta}$, it becomes

$$
\begin{aligned}
&-\int_{\Gamma_{lat\cup+\cup-}^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta, x^1\right)+\phi_b^{\vartheta}\right) v_b^{\vartheta} n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&-\sum_{\eta}\int_{\Gamma_{lat\cup+\cup-}^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}\right) v_b^{interf} n_i^{\infty}\,\mathrm{d}s\left(x^1\right)=0.
\end{aligned}
$$

Applying the interpretation of weak equality, we get

$$
\frac{\partial \phi_b^{\vartheta}}{\partial x_i^1} n_i^{+\infty} = -\frac{\partial \phi^0\left(\vartheta, x^1\right)}{\partial x_i^1} n_i^{+\infty} = 0 \text{ on } \Gamma_{lat\cup+\cup-}^{1,vac,+\infty} \text{ and } \frac{\partial \phi_b^{interf}}{\partial x_i^1} n_i^{\infty,\eta} = -\frac{\partial \phi^0\left(L_1^m, x^1\right)}{\partial x_i^1} n_i^{\infty,\eta} = 0 \text{ on } \Gamma_{lat\cup+\cup-}^{1,vac,\infty,\eta}. \tag{3.80}
$$

Applying (3.80) to above equation, we get

$$
\begin{aligned}
&-\int_{\Gamma_{end,\vartheta}^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta, x^1\right)+\phi_b^{\vartheta}\right) v_b^{\vartheta} n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&-\sum_{\eta}\int_{\Gamma_{interf}^{1,vac,\infty,\eta}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}\right) v_b^{interf,\eta} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)\\
&+\sum_{\eta}\int_{\Gamma_{interf}^{1,vac,\infty,\eta}} \left(\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}\right) \frac{\partial v_b^{interf,\eta}}{\partial x_i^1} n_i^{\infty,\eta}\,\mathrm{d}s\left(x^1\right)=0.
\end{aligned}
$$

If v_b^{interf} and $\partial_{x_i^1} v_b^{interf}$ is continuous on $\Omega^{1,vac,\infty}$, we get

$$
\begin{aligned}
&-\int_{\Gamma_{end,\vartheta}^{1,vac,+\infty}} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(\vartheta, x^1\right)+\phi_b^{\vartheta}\right) v_b^{\vartheta} n_i^{+\infty}\,\mathrm{d}s\left(x^1\right)\\
&-\int_{\Gamma_{interf}^{1,vac,\infty}} \left[\sum_{\eta} \frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}\right)_{|\Omega^{1,vac,\infty,\eta}} n_i^{\infty,\eta}\right] v_b^{interf}\,\mathrm{d}s\left(x^1\right)\\
&+\int_{\Gamma_{interf}^{1,vac,\infty}} \left[\sum_{\eta} \left(\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}\right)_{|\Omega^{1,vac,\infty,\eta}} n_i^{\infty,\eta}\right] \frac{\partial v_b^{interf}}{\partial x_i^1}\,\mathrm{d}s\left(x^1\right)=0.
\end{aligned}
$$

Applying the interpretation of weak equality with $n^{\infty,1} = (-1,0,0)$ and $n^{\infty,2} = (1,0,0)$, we get

$$
\begin{aligned}
\frac{\partial \phi_b^{\vartheta}}{\partial x_i^1} n_i^{+\infty} &= -\frac{\partial \phi^0\left(\vartheta, x^1\right)}{\partial x_i^1} n_i^{+\infty} \text{ on } \Gamma_{end,\vartheta}^{1,vac,+\infty}, \qquad (3.81)\\
[\![\frac{\partial}{\partial x_i^1}\left(\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}\right)]\!] &= 0 \text{ on } \Gamma_{interf}^{1,vac,\infty} \text{ and}\\
[\![\phi^0\left(L_1^m, x^1\right)+\phi_b^{interf}]\!] &= 0 \text{ on } \Gamma_{interf}^{1,vac,\infty},
\end{aligned}
$$

where $[\![f(x)]\!]$ is the "jump" of $f(x)$ at the interface. Finally, from (3.73), (3.77), (3.79), (3.80) and (3.81), the governing system for boundary layers at the two ends and at the interface are given as

$$
\left\{\begin{array}{lll}
-\mathrm{div}_{x^1}\left(\nabla_{x^1}\phi_b^{\vartheta}\right)=0 & \text{in} & \Omega^{1,vac,+\infty}\\
\phi_b^{\vartheta}=V_b^{\vartheta} & \text{on} & \Gamma_{int}^{1,vac,+\infty}\\
\nabla_{x^1}\phi_b^{\vartheta}\cdot\mathbf{n}^{+\infty}=0 & \text{on} & \Gamma_{lat\cup+\cup-}^{1,vac,+\infty}\\
\nabla_{x^1}\phi_b^{\vartheta}\cdot\mathbf{n}^{+\infty}=-\nabla_{x^1}\phi^0\left(\vartheta, x^1\right)\cdot\mathbf{n}^{+\infty} & \text{on} & \Gamma_{end,\vartheta}^{1,vac,+\infty}
\end{array}\right. \text{ and} \tag{3.82}
$$

$$\begin{cases} -\mathrm{div}_{x^1}\left(\nabla_{x^1}\phi_b^{interf}\right) = 0 & \text{in} \quad \Omega^{1,vac,\infty} \\ \phi_b^{interf} = V_b^{interf} & \text{on} \quad \Gamma_{int}^{1,vac,\infty} \\ \nabla_{x^1}\phi_b^{interf}\cdot\mathbf{n}^{\infty} = 0 & \text{on} \quad \Gamma_{lat\cup+\cup-}^{1,vac,\infty} \\ [\![\partial_{x_1^1}\left(\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}\right)]\!] = 0 & \text{on} \quad \Gamma_{interf}^{1,vac,\infty} \\ [\![\phi^0\left(L_1^m, x^1\right) + \phi_b^{interf}]\!] = 0 & \text{on} \quad \Gamma_{interf}^{1,vac,\infty} \end{cases} . \tag{3.83}$$

3.3/ Multi-scale Model Implementation

In this section, the main solution ϕ^0 is calculated on the microscopic domain Ω^1 for the two voltages $V^0 = 20V$ and $30V$, see Figure 3.6 for a solution with $V^0 = 20V$. The computation of the boundary layer corrector ϕ_b^{ϑ} is performed on domains starting from $x_1^1 = 0$ and being two cell long only instead of infinite domains. It is possible to restrict simulation to one or two cells because the boundary layer correctors are exponentially vanishing and their value in the second cell is already negligible, see Figure 3.7. The same principle holds for the computation of the boundary layer corrector ϕ_b^{interf} at the interface which is computed on four cells, see Figure 3.8. The full solution is built by superimposing the periodic solution ϕ^0 and the three boundary layer correctors, see Figure 3.9. The simulation time for a large array is related to the number of different voltages that are applied but not to its number of cells.

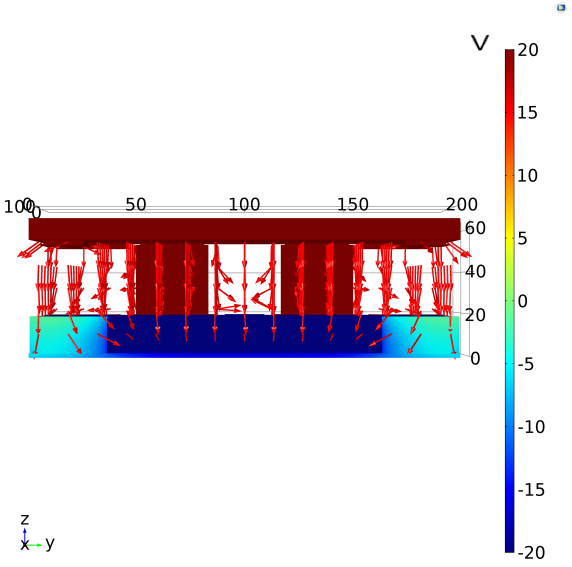


Figure 3.6: Front view of a plot of ϕ^0 in the microscopic domain. The mirror and the pillars are in red while the bottom electrode is in blue. The imposed voltages are $20V$ and $-20V$. The vector of electric field is materialized by red arrows. The electric field lines are vertical almost everywhere, with few tilted arrows visible on the edges; this means that the electric field is mainly localized in each cell, reducing to a very low value the crosstalk with neighboring cells.

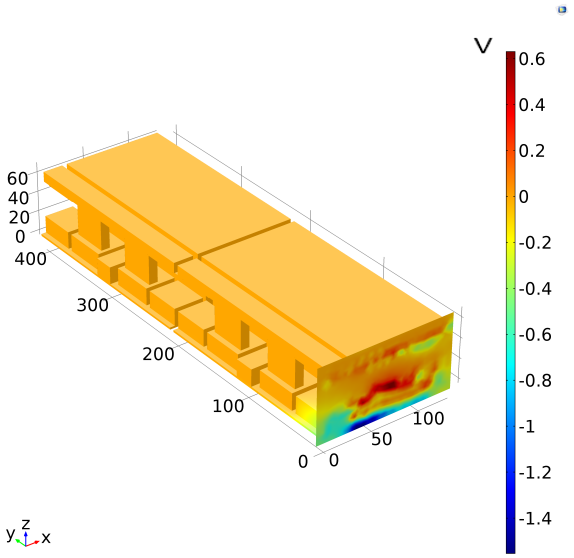


Figure 3.7: One of the two boundary layer corrections ϕ_b^0 simulated in two cells at one end of the array.

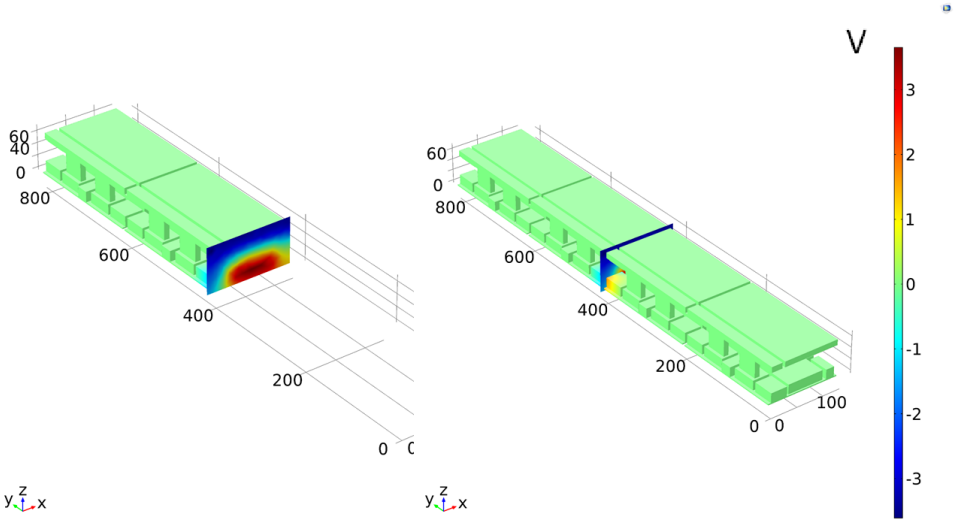


Figure 3.8: Boundary layer correction ϕ^{interf} at the interface. It is computed in four cells centered to the interface.

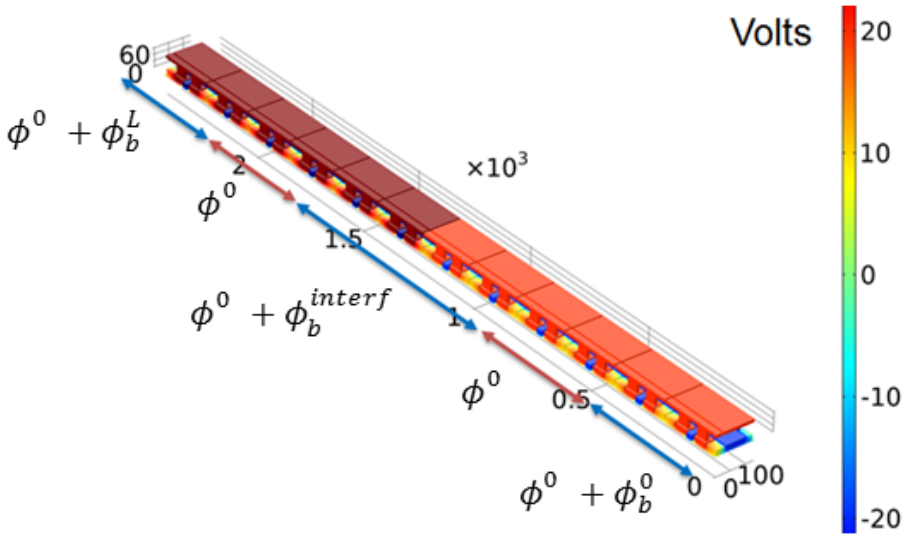


Figure 3.9: Simulation result for a twelve-cell array. The imposed voltages are $\pm 20V$ in the left part and $\pm 30V$ in the right part. The figure shows the zones of superimposition of the solutions ϕ^0, ϕ_b^ϑ and ϕ_b^{interf}.

4

EXTENSION AND COMBINATION IN MEMSALAB

4.1/ INTRODUCTION

We continue on the research path initiated in [Yang et al., 2014] where the concepts of extension and their combination were introduced for the first time. In this seminal work proofs were formalized as rewriting strategies and extensions were formalized as second-order rewriting strategies. However the combination of extensions was done via composition, not allowing for conflicts between extensions. The complete principle of the extension-combination method was introduced in [Belkhir et al., 2015]. In this work, we have presented the design and implementation of a user language for the specification of rewriting strategies based proofs and extensions. We also stated computation rules for combinations of extensions. Although we considered combinations for a small class of usual rewriting strategies as `OuterMost` and `InnerMost`, the question whether this class, or possibly a wider class, is closed under combination was left open, as well as the question of the correctness and soundness of the combination formulae.

This question was addressed in [Belkhir et al., 2016] where the authors introduced a larger the class of context embedding strategies, or CES-strategies for short. This framework involves more elementary operations but generating a wider class of rewriting strategies. Although the idea of combination is kept the same, the tools and the techniques are different. The elementary extension operation on a term is still an enrichment by context insertion. However, the traversal strategies in a CE-strategy are built with a jump operator and an iterator/fixed-point operator instead of `OuterMost` a more complex strategy. This class is indeed closed under combination and the correctness of the combination operation was proved.

Although the class of CE-strategies enjoys nice algebraic properties, it has a major practical drawback: it is built up with low level strategy constructors making it hard to use in practice. In particular, the definition of the traversal navigation strategies such as `OuterMost` yields a CE-strategy whose size depends on the signature. Even worse, the size of the resulting combined CE-strategy can be exponential with respect to the size of the two input CE-strategies. In this chapter we overcome these difficulties by finding another class of strategies, called high level context embedding strategies, or HCE-strategies for short, which is a strict subclass of the class of CE-strategies. It enjoys similar algebraic properties and seems reasonably easy to use in practice. In particular, the class of HCE-strategies is closed by combination, and the size of the resulting com-

bined HCE-strategy is polynomial with respect of the size of the two input HCE-strategies.

The strategy language underlying both the CE-strategies and the HCE-strategies is inspired by the modal μ-calculus [Arnold et al., 2001]. Instead of formulating the strategy language as in [Cirstea et al., 2003], the μ-calculus-like approach makes the strategy constructors more rudimentary and therefore tractable the question of language closure for combinations. Moreover, the formulae of combination of HCE-strategies together with their verification is also much simplified.

Organization of the Chapter: The Chapter is structured as follows. In Section 4.3 we introduce the class of elementary HCE-strategies, which is a subclass of HCE-strategies. It provides an illustration of the concept of unification and combination in simple cases and serves as a set of basic building blocks for the class of HCE-strategies. The syntax and the semantics of the latter as well as their unification and combination are introduced in Section 4.4. In Section 4.5 we argue that the unification and combination of HCE-strategies is sound and complete, and state its main algebraic properties. The proofs of the claims are similar to the ones for the class of CE-strategies [Belkhir et al., 2016].

4.2/ PRELIMINARIES

We introduce preliminary definitions and notations.

Terms, contexts. Let $\mathcal{F} = \cup_{n\geq 0}\mathcal{F}_n$ be a set of symbols called function symbols. The arity of a symbol f in $\mathcal{F}_n$ is n and is denoted $ar(f)$. Elements of arity zero are called constants and often denoted by the letters a, b, c, etc. The set $\mathcal{F}_0$ of constants is always assumed to be not empty. Given a denumerable set $\mathcal{X}$ of variable symbols, the set of terms $\mathcal{T}(\mathcal{F},\mathcal{X})$, is the smallest set containing $\mathcal{X}$ and such that $f(t_1,\ldots,t_n)$ is in $\mathcal{T}(\mathcal{F},\mathcal{X})$ whenever $ar(f) = n$ and $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ for $i \in [1..n]$. Let the constant $\Box \notin \mathcal{F}$, the set $\mathcal{T}_\Box(\mathcal{F},\mathcal{X})$ of "*contexts*", denoted simply by $\mathcal{T}_\Box$, is made with terms with symbols in $\mathcal{F} \cup \mathcal{X} \cup \{\Box\}$ which includes exactly one occurence of $\Box$. Evidently, $\mathcal{T}_\Box(\mathcal{F},\mathcal{X})$ and $\mathcal{T}(\mathcal{F},\mathcal{X})$ are two disjoint sets. For a term t and a context τ, we shall write $\tau[t]$ for the term that results from the replacement of $\Box$ by t in τ. We shall write simply $\mathcal{T}$ (resp. $\mathcal{T}_\Box$) instead of $\mathcal{T}(\mathcal{F},\mathcal{X})$ (resp. $\mathcal{T}_\Box(\mathcal{F},\mathcal{X})$). We denote by $\mathcal{V}ar(t)$ the set of variables occurring in t.

Example 1:

Out of the definition of term and context, we introduce a term t and two contexts τ_1, τ_2 as an example. Consider x as a variable which is defined on a domain Ω in $\mathbb{R}$. A term t corresponding to x having the grammar is given by [Yang, 2014]

$$t = \mathtt{Var}\left(x, \mathtt{Reg}\left(\Omega, 1\right)\right),$$

where `Var` and `Reg` are functions symbols belonging to $\mathcal{F}_2$, and x, Ω,1 are constants belonging to $\mathcal{F}_0$. The constant 1 represents the dimension of the space. Besides, a context τ_1 providing an ability to extend a term into a vector of three elements and a context τ_2 having an ability to extend a term into a vector of any size are given by the following equations

$$\tau_1 = \mathtt{List}\left(\square, \mathtt{Index}\left(i, [1,2,3]\right)\right), \tau_2 = \mathtt{List}\left(\square, \mathtt{Index}\left(j, X\right)\right),$$

where `List` and `Index` are functions symbols belonging to $\mathcal{F}_2$, and $i, j, 1, 2, 3$ are constants and X is variable symbol belong to $\mathcal{X}$. The tree structure of t, τ_1 and τ_2 are shown in Figure 4.1.

Positions, prefix-order, combination of contexts. Let t be a term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A position in a tree is a sequence of integers of $\mathbb{N}_\epsilon^\omega = \{\epsilon\} \cup \mathbb{N} \cup (\mathbb{N} \times \mathbb{N}) \cup \cdots$. In particular we shall write $\mathbb{N}_\epsilon$ for $\{\epsilon\} \cup \mathbb{N}$. Given two positions $p = p_1 p_2 \ldots p_n$ and $q = q_1 q_2 \ldots q_m$, the concatenation of p and q, denoted by $p \cdot q$ or simply pq, is the position $p_1 p_2 \ldots p_n q_1 q_2 \ldots q_m$. The set of positions of the term t, denoted by $\mathcal{P}os(t)$, is a set of positions of positive integers such that, if $t \in \mathcal{X}$ is a variable or $t \in \mathcal{F}_0$ is a constant, then $\mathcal{P}os(t) = \{\epsilon\}$. If $t = f(t_1, ..., t_n)$ then $\mathcal{P}os(t) = \{\epsilon\} \cup \bigcup_{i=1,n} \{ip \mid p \in \mathcal{P}os(t_i)\}$. The position ϵ is called the root position of term t, and the function or variable symbol at this position is called root symbol of t.

The prefix order defined as $p \leq q$ iff there exists p' such that $pp' = q$, is a partial order on positions. If $p' \neq \epsilon$ then we obtain the strict order $p < q$. We write $(p \parallel q)$ iff p and q are incomparable with respect to $\leq$. The binary relations $\sqsubset$ and $\sqsubseteq$ defined by $p \sqsubset q$ iff $(p < q$ or $p \parallel q)$ and $p \sqsubseteq q$ iff $(p \leq q$ or $p \parallel q)$, are total relations on positions.

For any $p \in \mathcal{P}os(t)$ we denote by $t_{|p}$ the subterm of t at position p, that is, $t_{|\epsilon} = t$, and $f(t_1, ..., t_n)_{|iq} = (t_i)_{|q}$. For a term t, we shall denote by $\delta(t)$ the depth of t, defined by $\delta(t_0) = 0$, if $t_0 \in \mathcal{X} \cup \mathcal{F}^0$ is a variable or a constant, and $\delta(f(t_1, \ldots, t_n)) = 1 + max(\delta(t_i))$, for $i = 1, \ldots, n$. For any position $p \in \mathcal{P}os(t)$ we denote by $t[s]_p$ the term obtained by replacing the subterm of t at position p by s: $t[s]_\epsilon = s$ and $f(t_1, ..., t_n)[s]_{iq} = f(t_1, ..., t_i[s]_q, ..., t_n)$.

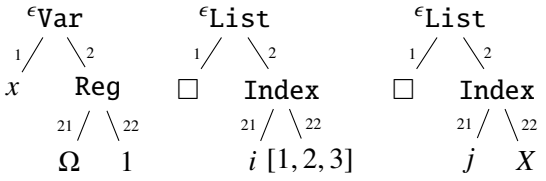


Figure 4.1: Complete tree structure of a variable $x \in \Omega \subset \mathbb{R}$, of context τ_1 and of context τ_2, given in Example 1, in MEMSALab.

Example 2:

To clarify how the depth of a tree structure is defined, we consider the case of the term t presented in Example 1. Its depth is inductively computed as

$$\delta(t_1) = 0, \delta(t_{21}) = \delta(t_{22}) = 0,$$
$$\delta(t_2) = 1 + \max\{\delta(t_{21}), \delta(t_{22})\} = 1, \text{ so that } \delta(t) = 1 + \max\{\delta(t_1), \delta(t_2)\} = 2.$$

Besides, the concept of replacement of a sub-term of a term or of a context can be recognized obviously, for instance, the new term obtained by replacing the name x by y is given as

$$t[y]_1 = \mathtt{Var}(y, \mathtt{Reg}(\Omega, 1)),$$

the tree structure of $t[y]_1$ can be found in Figure 4.2.

A substitution is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $\sigma(x) \neq x$ for only finitely many xs. The finite set of variables that σ does not map to themselves is called the domain of σ: $\mathrm{Dom}(\sigma) \overset{def}{=} \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$. If $\mathrm{Dom}(\sigma) = \{x_1, ..., x_n\}$ then we write σ as: $\sigma = \{x_1 \mapsto \sigma(x_1), ..., x_n \mapsto \sigma(x_n)\}$.

A substitution $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ uniquely extends to an endomorphism $\widehat{\sigma} : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ defined by: $\widehat{\sigma}(x) = \sigma(x)$ for all $x \in \mathrm{Dom}(\sigma)$, and $\widehat{\sigma}(x) = x$ for all $x \notin \mathrm{Dom}(\sigma)$, and $\widehat{\sigma}(f(t_1, \ldots, t_n)) = f(\widehat{\sigma}(t_1), \ldots, \widehat{\sigma}(t_n))$ for $f \in \mathcal{F}$. In what follows we do not distinguish between a substitution and its extension.

Example 3:

Consider a substitution σ of τ_2 presented in Example 1, the domain of σ is given as $\mathrm{Dom}(\sigma) = \{X\}$. We want to substitute the variable X in τ_2 by a list of constants, for instance, $[1, 2]$. Since $\mathrm{Dom}(\sigma)$ is finite, σ can be rewritten as $\sigma = \{X \to [1, 2]\}$. The result of the application of σ to τ_2 is given as

$$\sigma(\tau_2) = (X \to [1, 2])(\tau_2) = \tau_2[[1, 2]]_{22}.$$

The complete tree structure of $\sigma(\tau_2)$ can be found in Figure 4.2.

