Homogenization for Multi Field Modelling

Part I: Theories and FEniCS

Yi Hu

March 28, 2016

Contents

In this student project, Homogenization Method is utilized to investigate behaviour of composites under multiple fields. Many applications can be found in the field of Material Modelling, specifically for coupled problems of composites, e.g. Elastroactive Polymers. As for implementation, a novel Finite Element framework, FEniCS, is investigated. FEniCS is a collection of libraries and modules, which uses Python (or C++) as its interface language. The most important feature about FEniCS is its specialization in code generation with respect to bilinear forms, linear forms and function spaces in the formulation of computational problems, which can translate the mathematical language directly into codes and accelerates the trial process of new models and new computation methods. Moreover, Python make the implementation of a problem even faster, as it is a dynamical language for prototyping. In the framework of FEniCS a Unit Cell module is realized, where the calculation (include homogenized properties) of composites in micro scale under multiple fields could be performed.

1 Introduction

Composites play an important role in engineering, as they combine the advantages of each components in materials. They finds many applications in the field of Civil Engineering, Aerospace Engineering, etc. In order to understand composites thoroughly, experiments and simulations are accomplished. The current work specialized in the simulation of composites.

A multi scale model is conducive in modelling, where the deformation in small scale will be reflected in the corresponding macro scale. Micro structures in composites are essential for their properties. Material behaviours are determined by these micro structures. If an energy function is used to describe the composites, material components and geometry configuration in micro scale governs the material energy function. Under this condition, an appropriate energy function is sought after. However, the exact expression for composites under different loading and with various inclusion geometry is hard to obtain. Hence the consideration shift to acquire this description on the fly, i.e. to embed the result of micro scale simulation into macro scale simulation. From a practical perspective, models in multiple scales would lead to full understanding of composite, which will in return be beneficial for material design. Moreover, in computing scale separation would benefit from various computing techniques such as parrellelization and model reduction. These techniques would possibly boost the efficiency of computation.

Intuitively a full simulation can be realized, where micro structures and different materials would be represented explicitly in the simulated object. This results in a tremendous many degrees of freedom consuming large amount of computation resources. Hence a full simulation accounting for all the micro structures and inclusions is not an efficient way.

In this work Homogenization Method is used to achieve multi scale modelling. The key task of homogenization is to calculate homogenized parameters for macro scale model, which is mainly based on the micro scale result. The formulation of homogenized parameters is carried out through material energy. Using material energy in the formulation is of great importance, as it simplifies the derivation of equilibrium and unifies the calculation of homogenized parameters. In the case of composites in multiple fields, energy formulation will result in relative simple formulation in the homogenized context.

To avoid the lengthy derivation of equations for coupled fields, calculation under FEniCS framework is realized. Its strength lies in the field of coupled problem, which will be seen in the later chapter.

The whole report will fall into two parts. The first part concentrates on the theory basics of simulation, while the second part focus more on the derivation and implementation for the composite simulation under multiple fields. Numerical examples and summary are given in the end of this report.

2 Homogenization Method

The method used in the current work is Homogenization Method. It was proposed in 1970s by Babuska and collaborators [1]. The main purpose of this method is to make use of the scale separation, in order that a reduced PDE for the macroscopic problem is obtained. The macroscopic problem always contains the information from the corresponding micro scale, and it is often represented as "effective parameters". These effective parameters are often calculated in the sense of "averaging procedure" or "homogenization". In order to obtain homogenized quantities a micro scale problem needs to be solved. It could be solved with numerical methods such as Finite Element Method, Finite Volume Method or theoretical results in simple cases. An extensive review can be found in book [3]. A more general framework, the Heterogeneous Multiscale Method (HMM), was proposed by Bjorn Engquist. This method extend the idea of homogenization and introduce generic methodology between macro scale and micro scale. An introductory review could be found in [8].

In this part the basic idea of Homogenization Method is presented with a one dimensional example. Then the application to the 3D elliptic PDE is briefly discussed. Hill Mandel requirements should be fulfilled in the energy conserving problem. Hence they are stated in the end of this part. We confine our discussion here mainly on materials with periodic structures.