For two terms $t, t' \in \mathcal{T}$, we say that t matches t', written $t \ll t'$, iff there exists a substitution σ, such that $\sigma(t) = t'$. It turns out that if such a substitution exists, then it is unique. A term t' is subsumed by a term t iff there exists a substitution σ such that $\sigma(t) = t'$. A substitution σ' is subsumed by a substitution σ iff $\sigma'(t)$ is subsumed by $\sigma(t)$ for each term t. The most general unifier of the two terms u and u' is a substitution γ such that $\gamma(u) = \gamma(u')$ and, for any other substitution γ' satisfying $\gamma'(u) = \gamma'(u')$, we have that γ' is subsumed by γ. Besides, we shall write $u \wedge u'$ to denote the term $\gamma(u)$. The composition of functions will

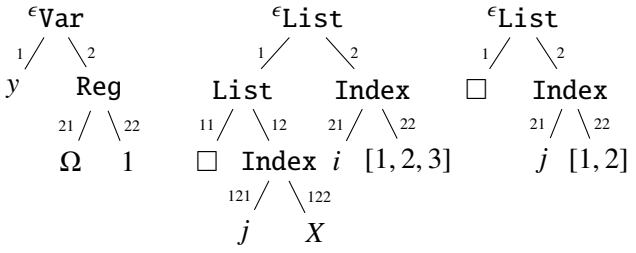


Figure 4.2: Complete tree structure of the term $t[y]_1$ presented in Example 2, of the combination of τ_1 and τ_2 and of $\sigma(\tau_2)$ discussed in Example 3.

be denoted by "$\circ$". If l_1 and l_2 are lists, then we denote by $l_1 \sqcup l_2$ their concatenation Sometimes we shall write $\sqcup_{i=1,n} e_i$ to denote the list $[e_1, \ldots, e_n]$. For any $n \in \mathbb{N}$ we simply denote by $[n]$ the interval $[1, \ldots, n]$.

Example 4:

Consider two terms t_1 and t_2 representing two mathematical variables given as the following equations

$$t_1 = \mathtt{Var}(X, \mathtt{Reg}(\Omega, 1)), t_2 = \mathtt{Var}(y, \mathtt{Reg}(Y, 1)),$$

where X and Y are variables, t_1 matches t_2 under a unique substitution $\gamma = \{X \to y, Y \to \Omega\}$.

4.3/ POSITION-BASED HCE-STRATEGIES AND THEIR COMBINATION

We need to consider the combination of contexts when they are inserted at the same position.

Definition 8: Combination of contexts

For any $\tau, \tau' \in \mathcal{T}_\Box$, the combination of two contexts, denoted by "$\cdot$", is defined by

$$\tau \cdot \tau' = \tau[\tau']_{\mathcal{P}os(\tau,\Box)},$$

where $\mathcal{P}os\,(t, \Box)$ is the position of $\Box$ in t.

Example 5:

The combination of the two contexts τ_1 and τ_2 given in Example 1 is given as

$$\tau_1 \cdot \tau_2 = \tau_1[\tau_2]_1 = \mathtt{List}(\mathtt{List}(\Box, \underline{j}), \underline{i})$$

where $\underline{i}$ and $\underline{j}$ are shortcut terms which represent $\mathtt{Index}(i, [1, 2, 3])$ and $\mathtt{Index}(j, X)$ respectively. This concept has already been presented in [Yang, 2014]. The complete tree structure of this term is shown in Figure 4.2.

To define the position-based HCE-strategies, we introduce two position-based strategies.

For a position p and a context τ, the jump strategy $@p.\tau$ applied to a term t inserts τ at the position p of the input term t. The failing strategy $\varnothing$ fails when applied to any term. Their precise semantics are given in Definition 9 below for Semantics of position-based HCE-strategies.

Definition 9: Position-based HCE-strategies

A position-based HCE-strategy is either the failing strategy $\varnothing$ or the list $[@p_1.\tau_1, \ldots, @p_n.\tau_n]$, where $n \geq 1$, each p_i is a positions and each τ_i is a context in $\mathcal{T}_{\Box}$.

We impose that the position-based HCE-strategies respect some constraints on positions of insertions to avoid conflicts: the order of context insertions goes up from the leaves to the root.

Definition 10: Well-founded position-based HCE-strategy

A position-based HCE-strategy $E = [@p_1.\tau_1, ..., @p_n.\tau_n]$ is well-founded iff

i.) a position occurs at most once in E, i.e. $p_i \neq p_j$ for all $i \neq j$, and

ii.) insertions at lower positions occur earlier in E, i.e. $i < j$ if $p_i \sqsubset p_j$, for all $i, j \in [n]$.

In particular, the empty position-based HCE-strategy $\varnothing$ is well-founded.

In all what follows we work only with the set of well-founded position-based HCE-strategies, denoted by $\mathcal{E}$. For two position-based HCE-strategies E and E', we shall abuse of notation and write $E = E'$ to mean that they are equal up to a permutation of their parallel positions. We shall simply write $@p.\tau$ instead of $[@p.\tau]$. For a position p, we let $p.[@p_1.\tau_1, \ldots, @p_n.\tau_n] = [@pp_1.\tau_1, \ldots, @pp_n.\tau_n]$. We next define the semantics of a position-based HCE-strategy as a function in $\mathcal{T} \cup \{\mathbb{F}\} \to \mathcal{T} \cup \{\mathbb{F}\}$, with the idea that if the application of a position-based HCE-strategy to a term fails, the result is $\mathbb{F}$. Besides, we adopt a stronger version of failure, that is, $[@p_1.\tau_1, \ldots, @p_n.\tau_n]$ fails when each of $@p_i.\tau_i$ fails. To formalize this notion of failure we need to introduce an intermediary function $\eta : (\mathcal{T} \cup \{\mathbb{F}\} \to \mathcal{T} \cup \{\mathbb{F}\}) \to \mathcal{T} \cup \{\mathbb{F}\} \to \mathcal{T} \cup \{\mathbb{F}\}$, that stands for the fail as identity. It is defined for any function f in $\mathcal{T} \cup \{\mathbb{F}\} \to \mathcal{T} \cup \{\mathbb{F}\}$ and any term $t \in \mathcal{T} \cup \{\mathbb{F}\}$ by

$$(\eta(f))(t) = \begin{cases} f(t) & \text{if } f(t) \neq \mathbb{F}, \\ t & \text{otherwise.} \end{cases}$$

The semantics of position-based HCE-strategies follows.

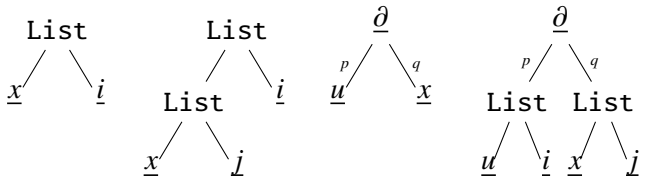


Figure 4.3: The tree structure of the terms $@\epsilon.\tau_1(t)$, $@\epsilon.\tau(t)$, t' and $[\![@p.\tau_1, @q.\tau_2]\!](t')$ discussed in Example 6.

Definition 11: Semantics of position-based HCE-strategies

The semantics of a position-based HCE-strategy E is a function $[\![E]\!]$ in $\mathcal{T} \cup \{\mathbb{F}\} \to \mathcal{T} \cup \{\mathbb{F}\}$ inductively defined by:

$$[\![\varnothing]\!](t) \stackrel{def}{=} \mathbb{F},$$

$$[\![E]\!](\mathbb{F}) \stackrel{def}{=} \mathbb{F},$$

$$[\![@p.\tau]\!](t) \stackrel{def}{=} \begin{cases} t[\tau[t_{|p}]]_p & \text{if } p \in \mathcal{P}os(t) \\ \mathbb{F} & \text{otherwise,} \end{cases}$$

$$[\![[@p_1.\tau_1, \ldots, @p_n.\tau_n]]\!](t) \stackrel{def}{=} \begin{cases} \Big((\eta([\![@p_n.\tau_n]\!])) \circ \cdots \circ (\eta([\![@p_1.\tau_1]\!]))\Big)(t) & \text{if } \exists p_i \in \{p_1, \ldots, p_n\} \\ & \text{s.t. } p_i \in \mathcal{P}os(t) \\ \mathbb{F} & \text{otherwise.} \end{cases}$$

Example 6:

We illustrate the idea and the interest of position-based HCE-strategies through the term t and the two contexts $\tau_1 = \texttt{List}(\square, \underline{i})$ and $\tau_2 = \texttt{List}(\square, \underline{j})$ defined in Example 1 but with the short-cut notation used in Example 5. Applying the strategy of $@\epsilon.\tau_1$ to the term $t = \texttt{Var}(x, \texttt{Reg}(\Omega, 1))$ gives the transformation of one-dimensional space coordinate variable x to indexed multi-dimensional space coordinate variable x_i. The procedure is given as the following equation

$$\begin{aligned}[\![@\epsilon.\tau_1]\!](t) &= t[\tau_1[t_{|\epsilon}]]_\epsilon = t[\tau_1[t]]_\epsilon = \tau_1[t] = \tau_1[t]_{\mathcal{P}os(\tau_1,\square)} \\ &= \texttt{List}\left(\texttt{Var}\left(x, \texttt{Reg}\left(\Omega, 1\right)\right), \underline{i}\right)\end{aligned}$$

Let $\tau = \tau_1 \cdot \tau_2$. The application of $@p.\tau$ to the term t is given as

$$\begin{aligned}[\![@\epsilon.\tau]\!](t) &= \tau[t]_{\mathcal{P}os(\tau_1[\tau_2],\square)} \\ &= \texttt{List}\left(\texttt{List}\left(\texttt{Var}\left(x, \texttt{Reg}\left(\Omega, 1\right)\right), \underline{j}\right), \underline{i}\right)\end{aligned}$$

Consider a term $t' = \underline{\partial}_{\underline{x}}\underline{u}$ having shortcut terms $\underline{\partial}$ and $\underline{u}$. Let p and q belong to $\mathcal{P}os(t')$ be positions of $\underline{u}$ and $\underline{x}$ in t', the application of $[\![@p.\tau_1 \circ @q.\tau_2]\!](t')$ yields the term $\underline{\partial}_{\underline{x}_{\underline{j}}}\underline{u}_{\underline{i}}$. Since the $p \parallel q$, the list of HCE-strategies is well-founded, the semantic of the application is given as

$$[\![[@p.\tau_1, @q.\tau_2]]\!](t') = ([\![@p.\tau_1]\!] \circ [\![@q.\tau_2]\!])(t') = [\![@p.\tau_1]\!]([\![@q.\tau_2]\!](t')).$$

The complete tree structures of $\underline{x}_{\underline{i}}$, $\underline{x}_{\underline{i}\underline{j}}$, t' and $\underline{\partial}_{\underline{x}_{\underline{j}}}\underline{u}_{\underline{i}}$ are depicted in Figure 4.3.

The unification of two position-based HCE-strategies amounts to sort and merge their positions, and to combine their contexts if they are inserted at the same position.

Definition 12: Unification of two position-based HCE-strategies

The unification of two position-based HCE-strategies is the binary operation $\curlywedge : \mathcal{E} \times \mathcal{E} \longrightarrow \mathcal{E}$ defined as

$$E \curlywedge E' = \begin{cases} E'' & \text{if } E \neq \varnothing \text{ and } E' \neq \varnothing \\ \varnothing & \text{if } E = \varnothing \text{ or } E' = \varnothing \end{cases}$$

where the first case $E = [@p_1.\tau_1, \ldots, @p_n.\tau_n]$, $E' = [@p'_1.\tau'_1, \ldots, @p'_m.\tau'_m]$ and $E'' = [@p''_1.\tau''_1, \ldots, @p''_r.\tau''_r]$ with sets of positions P, P' and $P'' = P \cup P'$ and the contexts τ''_k defined as follows. For a position $p''_k \in P'' \setminus P \cap P'$,

$$\tau''_k = \tau_i \text{ if } p''_k = p_i \in P \qquad \text{and} \qquad \tau''_k = \tau'_j \text{ if } p''_k = p'_j \in P'.$$

Otherwise, $p''_k = p_i = p'_j \in P \cap P'$ for some i, j and $\tau''_k = \tau_i \cdot \tau'_j$. Besides, the order of the positions in P'' is chosen so that E'' is well-founded.

Example 7:

Let the list of HCE-strategies $E = [@p_1.\tau_1, @p_2.\tau_2, @p_3.\tau_3]$ and $E' = [@p_1.\tau'_1, @q_1.\tau'_2, @q_2.\tau'_3]$, the set of positions of E and E' be $P = \{p_1, p_2, p_3\}$ and $P' = \{p_1, q_1, q_2\}$, $P \cup P' = \{p_1, p_2, p_3, q_1, q_2\}$, $P \cap P' = \{p_1\}$. The unification of E and E' is given as the following equation

$$E'' = [@p_1.\tau'_1.\tau_1, @p_2.\tau_2, @p_3.\tau_3, @q_1.\tau'_2, @q_2.\tau'_3].$$

The combination of two position-based HCE-strategies is the same as their unification apart that it is defined on non-failing position-based HCE-strategies.

Definition 13: Combination of two position-based HCE-strategies

The combination of two position-based HCE-strategies is a binary operation $\curlyvee : \mathcal{E} \times \mathcal{E} \longrightarrow \mathcal{E}$ defined for any E and E' in $\mathcal{E}$ by

$$E \curlyvee E' = \begin{cases} E \curlywedge E' & \text{if } E \neq \varnothing \text{ and } E' \neq \varnothing \\ E & \text{if } E \neq \varnothing \text{ and } E' = \varnothing \\ E' & \text{if } E = \varnothing \text{ and } E' \neq \varnothing \\ \varnothing & \text{if } E = \varnothing \text{ and } E' = \varnothing \end{cases}$$

Proposition 4:

The following hold.

1. The set $\mathcal{E}$ of position-based HCE-strategies together with the unification and combination operations enjoy the following properties.
   1. The neutral element of the unification is $@_\epsilon.\Box$, and the absorbing element is $\varnothing$.
   2. The neutral element of the combination is $\varnothing$.
   3. The unification and combination are associative, i.e. $(E \curlywedge E') \curlywedge E'' = E \curlywedge (E' \curlywedge E'')$ and $(E \curlyvee E') \curlyvee E'' = E \curlyvee (E' \curlyvee E'')$.
2. The unification and combination of position-based HCE-strategies is non commutative, i.e $E \curlywedge E' \neq E' \curlywedge E$ and $E \curlyvee E' \neq E' \curlyvee E$.

The associativity follows from the equality $(\tau_1 \cdot \tau_2) \cdot \tau_3 = \tau_1 \cdot (\tau_2 \cdot \tau_3)$, and the non-commutativity is a consequence of the fact that $\tau_1 \cdot \tau_2 \neq \tau_2 \cdot \tau_1$ in general, for any contexts τ, τ_1, τ_2 and τ_3.

4.4/ THE CLASS OF CONTEXT-EMBEDDING STRATEGIES (HCE-STRATEGIES)

We introduced the position-based HCE-strategies to clarify the ideas behind contexts, their insertion as well as their combination. However, position-based HCE-strategies are not satisfactory for practical applications, since the positions are generally not accessible and cannot be used on a regular basis in applications. So, we enrich this framework by introducing navigation strategies to form a class of HCE-strategies that is closed under combination.

4.4.1/ SYNTAX AND SEMANTICS OF HCE-STRATEGIES

A HCE-strategy is composed of two parts: a navigation of the input term without changing it, and an insertion of contexts at certain positions. We shall introduce the left-choice strategy constructor ($\oplus$), a conditional constructor "`if-then`", a restricted form of the composition, and the fixed-point constructor ("μ") allowing the recursion in the definition of strategies. The resulting class is called the class of HCE-strategies. In what follows we assume that there is a denumerable set of fixed-point variables denoted by $\mathcal{Z}$. Fixed-point variables in $\mathcal{Z}$ will be denoted by $X, Y, Z, \ldots$

Definition 14: HCE-strategies

The class of HCE-strategies is defined by the following grammar:

$$\mathcal{S} ::= \varnothing \mid X \mid \tau \mid (u, \mathcal{S}) \mid \mathcal{S} \oplus \mathcal{S} \mid \mu X.\mathcal{S} \mid [@i_1.\mathcal{S}, \ldots, @i_n.\mathcal{S}] \mid \mathtt{Most}(\mathcal{S}) \mid \mathbf{If}\ C\ \mathbf{then}\ S$$

$$C ::= \mathcal{S} \mid \mathcal{S}\ \text{and}\ \mathcal{S}$$

where X is a fixed-point variable in $\mathcal{Z}$, and τ is a context in $\mathcal{T}_\Box$, and u is a term in $\mathcal{T}$, and i is a position in $\mathbb{N}_\epsilon$, and $n \geq 1$. The set of HCE-strategies will be denoted by C.

We shall simply write $@i.\mathcal{S}$ instead of $[@i.\mathcal{S}]$ We notice that extending the class of HCE-strategies by allowing the position i of the jump operator $@i.S$ to range over $\mathbb{N}_\epsilon^\omega$ instead of $\mathbb{N}_\epsilon$ does not increase the expressiveness of the strategy language.

The design of the class of HCE-strategies is inspired by the μ-calculus formalism [Arnold et al., 2001] since we need very rudimentary strategy constructors. In particular the jumping into the immediate positions of the term tree is morally similar to the diamond and box modalities ($\langle\cdot\rangle$ and $[\cdot]$) of the propositionsal modal μ-calculus. And the fixed-point constructor is much finer than the iterate operator of e.g. [Cirstea et al., 2003]. Besides, we incorporate the left-choice strategy constructor and a restricted form of the composition.

We shall sometimes write $\mu X.\mathcal{S}(X)$ instead of $\mu X.\mathcal{S}$ to emphasize that the fixed-point variable X is free in $\mathcal{S}$.

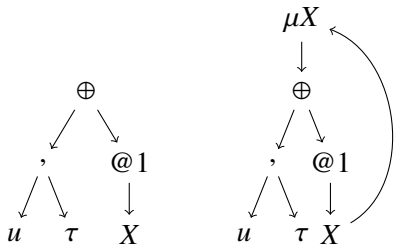


Figure 4.4: The tree-like structure of the HCE-strategy $\mathcal{S}(X) = (u, \tau) \oplus (@1.X)$ (left) and $\mu X.\mathcal{S}(X)$ (right) discussed in Example 8.

Example 8:

Consider the two HCE-strategies defined by $\mathcal{S}(X) = (u, \tau) \oplus (@1.X)$ and $\mu X.\mathcal{S}(X)$, where u is a term and τ is a context. When applied to a term t, the HCE-strategy $\mu X.\mathcal{S}(X)$ checks first whether u matches with t. If it is the case, then the context τ is inserted at the root of t, yielding the term $\tau[t]$. Otherwise, the HCE-strategy jumps to the position 1 of t and restarts again. If it reaches the left-most leaf of t and u does not match with this leaf, then the HCE-strategy $\mu X.\mathcal{S}(X)$ fails. Roughly speaking, the HCE-strategy $\mu X.\mathcal{S}(X)$ is equivalent to its unfolding:

$$\begin{aligned}\mu X.\mathcal{S}(X) &= (u, \tau) \oplus (@1.((u, \tau) \oplus (@1.(u, \tau) \oplus (@1...))) \\ &= (u, \tau) \oplus @1.(u, \tau) \oplus @11.(u, \tau) \oplus @111.(u, \tau) \oplus \dots\end{aligned}$$

It is helpful to view HCE-strategies as trees with back-edges. A tree with back-edges is an oriented tree with possible edges going from a node to at most one of its ancestors in the tree. For instance, $\mathcal{S}(X) = (u, \tau) \oplus (@1.X)$ is depicted on the left of Figure 4.4, while the tree with back-edges related to $\mu X.\mathcal{S}(X)$ is depicted on the right.

For any HCE-strategies $\mathcal{S}(X)$ and $\mathcal{S}'$ in C, and $i \geq 1$, we define $\mathcal{S}^1(\mathcal{S}') \stackrel{def}{=} \mathcal{S}(\mathcal{S}')$, which stands for the syntactic replacement of X by $\mathcal{S}'$ in $\mathcal{S}$, and $\mathcal{S}^{i+1}(\mathcal{S}') \stackrel{def}{=} \mathcal{S}^i(\mathcal{S}(\mathcal{S}'))$. A HCE-strategy strategy is closed if all its fixed-point variables are bound.

Definition 15: Semantics of HCE-strategies

The semantics of a closed HCE-strategy $\mathcal{S}$ is a function $[\![\mathcal{S}]\!] : \mathcal{T} \cup \mathbb{F} \to \mathcal{T} \cup \mathbb{F}$, which is defined inductively as follows.

$$[\![\varnothing]\!](t) \stackrel{def}{=} \mathbb{F}.$$

$$[\![\mathcal{S}]\!](\mathbb{F}) \stackrel{def}{=} \mathbb{F}.$$

$$[\![(u, \mathcal{S}')]\!](t) \stackrel{def}{=} \begin{cases} [\![\mathcal{S}']\!](t) & \text{if } u \ll t, \\ \mathbb{F} & \text{otherwise.} \end{cases}$$

$$[\![\tau]\!](t) \stackrel{def}{=} \tau(t),$$

$$[\![\mathcal{S}_1 \oplus \mathcal{S}_2]\!](t) \stackrel{def}{=} \begin{cases} [\![\mathcal{S}_1]\!](t) & \text{if } [\![\mathcal{S}_1]\!](t) \neq \mathbb{F}, \\ [\![\mathcal{S}_2]\!](t) & \text{otherwise.} \end{cases}$$

$$[\![\mu X.\mathcal{S}(X)]\!](t) \stackrel{def}{=} [\![\mathcal{S}^{\delta(t)}(\varnothing)]\!](t).$$

$$[\![\textbf{If } (\mathcal{S}_1 \text{ and } \ldots \text{ and } \mathcal{S}_n) \textbf{ then } \mathcal{S}]\!](t) \stackrel{def}{=} \begin{cases} [\![\mathcal{S}]\!](t) & \text{if } \forall i \in [n],\ [\![\mathcal{S}_i]\!](t) \neq \mathbb{F}, \\ \mathbb{F} & \text{otherwise.} \end{cases}$$

$$[\![@p.\mathcal{S}]\!](t) \stackrel{def}{=} \begin{cases} t[[\![\mathcal{S}]\!](t_{|p})]_p & \text{if } [\![\mathcal{S}]\!](t_{|p}) \neq \mathbb{F} \text{ and } p \in \mathcal{P}os(t), \\ \mathbb{F} & \text{otherwise.} \end{cases}$$

$$[\![[@p_1.\mathcal{S}_1, \cdots, @p_n.\mathcal{S}_n]]\!](t) \stackrel{def}{=}$$

$$\begin{cases} (\eta([\![@p_n.\mathcal{S}_n]\!]) \circ \cdots \circ \eta([\![@p_1.\mathcal{S}_1]\!]))(t) & \text{if } \exists i \in [n] \text{ s.t. } [\![@p_i.\mathcal{S}_i]\!](t) \neq \mathbb{F}, \\ \mathbb{F} & \text{otherwise.} \end{cases}$$

$$[\![\texttt{Most}(\mathcal{S})]\!](t) \stackrel{def}{=} \begin{cases} f(\eta([\![\mathcal{S}]\!])(t_1), \cdots, \eta([\![\mathcal{S}]\!])(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and} \\ & \exists i \in [n] \text{ s.t. } [\![\mathcal{S}]\!](t_i) \neq \mathbb{F}, \\ \mathbb{F} & \text{otherwise.} \end{cases}$$

The general definition of the fixed-point constructor requires a heavy machinery involving Knaster-Tarski fixed-point theorem [Tarski, 1955]. However, due to the particular nature of HCE-strategies, we gave an adhoc definition of the fixed-point HCE-strategy by $[\![\mu X.\mathcal{S}(X)]\!](t) \stackrel{def}{=} [\![\mathcal{S}^{\delta(t)}(\varnothing)]\!](t)$. The justification of the iteration of $\mathcal{S}(\varnothing)$ at most $\delta(t)$ times, the depth of t, is that the navigation part of a HCE-strategy does not change the input term t. Therefore, either the HCE-strategy $\mathcal{S}$ progresses on the term t and will reach the leaves of t after $\delta(t)$ iterations, or $\mathcal{S}$ does not progress and in this case it fails after any iteration. Examples of HCE-strategies that do not progress are $\mathcal{S} = \mu X.X$ and $\mathcal{S} = \mu X.(u, X)$ for a term u. In technical terms, one can show that $\mathcal{S}^{\delta(t)}(\varnothing)$ is a fixed-point of $\mathcal{S}(X)$ in the sense that, for every term t, we have $[\![\mathcal{S}(\mathcal{S}^{\delta(t)}(\varnothing))]\!](t) = [\![\mathcal{S}^{\delta(t)}(\varnothing)]\!](t)$.

For any HCE-strategies $\mathcal{S}, \mathcal{S}'$ in $\mathcal{C}$, we shall write $\mathcal{S} \equiv \mathcal{S}'$ iff $[\![\mathcal{S}]\!] = [\![\mathcal{S}']\!]$.

Example 9:

We show how to encode some standard traversal strategies in our formalism using the fixed-point constructor. In what follows we assume that $\mathcal{S}$ is a HCE-strategy. We recall that, when applied to a term t, the HCE-strategy $\texttt{OuterMost}(\mathcal{S})$ tries to apply $\mathcal{S}$ to the maximum of the sub-terms of t starting from the root of t, it stops when it is successfully applied. Hence,

$$\texttt{OuterMost}(\mathcal{S}) := \mu X.(\mathcal{S} \oplus \texttt{Most}(\mathcal{S}))$$

Example 10:

We illustrate the semantics of the fixed-point constructor. Consider the HCE-strategy

$$\mathcal{S}(X) = (a, \tau) \oplus @1.X,$$

where a is a constant and $\tau = \texttt{List}(\square, j)$. Let $t = (a + b * c)^2$ be a term where b and c are constants. If the expression $(a + b * c)^2$ is encoded by the term $\texttt{Power}(a + b * c, 2)$, then we have that the depth of t is $\delta(t) = 3$. Hence, the semantics of $\mu X.\mathcal{S}(X)$ when applied to t is

$$[\![\mu X.\mathcal{S}(X)]\!](t) \quad = \quad \mathcal{S}^3(\varnothing)(t),$$

where $[\![\mathcal{S}^3(\varnothing)]\!]$ is defined by means of $[\![\mathcal{S}^2(\varnothing)]\!]$ and $[\![\mathcal{S}^1(\varnothing)]\!]$ as follows:

$$\begin{array}{lll}
[\![\mathcal{S}^1(\varnothing)]\!] = [\![\mathcal{S}(\varnothing)]\!] & = (a,\tau) \oplus @1.\varnothing & = (a,\tau).\\
[\![\mathcal{S}^2(\varnothing)]\!] = [\![\mathcal{S}(\mathcal{S}(\varnothing))]\!] & = [\![\mathcal{S}((a,\tau))]\!] & = (a,\tau) \oplus @1.(a,\tau).\\
[\![\mathcal{S}^3(\varnothing)]\!] = [\![\mathcal{S}(\mathcal{S}^2(\varnothing))]\!] & = [\![\mathcal{S}((a,\tau) \oplus @1.(a,\tau))]\!] & = (a,\tau) \oplus @1.((a,\tau) \oplus @1.(a,\tau))\\
 & & = (a,\tau) \oplus @1.(a,\tau) \oplus @11.(a,\tau).
\end{array}$$

Thus we get,

$$\begin{array}{lll}
[\![\mathcal{S}^3(\varnothing)]\!](t) = (@1.(a,\tau) \oplus @11.(a,\tau))(t) & & \text{(since } a \text{ does not match with “}\texttt{Power}\text{”)}\\
= t[((a,\tau) \oplus @1.(a,\tau))(t_{|1})]_1 & & \\
= t[(@1.(a,\tau))(t_{|1})]_1 & & \text{(since } a \text{ does not match with “+”)}\\
= t[t_1[(a,\tau)(t_{|11})]_1]_1 & & \\
= t[t_1[(a \to u; \tau[u])(t_{|11})]_1]_1 & & \\
= t[t_1[t_{|11}[\tau[u]]_\epsilon]_1]_1 & & \text{(since } a \text{ matches with } a\text{)}\\
= t[\tau[a]]_{11} & & \\
= \texttt{Power}(\texttt{List}(a, j) + b * c, 2). & &
\end{array}$$

We generalize next the condition of well-foundedness from position-based HCE-strategies to HCE-strategies.

Definition 16: Well-founded HCE-strategies.

A HCE-strategy $\mathcal{S}$ is well-founded iff every position-based HCE-strategy that is a sub-strategy of $\mathcal{S}$ is well-founded in the sense of Definition 10.

4.4.2/ FROM HCE-STRATEGIES TO POSITION-BASED HCE-STRATEGIES

Out of a HCE-strategy and a term it is possible to construct a position-based HCE-strategy. The main purpose of this mapping is to formulate a correctness-completeness criterion for the unification and combination of HCE-strategies in terms of position-based HCE-strategies. Roughly speaking, this criterion imposes that the mapping of the combination of two HCE-strategies is equivalent to the combination of their respective mappings. The definition of this mapping follows.

Definition 17:

Define the function $\Psi : \mathcal{C} \times \mathcal{T} \longrightarrow \mathcal{E}$, that associates to each closed HCE-strategy $\mathcal{S}$ in $\mathcal{C}$ and a term t in $\mathcal{T}$ a position-based HCE-strategy $\Psi(\mathcal{S}, t)$ in $\mathcal{E}$ by

$$\Psi(\varnothing, t) = \varnothing. \qquad \Psi(\tau, t) = @\epsilon.\tau.$$

$$\Psi((u,\tau), t) = \begin{cases} (\epsilon, \tau) & \text{if } u \ll t, \\ \varnothing & \text{otherwise.} \end{cases} \qquad \Psi((u, \mathcal{S}), t) = \begin{cases} \Psi(\mathcal{S}, t) & \text{if } u \ll t, \\ \varnothing & \text{otherwise.} \end{cases}$$

$$\Psi(@p.\mathcal{S}, t) = @p \cdot \Psi(\mathcal{S}, t_{|p}).$$

$$\Psi(\mathcal{S} \oplus \mathcal{S}', t) = \begin{cases} \Psi(\mathcal{S}, t) & \text{if } \Psi(\mathcal{S}, t) \neq \varnothing, \\ \Psi(\mathcal{S}', t) & \text{otherwise.} \end{cases} \qquad \Psi(\mu X.\mathcal{S}(X), t) = \Psi(\mathcal{S}^{\delta(t)}(\varnothing), t).$$

$$\Psi(\textbf{If } (\mathcal{S}_1 \text{ and } \ldots \text{ and } \mathcal{S}_n) \textbf{ then } \mathcal{S}, t) \stackrel{def}{=} \begin{cases} \Psi(\mathcal{S}, t) & \text{if } \forall i \in [n],\ \Psi(\mathcal{S}_i, t) \neq \varnothing, \\ \varnothing & \text{otherwise.} \end{cases}$$

$$\Psi([@p_1.\mathcal{S}_1, \cdots, @p_n.\mathcal{S}_n], t) \stackrel{def}{=} [\Psi(@p_1.\mathcal{S}_1, t), \cdots, \Psi(@p_n.\mathcal{S}_n, t)].$$

$$\Psi(\texttt{Most}(\mathcal{S}), t) = \begin{cases} \bigsqcup_{i=1,n} @i.(\Psi(\mathcal{S}, t_i)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } \exists i \in [n] \text{ s.t. } \Psi(\mathcal{S}, t_i) \neq \varnothing, \\ \varnothing & \text{otherwise.} \end{cases}$$

The application of the position-based HCE-strategy $\Psi(\mathcal{S}, t)$ to the term t will be simply written as $\Psi(\mathcal{S}, t)(t)$ instead of $[\![\Psi(\mathcal{S}, t)]\!](t)$.

It turns out that the function Ψ (Definition 17) preserves the semantics of HCE-strategies in the following sense.

Lemma 1:

For any HCE-strategy S in $\mathcal{C}$ and any term t in $\mathcal{T}$, we have $[\![\mathcal{S}]\!](t) = \Psi(\mathcal{S}, t)(t)$.

The proof of this Lemma does not provide any difficulties since the definition of Ψ is close to the definition of the semantics of HCE-strategies.

Lemma 2:

The function Ψ enjoys the following properties.

i.) For any position-based HCE-strategies E, E' in $\mathcal{E}$, we have that $E = E'$ iff $\Psi(E,t) = \Psi(E',t)$ for any term t.

ii.) For any HCE-strategies $\mathcal{S}, \mathcal{S}'$ in C, we have that $\mathcal{S} \equiv \mathcal{S}'$ iff $\Psi(\mathcal{S},t) = \Psi(\mathcal{S}',t)$ for any term t.

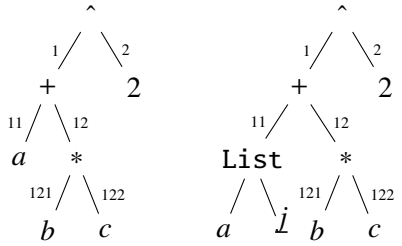


Figure 4.5: The complete tree structure of the term t and $\mu X.\mathcal{S}(X)(t)$ discussed in Example 10.

4.5/ UNIFICATION AND COMBINATION OF HCE-STRATEGIES

We define the combination of HCE-strategies (Definition 19) by means of their unification (Definition 18) together with an example. The first main result of this section is Theorem 1 that guarantees the correctness of the combination of HCE-strategies. The correctness is given in terms of the position-based HCE-strategies, it imposes that the mapping (via the homomorphism Ψ of Definition 17) of the combination of two HCE-strategies is equivalent to the combination of their respective mapping. Besides, Theorem 2 is a consequence of Theorem 1 which is more difficult and proves the same result but for the unification of HCE-strategies instead of the combination. The second main result is the nice algebraic properties of the unification and combination of HCE-strategies stated in Proposition 5. In particular, the combination and unification are associative, which is an important property in the applications, and are a congruence.

We omit the symmetric cases in the following definition which is given by an induction on the HCE-strategies by exhibiting all the possible cases. Besides, to reduce the number of formulas in Definition 18, the elementary HCE-strategy τ is simply written $@\epsilon.\tau$, where τ is a context.

Definition 18: Unification of HCE-strategies

The unification of HCE-strategies is a binary operation $\curlywedge : C \times C \longrightarrow C$ inductively defined as follows.

$$\begin{aligned}
\varnothing \curlywedge S &= \varnothing. & S \curlywedge \varnothing &= \varnothing.\\
@i.\tau \curlywedge @i.\tau' &= @i.(\tau \cdot \tau'). & @i.\tau \curlywedge @j.\tau' &= [@i.\tau, @j.\tau'], \text{ if } j \sqsubset i.\\
@i.\tau \curlywedge @i.S &= @i.(@\epsilon.\tau \curlywedge S). & @i.\tau \curlywedge @j.S &= \mathbf{If}\ @j.S\ \mathbf{then}\ [@i.\tau, @j.S], \text{ if } j \sqsubset i.
\end{aligned}$$

$$\begin{aligned}
(u,\tau) \curlywedge @i.\tau' &= (u, @\epsilon.\tau \curlywedge @i.\tau'), & (u,\tau) \curlywedge @i.\tau' &= \varnothing,\\
&\text{if } i \in [ar(u)] \cup \{\epsilon\}. & &\text{if } i \notin [ar(u)] \cup \{\epsilon\}.\\
@i.\tau \curlywedge (u,S) &= (u, (@i.\tau) \curlywedge S), & @i.\tau \curlywedge (u,S) &= \varnothing,\\
&\text{if } i \in [ar(u)] \cup \{\epsilon\}. & &\text{if } i \notin [ar(u)] \cup \{\epsilon\}.\\
(u,\tau) \curlywedge (u',S') &= (u \wedge u', (@\epsilon.\tau \curlywedge S')). & (u,S) \curlywedge (u',S') &= (u \wedge u', S \curlywedge S').
\end{aligned}$$

For the rest, assume $\mathcal{L} = \bigsqcup_{i\in\mathcal{I}} @i.S_i$ and $\mathcal{L}' = \bigsqcup_{j\in\mathcal{J}} @j.S'_j$.
Let $\mathcal{L}_1 = \bigsqcup_{i\in\mathcal{I}\cap\mathcal{J}} @i.(S_i \curlywedge S'_i)$ and $\mathcal{L}_2 = \bigsqcup_{i\in\mathcal{I}\setminus\mathcal{J}} @i.S_i$ and $\mathcal{L}_3 = \bigsqcup_{i\in\mathcal{J}\setminus\mathcal{I}} @i.S'_i$. Define

$$\begin{aligned}
\mathcal{L} \curlywedge \mathcal{L}' &= \mathcal{L}_1 \sqcup \mathcal{L}_2 \sqcup \mathcal{L}_3.\\
(u,S) \curlywedge \mathcal{L} &= (u,S) \curlywedge \mathcal{L}.\\
(S_1 \oplus S_2) \curlywedge S &= (S_1 \curlywedge S) \oplus (S_2 \curlywedge S).
\end{aligned}$$

$$\begin{aligned}
(\mathbf{If}\ C_1\ \mathbf{then}\ \mathcal{S}_1) \curlywedge (\mathbf{If}\ C_2\ \mathbf{then}\ \mathcal{S}_2) &= \mathbf{If}\ (C_1 \text{ and } C_2)\ \mathbf{then}\ (\mathcal{S}_1 \curlywedge \mathcal{S}_2)\\
(\mathbf{If}\ C_1\ \mathbf{then}\ \mathcal{S}_1) \curlywedge \mathcal{S}_2 &= \mathbf{If}\ C_1\ \mathbf{then}\ (S_1 \curlywedge \mathcal{S}_2)
\end{aligned}$$

For the fixed-point HCE-strategies,

$$\begin{aligned}
\mu X.\mathcal{S}(X) \curlywedge \mu X'.\mathcal{S}'(X') &= \mu Z.\mathcal{S}''(\mu X.\mathcal{S}(X), \mu X'.\mathcal{S}'(X'), Z),\\
\text{where } \mathcal{S}''(X,X',Z) &= [\mathcal{S}(X) \curlywedge \mathcal{S}'(X')]_{|X \curlywedge X' := Z}, \text{ and } Z \text{ is fresh.}\\
(\mu X.\mathcal{S}(X)) \curlywedge \mathcal{S}' &= \mathcal{S}''(\mu X.\mathcal{S}(X)), \text{ where } \mathcal{S}''(X) = \mathcal{S}(X) \curlywedge \mathcal{S}'.\\
X \curlywedge \mathcal{S}' &= \varnothing.
\end{aligned}$$

Finally,

$$\begin{aligned}
\underbrace{\mathtt{Most}(\mathcal{S}_1)}_{\mathcal{R}_1} \curlywedge \underbrace{\mathtt{Most}(\mathcal{S}_2)}_{\mathcal{R}_2} &= \mathbf{If}\ (\mathcal{R}_1 \text{ and } \mathcal{R}_2)\ \mathbf{then}\ \mathtt{Most}((\mathcal{S}_1 \curlywedge \mathcal{S}_2) \oplus \mathcal{S}_1 \oplus \mathcal{S}_2)\\
\mathtt{Most}(S) \curlywedge (u; \bigsqcup_{i\in I\subset\mathbb{N}} @i.S_i) &= u; (\bigsqcup_{i\in[1,arity(u)]} @i.S \curlywedge \bigsqcup_{i\in I\subset\mathbb{N}} @i.S_i)\\
\mathtt{Most}(S) \curlywedge (u; S') &= u; (\mathtt{Most}(S) \curlywedge S')\\
\mathtt{Most}(S) \curlywedge @\epsilon.\tau &= [\mathtt{Most}(S), @\epsilon.\tau]
\end{aligned}$$

Definition 19: Combination of HCE-strategies

The combination of HCE-strategies is a binary operation
$\curlyvee : C \times C \longrightarrow C$, defined for any $\mathcal{S}$ and $\mathcal{S}'$ in C by $\mathcal{S} \curlyvee \mathcal{S}' \stackrel{def}{=} (\mathcal{S} \curlywedge \mathcal{S}') \oplus \mathcal{S} \oplus \mathcal{S}'$.

Example 11:

Let $\mathcal{S}(X) = (u, \tau) \oplus @1.X$ and $\mathcal{S}'(X') = (u', \tau') \oplus @1.X'$, be two HCE-strategies. We compute the $\mu X.\mathcal{S}(X) \curlywedge \mu X'.\mathcal{S}'(X')$ by applying the formulas given in Definition 18. Firstly, the unification $(*)$ of $\mathcal{S}(X)$ and $\mathcal{S}'(X')$ is:

$$
\begin{aligned}
(*) &= \mathcal{S}(X) \curlywedge \mathcal{S}'(X') \\
&= ((u, \tau) \oplus @1.X) \curlywedge ((u', \tau') \oplus @1.X') \\
&= ((u, \tau) \curlywedge ((u', \tau') \oplus @1.X')) \oplus (@1.X \curlywedge ((u', \tau') \oplus @1.X')) \\
&= ((u, \tau) \curlywedge (u', \tau')) \oplus (@1.X \curlywedge (u', \tau')) \oplus ((u, \tau) \curlywedge @1.X') \oplus (@1.X \curlywedge @1.X') \\
&= (u \wedge u', \tau' \cdot \tau) \oplus (u, @\epsilon.\tau \curlywedge @1.X') \oplus (u', @1.X \curlywedge @\epsilon.\tau') \oplus (@1.(X \curlywedge X')) \\
&= (u \wedge u', \tau' \cdot \tau) \oplus (u, \mathbf{If}\,(@1.X')\,\mathbf{then}\,[@1.X', @\epsilon.\tau]) \\
&\qquad \oplus (u', \mathbf{If}\,(@1.X)\,\mathbf{then}\,[@1.X, @\epsilon.\tau']) \\
&\qquad \oplus (@1.(X \curlywedge X')).
\end{aligned}
$$

Secondly, the replacement of $X \curlywedge X'$ in $(*)$ by a fresh fixed-point variable Z yields:

$$
\begin{aligned}
\mathcal{S}''(X, X', Z) &= [\mathcal{S}(X) \curlywedge \mathcal{S}'(X')]_{|X \curlywedge X' := Z} \\
&= \Big[(u \wedge u', \tau' \cdot \tau) \oplus (u, \mathbf{If}\,(@1.X')\,\mathbf{then}\,[@1.X', @\epsilon.\tau]) \\
&\qquad \oplus (u', \mathbf{If}\,(@1.X)\,\mathbf{then}\,[@1.X, @\epsilon.\tau']) \\
&\qquad \oplus (@1.(X \curlywedge X'))\Big]_{|X \curlywedge X' := Z} \\
&= (u \wedge u', \tau' \cdot \tau) \oplus (u, \mathbf{If}\,(@1.X')\,\mathbf{then}\,[@1.X', @\epsilon.\tau]) \\
&\qquad \oplus (u', \mathbf{If}\,(@1.X)\,\mathbf{then}\,[@1.X, @\epsilon.\tau']) \\
&\qquad \oplus (@1.Z).
\end{aligned}
$$

Finally, the unification $(**)$ of $\mu X.\mathcal{S}(X)$ and $\mu X'.\mathcal{S}'(X')$ is:

$$
\begin{aligned}
(**) &= \mu X.\mathcal{S}(X) \curlywedge \mu X'.\mathcal{S}'(X') \\
&= \mu Z.\mathcal{S}''(\mu X.\mathcal{S}(X), \mu X'.\mathcal{S}'(X'), Z) \\
&= (u \wedge u', \tau' \cdot \tau) \oplus (u, \mathbf{If}\,(@1.X')\,\mathbf{then}\,[@1.X', @\epsilon.\tau]) \\
&\qquad \oplus (u', \mathbf{If}\,(@1.X)\,\mathbf{then}\,[@1.X, @\epsilon.\tau']) \\
&\qquad \oplus (@1.Z).
\end{aligned}
$$

The application of $(**)$ to a term t features four cases.

i.) Either t matches with both u and u', and in this case the context $\tau' \cdot \tau$ is inserted at the root of t.

ii.) Or only u matches with t, and in this case τ is inserted at the position 1 of t provided the HCE-strategy $\mu X'.\mathcal{S}'(X')$ is applied successfully at the position 1 of t.

iii.) Or only u' matches with t, and in this case τ' is inserted at the position 1 of t provided the HCE-strategy $\mu X.\mathcal{S}(X)$ is applied successfully at the position 1 of t.

iv.) Or both $\mu X.\mathcal{S}(X)$ and $\mu X'.\mathcal{S}'(X')$ are applied at the position 1 of t.

4.5.1/ THE CORRECTION AND COMPLETENESS OF THE UNIFICATION AND COMBINATION OF HCE-STRATEGIES

Now we are ready to state the main results of this Chapter. Namely, the unification and combination of HCE-strategies is sound and complete.

Theorem 1:

For every term $t \in \mathcal{T}$ and for every HCE-strategies $\mathcal{S}$ and $\mathcal{S}'$ in $\mathcal{C}$, we have that $\Psi(\mathcal{S} \curlywedge \mathcal{S}', t) = \Psi(\mathcal{S}, t) \curlywedge \Psi(\mathcal{S}', t)$.

Theorem 2:

For every term $t \in \mathcal{T}$ and for every HCE-strategies $\mathcal{S}$ and $\mathcal{S}'$ in $\mathcal{C}$, we have that $\Psi(\mathcal{S} \curlyvee \mathcal{S}', t) = \Psi(\mathcal{S}, t) \curlyvee \Psi(\mathcal{S}', t)$.

Thanks to the fact that the function Ψ is an homomorphism (in the first argument), one can transfer all the properties of the combination and unification of position-based HCE-strategies (stated in Proposition 4) to HCE-strategies.

Proposition 5:

The following hold.

1. The set $\mathcal{C}$ of HCE-strategies together with the unification and combination operations enjoy the following properties.
   1. The neutral element of the unification is $@_\epsilon.\Box$, and the absorbing element is $\varnothing$.
   2. The neutral element of the combination is $\varnothing$.
   3. The unification and combination of HCE-strategies are associative.
2. The unification and combination of HCE-strategies is non commutative.
3. For any HCE-strategies $\mathcal{S}$ and $\mathcal{S}'$ in $\mathcal{C}$, and for any term t in $\mathcal{T}$, we have that

$$\begin{array}{lll} \Psi(\mathcal{S} \curlywedge \mathcal{S}', t) = \varnothing & \text{iff} & \Psi(\mathcal{S}, t) = \varnothing \text{ or } \Psi(\mathcal{S}', t) = \varnothing. \\ \Psi(\mathcal{S} \curlyvee \mathcal{S}', t) = \varnothing & \text{iff} & \Psi(\mathcal{S}, t) = \varnothing \text{ and } \Psi(\mathcal{S}', t) = \varnothing. \end{array}$$

4. For any HCE-strategies $\mathcal{S}$ and $\mathcal{S}'$ in $\mathcal{C}$, we have that

$$\begin{array}{lll} \mathcal{S} \curlywedge \mathcal{S}' \equiv \varnothing & \text{iff} & \mathcal{S} \equiv \varnothing \text{ or } \mathcal{S}' \equiv \varnothing. \\ \mathcal{S} \curlyvee \mathcal{S}' \equiv \varnothing & \text{iff} & \mathcal{S} \equiv \varnothing \text{ and } \mathcal{S}' \equiv \varnothing. \end{array}$$

5. The unification and combination of HCE-strategies is a congruence, that is, for any HCE-strategies $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}$ in $\mathcal{C}$, we have that:

$$\begin{array}{lll} \text{If } \mathcal{S}_1 \equiv \mathcal{S}_2 & \text{then} & \mathcal{S}_1 \curlywedge \mathcal{S} \equiv \mathcal{S}_2 \curlywedge \mathcal{S} \text{ and } \mathcal{S} \curlywedge \mathcal{S}_1 \equiv \mathcal{S} \curlywedge \mathcal{S}_2. \\ \text{If } \mathcal{S}_1 \equiv \mathcal{S}_2 & \text{then} & \mathcal{S}_1 \curlyvee \mathcal{S} \equiv \mathcal{S}_2 \curlyvee \mathcal{S} \text{ and } \mathcal{S} \curlyvee \mathcal{S}_1 \equiv \mathcal{S} \curlyvee \mathcal{S}_2. \end{array}$$

We notice that the neutral and absorbing element, and the associativity property of

the unification and combination must be understood at the semantic level and not at the syntactic level since there are HCE-strategies which are syntactically different but semantically equivalent. For instance, the HCE-strategies $@\epsilon.\Box$ and $(x, @\epsilon.\Box)$ and $(x, @\epsilon.\Box) \oplus (y, @\epsilon.\Box)$, where x, y are variables, are all equivalent. Therefore, saying that $@\epsilon.\Box$ is the neutral element for the unification of HCE-strategies must be understood as follows. For any HCE-strategies $e, \mathcal{S} \in C$ such that $e \equiv @\epsilon.\Box$, we have that $e \curlyvee \mathcal{S} \equiv \mathcal{S} \curlyvee e \equiv \mathcal{S}$. And the associativity of the unification must be understood as follows. For any HCE-strategies $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \in C$, we have that $(\mathcal{S}_1 \curlyvee \mathcal{S}_2) \curlyvee \mathcal{S}_3 \equiv \mathcal{S}_1 \curlyvee (\mathcal{S}_2 \curlyvee \mathcal{S}_3)$.