2.1 Periodic Structures

Periodicity appears frequently in composites, for instance material with fiber or particle reinforcement. In these materials inclusions are arranged periodically. Concerning about material parameters they could be expressed with periodic functions of coordinates. For example, Young's Modulus can be written in the following form,

$$\mathbb{C}(\mathbf{x} + \mathbf{Y}) = \mathbb{C}(\mathbf{x}). \tag{2.1}$$

2.2 Scale Separation

When a two scale problem is addressed, a corresponding field variable could be expanded as follows,

$$\mathbf{\Phi}^{\epsilon}(\mathbf{x}) = \mathbf{\Phi}^{0}(\mathbf{x}, \mathbf{y}) + \epsilon\mathbf{\Phi}^{1}(\mathbf{x}, \mathbf{y}) + \epsilon\mathbf{\Phi}^{2}(\mathbf{x}, \mathbf{y}) + \cdots, \tag{2.2}$$

In the above formula, $\mathbf{x}$ is the position vector of a point, which is deemed as the *macroscopic* coordinate. $\mathbf{y} = \mathbf{x}/\epsilon$ is a *microscopic* coordinate, which stands for *rapid* oscillation. The physical nature of the right hand side is the decomposition of macro scale dependency and micro scale dependency with respect to a reference cell. The purpose of setting $\mathbf{y} = \mathbf{x}/\epsilon$ is achieving a closed form expressed with the original coordinates. The ratio ϵ means that the micro quantity will vary $1/\epsilon$ faster than macroscopic level. When ϵ goes to 0, functions $\mathbf{\Phi}^{0}(\mathbf{x}, \mathbf{y}), \mathbf{\Phi}^{1}(\mathbf{x}, \mathbf{y}), \cdots$ are smooth in $\mathbf{x}$ and $\mathbf{Y}$-periodic in $\mathbf{y}$.

The characteristic of field variable is illustrated in the following figure.

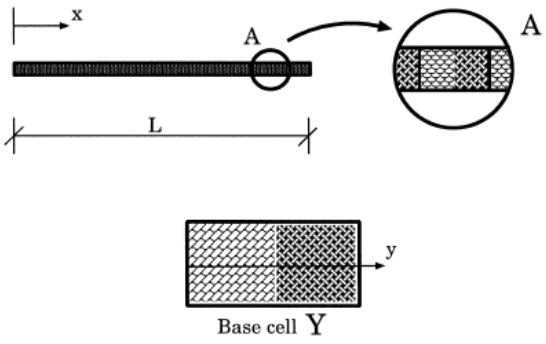

Figure 2.1: Micro and Macro Coordinate of Field Variable [5]

2.3 One Dimensional Problem

Many books and review papers list one dimensional problem, such as [2]. Here we briefly go through one dimensional elasticity problem. More detailed derivation could be referred to [5].

The governing equations, i.e. the equilibrium and Hooke's law are,

$$\begin{cases} \dfrac{\partial \sigma^\epsilon}{\partial x} + \gamma^\epsilon = 0 \\ \sigma^\epsilon = E^\epsilon \dfrac{\partial u^\epsilon}{\partial x}, \end{cases} \tag{2.3}$$

Noting that ϵ in superscript represents its periodic property. γ^ϵ is the body weight of material. If E^ϵ and γ^ϵ are uniform in macro coordinate and only differ inside each cell, then the following relation holds,

$$E^\epsilon(x, x/\epsilon) = E^\epsilon(x/\epsilon) = E(y), \tag{2.4}$$

The relation with respect to body weight is likewise. Regarding the double scale expansion according to (2.2) it follows,

$$\begin{cases} u^\epsilon(x) = u^0(x,y) + \epsilon u^1(x,y) + \epsilon^2 u^2(x,y) + \cdots \\ \sigma^\epsilon(x) = \sigma^0(x,y) + \epsilon \sigma^1(x,y) + \epsilon^2 \sigma^2(x,y) + \cdots, \end{cases} \tag{2.5}$$