4.6/ REMARKS

The class of HCES-strategies introduced in this Chapter is indeed a strict subclass of the class of context embedding strategy, CES-strategies for short, introduced in [Belkhir et al., 2016]. The strategy constructors of the class of CES-strategies feature the insertion of contexts, the jump operator "@", the left-choice "$\oplus$", the fixed-point operator "μ" and a mechanism to specify and handle the failure. While the constructors of the class of HCES-strategies feature the insertion of contexts, the jump operator "@", the left-choice "$\oplus$", the fixed-point operator "μ" and the `Inside`strategy. This makes the class of HCES-strategies less expressive than the class of CES-strategies but, on the other hand, the encoding of the (HCES-strategy) `Inside` in the class of CES-strategies yields a strategy whose size depends on the signature. This makes the class of HCES-strategies more practical although its constructors are less rudimentary than the constructors of the class of CES-strategies.

5

IMPLEMENTATION IN MEMSALAB

5.1/ INTRODUCTION

We implemented the HCE-strategy language as a part of a user-friendly language which is integrated in `MEMSALab`. It makes the implementation of derivation of asymptotic models easier by considering the whole process as a sequence of mathematical properties which transforms an input reference PDE into an asymptotic model.

For sake of organization, there are three kinds of user files distinguished by their extension `".pde"`, `".proof"` and `".ext"`. The `".pde"` file is used for the specification of PDEs. It is composed of many sections such as `Constant`, `Index`, `Region`, `Variable`, `Function`, `Operator`, `Expression` and `PDE`. These sections define respectively constants, indices, domains, mathematical variables, functions, operators operating on functions, mathematical expressions and the PDE. It is possible to introduce and use ”shortcut” names. Beside, commonly used operations and operators, such as ∂ and $\int$, are predefined, the user does not need to define them. Both `Unicode` and LaTeX symbols can be used.

The `".proof"` file is a collection of mathematical properties written rewriting strategies. A strategy is applied to the PDE defined in the PDE file and transforms it to another asymptotic model. The `".proof"` file has the same structure as the PDE file, except it does not contain the `PDE` section but new sections such as `Rule` and `Step`. The `Rule` section allows one to define rewriting rules. The `Step` section contains a list of strategies.

The `".ext"` file allows one to define HCE-strategies with the possibility to introduce and use shortcut names.

The `".pde"`, `".proof"` and `".ext"` files are parsed and an `OCaml` code is generated. However, the user can manage and visualize PDEs, proofs, extensions, the application of a proof to a PDE, the application of an extension to a proof or a PDE, and the combination of two extensions via `Matlab`. The visualization is possible in many formats: LaTeX PDF, html or Unicode text. Finally, to illustrate how to use the User Language for writing PDEs, proofs and extensions. Figures are reproduced in the last section to illustrate the display tools, the application of a proof to a PDE, the application of extensions to a PDE and the combination of two extensions and its result when applying to the PDE.

Organization of the Chapter: The User Language is introduced in Section 5.2. The structure of PDE, Proof and Extension files as well as the definitions of first and rewriting variables, patterns, rules, strategies are detailed in Sections 5.3, 5.4 and 5.5. In Section 5.7 we introduce Matlab functions to manage PDE, Proof and Extension files. The outputs are illustrated by figures in Section 5.9.

5.2/ GENERALITIES

This section describes the User Language that allows for the implementation of derivation of asymptotic models for partial differential equations (PDE) based on the Extension-Combination method, a method grounded on a concept of reusability. Roughly speaking, its principle consists in starting from a reference PDE and in its transformation into an asymptotic model through the application of a sequence of mathematical properties forming a proof. For instance, the reference PDE in (5.1) is the one-dimensional second order differential equation

$$-\frac{d}{dx}(a^{\varepsilon}(x)\frac{du}{dx}) = f \text{ in }]a,b[\tag{5.1}$$

with a periodic coefficient $a^{\varepsilon}(x) = a(x/\varepsilon)$ and the homogeneous Dirichlet boundary conditions

$$u(a) = u(b) = 0. \tag{5.2}$$

The reference proof transforms the reference boundary value problem (5.1, 5.2) into its homogenization by passing to the limit $\varepsilon \to 0$.

Then, the method consists in transforming the reference proof into another proof that applies to a more complex input PDE. The latter should come from a script or a software package e.g. FreeFEM++ or COMSOL. Such a transformation is refered to as an Extension. In principle an extension is designed to be applied to the reference proof, but it should also be applicable to the reference PDE to generate a more complex PDE. By construction, an extension might be 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.

After building several extensions, it is possible to combine them by a Combination to build a new extension that is inheriting all the features of the extensions.

5.2.1/ THE FOUR KINDS OF FILES

Four kinds of user files are distinguished with their extensions. A ".pde" file is building an equation taken as an input of a model derivation. The latter is described in a ".proof" file. The equations and the proof are extended with ".ext" files.

5.2.2/ INCLUSION OF FILES

Included files can be at any position of a file with the command `#Include "file"` where `file` is a filename of the same kind as the main file with.

5.2.3/ LIST OF UNICODE CHARACTERS

The four kinds of files can make use of unicode characters that are gathered in Table 5.2.3.

In Windows, the edition is done with Notepad++. A special choice of font must be done for correct edition of unicode characters. The illustrations presented in the following are done with the font Arial Unicode MS.

∂	đ	$\int$	$\sum$	$\prod$	$\bullet$	$\rightarrow$	$\rightrightarrows$	$\downarrow$	$\mathrm{I}\!\downarrow$	$\uparrow$	$\mathrm{I}\!\uparrow$	$\circlearrowleft$
2202	0111	222B	2211	220F	2219	2192	21C9	2193	21A7	2191	21A5	21BA
$\in$	$\notin$	$\subset$	$\not\subset$	$\subseteq$	$\not\subseteq$	$\cap$	$\cup$	$\setminus$	$\neq$	$\leq$	$\geq$	$\emptyset$
2208	2209	2282	2284	2286	2288	2229	222A	2216	2260	2264	2265	2205
$\square$												
25A1												

Figure 5.1: List of Unicode Characters

5.3/ PDE-FILES

5.3.1/ STRUCTURE OF A PDE FILE

A pde-file begins with the name of the file stated as

`PDE "file_name"`

and ends with the Partial Differential Equation declared with the keyword `PDE` that must appear exactly one time in a pde-file. Comments begin with the character `%` and apply until the end of the line. They can be put anywhere. The file is organized in Sections, each refering to a type of data. Each section begins with a keyword and includes a list of shortcut statements, see Section 5.3.2. Any of these statements comprises a shortcut name and a set of fields defined in Table 1. The shortcuts and their fields can be used in subsequent instructions. All kinds of sections are optional, they can be introduced in any order and can be repeated. The last line of the file is the declaration of the PDE that begins with the keyword `PDE`.

5.3.2/ SHORTCUTS AND THEIR FIELDS

Excepted `PDE` and `Expression` defined in Section 5.3.4, the six other kinds of sections defined in Pde-files are `Constant`, `Index`, `Region`, `Variable`, `Function` and `Operator`. They define constants, indices, domains where partial differential equations are posed, mathematical variables (to distinguish with rewriting variables used in rules for proofs) each being defined in a domain, functions depending on possibly several variables, and operators operating on functions. The fields of each of these kind of shortcuts are summarized in the following table.

Keywords	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
`Constant`	`"Value"`	–	–	–	–	–
`Index`	`"Name"`	`[Range]`	`"Quantifier"`	–	–	–
`Region`	`"Name"`	`[Index]`	`[Axes]`	`[Subregion]`	`Boundary`	`Normal`
`Variable`	`"Name"`	`[Index]`	`[Region]`	–	–	–
`Function`	`"Name"`	`[Index]`	`[Variable]`	`[BC]`	`"Type"`	–
`Operator`	`"Name"`	`[Index]`	`[Expr]`	`[Inputvar]`	`[Outputvar]`	`[Parameter]`

Table 1: the keywords of the sections and the description of their fields

Below is the definition of two shortcuts of contants that can be a number or a string.

```
Constant
pi :  3
e :  "e"
```

or equivalently

```
Constant pi :  "3"  e :  "e"
```

Let us explain the meaning of each field, the type of the shortcuts being detailed in Table 3 below. All kind of shortcuts, excepted `Constant` and `Index` can be indexed thanks to their field `Index`.

In the type `Index`, the field `Quantifier` describe in which sense the index is used, see Table 3.

In the type `Region`, the field `[Axes]` is the list of directions in which a region is defined. It determines the dimension of the region. For example, for a three-dimensional region it can be `[1,2,3]` and for a two-dimensional region it can be `[1,3]`. If subregions are used, they can be declared in the list `[Subregion]`. The boundary of a region may be described in `Boundary` and its outward unit normal direction is a function in `Normal`.

In the type `Variable`, the field `Region` refers to the domain where it is defined. If the variable is a vector, each of its components is defined in a region specified by a component of the vector of regions.

In the type `Function`, the field `Variable` refers to the variable on which the function operates. The possible boundary conditions that it satisfies are in `BC`. Each `BC` is a rewriting rule defined in Section 5.4.2. The "`Type`" of a function must be given. It is a "`Given`" function, or a "`Test`" function or an "`Unknown`" function (solution to an equation) or a $O(\varepsilon)$-function that is expected to be replaced by zero when ε vanishes.

For the type `Operator`, the field `Expr` is the expression to which the operator is applied, meaning that the same shortcut cannot be used for different applications of the same operator. For instance Δu and Δv requires to define

```
Function u :  "u" [] [] [] "Unknown"
          v :  "v" [] [] [] "Test"
Operator opDelta_to_u :  "Delta" [] [u] [] [] []
          opDelta_to_v :  "Delta" [] [v] [] [] []
```

The fields `Inputvar` and `Outputvar` refer to the variables of the input and output functions on which the operator actually operates. For instance, for the partial derivative about a variable `x` of a function `f(x,y)` of two variables, `Inputvar` and `Outputvar` can be taken as `[x]` only. But for a partial integration about `x`, `Inputvar` can be taken as `[x]` but the `Outputvar` is the empty list `[]`. Finally, `Parameter` allows to take into account any parameter.

The name of a shortcut is also called an identifier and is a sequence of characters that may be a letter, a digit, the underscore character "_", or the single quote "'". It starts with a lowercase letter or an underscore. For instance `_n` and `a_i` are admissible identifiers but `A` is not. The set of letters includes the 52 lowercase and uppercase letters from the ASCII set. The current implementation accepts identifiers up to 16.000.000 characters in length. Table 2 summarizes the specifications of the identifiers together with the other

primary objects used in a Pde-file

Objects	Specification
Integer	[0-9]+
Float	[0-9]+.[0-9]*([eE][-+]?[0-9]+)?
Identifier	[a .. z _] [A .. Z a .. z 0 .. 9 _ ']*

Table 2: specification of the integers, float and identifiers used in a Pde-file

In Table 1, we distinguish between three kinds of fields: lists of shortcuts denoted with brackets, strings denoted with double quotes or simple shortcuts. A shortcut appearing in a field must have already been defined. The sets of definitions of all shortcut fields appearing in Table 1 are detailed in Table 3.

Field names	Sets of Definition	Field Names	Sets of Definition
Name	Id.	Expr	Id. of a sc "Expression"
Range	Integer | Id. of a sc "Constant"	BC	Id of Rules
Quantifier	{Given, For_all, There_exists}	Type	{Given, Unknown, Test, Bigo}
Index	Integer | Id. of a sc "Index"	BCRegion	Id. of a sc "Region"
Axes	Integer | Id. of a sc "Constant"	BCLhsExpr	Id. of a sc "Expression"
Subregion	Id. of a sc "Region"	BCRhsExpr	Id. of a sc "Expression"
Boundary	Id. of a sc "Region"	Inputvar	Id. of a sc "Variable"
Normal	Id. of a sc "Function"	Outputvar	Id. of a sc "Variable"
Region	Id. of a sc "Region"	Parameter	Integer | Id. of a sc of any type
Variable	Id. of a sc "Variable"		

Table 3: the sets of definition of the shortcut fields of Table 1. Here Id. stands for Identifier and sc for shortcut

Moreover, in a shortcut a null list is represented by [] and a null simple shortcut by an underscore, but a field in quotes cannot be with an empty component. The access to a shortcut field is done according to the rail diagram:

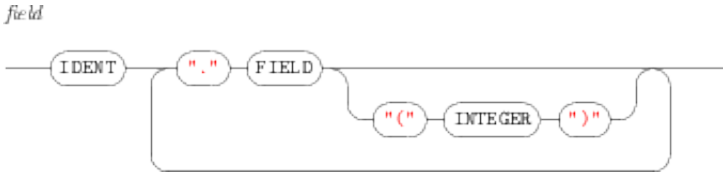


Figure 5.2: Rail diagram for the access to fields of a shortcut

where IDENT is the name of the shortcut, FIELD is the name of the field and INTEGER is the rank of the field in the list if neccessary. For instance, for the shortcuts

```
Index i :  "i" [1,2] "Given"

Index j :  "j" [1,2] "For_all"

Region omega :  "omega" [i,j] [] [] _ _
```

the code `omega.Index(2).Quantifier` is equal to the string `"For_all"`.

5.3.3/ PREDEFINED OPERATORS

A number of operations and operators are predefined. They are listed in the table below.

Operations and operators
`expr + expr`
`expr - expr`
$\sum_{\_}$ `ident expr` or `\sum_ ident expr`
$\prod_{\_}$ `ident expr` or `\prod_ ident expr`
$\int$ `expr d ident` or `\int expr \dj ident`
∂ `expr / ∂ ident` or `\partial expr / \partial ident`
`expr * expr`
`expr ·  expr` or `expr \cdot expr`
`expr / expr`
`+ expr`
`- expr`
`exprˆexpr` or `expr ** expr`
`( expr )`

Table 4: all predefined operations and operators with their associativity property

5.3.4/ EXPRESSIONS

The shortcuts of the kind `Expression` define mathematical expressions. Their grammar in Unicode characters or Latex commands is defined in Table 4. Their precedence rules and their associativity rules follow the usual mathematical rules.

5.3.5/ PDE

The declaration of the model is done under the section `PDE` and has the form

```
PDE

pde_1D : lefthandside = righthandside
```

where `lefthandside` and `righthandside` are two expressions.

5.3.6/ EXAMPLES

The weak formulation of the model problem posed in a domain ω,

$$-\frac{d^2u}{dx^2} + u = f \text{ in } \omega$$

with Neumann boundary conditions is written as: $u \in H^1(\omega)$

$$\int_\omega \frac{du}{dx}\frac{dv}{dx}\,dx = \int_\omega f\,v\,dx \text{ for all } v \in H^1(\omega).$$

A coding with a minimum specification is as follows.

```
PDE "pde_1D" Region
  %       Name     Index  Axes Subregion Boundary Normal
  omega : "omega" []    []    []        _        _
Variable
  %    Name Index Region
  x  : "x"  []    omega
Function
  %    Name Index Variable [BCRegion,BCLhsExpr,BCRhsExpr] Type
  u  : "u"  []    [x]      []                              "Unknown"
  v  : "v"  []    [x]      []                              "Test"
  f  : "f"  []    [x]      []                              "Given"
PDE
```

pde_1D : $\int \partial u/\partial x * \partial v/\partial x + u * v \; dx = \int f * v \; dx$

The PDE can also be defined from a predefined expression,

Expression exp : $\int \partial u/\partial x * \partial v/\partial x + u * v \; dx = \int f * v \; dx$

```
PDE
pde_1D : exp
```

5.4/ PROOF FILES

A proof file implements a proof that applies to a PDE. It includes patterns that have the same form as the shortcuts in a Pde-file excepted that they include rewriting variables. They are defined in Section 5.4.1. Additionally, there are other kinds of shortcuts, namely `Rule` defined in Section 5.4.2, `Strategy` in Section 5.4.3, and `Step` and `Lemma` in Section 5.4.4. It ends with the definition of the model that starts with the keyword `Model`. Figure **??** represents a global view of the tree structure of a proof after all shortcuts have been assembled to form a single expression.

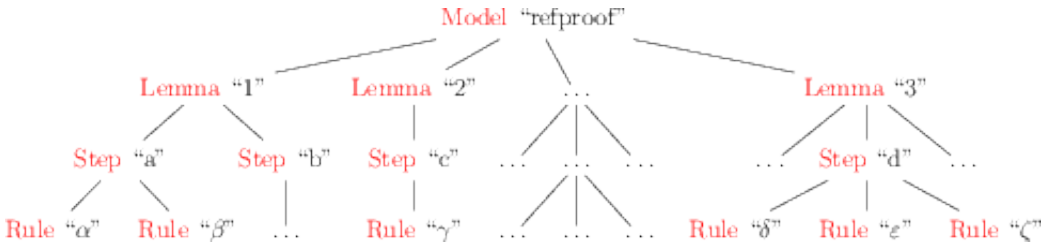


Figure 5.3: Global tree representation of a proof

5.4.1/ REWRITING VARIABLES, PATTERNS AND PATTERN MATCHING

A pattern has the same structure as the shortcuts in PDEs excepted that it can also include rewriting variables. A rewriting variable can represent any expression. It is denoted by a trailing underscore, e.g. `x_` where `x` is any identifier. A pattern is made with any shortcut definition or any expression that may occur in a Pde file but with some parts replaced by rewriting variables. For instance,

`Function u : "u" [] [x_] [] "Unknown"`

is a pattern representing a shortcut of the function u that is an unknown in the problem and which variable can be any variable. We notice that the fields `Type` and `Quantifier` cannot be replaced by a rewriting variable. For instance, the declaration

`Function u : "u" [] [x_] [] y_`

is wrong.

The pattern

`Expression exp1 : z_+t_`

represents any sum as for instance "1+2".

A pattern ℓ is used to be compared with a term t through an operation called pattern matching. It consists in testing if ℓ and t are equal modulo replacement of the variables in ℓ by some subexpressions of t. When the equality holds, the pattern ℓ is said to match with the term t or vice versa. Addition and multiplication being commutative and associative, the implementation of the pattern matching procedure can operate up to these properties. The list of commutative and associative symbols is defined in Section 5.3.3.

The internal representation of a list in a shortcut field e.g. `[1,2,3]` is made with nested two element lists whose left element is an expression and right element a list e.g. `[1,[2,3]]`. The following examples illustrate how a rewriting variable matches in a list.

- `x_` can match any list

A variable in a list matches with a list only if it is in last position. Otherwise it matches only with an expression.

- `[1,x_]` matches `[1,2,3]` and `x_` matches `[2,3]`.
- `[x_,2,3]` matches `[1,2,3]` and `x_` matches `1`.
- `[1,x_,3]` matches `[1,2,3]` and `x_` matches `2`.
- `[x_,y_]` matches `[1,2,3]`, `x_` matches `1` and `y_` matches `[2,3]`.

The field of a rewriting variable can be used in the same way as for any shortcut. If `omega_` is a rewriting variable that is used to match with a `Region`, the code `omega_.Index(2).Quantifier` is equal to the `Quantifier` of the `Index` of the `Region` matched with `omega_`.

5.4.2/ RULES

A rule transforms a pattern ℓ into another one r under a condition c. A condition follows the grammar detailed in Section 5.4.2.1. It can call the Var-function defined in Section 5.4.2.2 that returns the set of mathematical variables involved in an expression. It can also include rewriting variables.

A rule is declared under the section name `Rule` and is expressed by an arrow "→": $\ell \rightarrow r$ `if` c. The rewriting variables included in r must be parts of ℓ. The rule is designed

to be applied to an expression t that matches with ℓ. Applying a rule to an expression t generates an expression r where the rewriting variables have been replaced by the subexpressions determined in the comparison between ℓ and t. The terms ℓ, r and c can include shortcuts previously defined. For instance the rule

`Rule anyrule : x_ → x_`

transforms any expression into the same expression. The rule

`Rule plusone : x_ → x_+1 if x_>0`

adds 1 to an expression provided that it is positive. So it transforms a+b into a+b+1 if one knows that a+b>0.

The next example is a proof file named ”simple_proof.proof” that applies only the rule ”plusone”,

`Model simple_proof : plusone`

The three fields of a rule are `Left`, `Right` and `Condition`. For instance `plusone.Right` is `x_+1`.

5.4.2.1/ GRAMMAR OF RULE'S CONDITIONS

Table 5 below summarizes the usual mathematical operations on sets.

∅ or \varnothing
{ }
{ident,...,ident}
VarOf expr
set ∪ set or expr \cup expr
set \ set
set ∩ set or expr \cap expr

Table 5: notations used in the rule conditions for usual operations on sets

Table 6 summarizes the usual logical operators. The function `VarOf` is defined in Section 5.4.2.2.

Operators	**Operators**
true	expr = expr
false	expr ≠ expr or expr \neq expr
cond or cond	ident ∈ set or ident \in set
cond and cond	ident ∉ set or expr \not\in expr
not cond	set ⊂ set or expr \subset expr
(cond)	set ⊄ set or expr \not\subset expr
expr < expr	set ⊆ set or expr \subseteq expr
expr ≤ expr or expr \leq expr	set ⊈ set or expr \nsubseteq expr
expr > expr	set = set
expr ≥ expr or expr \geq expr	

Table 6: notations used in the rule conditions for usual logic operations

In some cases, the corresponding Latex commands are also recalled since they are planed to be implemented in the language.

5.4.2.2/ THE VAROF--FUNCTION

The `VarOf--`function determines the sets of mathematical variables of an expression. It is used in the condition part of rewriting rules. It is recursively defined according to the table below, where `expr` is an expression, `d` an integer, `f` a float, `s` a string, `x` a mathematical variable and `A` an operator. A field e.g. `f.Variable` is considered as an *evaluable expression* and must be evaluated, i.e. replaced by its value, before computing the `VarOf--`function on it. An expression `VarOf(.)` is also an evaluable expression.

Expression	**Set of mathematical variables**
`VarOf(exp1+exp2), VarOf(exp1-exp2)`	`VarOf(exp1)` ∪ `VarOf(exp2)`
`VarOf(`$\sum_i$`exp), VarOf(`$\prod_i$`exp)`	`VarOf(exp)` \ `VarOf(i)`
`VarOf(`$\int$`exp dx)`	`VarOf(exp)` \ `VarOf(x)`
`VarOf(`∂`exp/`∂`x)`	`VarOf(exp)` if `VarOf(x)`$\subseteq$ `VarOf(exp)`, and ∅ otherwise
`VarOf(exp1*exp2), VarOf(exp1·exp2)`	`VarOf(exp1)` ∪ `VarOf(exp2)`
`VarOf(exp1/exp2)`	`VarOf(exp1)` ∪ `VarOf(exp2)`
`VarOf(+exp), VarOf(-exp)`	`VarOf(exp)`
`VarOf(exp1^exp2), VarOf(exp1**exp2)`	`VarOf(exp1)` ∪ `VarOf(exp2)`
`VarOf("("exp")")`	`VarOf(exp)`
`VarOf(d), VarOf(f), VarOf(s)`	∅
`VarOf(x)`	{x}
`VarOf(A)`	`VarOf(A.Expr)` ∪ `VarOf(A.Outputvar)` \ `VarOf(A.Inputvar)`
`VarOf(evaluable expression)`	`VarOf(evaluation of the expression)`

Table 7: Definition of the `VarOf`-function that returns the set of variables included in an expression

5.4.3/ STRATEGIES

A strategy is a transformation applied to an input expression. It specifies how the rewriting rules are applied. It often consists in a navigation in the input expression followed by an application of rewriting rules. The application of a strategy to an input expression either succeeds and in this case it yields another expression, or it fails and the input expression is returned as the output expression.

- **Composition:** The strategy "`;`" stands for the composition of two strategies. That is, the strategy `s1;s1` consists of the application of `s1` followed by the application of `s2`.
- **Or:** The strategy "|" or `lor` stands for left-choice. The strategy `s1|s2` applies `s1`. If this application fails then `s2` is applied. Hence, `s1|s2` fails when applied to a given term `t` iff both `s1` and `s2` fail when applied to `t`. In other words, this strategy applies

at maximum one of the two strategies `s1` or `s2` with a priority given to `s1`. This is why it is called `lor` for "or with the left-priority".

- **OuterMost:** The outermost strategy `s`$\Downarrow$ applies the strategy `s` once to all the subterms of `t` for `s` that are the closest ones to the root of `t`, i.e. to the largest subterms of `t` on which `s` succeeds. In other words the strategy outermost traverses the expression `t` down from its root and tries to apply `s` to each traversed subexpressions. If the strategy `s` succeeds on some subexpression `t'` of `t`, then it is not applied to the proper subexpressions of `t'` neither to the result of the application. In particular, `s`$\Downarrow$ fails if and only if `s` fails on all the subexpressions of `t`.

- **InnerMost:** The innermost strategy `s`$\Uparrow$ works similarly, but in the opposite direction, i.e. it traverses a term `t` up from its smallest subexpressions and tries to apply the strategy `s` once to the smallest redexes of `t` for `s`.

- **TopDown:** The topdown strategy `s` $\downarrow$ tries to apply the strategy `s` to all the subexpressions of any expression `t`, at any depth, by starting with the root of `t`. It fails when there is a subexpression of `t` where `s` fails.

- **BottomUp:** The bottomup strategy `s` $\uparrow$ behaves similarly, but works in the opposite direction, i.e. it starts from the leaves (the smallest subexpressions) and goes up.

- **Iteration:** The iterate strategy `s` $\circlearrowleft$ iterates the application of the strategy `s` until the latter fails, in this case the full strategy $\circlearrowleft$ fails, or a fixed point is reached. It fails if and only if the strategy `s` fails during these iterated applications.

- **Co-strategy:** The co-strategy `-s` permutes the left and right sides of all rules in `s`. This yields a correct strategy only if the set of variables of the left and right sides are the same. This constraint is not verified.

The next program applies the rule `plusone` everywhere starting from the root,

```
Strategy strat_plusone :  plusone↓
```

or

```
Strategy strat_plusone :  plusone \td
```

and

```
Model simple_proof : strat_plusone
```

or simply, in a single line

```
Model simple_proof : plusone↓
```

The next table summarizes the strategies.

Strategies	Notation	Meaning	Associativity
Compose	`s1; s2` or `s1 and s2`	apply `s1` and then `s2`	left
LeftChoice	`s1 | s2` or `s1 lor s2`	apply `s1` and if does not work apply `s2`	left
TopDown	`s` ↓ or `s \td`	apply `s1` everywhere starting from the root	
OuterMost	`s` ⇓ or `s \om`	same as TopDown but stops at the first success	
BottomUp	`s` ↑ or `s \bu`	apply `s1` everywhere starting from the leaves	
InnerMost	`s` ⇑ or `s \im`	same as BottomUp but stops at the first success	
Iterate	`s` ↻ or `s \it`	apply `s1` as long as it is possible	non associative
Co-strategy	`-s`	permute left and right rule parts in all rules of `s`	

Table 8: the strategies, their notation and their possible associativity property

5.4.4/ STEPS, LEMMA, MODELS

In a Proof-file, the final instruction is in the section `Model` and is made with strategies. Lemma are made with strategies used to build intermediary results. An additionnal strategy is available to transform the result of a lemma into a new rule so to be used by subsequent lemmas. The Lemma can refer to steps that are themselves made with strategies. The use of steps is for adding more structure to the proof of a Lemma.

An example of a section `Step` that includes a single step which applies a single strategy is

```
Step step1 : strat_plusone
```

Then, a possible section `Lemma` with a single step is

```
Lemma lemma1 : step1
```

Avoiding the use of the sections `Strategy` and/or `Step`, a strategy can be

```
Lemma lemma1 : plusone↓
```

The strategy `generateRule` is used to transform the result of the application of a lemma to a special term that yields an equality `l=r if c` to the rule `l → r if c`. If the name of the lemma is `lemma_id`, the generated rule is name `lemma_id_Rule`. For instance, applying `lemma1` to the term

```
t:= x+1=y+2
```

Applying the lemma

```
Lemma lemma1 : step1; generateRule "nameOfGeneratedRule"
```

creates the new rule

```
Rule lemma1_Rule :  x+2 → y+3
```

that can be used in another lemma

```
Lemma lemma2 : lemma1_Rule
```

and a proof can be made with the successive call of the two lemmas

```
Model proof1 : lemma1; lemma2
```

5.4.5/ EXAMPLE OF A PROOF FILE

```
Model "refproof"

% Index, Region, Constant, Variable, Function, Operator, Expression
% ...

Rule
rule01 : ∫ u_v_ dx_ = 0 → u_ = 0 if v_.Type = "Test"
rule02 : ∫ u_v_ dx_ = 0 → v_ = 0 if u_.Type = "Test"

Strategy
newRule : rule01 | rule02

Step
step11 : rule01  ; newRule ; rule02
step12 : rule01

Lemma
lemma01 : step11 ; step12 ; ... ; step19
lemma02 : step21 ; step22 ; ... ; step29

Model
refproof : lemma01 ; lemma02 ; ... ; lemma07
```

5.5/ EXTENSION FILES

An extension file implements an extension that operates on a PDE or a Proof to transform it into another PDE or another Proof. The elementary operations in these transformations are the second order rules that use second order patterns matching and consists in adding subterms named contexts at some positions, as explained in Section 5.5.1. To do so, an extension file may include the same sections as the kind of file to which it applies. It also include two sections `SORule` and `SOStrategy` for second order rules and strategies. The second order rules are explained in Section 5.5.2. The file ends with the description of the extension begining with the keyword `Extension`.

5.5.1/ SECOND ORDER PATTERN MATCHING, CONTEXTS AND SECOND ORDER RULES

The patterns used for pattern matching in Extensions have the same structure as the patterns used in pattern matching in proofs however they can also include shortcuts met

in proofs as shotcuts of rules or strategies. In the pattern matching of extensions, a rewriting variables matches any rewriting variable as well as any term. For instance, in an extension the rewriting rule

`Rule plusone : y_ → y_+1`

matches the first order rule `Rule plusone : x_ → x_+1` since the first order variable `x_` matches `y_`. It also matches `Rule plusone : x → x+1`.

A second order rewriting rule is on the form $\alpha \rightrightarrows \beta$ where α and β are any patterns of extensions, and β is equal to α excepted embedded additional terms called contexts. A context can be any expression where exactely one subterm is missing which is materialized by the symbol □ or `\ square`. For example

`2 * □`

`□ + b`

`Function u : "u_" [] [□] [] y_`

`Step step1: □`

are four contexts. Since the order of common parts of α and β is maintained, the added terms are automatically identified by the compiler. However, the unique determination of added terms requires additional rules. In case of more than one occurence of a same associative operation, the priority is given to the most right operations. For instance `a + c + b` is transformed into `a + (c + b)`, and in

`Rule so_rule : a+b ⇉ a+c+b`

`c + □` is identified as a context of b. In the example

`Rule so_rule : a+b ⇉ a*c+b`

due to the priority rule of the multiplication over the addition, $a * c + b$ is understood as $(a * c) + b$, so $\square + c$ is a context of a. An arbitrary number of contexts can be taken into consideration. A case with two contexts is

`Rule so_rule : a+b ⇉ 2+a*c+b`

where $2 + \square$ and $\square * c$ are contexts for the term $a + b$ and a respectively. Finally, the grammar of contexts is summarized as

$$\tau ::= \square \mid \texttt{f}(\alpha, \ldots, \alpha)$$

where α stands for any extension pattern and `f` for any function.

5.5.2/ Second Order Strategies

The second order strategies or SO-strategy or extensions constitute a very special class of strategies. This class includes the second order rules and is built to be stable under combination as defined in Section 5.6. An extension behaves like a rewriting strategy when applied to an input expression: either it succeeds and in this case it returns a new expression, or it fails.

The class SO-strategies is semantically equivalent to the class of HCE-strategies introduced in Section 4.4. The only difference is that the grammar of SO-strategies is user-

oriented making this class more like a programming language.

A labeled SO-pattern U is an ordinary SO-pattern in which some term-labels are introduced e.g. f(Loc1 Of x_,Loc2 Of y_,z_) is the SO-pattern f(x_,y_,z_) where the variables x_ and y_ are labeled with the label names Loc1 and Loc2 while the variable z_ is not labeled. We mention that the matching of a labeled pattern with an input expression is made by ignoring the labels. For instance, f(Loc1 Of x_,Loc2 Of y_,z_) matches with f(a_,b_,c_).

All elementary expressions are listed hereafter where α and β stand for any labeled second order patterns or their shortcuts, σ for any labeled second order patterns that include strategies or its shortcut, γ for any labeled second order patterns that does not include any strategy or its shortcut, S, S1 and S2 for any second order strategies, Loc, Loc1, Loc2 for any labels at positions, Label_Name for any label on strategies, and τ for contexts.

- **SO-rule:** The SO-rule $\alpha \rightrightarrows \beta$ is defined in Section 5.5.1. This extension fails if and only if α does not match with the input expression or if β can not be obtained from α by adding contexts, for instance f(a) $\rightrightarrows$ g(b) always fails.
- **Id(γ):** Id(γ);S specifies that the labeled SO-pattern γ is matched with the input expression. That is, Id(γ) is basically the identity rewriting rule $\gamma \to \gamma$. If the matching is successful then S is applied to the input expression. This extension fails if and only if γ does not match with the input expression or S fails when applied to the input expression.
- **Id(σ):** The extension Id(σ);;S behaves exactly like Id(γ);S but for σ that includes strategies.
- **Inside(.):** The extension Inside(S) applies S to all the fields of the input expression. For instance the fields of f(a,b,g(c)), where a,b,c are constants, are a,b and g(c). One has to be careful since the fields of some expressions e.g. $\int$ f(x)dx where x is a mathematical variable, are defined implicitly. This extension fails when applied to an input expression t if and only if it fails on all the fields of t. Besides, if S fails on some fields and succeeds on others, then it behaves like the identity on the failed fields by letting them unchanged.
- **Or:** The extension S1 Or S2 stands for the left-choice : S1 is applied to the input expression t, if it does not succeed S2 is applied to t. This extension fails if and only if both S1 and S2 fail when applied to t. The constructor Or is associative.
- **If · then ·:** The extension **If** (S1 and ... and Sn) **then** S, when applied to an input expression t, tests if each application of Si to t succeeds, if it is the case, then it applies S to t. Otherwise (i.e. one of the applications of Si to t fails), then this extension fails.
- **Insert:** The extension Insert τ, when applied to an input expression t, replaces t by τ and inserts t in the empty position of τ. When the extension Insert τ At Loc1 is applied to t, it applies Insert τ at the position Loc1.
- **At Loc:** The extension S At Loc_Name must be preceded by Id(γ); or Id(σ);; and the label declaration "Loc_Name Of" **must appear exactly once**[1] in γ and σ.

[1] We can relax this constraint by allowing the label declaration to refers to many non-nested subterms of γ and σ but this makes the unification formulas of extensions more complicated. So we keep this assumption to simplify the exposition.

Besides, we do not allow a location name to be defined in more than one SO-pattern in a given SO-strategy. The strategy S is applied to the subterm of the input term t that matches with the subterm labeled by Loc_Name in γ or σ.

- **Goto Label:** The extension Label_Name Of S introduces the label Label to which the extension S might refer via the command Goto. This extension fails if and only if S fails. The extension Goto Label_Name stands for going back where the label Label_Name was introduced. The purpose of the labels on SO-strategies together with the Goto instruction is to define recursive extensions. It is like the "goto " instruction in some languages. It is also similar to the "while" loop in the programming languages with the difference that there is no explicit halting condition. It halts when it fails or when there is a branch in a left-biased-choice in which there is no Goto.
- **. At Loc And . At Loc:** The extension S1 At Loc1 And S2 At Loc2 must be preceded by Id(γ); or Id(σ);; and it stands for the application of S1 At Loc_1 followed by the application of S2 At Loc_2. We impose that the labels Loc_1 and Loc_2 refer to parallel sub-SO-patterns of the same SO-pattern excepted when S2 is the Insert strategy. In such a case Loc_2 can refer to a position that is higher than Loc_1 or in a parallel branch. For more than two And this condition must be satisfied for any adjacent pair of And. Thus, this constructor is associative.
- **. And Insert:** The extension S And Insert(τ) when S does not insert a context to the root. This extension applies S and inserts τ.

The priority rules are as usual. The restricted form of compositions ";" and ";;" have a higher priority than the left-biased-choice "Or".

The grammar of the extensions is defined in (5.3),

$$
\begin{aligned}
&\texttt{S::=Fail} \mid \alpha \rightrightarrows \alpha \mid \texttt{Id}(\alpha)\texttt{;S} \mid \texttt{Id}(\alpha)\texttt{;;S} \mid \texttt{Inside(S)} \mid \texttt{S Or S} \mid \ \mid \textbf{If } C \textbf{ then } S \\
&\quad \texttt{Id}(\alpha)\texttt{;Insert}(\tau) \mid \texttt{Id}(\alpha)\texttt{;;insert}(\tau) \mid \texttt{S And Insert}(\tau) \\
&\quad \texttt{Label\_Name Of S} \mid \texttt{Goto Label\_Name} \mid \texttt{Id}(\alpha)\texttt{;R} \mid \texttt{Id}(\alpha)\texttt{;;R} \mid \\
&\quad \texttt{Id}(\alpha)\texttt{;(R And Insert}(\tau)\texttt{))} \mid \texttt{Id}(\alpha)\texttt{;;(R And Insert}(\tau)\texttt{))} \mid \\
&\quad \texttt{Id}(\alpha)\texttt{;(R Or R)} \mid \texttt{Id}(\alpha)\texttt{;;(R Or R)} \\
&\texttt{R::= R And R} \mid \texttt{S At Loc\_Name} \\
&C\texttt{::= S} \mid \texttt{S and S}
\end{aligned}
\tag{5.3}
$$

where α is a SO-pattern of expressions defined by the grammar (5.4),

$$\alpha\texttt{::= x} \mid \texttt{f}(\alpha\texttt{,...,}\alpha\texttt{)} \mid \texttt{Loc\_Name Of } \alpha \tag{5.4}$$

Here, x stands for any SO-variable and f for any function.

For instance the next SO-rule and SO-strategy

```
SORule plusone1 exp ⇉ exp + 1
```

```
SOStrategy plusone2 Id(Loc Of exp); Insert □ + 1 At Loc
```

are equivalent. Here exp is a SO-pattern of the kind Expression assumed to be previously defined. The difference between these two implementations is that plusone1 is implicit that is it uses an algorithm to find the contexts and their position while plusone2 is explicit since the positions and the contexts are made explicit by the programmer.

The SO-strategy

```
SOStrategy applyS Id(Loc Of exp+1); S At Loc
```

matches the input term with `exp+1` and if it is successful, it applies `S` to `exp`.

The SO-strategy

```
SOStrategy outermost Label Of (S Or Inside(Goto Label))
```

implements the strategy `OuterMost(S)`.

The SO-strategy

```
SOStrategy sOrAndIter Label1 Of (S1;Goto Label1 Or Label2 Of S2; Goto Label2)
```

either `S1` is applied successfully and in this case the strategy is re-applied, or `S1` fails, and in this case `S2` is applied repeatidely until it fails.

5.5.3/ CORRESPONDENCE BETWEEN SO-STRATEGIES AND HCE-STRATEGIES

The correspondence between the constructors of the grammar of the HCE-strategies (defined in Section 4.4, Definition 14) and those of the user-friendly SO-strategies is illustrated in the following table. Besides the fact that the constructors of SO-strategies are user-friendly, the positions of the patterns in SO-strategies are made explicit and need to be defined by means of the pattern labels (`Locs`). For instance, the SO-strategy `Id(f(LocOf x,x));S` corresponds to the HCE-strategy `f(x,x);@1.S`.

HCE-strategies	SO-strategies
μ`X.S`	`Label_Name Of S`
`X`	`Goto Label_Name`
`@i.`	`S At Loc_Name`
`@i.`τ	`Insert(`τ`) At Loc_Name`
Explicit positions	Implicit location names: `Loc_Name Of`α
α`;S`	`Id(`α`);S`
`[S,S']`	`S And S'`
`Most(S)`	`Inside(S)`
If (S1 and ... and Sn) **then** S	**If** (S1 and ... and Sn) **then** S

Table 9: Correspondence between the HCE-strategies and the User Language strategies

5.5.4/ UNIFICATION OF TERMS CONTAINING LOCATIONS

It remains to make explicit the unification algorithm of terms that contain location declarations since we want that the resulting term contains location declarations as well. We recall that in Section 4.2, we defined the unification of terms (that do not contain location declarations), we apart it next to the terms with location. When unifying two terms α_1, α_2 that contain locations, we firstly need to compute the concrete positions in α_1 and α_2 associated to the location names. Then we remove the location names from α_1 and α_2 and

unify the resulting terms. Then we restore the location names by paying an attention to the possible pair of locations that points towards the same position of the unified term. In such pair exists, we create a new location name that points to this position, and ommit the location name coming from α_1 and the one from α_2.

We denote by $\texttt{Loc\_Set}(\alpha)$ the set of location names defined in the term α. We write $\texttt{Loc\_Set}(\alpha_1) \cdot \texttt{Loc\_Set}(\alpha_2)$ for the location names obtained by contacetanation of each location name in $\texttt{Loc\_Set}(\alpha_1)$ with each location name in $\texttt{Loc\_Set}(\alpha_2)$.

We define the function that remove locations from terms by

$$\texttt{Remove\_Loc}(t) = \begin{cases} t & \text{if } t \text{ is a constant or a variable,} \\ \texttt{Remove\_Loc}(t') & \text{if } t = \texttt{Loc\_Name Of } t', \\ f(t'_1, \ldots, t'_n) & \text{if } t = f(t_1, \ldots, t_n), \text{ where } t'_i = \texttt{Remove\_Loc}(t_i). \end{cases} \tag{5.5}$$

If the location name $\texttt{Loc\_Name}$ is in $\texttt{Loc\_Set}(\alpha)$, then we write $\texttt{Position\_Of\_Loc}(\alpha, \texttt{Loc\_Name})$ for the position in α pointed by $\texttt{Loc\_Name}$. For instance, if $\alpha = f(g(a, \texttt{Loc\_Name Of } b))$, then $\texttt{Position\_Of\_Loc}(\alpha, \texttt{Loc\_Name}) = 1 \cdot 2$. Notice that since we don't allow two label names pointing to the same position, the function $\texttt{Position\_Of\_Loc}(\alpha, \cdot)$ is injective. Given two terms α_1, α_2, with $\texttt{Loc\_Set}(\alpha_1) \cap \texttt{Loc\_Set}(\alpha_2) = \emptyset$, we extend the function $\texttt{Position\_Of\_Loc}(\alpha, \cdot)$ to operate on two input terms:

$$\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \cdot) : \texttt{Loc\_Set}(\alpha_1) \cup \texttt{Loc\_Set}(\alpha_2) \cup (\texttt{Loc\_Set}(\alpha_1) \cdot \texttt{Loc\_Set}(\alpha_2)) \longrightarrow \mathbb{N}_\epsilon^\omega$$

as follows

$\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \texttt{Loc\_Name}) =$

if $\texttt{Loc\_Name} \in \texttt{Loc\_Set}(\alpha_1) \cdot \texttt{Loc\_Set}(\alpha_2)$ and
$\texttt{Loc\_Name} = \texttt{Loc\_Name1} \cdot \texttt{Loc\_Name2}$ and
$\texttt{Position\_Of\_Loc}(\alpha_1, \texttt{Loc\_Name1}) = \texttt{Position\_Of\_Loc}(\alpha_2, \texttt{Loc\_Name2})$ **then**
$\texttt{Position\_Of\_Loc}(\alpha_1, \texttt{Loc\_Name1})$
elseif $\texttt{Loc\_Name} \in \texttt{Loc\_Set}(\alpha_1)$ **then** $\texttt{Position\_Of\_Loc}(\alpha_1, \texttt{Loc\_Name})$
elseif $\texttt{Loc\_Name} \in \texttt{Loc\_Set}(\alpha_2)$ **then** $\texttt{Position\_Of\_Loc}(\alpha_2, \texttt{Loc\_Name})$

Notice that $\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \cdot)$ is injective.

For a term β, we shall write $\texttt{Restore\_Loc}(\alpha_1, \alpha_2, \beta)$ for the function that introduces in β every location name $\texttt{Loc\_Name} \in \texttt{Dom}(\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \cdot))$ at the position $\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \texttt{Loc\_Name})$.

We are ready to define the unification algorithm of terms containing locations.

Algorithm 1 Unification of terms containing locations

Input: Two terms α_1, α_2 with locations with $\texttt{Loc\_Set}(\alpha_1) \cap \texttt{Loc\_Set}(\alpha_2) = \emptyset$
Output: A term β with locations such that

$$\begin{cases} \texttt{Remove\_Loc}(\beta) & = \texttt{Remove\_Loc}(\alpha_1) \wedge \texttt{Remove\_Loc}(\alpha_2), \text{ and} \\ \texttt{Loc\_Set}(\beta) & = \texttt{Dom}(\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \cdot)) \end{cases}$$

1: **function** UNIFIY(α_1, α_2)
2: $\quad \alpha_1' \leftarrow \texttt{Remove\_Loc}(\alpha_1)$
3: $\quad \alpha_2' \leftarrow \texttt{Remove\_Loc}(\alpha_2)$
4: $\quad \beta' \leftarrow \alpha_1' \wedge \alpha_2'$
5: $\quad \beta \leftarrow \texttt{Restore\_Loc}(\alpha_1, \alpha_2, \beta')$
6: $\quad$ **return** β
7: **end function**

5.6/ FORMULAS OF UNIFICATION OF SECOND ORDER STRATEGIES

It is recalled that in the theoretical framework of Extension-Combination, the combination of two extensions involves their unification. The unification of the theoretical has its counterpart when combining extensions expressed in the User Language. Table below provides the unification formulas that has been established and implemented. However, we deliberately omit the symmetric cases. For instance the symmetric case of Case 8 in the table would be the unification of S with S' `Or` S'' which is omitted.

Besides, in the unification table 10 we assume that, for each extension of the form $\texttt{Id}(\alpha)$; `S At Loc_Name`, the label `Loc_Name` points towards an immediate subterm (i.e. child) of the term α. This is not really a restriction on the extensions that we unify since one can turn any extension into an equivalent one that fulfills this assumption. Finally we mention that the formulas are arranged by highest priority order from the top of the table to the bottom. As a consequence, Formula 11 has to be applied before Formula 12. That is, we apply Formula 12 only if $\texttt{S}_2$ is not of the form $\texttt{Id(u)}; \texttt{S}''$. Finally we mention that we omitted the case when the first extension is of the form $\alpha; ; \texttt{S}'$ since it is similar to Case 7.

Before the definition of the unification of extensions, we introduce some notations. If S, S', S'' are extensions, then we write $S[S' := S'']$ for the extension that results from S by replacing each occurence S' in S by S''. Besides, if `Loc_Name1, Loc_Name2` are location names, we shall write $S[\texttt{Loc\_Name1} := \texttt{Loc\_Name2}]$ for the replacement of `Loc_Name1` by `Loc_Name1` in S. We shall overload notations and write $\texttt{Loc\_Set}(S)$ for the set of location names defined in all terms in the extension S. It follows from the constraints imposed on the locations, that for each location name `Loc_Name` in an extension S there corresponds a unique term in which it is defined. This term will be denoted by $\texttt{Term\_Of\_Loc}(\texttt{Loc\_Name}, S)$.

Given two terms α_1, α_2 with locations that can be unified, we define the renaming function $\texttt{Rename\_Loc}(\alpha_1, \alpha_2, \cdot)$ that renames the location names in $\texttt{Loc\_Set}(\alpha_1)$ and $\texttt{Loc\_Set}(\alpha_2)$ that point towards the same position:

$$\texttt{Rename\_Loc}(\alpha_1, \alpha_2, \cdot) : \texttt{Loc\_Set}(\alpha_1) \cup \texttt{Loc\_Set}(\alpha_2) \longrightarrow \texttt{Dom}(\texttt{Position\_Of\_Loc}(\alpha_1, \alpha_2, \cdot))$$

by $\texttt{Rename\_Loc}(\alpha_1, \alpha_2, \texttt{Loc\_Name}) = \texttt{Loc\_NameX}$, where $\texttt{Loc\_NameX} \in$

$\texttt{Loc\_Set}(\texttt{Unifiy}(\alpha_1, \alpha_2))$ and either

1. $\texttt{Loc\_Name} \in \texttt{Loc\_Set}(\alpha_1)$ and $\texttt{Position\_of\_Loc}(\alpha_1, \texttt{Loc\_Name}) = \texttt{Position\_of\_Loc}(\texttt{Unify}(\alpha_1, \alpha_2), \texttt{Loc\_NameX})$, or
2. $\texttt{Loc\_Name} \in \texttt{Loc\_Set}(\alpha_2)$ and $\texttt{Position\_of\_Loc}(\alpha_2, \texttt{Loc\_Name}) = \texttt{Position\_of\_Loc}(\texttt{Unify}(\alpha_1, \alpha_2), \texttt{Loc\_NameX})$.