After substitution and equating the correspondent terms, we have

$$\begin{cases} 0 = E(y)\left(\dfrac{\partial u^0}{\partial y}\right) \\ \sigma^0 = E(y)\left(\dfrac{\partial u^0}{\partial x} + \dfrac{\partial u^1}{\partial y}\right) \\ \sigma^1 = E(y)\left(\dfrac{\partial u^1}{\partial x} + \dfrac{\partial u^2}{\partial y}\right), \end{cases} \tag{2.6}$$

and

$$\begin{cases} \dfrac{\partial \sigma^0}{\partial y} = 0 \\ \dfrac{\partial \sigma^0}{\partial x} + \dfrac{\partial \sigma^1}{\partial y} + \gamma(y) = 0, \end{cases} \tag{2.7}$$

Simplification of (2.6) and (2.7) yields

$$\sigma^0(x) = \left(Y / \int_Y \frac{\mathrm{d}y}{E(y)}\right) \frac{\mathrm{d}u^0(x)}{\mathrm{d}x}. \tag{2.8}$$

Define the *homogenized modulus of elasticity* as follows,

$$E^H = 1 / \left(\frac{1}{Y}\int_0^Y \frac{\mathrm{d}\eta}{E(\eta)}\right). \tag{2.9}$$

Then the original problem is transformed to

$$\begin{cases} \sigma^0(x) = E^H \dfrac{\mathrm{d}u^0(x)}{\mathrm{d}x} \\ \dfrac{\mathrm{d}\sigma^0}{\mathrm{d}x} + \bar{\gamma} = 0, \end{cases} \tag{2.10}$$

where $\bar{\gamma} = 1/Y \int_Y \gamma(y)$ is the average of γ inside the cell. From (2.10) the differential equation for displacement holds as

$$\frac{\mathrm{d}^2 u^0(x)}{\mathrm{d}x^2} = -\frac{\bar{\gamma}}{E^H} \tag{2.11}$$

Accounting the boundary conditions on both ends gives the result

$$u(x) = -\frac{\bar{\gamma}}{E^H}\frac{x^2}{2} + \frac{\bar{\gamma}}{E^H} L x$$

2.4 General Elliptical PDE

If a general PDE for three dimensional problem is taken into account, it would be more intricate, as the solution is often sought in the sense of weak form. In this circumstance a homogenized weak form is considered instead of the homogenized differential operator. Then the limit of homogenized weak form should converge to the weak form without homogenization, which is called *G-convergence*, [6]. As for the case of elasticity tensor in the sense of differential operator G-convergence is expressed as

$$\lim_{\epsilon\to 0}\frac{\partial}{\partial x_i}\left[C^{\epsilon}_{ijkl}\frac{\partial u^{\epsilon}_k}{\partial x_l}\right]\to\frac{\partial}{\partial x_i}\left[\bar{C}_{ijkl}\frac{\partial u_k}{\partial x_l}\right] \tag{2.12}$$

A quick overview of the general problem is given in the review [5]. Several key points of the general problem are listed here. With the notation of general elliptical operator using

$$\mathcal{A}^{\epsilon}=\frac{\partial}{\partial x_i}\left(a_{ij}(\mathbf{y})\frac{\partial}{\partial x_j}\right). \tag{2.13}$$

general problem could then be described as,

$$\begin{cases}\mathcal{A}^{\epsilon}\mathbf{u}^{\epsilon}=\mathbf{f} & \text{in } \Omega\\ \mathbf{u}^{\epsilon}=\mathbf{0} & \text{on } \partial\Omega\end{cases} \tag{2.14}$$

Employing the double scale expansion for both the field variable $\mathbf{u}^{\epsilon}$ and the differential operator $\mathcal{A}^{\epsilon}$, namely (notice that chain rule is applied when differentiating)

$$\begin{cases}\mathbf{u}^{\epsilon}(\mathbf{x})=\mathbf{u}^0(\mathbf{x},\mathbf{y})+\epsilon\mathbf{u}^1(\mathbf{x},\mathbf{y})+\epsilon^2\mathbf{u}^2(\mathbf{x},\mathbf{y})+\cdots\\ \mathcal{A}^{\epsilon}=\dfrac{1}{\epsilon^2}\mathcal{A}^1+\dfrac{1}{\epsilon}\mathcal{A}^2+\mathcal{A}^3\end{cases} \tag{2.15}$$