Given an extension S we shall write $\texttt{Rename\_Loc}(\alpha_1, \alpha_2, \cdot)(S)$ for the extension that results from the replacing in S of each location name $\texttt{Loc\_Name} \in \texttt{Dom}(\texttt{Rename\_Loc}(\alpha_1, \alpha_2, \cdot))$ by $\texttt{Rename\_Loc}(\alpha_1, \alpha_2, \texttt{Loc\_Name})$.

	Extension 1	**Extension 2**	**Unification: Extension 1 ∩ Extension 2**
1	Insert τ	Insert τ'	Insert $\tau \cdot \tau'$
2	Insert τ	S_2 At Loc_Name	**If**(S_2 At Loc_Name) **Then** (S_2 At Loc_Name) And Insert τ
3	S_1 At Loc_Name	Insert τ	**If**(S_1 At Loc_Name) **Then** (S_1 At Loc_Name) And Insert τ
4	Insert τ	S_2 And Insert(τ')	S_2 And (Insert $\tau \cap$ Insert τ')
5	Insert τ	S_2 And S_2'	S_2 And S_2' And Insert τ
6	S_1 And Insert(τ)	Insert τ'	S_1 And (Insert $\tau \cap$ Insert τ')
7	S_1 And S_1'	Insert τ'	S_1 And S_1' And Insert τ'
8	$\underbrace{S_1 \text{ At Loc\_Name1}}_{R_1}$ And $\underbrace{S_1' \text{ At Loc\_Name1}'}_{R_1'}$	$\underbrace{S_2 \text{ At Loc\_Name2}}_{R_2}$	$(R_1 \cap R_2)$ And R_1' if Loc_Name1 ≡ Loc_Name2 R_1 And $(R_1' \cap R_2)$ if Loc_Name1′ ≡ Loc_Name2 R_1 And R_1' And R_2 otherwise
9	S_1	S_2 And S_2'	$(S_1 \cap S_2) \cap S_2'$
10	$\underbrace{S_1 \text{ At Loc\_Name1}}_{S_1'}$	$\underbrace{S_2 \text{ At Loc\_Name2}}_{S_2'}$	R At Loc_Name1Name2 if Loc_Name1 ≡ Loc_Name2 S_1' And S_2' otherwise where R = $(S_1 \cap S_2)$[Loc_Name1 := Loc_Name1Name2, Loc_Name2 := Loc_Name1Name2]
11	Id(α_1); S_1	Id(α_2); S_2	Id(Unify(α_1, α_2)); $(S_1' \cap S_2')$ if ∃ mgu(α_1, α_2) Fail otherwise where $S_i' = \text{Rename\_Loc}(\alpha_1, \alpha_2, \cdot)(S_i)$, $i = 1, 2$
12	Id(α); S_1	S_2	Id(α); $(S_1 \cap S_2)$
13	S_1 Or S_1'	S_2	$(S_1 \cap S_2)$ Or $(S_1' \cap S_2)$
14	Goto Label_Name1	Goto Label_Name2	Goto Label_Name1_Name2
15	$\underbrace{\text{Label\_1 Of } S_1}_{S}$	$\underbrace{\text{Label\_2 Of } S_2}_{S'}$	Label_1_2 Of $(S_1 \cap S_2)$[Goto(Label_1) := S, Goto(Label_2) := S′]
16	$\underbrace{\text{Label\_1 Of } S_1}_{S}$	S_2	S_1[Goto(Label_1) := S] ∩ S_2
17	Inside(S_1)	Inside(S_2)	Inside($(S_1 \cap S_2)$ Or S_1 Or S_2)
18	Id(α); S_1	Inside(S_2)	Id(α); $(S_1 \cap S_2)$
19	Insert(τ)	Inside(S_2)	Inside(S_2) And Insert(τ)
20	S At Loc_Name	Inside(S_2)	(S At Loc_Name) ∩ (Id(α'); ((S At Loc_Name_1) And ... And (S At Loc_Name_n))) where α = Term_Of_Loc(Loc_Name) and $\alpha = f(t_1, \ldots, t_n)$, and $\alpha' = f$(Loc_Name_1 Of $t_1, \ldots,$ Loc_Name_n Of t_n)
21	S where S = S_1 At Loc_N1 And S_2 At Loc_N2	Inside(S_2)	S ∩ (Id(α'); ((S At Loc_Name_1) And ... And (S At Loc_Name_n))) where α = Term_Of_Loc(Loc_N1) and $\alpha = f(t_1, \ldots, t_n)$, and $\alpha' = f$(Loc_Name_1 Of $t_1, \ldots,$ Loc_Name_n Of t_n)

Table 10: Formulas of unification of SO-strategies. The formulas are arranged by highest priority order from the top of the table to the bottom. We assume that the location names defined in a term α point toward an immediate subterm (child) of α. For two input extentions S_1 and S_2 we assmue also that the sets Set_Of_Loc(S_1) and Set_Of_Loc(S_2) are disjoint. We have written Term_Of_Loc(Loc_Name) instead of Term_Of_Loc(Loc_Name, S) to mean the pattern in the input

extension S in which Loc_Name is defined. Besides, we write Loc_Name1 $\equiv$ Loc_Name2 to mean that Loc_Name1 and Loc_Name2 point towards the same position.

It is worth to explain Formula 11 on the unification of $\mathrm{Id}(\alpha_1);\mathrm{S}_1$ and $\mathrm{Id}(\alpha_2);\mathrm{S}_2$. When we unify the terms α_1 and α_2 that may contain location definitions by the algorithm given in Section 5.5.4, it is possible that a location name Loc_Name1 of α_1 and a location name Loc_Name2 of α_2 point towards the same position. In this case we replace both Loc_Name1 and Loc_Name2 by a new location name, say Loc_Name1Name2, in the resulting unified term. As a consequence, the location name Loc_Name1 (resp. Loc_Name2) has to be replaced by Loc_Name1Name2 in the extension S_1 (resp. S_2).

5.7/ A HIGH LEVEL USER LANGUAGE

Once the files of PDEs, proofs and extensions have been written, their use is managed at a programming level in MATLAB. Functions available for

- compiling that is to transform a PDE, a proof or an extension file written in the User Language into a file in the Processing Language,
- printing a PDE, a proof or an extension expressed in the processing language in three formats: Latex, PDF, html, and Unicode text,
- applying a proof to a PDE to form a new PDE and applying an extension to a PDE or a proof yielding a new PDE or a new proof,
- and combining two extensions generating a new extension.

5.7.1/ COMPILE

```
output_filename = compile("filename.xxx")
```

transforms any file `filename.pde` or `filename.proof` or `filename.ext` in the User Language into a file named `filename.ml` written in the Processing Language ready to be used by the other commands as `inspect` or `HtmlView` or `Applyxxx` .

5.7.2/ INSPECT

```
inspect("filename.ml", hidden_fields, "filename.txt" )
```

from the file `"filename.ml"` in the processing language display the corresponding expression into the matlab command window, into the text file `"filename.txt"`, into the Latex file `"filename.tex"`. The Latex file is compiled to build the pdf file `"filename.pdf"`. The fields listed in the list `hidden_fields` are hidden. The list of hidden fields has the form `["field_1";...;"field_n"]` and the fields are taken in the list of all possible fields, for instance for a pde file:

`Reg` for regions,

`MathVar` for mathematical variables,
`OperExpr` for mathematical expressions,
`Fun` for mathematical functions,
`Oper` for operators.

For instance, the pde file

```
PDE "pde"
Region
 omega:  "omega" [] [1] [] _ _
Variable
 x:  "x" [] omega
Function
 f:  "f" [] [x] [] "Given"
PDE
 pde:  f = x
```

is compiled with the command

```
pde = compile('pde.pde');
```

and the four display commands

```
inspect(pde,'["Fun","MathVar"]','pde.txt');
inspect(pde,'["Fun"]','pde.txt');
inspect(pde,'["MathVar"]','pde.txt');
inspect(pde,'["Reg"]','pde.txt');
```

produce

```
>> Source term :  Pde pde:  f=x
>> Source term :  Pde pde:  f=MathVar(x,∅,Reg(omega,∅,1,∅,⊥ ,⊥ ))
>> Source term :  Pde pde:  Fun(f,∅,x,∅,Given)=x
>> Source term :  Pde pde:  Fun(f,∅,MathVar(x,∅,omega),∅,Given)=MathVar(x,∅,omega)
```

in the matlab window.

The proof file

```
Model "proof"
Constant
 psi:  "psi"
Rule
 simplify:  a_ - a_ → 0
 create_source_term:  a_ = b_ → psi = a_ - a_
```

```
Step
 step1:  create_source_term ↑
 step2:  simplify ↑
Model
 proof:  step1; step2
```

is compiled with the command

```
proof = compile('proof.proof');
```

and it is displayed with

```
inspect(proof,'["Reg";"MathVar";"OperExpr";"Fun"]');
```

yielding

```
>> Printed proof :  Model(
(
 Step(step1,(create_source_term:  (a_=b_ → psi=(a_+-(a_))))↑) ;
 Step(step2,(simplify:  ((a_+-(a_)) → 0))↑)))
```

in the matlab window

5.7.3/ HTMLVIEW

```
HtmlView("filename.ml")
```

translates the content of the file `"filename.ml"` into a friendly writing ni a html window managed through matlab. One can hide or show any highlighted subterm by a click or operate a global hide or show of any given field; see Figure **??**. The hide/show operations on a single identifier are when all selected fields are ”blank” while the global ones are when selecting a field and clicking anywhere in the html window.

5.7.4/ APPLICATION OF A PROOF OR AN EXTENSION

Applying a proof to a PDE is done by

```
output_pde = apply_proof_to_pde("proof.ml","pde.ml","filename_out")
```

where `output_pde = "filename_out.ml".`

Applying an extension to a PDE or a proof is done by two different functions

```
output_pde = apply_ext_to_pde("extension.ml","pde.ml","filename_out")
```

where `output_pde="filename_out.ml"` and

```
output_proof = apply_ext_to_proof("extension.ml","proof.ml","filename_out")
```

where `output_proof = "filename_out.ml".`

5.7.5/ COMBINE

The combination of two extensions is built with the command

```
output_ext = combine_ext("extension1.ml","extension2.ml","extension_from_1_and_2.ml");
```

where `output_ext = "extension_from_1_and_2.ml"`.

It yields an extension that can therefore used to be applied to a PDE or a proof, as well to be combined with another extension.

For instance, given two extensions

```
Extension "extofproof_1"
Rule
 rule :  a_ + b_ → a_ + b_
 sorule :  a_ ⇉ a_ + 1
Extension
 extofproof_1 :  sorule
```

and

```
Extension "extofproof_2"
Rule
 rule :  a_ + b_ → a_ + b_
 sorule1 :  b_ ⇉ b_ + 2
Extension
 extofproof_2 :  sorule1
```

after compilation

```
output_ext_1 = compile("extofproof_1.ext");
output_ext_2 = compile("extofproof_2.ext");
```

they are combined

```
output_ext_1_2 = combine_ext(output_ext_1,output_ext_2,"output_ext_1_2.ml");
```

and the three extensions are displayed in the matlab command window with the commands

```
inspect(output_ext_1,'["Reg";"MathVar";"OperExpr";"Fun"]');
inspect(output_ext_2,'["Reg";"MathVar";"OperExpr";"Fun"]');
inspect(output_ext_1_2,'["Reg";"MathVar";"OperExpr";"Fun"]');
```

yielding

```
>> Extension :  a_ ⇉ (a_ + 1)
>> Extension :  b_ ⇉ (b_ + 2)
```

5.8/ MESSAGE OF THE DEBUGGER

Error: This expression has type strategy but an expression was expected of type tree : the type strategy is for `Rule` and `Strategy` and tree for the other keywords `Constants`, `Variable` etc.

5.9/ ILLUSTRATION OF A SEQUENCE OF OPERATIONS

The next figures illustrate the use of the language for writing PDEs, proofs and extensions. They also illustrate the display tools, the application of a proof to a PDE, the application of extensions to a PDE and the combination of two extensions. Finally, the extensions resulting from the combination is applied to the PDE.

```
1  PDE "simple_pde"
2  Region
3    gamma : "gamma" [] [] [] __ __
4  Variable
5    xg : "xg" [] gamma
6  Function
7    n : "n" [] [xg] [] "Given"
8  Region
9    omega : "omega" [] [] [] gamma n
10 Variable
11   x : "x" [] omega
```

```
12 Function
13   u : "u" [] [x] [] "Unknown"
14   v : "v" [] [x] [] "Test"
15   f : "f" [] [x] [] "Given"
16 Operator
17   tr_u : "trace" [] [u] [x] [xg] []
18   tr_v : "trace" [] [v] [x] [xg] []
19 Expression
20   weak_form :
21 ∫ ∂ u / ∂ x • ∂ v / ∂ x đ x = ∫ f • v đ x
22 PDE simple_pde : weak_form
```

Figure 5.4: Description of a simple PDE in Memsalab.

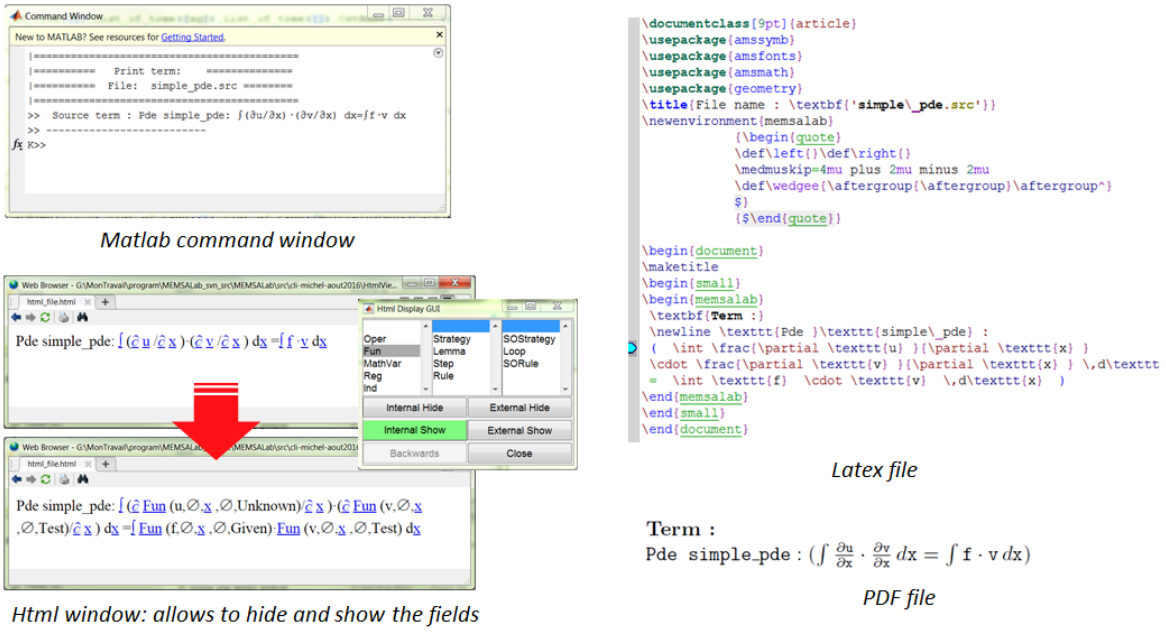


Figure 5.5: Three manners to display a PDE, a proof or an extension: in a matlab window, in a Html window or in a Latex file.

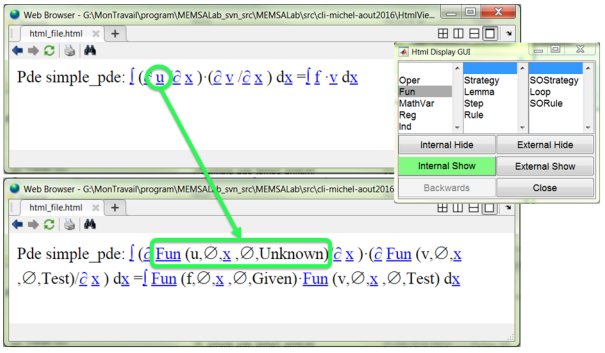


Figure 5.6: Hiding and showing selected fields in a Html window. Here the ”Functions” are expanded.

```
Model "simple_proof"
Rule
  green_rule : ∫ z_ • ∂ w_ / ∂ x_ đ x_
      → - ∫ ∂ z_ / ∂ x_ • w_ đ x_
  weak_form_interpretation :
   - ∫ f_ • r_ đ x_ = ∫ g_ • r_ đ x_
      → - f_ = g_ if r_ .Type = "Test"
Strategy
   green_rule_st : green_rule ↧
   weak_form_interpretation_st :
      weak_form_interpretation ↧
Model
  simple_proof :
     green_rule_st ;
     weak_form_interpretation_st
```

Figure 5.7: A proof in memsalab for deriving the strong formulation associated to a weak formulation.

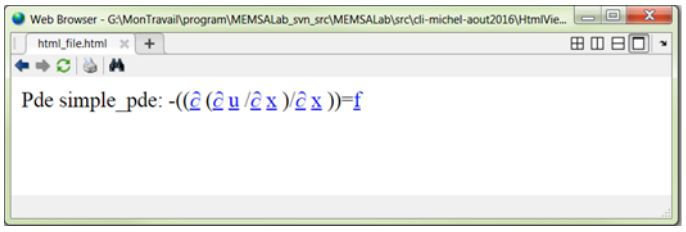


Figure 5.8: The strong form of the pde resulting from the application of the above proof to the above weak formulation.

```
% ext_neumann
Extension "pde_neumann"
#Include "pde.pde"
Function
  g : "g" [] [xg] [] "Given"    % added
Expression
  weak_form'neumann :
∫ ∂ u_ / ∂ x_ • ∂ v_ / ∂ x_ đ x_ = ∫ f_ • v_ đ x_
   + ∫ g • tr_v đ xg
Rule
  weak_form_ext : weak_form ⇉ weak_form'neumann
Extension
  pde_neumann : Label_neumann Of (
     Id (weak_form) ;
     weak_form_ext  Or Inside( Goto Label_neumann)
     )
```

Figure 5.9: An extension that takes into account a non-homogeneous Neumann condition in the weak formulation.

```
% ext_reaction
Extension "pde_reaction"
#Include "pde.pde"
Function
  a : "a" [] [x] [] "Given"    % added
Expression
  weak_form'reaction :
∫ ∂ u / ∂ x • ∂ v / ∂ x đ x + ∫ a • u • v đ x = ∫ f • v đ x
Rule
  weak_form_ext : weak_form ⇉ weak_form'reaction
Extension
  pde_reaction : Label_reaction Of (
      Id (weak_form) ;
      weak_form_ext  Or Inside( Goto Label_reaction)
      )
```

Figure 5.10: An extension that takes into account a reaction term in the weak formulation.

```
Extension  : Label_neumannLabel_reaction: [
    [(WF; [ Insert [∘+G]@2 And Insert [∘+A]@1])    % unification
        Or
        (WF;[
            Inside (
                Label_reaction: [(WF; [ Insert [∘+A]@1]) Or Inside ( Goto Label_reaction )]
            )
            And Insert [∘+G]@2
        ])
    ]
    Or
    [
        (WF;[
                Inside (
                        Label_neumann: [(WF; [ Insert [∘+G]@2]) Or Inside ( Goto Label_neumann)]
                    )
                And Insert [∘+A]@1
        ])
    Or Inside ( Goto Label_neumannLabel_reaction)
    ]
]
% where
A = ∫a·u·v dx
G = ∫g·Oper(trace,v) dxg
WF = ∫(∂u/∂x)·(∂v/∂x) dx=∫f·v dx
```

Figure 5.11: Result of the combination of the two extensions to take into account the Neumann boundary conditions and the reaction term in the weak formulation. The expression has been manually reformated so that to be easily readable.

```
Source term :
Pde pde:
∫ ( ∂ u/ ∂ x)·( ∂ v/ ∂ x) dx + ∫ a·u·v dx = ∫ f·v dx + ∫ g·Oper(trace,v) dxg
```

Figure 5.12: Result of the application of the combination of the two extensions that take into account the Neumann boundary conditions and the reaction term in the weak formulation. The expression has been manually reformated from the display in the matlab window.

6

CONCLUSIONS AND PERSPECTIVE

The leading physical phenomena governing the behavior of the micro-mirror cell have been simulated. Precisely, the simulations cover the following effects: electromechanics, pull-in, heat transfer and contact mechanics.

Then, the actuation voltage has been reduced by solving a minimization problem with respect to the thickness and the length of the suspended beams.

In another part, a two-scale model that includes boundary layer effects has been derived for the electrical field. Its derivation is carried out in the same framework as the reference proof of `MEMSALab`. This model has been implemented in COMSOL for numerical simulation. Apart of the thesis, its derivation has been expressed as a combination of extensions and implemented in `MEMSALab` which constitutes a first full-size application.

In my thesis, the strategy based combinations and extensions have been proven to be correct comparing to the position based combinations and extensions. The combination of strategy based extensions is shown to be closed under a given class of strategies.

In addition, the User Language have been introduced and developed in a manner that significantly simplifies program writing.

Finally, we suggest possible future works:

1. Perform numerical simulations of the contact problems occurring when the mirror touches the stopper beam and when the landing beams touch the landing pads.

2. Optimize the tilt angle of the mirror, the major parameters being the length and the thickness of the stopper beam and the height of the pillars.

3. Analyze the trade-off between the objectives of keeping a 20° tilted angle and of minimizing the pull-in voltage while the beam thickness is the common variable.

4. Derive a two-scale model of the mechanical behavior of the micro-mirror array in nonlinear elasticity that could be coupled with our two-scale electric model.

5. Implement the new two-scale models in `MEMSALab` as a combination of extensions of the Reference Proof.

6. Build a new evaluation tool to evaluate the extended Proof.

7. Build a link between `MEMSALab` and COMSOL.

BIBLIOGRAPHY

[Allaire, 1992] Allaire, G. (1992). **Homogenization and two-scale convergence**. *SIAM Journal on Mathematical Analysis*, 23(6):1482–1518.

[Allaire et al., 1998] Allaire, G., et Conca, C. (1998). **Bloch wave homogenization and spectral asymptotic analysis**. *Journal de mathématiques pures et appliquées*, 77(2):153–208.

[Arbogast et al., 1990] Arbogast, T., Douglas, Jr, J., et Hornung, U. (1990). **Derivation of the double porosity model of single phase flow via homogenization theory**. *SIAM Journal on Mathematical Analysis*, 21(4):823–836.

[Arnold et al., 2001] Arnold, A., et Niwinski, D. (2001). **Rudiments of μ-calculus**. Elsevier.

[Belkhir et al., 2016] Belkhir, W., Ratier, N., Nguyen, D. D., et Lenczner, M. (2016). **Closed combination of context-embedding iterative strategies**. Technical report, LORIA - Université de Lorraine ; FEMTO-ST.

[Belkhir et al., 2015] Belkhir, W., Ratier, N., Nguyen, D. D., Yang, B., Lenczner, M., Zamkotsian, F., et Cirstea, H. (2015). **Towards an automatic tool for multi-scale model derivation illustrated with a micro-mirror array**. In *SYNASC 2015*, pages 47–54. IEEE Computer Society.

[Blanchard et al., 2007] Blanchard, D., Gaudiello, A., et Griso, G. (2007). **Junction of a periodic family of elastic rods with a 3d plate. part i**. *Journal de mathématiques pures et appliquées*, 88(1):1–33.

[Blanchard et al., 2008] Blanchard, D., Gaudiello, A., et Mel'nyk, T. A. (2008). **Boundary homogenization and reduction of dimension in a kirchhoff-love plate**. *SIAM Journal on Mathematical Analysis*, 39(6):1764–1787.

[Bourgeat et al., 1996] Bourgeat, A., Luckhaus, S., et Mikelic, A. (1996). **Convergence of the homogenization process for a double-porosity model of immiscible two-phase flow**. *SIAM Journal on Mathematical Analysis*, 27(6):1520–1543.

[Canonica, 2012] Canonica, M. D. (2012). **Large Micromirror Array Based on a Scalable Technology for Astronomical Instrumentation**. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

[Casado-Diaz, 2000] Casado-Diaz, J. (2000). **Two-scale convergence for nonlinear dirichlet problems in perforated domains**. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 130(02):249–276.

[Casado-Diaz et al., 2004] Casado-Diaz, J., et Luna-Laynez, M. (2004). **Homogenization of the anisotropic heterogeneous linearized elasticity system in thin reticulated**

structures. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 134(06):1041–1083.

[Casado-Díaz et al., 2001] Casado-Díaz, J., Luna-Laynez, M., et Martín, J. D. (2001). **An adaptation of the multi-scale methods for the analysis of very thin reticulated structures**. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(3):223–228.

[Ciarlet, 1993] Ciarlet, P. G. (1993). **Mathematical elasticity: Three-dimensional elasticity**, volume 1. Elsevier.

[Ciarlet, 1997] Ciarlet, P. G. (1997). **Theory of plates**, volume 2. Elsevier.

[Cioranescu et al., 2008] Cioranescu, D., Damlamian, A., et Griso, G. (2008). **The periodic unfolding method in homogenization**. *SIAM Journal on Mathematical Analysis*, 40(4):1585–1620.

[Cirstea et al., 2003] Cirstea, H., Kirchner, C., Liquori, L., et Wack, B. (2003). **Rewrite strategies in the rewriting calculus**. In Gramlich, B., et Lucas, S., editors, *3rd International Workshop on Reduction Strategies in Rewriting and Programming* , volume 86(4) of *ENTCS*, pages 18–34, Valencia, Spain. Elsevier.

[De et al., 2004] De, S. K., et Aluru, N. (2004). **Full-lagrangian schemes for dynamic analysis of electrostatic mems**. *Microelectromechanical Systems, Journal of*, 13(5):737–758.

[El-Zafrany, 1997] El-Zafrany, A. (1997). **Non-linear finite element analysis of solids and structures: Volume 2: Advanced topics**. *Proceedings of the Institution of Mechanical Engineers*, 211(6):489.

[Fu et al., 2001] Fu, Y. B., et Ogden, R. W. (2001). **Nonlinear elasticity: theory and applications**, volume 281. Cambridge University Press.

[Griffiths et al., 1999] Griffiths, D. J., et College, R. (1999). **Introduction to electrodynamics**, volume 3. prentice Hall Upper Saddle River, NJ.

[Griso, 2002] Griso, G. (2002). **Estimation d'erreur et éclatement en homogénéisation périodique**. *Comptes Rendus Mathematique*, 335(4):333–336.

[Griso, 2004] Griso, G. (2004). **Error estimate and unfolding for periodic homogenization**. *Asymptotic Analysis*, 40(3, 4):269–286.

[Griso, 2006] Griso, G. (2006). **Interior error estimate for periodic homogenization**. *Analysis and Applications*, 4(01):61–79.

[Kovetz, 2000] Kovetz, A. (2000). **Electromagnetic theory**. Oxford University Press Oxford.

[Lenczner, 1997] Lenczner, M. (1997). **Homogénéisation d'un circuit électrique**. *Comptes Rendus de l'Académie des Sciences-Series IIB-Mechanics-Physics-Chemistry-Astronomy*, 324(9):537–542.

[Lenczner, 2006] Lenczner, M. (2006). **Homogenization of linear spatially periodic electronic circuits.** *NHM*, 1(3):467–494.

[Lenczner, 2007] Lenczner, M. (2007). **Multiscale model for atomic force microscope array mechanical behavior**. *Applied Physics Letters*, 90(9):091908.

[Lenczner et al., 1999] Lenczner, M., et Senouci-Bereksi, G. (1999). **Homogenization of electrical networks including voltage-to-voltage amplifiers**. *Mathematical Models and Methods in Applied Sciences*, 9(06):899–932.

[Lenczner et al., 2007] Lenczner, M., et Smith, R. C. (2007). **A two-scale model for an array of AFM's cantilever in the static case**. *Mathematical and Computer Modelling*, 46(5-6):776–805.

[Li et al., 2003] Li, G., et Aluru, N. R. (2003). **Efficient mixed-domain analysis of electrostatic mems**. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 22(9):1228–1242.

[Lukkassen et al., 2002] Lukkassen, D., Nguetseng, G., et Wall, P. (2002). **Two-scale convergence**. *Int. J. Pure Appl. Math*, 2(1):35–86.

[Mase et al., 1970] Mase, G. E., et Mase, G. (1970). **Continuum mechanics**, volume 970. McGraw-Hill New York.

[Nguetseng, 1989] Nguetseng, G. (1989). **A general convergence result for a functional related to the theory of homogenization**. *SIAM Journal on Mathematical Analysis*, 20(3):608–623.

[Nguyen, 2014] Nguyen, T. t. (2014). **Contribution to peroidic homogenization of a spectral problem and of the wave equation**. Theses, Université de Franche-Comté.

[Tarski, 1955] Tarski, A. (1955). **A lattice-theoretical fixpoint theorem and its applications.** *The Journal of Symbolic Logic*, 5(4):370.

[Waldis, 2010] Waldis, S. (2010). **MEMS-based mirror array for astronomical instrumentation**. PhD thesis, Université de Neuchâtel.

[Yang, 2014] Yang, B. (2014). **Contribution to a kemel of symbolic asymptotic modeling software.** PhD thesis, Université de Franche-Comté.

[Yang et al., 2014] Yang, B., Belkhir, W., et Lenczner, M. (2014). **Computer-aided derivation of multi-scale models: A rewriting framework**. *International Journal for Multiscale Computational Engineering.*, 12(2):91–114.

List of Figures

LIST OF TABLES

List of Definitions

I

APPENDIX

A

Simulation Demonstrations

A.1/ Stationary Simulation

The Model Definition and Result are already discussed in Section 2.3.

A.1.1/ Model Instruction

We assume that the geometry, the material and the boundary conditions are already built.

PARAMETER

1 In the Model Builder window, expand the Global > Definitions node, then click Parameters. In the Settings window forParameters, locate the Parameters section. In the table, input the following parameter:

Name	Expression	Value	Description
VE	$-30[V]$	30	Imposed potential on electrode
VM	$60[V]$	60	Imposed potential on mirror

ELECTROMECHANICS

1 In the Physics toolbar, click Domains and choose Linear Elastic Material. In the Settings window for Linear Elastic Material, locate the Domain Selection section. From the Selection list, choose Domain then enter 1-3, 5-17.

2 In the Physics toolbar, click Boundaries and choose Fixed Constraint. From the Selection list, choose Boundary then enter 3.

3 In the Physics toolbar, click Boundaries and choose Symmetry. From the Selection list, choose Manual then enter 2, 5, 8, 21, 73.

4 In the Physics toolbar, click Boundaries and choose Prescribed Mesh Displacement. From the Selection list, choose Manual then enter 10-11, 19, 115. Un-check Prescribed z Displacement to allow the mesh to move vertically.

5 In the Physics toolbar, click Boundaries and choose Electric Potential. From the Selection list, choose Manual then enter 12, 16, 82, 84, 96-97, 108. Locate Electric Potential section, in the V0 text field, type VE. Rename it as VE.

6 Add another Electric Potential node for VM with imposed boundaries 20, 22-23, 27-28, 30, 32, 42-45, 47-49, 52-54, 57-59, 63-72, 74-76, 78, 85-90, 93, 109, 113-114.

STUDY

1 To add a new study, on the Model toolbar, click Add Study to open the Add Study window. Then go to the Add Study window, find the Studies subsection. In the Select study tree, select Preset Studies > Stationary. Then, click Add Study in the window toolbar.

2 Rename it to Stationary then click Compute.

A.2/ PULL-IN ANALYSIS

These following models perform pull-in analysis of a micro-mirror cell, to predict the point at which the biased system becomes unstable. They are different and the second one is more efficient than the first one. The geometry and operation of the device are already discussed in previous chapters.

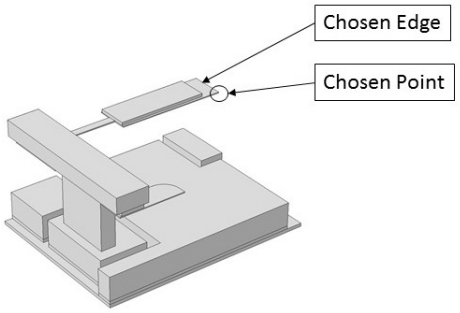
A.2.1/ MODEL 1

A.2.1.1/ MODEL DEFINITION

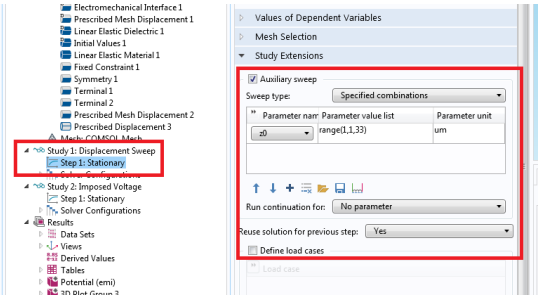
Consider an edge and a point as shown in Figure A.1(a). The Chosen Egde will be imposed by a given displacement, $z0$, while the Chosen Point will be used to measure the generated stress. Because of the deformation, there is a restoring force pushing the beam back to the original position. As a consequence, a positive stress in the $z-$direction is created at the Chosen Point. Then, a negative potential VE is set on the surface of the electrode and a positive voltage VM is set on the surfaces of the mirror and the beams. Because of opposite potentials, there is a force of attraction between the mirror, the beams and the electrode. This force will eliminate the restoring force and also the $z-$stress when VM is increased, as shown in Figure A.2(b). The value of VM vanishing the $z-$stress at the Chosen Point is the requirement of equilibrium.

To find VM, an Matlab algorithm using loops **for** and **while** is employed. A simple solution is increasing VM by a certain value and then measuring the z-stress at the Chosen Point in each loop until the stress changes sign from positive to negative. Then, the loop is stopped and VM is returned. Another solution is adding a big amount of potential such as $10\ V$ to VM if it makes positive $z-$stress at the Chosen Point; and in contrast, the added potential will be a half of the previous VM and the current VM if it makes negative $z-$stress. This process is continued until the difference between the two VM that make positive and negative $z-$stress is less than $1\ V$. For instance, Figure A.2(b) shows the convergence of VM regarding to $z0 = 8\mu m$.

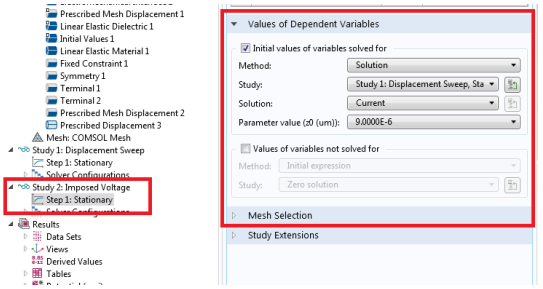
The displacement is given as a range from a position which is close to the original position of the mirror and to over one third of the gap between the two conductors. Solving these problems will return a set of VM in which the Pull-In voltage is the maximum. For example, Figure A.2(c) shows the Pull-In voltage of $poly = 500\ nm$ and $msux = 40\ \mu m$.



(a) The Chosen Point and the Chosen Edge.

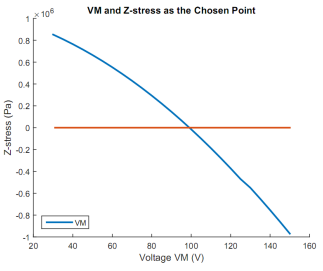


(b) Setting for Study 1: Displacement Sweep.

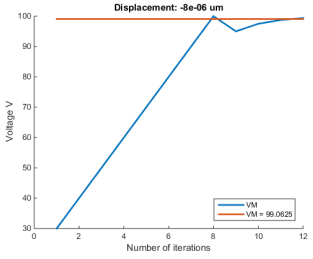


(c) Setting for Study 2: Imposed Voltage.

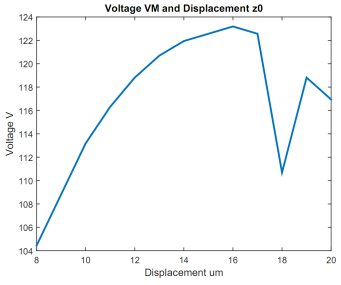
Figure A.1: Model geometry of half micro-mirror cell with symmetry on the $zx-$plane and COMSOL Study Setting.



(a) Plot of VM versus $z-$stress at the Chosen Point at displacement -8 μm.



(b) The convergence of VM at $z0 = 8\ \mu m$ in the fist simulation with the beam thickness is $500\ nm$.



(c) Plot of VM versus $z0$ in the case of $poly = 500\ nm$ and $msux = 40\ \mu m$.

Figure A.2: Numerrical results.

A.2.1.2/ Modeling Instructions

We assume that the geometry, the material and the boundary conditions are already built. In this model, we first define a study, Study 1: Displacement Sweep, which simulate mechanical behavior only, i.e. we do not set any potential on the mirror or on the electrode. Then, we define another study, Study 2: Imposed Voltage, which uses Study 1 as initial values.

PARAMETER

1 In the Model Builder window, expand the Global > Definitions node, then click Parameters. In the Settings window forParameters, locate the Parameters section. In the table, input the following parameter:

Name	Expression	Value	Description
$z0$	$2[um]$	$2.0000E-6m$	Imposed displacement on Chosen Edge

DEFINITIONS

1 On the Definitions toolbar, click Explicit. In the Setting window for Explicit, from the Geometric entity level, chose Point. and enter. Select Point 141 only. Rename it Chosen Point

2 Add another Explicit for Edge. Select Edge 231 only and rename it Chosen Edge.

ELECTROMECHANICS

1 Add boundaries like what have done in Section A.1.

2 In the Physics toolbar, click Egdes and choose Prescribed Displacement. From the Selection list, choose Chosen Edge. Locate Prescribed Displacement section, in the Prescribed in z Direction text field, type z0.

STUDY 1

1 To add a new study, on the Model toolbar, click Add Study to open the Add Study window. Then go to the Add Study window, find the Studies subsection. In the Select study tree, select Preset Studies > Stationary. Then, click Add Study in the window toolbar.

2 In the Model Builder window, under Study 1 click Step 1: Stationary. Then, in the Settings window for Stationary, click to expand the Study extensions section. Locate the Study Extensions section. Select the Auxiliary sweep check box and click Add. Select $z0$ and define it as a range from $1\ \mu m$ to $24\ \mu m$ by a small step of $1\ \mu m$ by clicking Range, in the Range dialog box, type 1 in the Start text field, in the Step text field, type 1 and in the Stop text field, type 24, in the Unit text field type um, then click Replace.

3 Selecting the option Yes from the Reuse solution for previous step list to always use the converged solution from the previous step. This option improves the convergence of the problem. Because the convergence of non-linear problem is dependent upon the starting point, the starting point should be close to the solution otherwise the solver may go to the wrong direction which is far from the real solution and thus lose the convergence.

4 In the Model Builder window, right-click Study 1 and choose Rename. In the Rename Study dialog box, type Displacement Sweep in the New label text field. Then, click OK.

5 On the Study toolbar, click Compute.

STUDY 2

1 Repeat the same process as before to add a new stationary study.

2 Locate the Settings window for Stationary, click to expand the Values of Dependent Variables section and select the Initial values of variables solved for check box. We are now able to specify the initial values of the dependent variables that we want to solve for. For example, if we want to solve a problem which has displacement set at $10\ \mu m$, we can chose the solution of $9\ \mu m$ from the drop list Parameter value of the Study 1 as an initial

value.

3 Provide the potentials on the mirror part and on the electrode part.

4 Rename the study to Imposed Voltage.

5 Click Compute.

The setting of Study 1 and Study 2 should look like Figures A.1(b) and A.1(c).

A.2.2/ COMSOL LIVELINK WITH MATLAB

In this subsection, we introduce a Matlab code that controls COMSOL and create loops to find appropriate VM.

1 Create a Matlab variable, called **model**, which links Matlab and COMSOL:

```
model = ModelUtil.model ('Model2');
```

2 Create a Matlab array, named **z0**, which is the imposed displacements on the **Chose Edge**, a Matlab array, named **Invalue**, which is a set of selection of initial values, a Matlab variable, named **VE**, which is the negative potential setting on the upper surface of the Electrode, a Matlab empty array, called **FullResult**, to store the all the results:

```
z0 = [8e-6:1e-6:24e-6]; InValue = [7:1:23]; VE = 30;
   FullResult = [];
```

3 Update the parameter VE in COMSOL by the Matlab variable **VE**:

```
model.param.set('VE',VE);
```

4 Create the first **for**-loop to set the COMSOL parameter z0 by Matlab array **z0(i)**

```
for i=1:size(z0,2)
        model.param.set('z0',z0(i));
```

5 Update the initial values during the loop:

```
model.study('std2').feature('stat').set('solnum',
    num2str(InValue(i)));
```

where std2 is COMSOL tag name for the Study 2: Imposed Voltage.

6 Create a Matlab variable, named **VM**, which is the positive voltage imposed on the surface of the mirror and the beam, a Matlab variable, named **VPlus**, which will store the value of **VM** if it causes positive z-stress on the **Chosen Point**. In contrast, create a Matlab variable, named **VMinus**, which will store the value of **VM** if it causes negative z-stress on the **Chosen Point**. All of them may be replaced after each loop.

```
VM = 1; VPlus = VM; VMinus = 0;
```

7 Create the second loop, update VM and execute the second study:

```
while(abs(VPlus-VMinus) >= 1)
        model.param.set('VM',VM);
        model.study('std2').run;
```

8 Create a Matlab structure, named **P0**, which stores the evaluation of the **Chosen Point** such as z-stress, position, unit, etc:

```
P0 = mpheval(model,{'emi.sz'},'dataset','
   dset2','edim',0,'Selection','sel1');
```

where emi.sz, dataset, dset2, edim and sel1 are COMSOL tag names representing the z-stress, the data set of a solution, data set of the Study 2, dimension and its given value, (in this case, it is point) and the tag name of the Chosen Point.

9 Add results to **FullResult**

```
FullResult = [FullResult;z0(i),VE,VM,P0.
   d1,P0.p(1),P0.p(2),P0.p(3)];
```

where P0.d1 is data of emi.sz and P0.p(1), P0.p(2), P0.p(3) are x,y,z-coordinates.

10 The increasing step of **VM** is depend on which sign of z-stress it makes at the Chosen Point. If it is negative, the step will be one half of the difference between the current **VM** and the previous **VM**. On the other hand, if the stress is positive and if we have the value that make negative stress as the Chosen Point (**VMinus** $\neq 0$), the step will be one half of the difference between them, otherwise the step is set by 10. The breaking condition of the **while** loop is **abs(VPlus-VMinus)**$\leq$**1** which means both types of **VM** should not be different than **1 V**.

11 If the **while** loop finishes, the appropriate **VM** for balancing the system as the displacement **z0(i)** is successfully returned. The loop **for** will continue the process with **z0(i+1)** until it reach the end of the **z0** array.

```
                if (P0.d1 < 0)
                        VMinus = VM; VM = VPlus + abs(
                           VPlus-VMinus)/2;
                else
                        VPlus = VM;
                        if (VMinus != 0)
                                VM = VPlus + abs(VPlus-
                                   VMinus)/2;
                        else
                                VM = VM + 10;
                        end
                end
        end
end
```

A.3/ MODEL 2

The Model Definition and Result are already discussed in Section 2.4.

A.3.1/ MODEL INSTRUCTIONS

We use the same instruction as COMSOL's example model, *Pull-In Voltage for a Biased Resonator-3D*, which can be found its library.

PARAMETER

1 In the Model Builder window, expand the Global > Definitions node, then click Parameters. In the Settings window forParameters, locate the **Parameters section**. In the table, input the following parameter:

Name	Expression	Value	Description
$zset$	$51.5[um]$	$5.15E-5m$	Set point's z-coordinates

DEFINITION

1 On the Definitions toolbar, click Component Couplings and choose Integration. In the Settings window for Integration, locate the Source Selection section. From the Geometric entity level list, choose Point. Select Point 141 only.

ELECTROMECHANICS

1 Set boundaries like what have been done in Section A.1.

2 In the boundary node VE, locate Electric Potential section, in the V0 text field, type VESP.

3 In the boundary node VM, locate Electric Potential section, in the V0 text field, type 1.

4 In the Model Builder window, expand the Component 1 (comp1), right-click Electromechanics (emi) node

5 In the Global, choose Global Equations 1. In the Settings window for Global Equations, locate the Global Equations section. In the table, enter the following equation:

Name	f(u, ut, utt, t)(1)	Initialvalue(u_0)(1)	Initialvalue(u_t0)(1/s)	Description
$VESP$	$intop1(z) - zset$	0	0	

Locate the Units section, in the Dependent variable quantily list, choose Electric Poatential (V).

MESH

The mesh is built manually. After meshing each part, right click Mesh and choose Statistics. In the Setting window for Mesh, locate Geometric entity level, in the selection list, choose Domain. Adding meshed domain and make sure that Minimum element quality is higher than 0.1.

STUDY

1 Add a Stationary study.

2 Click to expand Study 1 > Step 1: Stationary, in the Settings window for Stationary, expand the Mesh Selection section, make sure that the correct mesh is selected.

3 Expand the Study extensions section, select the Auxiliary sweep check box. Click Add, select zset from selection list of parameters. Click Range. In the Range dialog box, type 51.5 in the Start text field. In the Step text field, type -1. In the Stop text field, type 33. Click Replace.

4 On the Study toolbar, click Show Default Solver. In the Model Builder window, expand the Solution node, then click Dependent Variables 1. In the Settings window for Dependent Variables, locate the General section. In the Defined by study step list, choose User defined. In the Model Builder window, expand the Study 1 > Solver Configurations > Solution > Dependent Variables 1 node, then click Spatial coordinates (Material) (comp1.xyz). In the Settings window for Field, locate the Scaling section. From the Method list, choose Manual. In the Scale text field, type 1e-5. Continue to add 1e-5 as a scale factor for Displacement field (Material) (compl.u), 100 for Electric potential (comp1.V) and 100 for State variable VESP (compl.ODE1).

5 Right-click Model Builder window, expand the Study 1 > Solver Configurations > Solution 1 > Stationary Solver 1 node and choose Fully Coupled.In the Settings window for Fully Coupled, locate the General section. From the Linear solver list, choose Direct. Click to expand the Method and termination section. Locate the Method and Termination section. From the Nonlinear method list, choose Automatic highly nonlinear (Newton). Locate Maximum number of interations section, replace 25 by 10000.

6 Click Stationary Solver 1, in the Setting window for Stationary Solver 1, locate Linearity section, from selection list, choose Nonlinear.

7 Rename the study to Pull In then click Compute.

RESULTS

1 On the Model toolbar, click Add Plot Group and choose 1D Plot Group. In the Settings window for 1D Plot Group, locate the Data section. From the Data set list, choose Pull In/Solution 1. On the 1D plot group toolbar, click Global. In the Settings window for Global, click Replace Expression in the upper-right corner of the y-axis data section. From the selection list, choose Model > Component 1 > Electromechanics > VESP - State variable VESP. In the table, enter the following expression:

Expression	Unit	Description
$VESP - VM$	V	State variable VESP

2 Locate the x-Axis Data section. From the Parameter list, choose Expression, in the Expression text field, type zset.

3 Rename it to Pull-In Plot and click Plot.

A.4/ CONTACT PROBLEM IN VACUUM

A.4.1/ MODEL DEFINITION

When the applied voltage is greater than the pull-in voltage the mirror pulls down onto the base. The landing beams are going to contact with the landing pads and the mirror is going to contact with the stopper beam. The contact phenomena are handled by an approximate penalty or barrier method, nonlinear springs are used to represent the surfaces of the landing pads and the stopper beam. When the mirror is away from these surfaces these springs have low stiffness and consequently have a negligible influence on the deformation of the mirror. As the gaps are reduced and approaches a predefined distance the springs become much stiffer and resists further closure. The contact forces

Fc are given by:

$$F_c = \begin{cases} t_n - e_n g & \text{if } g < 0 \\ t_n + \exp\left(-\frac{e_n}{t_n} g\right) & \text{if } g \geq 0 \end{cases}$$

where t_n is the input estimate of the contact force, e_n is the penalty stiffness, g is the gap.

Figure 2.12 shows the two-dimensional geometry of the upper part of a Micro-Mirror.

A.4.2/ MODEL INSTRUCTION

We follow the instruction written in *Pull-in of an RF MEMS Switch (Application ID: 16379)* which can be found in COMSOL's library or on its website *www.comsol.fr/model*.

PARAMETER

1 In the Model Builder window, expand the Global > Definition node, then click Parameters. In the Settings window for Parameters, locate the Parameters Section. In the table, add the following table:

Name	Expression	Description
g	38[um]	Gap between Landing Beam and Landing Pad + Insulator height
insheight	5[um]	Insulator height
airheight	g-insheight	Gap between Landing Beam and Landing Pad
sb_height	poly+ox	Gap between Mirror and Stopper Beam

DEFINITIONS

1 On the Home toolbar, click Variable and choose Local Variables. In the Settings window for Variables, locate the Variables section. In the table, enter the following settings:

Name	Expression	Description
V_s	1260	Stop Voltage
V_m	1e7*(t/1[s])	Magnitude of Voltage
Va	((V_m<=V_s)*V_m+(V_m>V_s)*V_s)[V]	Increasing Voltage
gap	airheight+v	Gap between Landing Beam and Landing Pad
gap_sb	sb_height+v	Gap between Mirror and Stopper Beam
contactpressure_sb	(gap_sb<=0)*(tn-en*gap_sb)+(gap_sb>0)*tn*exp(-gap_sb*en/tn)	Fc on Mirror
contactpressure	(gap<=0)*(tn-en*gap)+(gap>0)*tn*exp(-gap*en/tn)	Fc on Landing Beam

In this table, t is time, V_m is defined by removing the unit of t and scaling it by 1e7, Va is increased with time when it is lower than V_s until it reaches V_s and keeps this value.

2 On the Home toolbar, click Functions and choose Local > Step. In the Settings window for Step, locate the Parameters section. In the location text field, type $1.05 * insheight$. Click to expand the Smoothing section. In the Size of transition zone text field, type $0.05 * insheight$.

3 On the Definitions toolbar, click Explicit. In the Model Builder window, right-click Explicit 1 and choose Rename. In the Rename Explicit dialog box, type Head Point b1 in the New Label text field. Click Ok. In the Settings Window for Explicit, locate Geometric entity level options and choose Point. Click Paste Selection and type 20.

4 Add another Point Explicit Selection, type 12 and rename Head Point m1.

MATERIAL

1 Add correct material for the Mirror and the Beam.

2 In the Model Builder window, under Component 1 (comp1) > Materials right click Materials and choose Blank Material. In the Settings window for Material, locate the Geometric Entity Selection section, click Paste Selections and type 1. Click to expand the Material properties section. Locate the Material Properties section. From the Material type list, choose Nonsolid. The Relative Permittivity of the landing pad is 11.7 since it is Silicon while in vacuum it is 1 by definition. Locate the Material Contents section. In the table, enter the following setting:

Property	Name	Value
Relative permittivity	epsilonr	11.7-step1(y)*10.7

The expression takes 1 when the landing beam is far away from the base and takes 11.7 when the landing beam comes close to the insulator. In the Setting window for Material, locate Geometric Entity Selection choose Domain, click Paste Selection and type 1. Right click Blank Material, choose Rename and type Vacuum and Insulator.

ELECTROMECHANICS (EMI)

1 On the Physics toolbar, click Domains and choose Linear Elastic Material. In the Settings window for Linear Elastic Material, locate the Domain Selection section, click Paste Selection and type $2 - 8$.

2 On the Physics toolbar, click Boundaries and choose Fixed Constraint. Select Boundary 8.

3 On the Physics toolbar, click Boundaries and choose Boundary Load. Select boundary 27. This boundary is surface of the landing beam. In the Settings window for Boundary Load, locate the Force section. Specify the F_A vector as

0	x
contactpressure	y

4 Add another Boundary Load for the stopper beam. Select boundary 15 and enter the F_A vector as

0	x
contactpressure_sb	y

5 On the Physics toolbar, click Boundaries and choose Prescribed Mesh Displacement. Select boundaries 1 and 31. In the Settings window for Prescribed Mesh Displacement,

locate the Boundary Selection section. Locate the Prescribed Mesh Displacement section. Clear the Prescribed y displacement check box.

6 On the Physics toolbar, click Boundaries and choose Terminal. In the Settings window for Teminal, locate the Boundary Selection section. Select boundary 5. Locate the Terminal section. From the Terminal type list, choose voltage. In the V_0 text field, type Va.

7 Add another Terminal, select boundary 2, select Voltage and type 0.

STUDY

1 On the Study toolbar, click Add Study choose Time Dependent. In the Settings window for Time Dependent, locate the Study Settings section. In the Time text field, type range(range(0,1e-7,80e-5)). Locate Time unit, choose s.

2 On the Study toolbar, click Show Default Solver. In the Model Builder window, expand the Solution 1 (sol1) node, then click Time-Dependent Solver 1. In the Settings window for Time-Dependent Solver, click to expand the Time stepping section. Locate the Time Stepping section. From the Method list, choose BDF.

3 Right-click Study 1>Solver Configuration>Solution 1 (sol1)>Time-Dependent Solver 1 and choose Fully Coupled. In the Settings window for Fully Coupled, click to expand the Method and termination section. Locate the Method and Termination section. From the Nonlinear method list, choose Automatic highly nonlinear (Newton).

4 Compute.

RESULTS

1 On the Home toolbar, click Add Plot Group and choose 1D Plot Group. Right-click 1D Plot Group and choose Point Graph, in the Settings window of Point Graph, locate Selection drop-down list and choose Head point b1. Locate Expression, type v. In the Model Builder window, right-click 1D Plot Group 1 and choose Rename. In the Rename 1D Plot Group dialog box, type Displacement at b1 in the New label text field. click Plot.

2 Add another 1D Plot Group for m_1.

A.5/ CONTACT PROBLEM IN AIR

A.5.1/ MODEL DEFINITION

We use both Fluid-Structure Interaction (fsi) interface and Electrostatics (es) interface. The first one is a multiphysics interface combines fluid flow with solid mechanics to capture the interaction between the fluid and the solid structure. The second one creates the electric force to pull down the mirror.

A.5.2/ MODEL INSTRUCTION

We follow the instruction written in *Fluid-Structure Interaction Application (Application ID: 361)* which can be found in COMSOL's library or on its website *www.comsol.fr/model*.

The **PARAMETER, GEOMETRY, MATERIAL** are defined like before.

1 On the Physic toolbar, click Add Physics. In the Add Physics window, navigate into AC/DC node and click Electrostatic (es).

2 In the Settings window of Electrostatics, locate Domain Selection, click Paste Selection and type $1-8$.

3 On Physics toolbar, expand the Boundaries drop list, and select Terminal. In the Setting window of Terminal, add boundary 5 and in the Voltage field, type Va.

4 Add another Terminal for boundary 2 with value 0.

5 On the Physics toolbar, click Add Physics. In the Add Physics widow, expand the Fluid Flow node and select Fluid-Structure Interaction (fsi).

6 In the Settings window of Fluid-Structure Interaction (fsi), locate Domain Selection, click Paste Selection and type $1-8$.

7 Add a Fixed Constraint boundary, enter boundaries $6, 9$.

8 Add a Prescribed Mesh Displacement boundary, enter boundaries $1, 31$.

9 Add a Boundary Load boundary, enter boundary 5. In the Settings window for Boundary Load, locate Fore section, in Load Type drop list, select Load defined as force per unit area. In the F_a text field type the following table:

es.unTex	x
es.unTey	y

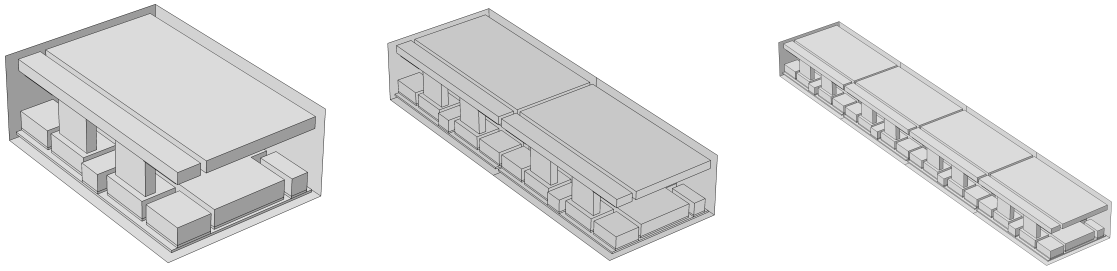
10 Add a Boundary Load for boundary 27 with value is force per unit area $contactpressure$.

11 Add a Boundary Load for boundary 15 with value is force per unit area $contactpressure\_sb$.

12 Click Compute.

A.6/ Asymptotic Model Implementation

A.6.1/ Model Definition



(a) Geometry used to implement ϕ^0.

(b) Geometry used to implement ϕ_b^ϑ.

(c) Geometry used to implement ϕ_b^{interf}.

Figure A.3: Model geometries used for implementation.

To implement the asymptotic model ϕ^0, ϕ_b^ϑ and ϕ_b^{interf} we use five components having geometries varied from one cell up to four cells, Figure A.3. It only needs one cell to simulate the periodic approximation and it only needs two to four cells to simulate the boundary layers and the interface because the contributions are vanished very soon. The implementation process is done by employing the Laplace Equation interface of COMSOL. The plotting process is done by adding ϕ^0 with ϕ_b^ϑ at the ends, adding ϕ^0 with ϕ_b^{interf} at the interface and repeat ϕ^0 for the internal cells.

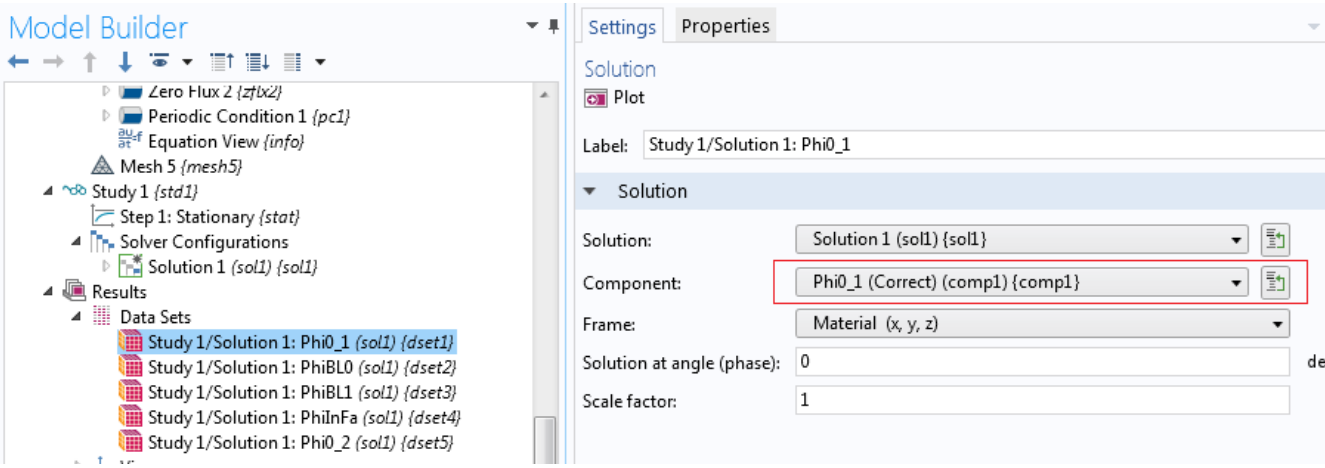


Figure A.4: Five solutions according to five components.

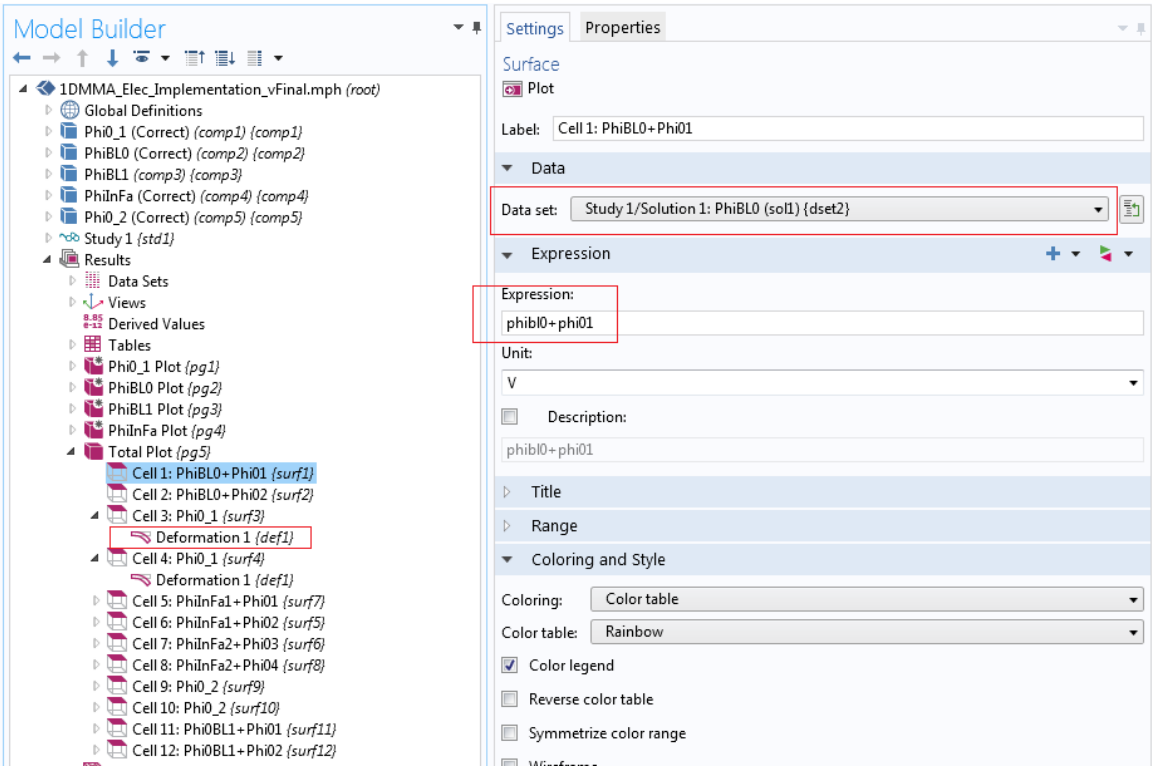


Figure A.5: Plot total array by using 3D Plot Group.

A.6.2/ MODEL INSTRUCTION

PARAMTERS

1 In the Model Builder window, expand the Global > Definitions node, then click Parameters. In the Settings window for Parameters, locate the Parameters section. In the table, enter the following settings:

Name	Expression	Description
VM1	20[V]	Potential on the Mirrors of the left part
VE1	20[V]	Potential on the Electrodes of the left part
VM2	30[V]	Potential on the Mirrors of the right part
VE2	30[V]	Potential on the Electrodes of the right part

COMPONENT 1: Phi0_1

On the Home toolbar, click Add Component. In the Model Builder window, right-click Component 1 select Rename, in the Rename dialog, write Phi0_1. This cell is used for simulating ϕ^0 of the left part of the array.

Geometry

1 In the Model Builder window, right-click Phi0_1 > Geometry and select Import. In the Settings window for Import, click Browse and navigate to Cell_Geo.mphbin.

2 Click Phi0_1 > Geometry, in the Settings window for Geometry, locate Length unit list, and choose μm.

Material

1 On the Materials toolbar, click Add Material. In Add Material window, expand MEMS model, navigate to Semiconductors and select Si - Silicon (single-crystal, isotropic). On the Model Builder toolbar, navigate to Materials > Si - Silicon (single-crystal, isotropic), in the Settings window, click Paste Selection and enter the following domains $2, 4, 6 - 7, 9, 11, 13, 18 - 19, 22, 26, 28$.

2 Add Si - Polycrystalline Silicon to domains $14 - 17, 23 - 24$.

3 Add SiO2 - Silicon oxide to domains $3, 5, 8, 10, 12, 25, 27$.

4 Add Au - Gold to domains $20 - 21$.

5 On Materials toolbar, click Blank Material. Select domain 1. Expand Material Properties, locate Material Type and choose Nonsolid. Expand Material Contents, locate Relative permittivity field and type 1. Rename this material by Vacuum.

Laplace Equation

1 On the Physics toolbar, click Add Physics. In the Add Physics window, expand Mathematics models, and expand Classical PDEs, select Laplace Equation. Rename it by phi0. On the Settings window for Laplace Equation, locate the Units section, in the Dependent variable quantity list, select Electric potential. In the Unit field, type V. Expand the Dependent Variables section, in the field Dependent variable, type $phi01$.

2 On the Physics toolbar, locate Boundary section, expand Boundaries and select Dirichlet Boundary Condition. Rename this node by VM. In the Settings window of Dirichlet Boundary Condition, locate the Boundary Selection and add $27 - 30, 36 - 37, 39, 41, 45 -$

$46, 48, 50, 60-72, 75-77, 80-82, 84-87, 89-92, 95-97, 100-104, 106-122, 126, 128, 139-148, 151-153, 156, 183, 186, 192-194$. Locate Dirichlet Boundary Condition section, in the field r, type VM1.

3 Add another Dirichlet Boundary Condition for boundaries $13-14, 16, 18, 133-134, 136, 138, 158, 160, 163-164, 185$. In the value field, type -VE1. And rename it by VE.

4 Add Periodic Condition and add boundaries $2, 5$.

5 Add Flux/Source and add boundaries $1, 3-4, 195$.

COMPONENT 2: PhiBL0

Add another component and rename it by PhiBL0. This component is used to simulate ϕ_b^0.

Geometry

1 Import the geometry of one cell.

2 Right-click the Geometry node, navigate to Transforms and select Array. In the Settings window for Array, locate the Size section and in the field Size type 2. In the Displacement field, locate y field and type $my + mg$. Click Build All Objects.

Materials

Add the materials like what are described in COMPONENT 1.

Laplace Equations

1 Add two Laplace Equation for ϕ^0 in two cells. Name them phi01 and phi02. In the Settings window, locate Domain Selection section and input 1 and 2 respectively. Both of them have VM1 on the mirrors and -VE1 on the electrodes.

2 Add another Laplace Equation. Rename it by $phibl0$. In the Settings window for Laplace Equation, locate Domain Selection section and input domains $1, 2$.

3 Add two Dirichlet Boundary Condition for the mirrors and electrodes which take value 0.

4 Add the Zero Flux boundary condition for boundaries $1, 3-5, 7-9, 388-389$.

5 Add Flux/Source boundary condition for boundary 2. Expand the Boundary Flux/Source section, in the g field, type $-(nx * phi01x + ny * phi01y + nz * phi01z)$.

COMPONENT 3: PhiBL1

Add another component and rename it by PhiBL1. This component is built to simulate ϕ_b^L. It follows the same approach as **COMPONENT 2** except all Dirichlet Boundary Conditions are imposed with $VM2$ and $-VE2$ and the boundary Flux/Source of ϕ_b^L is typed with $-(nx * phi02x + ny * phi02y + nz * phi02z)$.

COMPONENT 4: PhiInFa

Add another component and rename it by PhiInFa. This component is built to simulate ϕ_b^{interf}. It needs four cells to simulate the contributions of each parts of the array. Each part will occupy two cells.

Geometry

Import the geometry of one single cell and multiply it by 4, using Array node.

Materials

Add correct material like what are done in **COMPONENT 1**.

Laplace Equations

1 Add four Laplace Equations for ϕ^0 with related domain is the elementary cell. Note that the Dirichlet boundary condition should be imposed correctly according to which side the cell is belong to.

2 Add another Laplace Equation and rename it by PhiInFa1. Input domain $1, 2$.

3 Add two Dirichlet Boundary Conditions on the mirrors and electrodes with zero value.

4 Add Zero Flux boundary condition to boundaries $1-5, 7-8, 774-775$.

5 On Physics toolbar, locate Boundary section, expand Boundaries and select Constraint. In the Settings window for Constraint, add boundary 10. Locate the Constraint section in the R field, type $phi02y + phiInFa1y - phi03y - phiInFa2y$.

6 Add another Laplace Equation and rename it by PhiInFa2. Input domain $3, 4$.

7 Add boundary conditions like PhiInFa1 except the the Constraint boundary condition is impose with $phi02 + phiInFa1 - phi03 - phiInFa2$.

COMPONENT 5: Phi0_2

This component is built exactly like **COMPONENT 1** but its Dirichlet Boundary Conditions are imposed with $VM2$ and $-VE2$.

RESULT

1 In the Model Builder window, locate the sub-node Results > Data Sets, click Study 1/Solution 1. In the Setting window for Solution, expand the Component drop list and select Phi0_1 (comp1). Rename this solution by Study 1/Solution 1: Phi0_1.

2 Right-click the solution Study 1/Solution 1: Phi0_1 and choose Duplicate. In the Settings window of this solution, expand the Component drop list and select PhiBL0 (comp2). Rename this solution by Study 1/Solution 1: PhiBL0. Continue duplicate until five solutions are set according to five components, Figure A.4.

3 On the Results toolbar, locate Plot Group section, click 3D Plot Group. In the Settings window for 3D Plot Group, locate Data section and expand Data Set list, choose None. Rename it by Total Plot, Figure A.5.

4 *Plot the boundary layer at 0*: Right-click the Total Plot and choose Surface. In the Settings window for Surface, locate Data section, expand Data set drop list and select Study 1/Solution 1: PhiBL0 (sol1). In the Expression field, type $phibl0 + phi01$, in the Unit file, type V. Rename this plot by Cell 1: PhiBL0+Phi01. Duplicate this Surface and change the Expression field by $phibl0 + phi02$. Rename this plot by Cell 2: PhiBL0+Phi02.

5 *Plot the periodic solution* ϕ^0: Add another Surface plot, in the Data set list, choose Study 1/Solution 1: Phi_1 (sol1). In the Expression field, type $phi0$, and in the Unit field, type V. Rename this plot by Cell 3: Phi0_1. Right-click this plot and select Deformation, in the y component field, type $2 * (mg + my)$. Enable Scale factor field and enter 1.

6 Add another Surface plot for ϕ^0 and rename it by Cell 4: Phi0_1.

7 Continue to plot the interface with Data set list is Study 1/Solution 1: PhiInFa (sol1), the Expression are $phiInFa1 + phi01$, $phiInFa1 + phi02$, $phiInFa2 + phi03$ and $phiInFa2 + phi04$.

The y component field is $4 * (mg + my)$.

8 Plot other two cells for ϕ^0 with Data set is Study 1/Solution 1: Phi0_2 (sol1).

9 Plot the remain boundary layer solution with Data set is Study 1/ Solution 1: PhiBL1 (sol1). The y component is $10 * (200 + 5)[um]$.

10 Click Plot.

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

Abstract:

In this thesis, we contribute to the modeling, simulation and optimization of a new generation of micro mirror arrays designed by the Astrophysics Laboratory of Marseille (LAM). A contribution is also made to the development of MEMSALab a symbolic computation software package designed to assist multiscale model derivation for microsystem arrays. The coupling between the quasi-static nonlinear behavior of a cell of the micro-mirror array and the electrostatic field used for its actuation is simulated. This simulation is then used to study the phenomenon of pull-in and its optimization. Then, a homogenized model for the electrostatic field in the vacuum space surrounding the micro-mirror array has been built using an asymptotic method. The contributions to the development of MEMSALab consist in the introduction of an extension and combination theory that will be used to construct multiscale models based on various asymptotic approaches by a process of successive complexifications. Finally, a complete specification language for using MEMSALab is presented and illustrated by significant examples. In particular, it was used to encode the derivation of a homogenized model that serves as an initial state to the extension-combination method.

Keywords: MEMS Arrays, Micro-Mirrors, Asymptotic Modelling, Simulation, Two-Scale Convergence, Boundary Layers, Term Rewriting, Symbolic Computation, Extension-Combination.

Résumé :

Dans cette thèse, nous contribuons à la modélisation, la simulation et l'optimisation d'une nouvelle génération de matrices de micro-miroirs conçue par le Laboratoire d'Astrophysique de Marseille (LAM). Une contribution est également apportée au développement du logiciel de calcul symbolique MEMSALab qui assistera la construction de modèles multi-échelles pour des matrices de microsystèmes. Le couplage entre le comportement élastique quasi-statique non linéaire d'une cellule de la matrice de micro-miroirs et le champ électrostatique utilisé pour son actionnement a été simulé. Une fois validée, cette simulation a été utilisée pour étudier le phénomène de pull-in ainsi que son optimisation. Ensuite, un modèle homogénéisé du champ électrostatique dans le vide entourant la matrice de micro-miroirs est construit à l'aide d'une méthode asymptotique. Les contributions au développement de MEMSALab consistent en l'introduction d'une théorie d'extension et de combinaison de preuves qui sera utilisée pour construire des modèles asymptotiques par un procédé de complexifications successives. Enfin, un langage spécifique complet pour d'utilisation de MEMSALab est présenté et est illustré par des exemples significatifs. En particulier, il a été utilisé pour coder la construction d'un modèle homogénéisé qui sert d'état initial à la méthode d'extension-combinaison.

Mots-clés : Matrices de MEMS, Micro-Miroirs, Modélisation Asymptotique, Simulation, Convergence à Deux Echelles, Couches Limites, Réécriture, Calcul Symbolique, Méthode d'Extension-Combinaison.

SPIM

■ École doctorale SPIM 1 rue Claude Goudimel F - 25030 Besançon cedex

■ tél. +33 (0)3 81 66 66 02 ■ **ed-spim@univ-fcomte.fr** ■ www.ed-spim.univ-fcomte.fr

UFC
UNIVERSITÉ DE FRANCHE-COMTÉ