Here $\mathcal{A}^1,\mathcal{A}^2,\mathcal{A}^3$ is defined as follows

$$\mathcal{A}^1=\frac{\partial}{\partial y_i}\left(a_{ij}(\mathbf{y})\frac{\partial}{\partial y_j}\right);\ \mathcal{A}^2=\frac{\partial}{\partial y_i}\left(a_{ij}(\mathbf{y})\frac{\partial}{\partial x_j}\right)+\frac{\partial}{\partial y_i}\left(a_{ij}(\mathbf{y})\frac{\partial}{\partial x_j}\right);\ \mathcal{A}^3=\frac{\partial}{\partial x_i}\left(a_{ij}(\mathbf{y})\frac{\partial}{\partial x_j}\right).$$

Substitution with the above differential operators and comparing with the according terms it follows

$$\begin{cases}\mathcal{A}^1\mathbf{u}^0=\mathbf{0}\\ \mathcal{A}^1\mathbf{u}^1+\mathcal{A}^2\mathbf{u}^0=\mathbf{0}\\ \mathcal{A}^1\mathbf{u}^2+\mathcal{A}^2\mathbf{u}^1+\mathcal{A}^3\mathbf{u}^0=\mathbf{f}.\end{cases} \tag{2.16}$$

Referring [2] it is known that if a $\mathbf{Y}$-periodic function u has a unique solution in terms of $\mathcal{A}^1$ operator, i.e.

$$\mathcal{A}^1\mathbf{u}=\mathbf{F}\quad\text{in reference cell.} \tag{2.17}$$

Then the right hand side of the above equation, $\mathbf{F}$ should satisfy

$$\overline{\mathbf{F}}=\frac{1}{|Y|}\int_Y\mathbf{F}\,\mathrm{d}\mathbf{y}=\mathbf{0}. \tag{2.18}$$

Applying this proposition to (2.16) several times the field variable could be expressed with the following form,

$$\mathbf{u}^1(\mathbf{x},\mathbf{y})=\chi^i(\mathbf{y})\frac{\partial\mathbf{u}(\mathbf{x})}{\partial x_j}+\xi(\mathbf{x}) \tag{2.19}$$

Function $\chi^i(\mathbf{y})$ is the local solution of this problem, which has $\mathbf{Y}$-periodic property. The local problem is

$$\mathcal{A}^1\chi^j(\mathbf{y})=\frac{\partial a_{ij}(\mathbf{y})}{\partial y_i}\quad\text{in reference cell.} \tag{2.20}$$

Hence the macro scale problem (homogenized problem) can be written as

$$\mathcal{A}^H\mathbf{u}=\mathbf{f}, \tag{2.21}$$

with

$$\mathcal{A}^H=a^H_{ij}\frac{\partial^2}{\partial x_i\partial x_j}. \tag{2.22}$$

And the effective coefficients are related with the solution of micro scale problem, i.e.

$$a^H_{ij}=\frac{1}{|Y|}\int_Y\left(a_{ij}(\mathbf{y})+a_{ik}(\mathbf{y})\frac{\partial\chi^j}{\partial y_k}\right)\,\mathrm{d}\mathbf{y} \tag{2.23}$$

2.5 Hill-Mandel Condition

After introducing the general mathematical concepts about homogenization methods, we move to its application in material modelling. In this case a Representative Volume Element (RVE) is always investigated. Homogenization of the coefficients is then obtained through calculation on RVE. As RVE represents a material in the micro scale, the behaviour of RVE should resemble the material in this scale. Therefore the model for micro scale should be able to capture specific features, for instance the continuum mechanical equilibrium of composites in the micro scale. Besides the boundary condition of micro scale model should also be compatible with macro scale. This is the content of Hill-Mandel condition [4].

The Hill-Mandel condition states that the total stress power on the micro scale should be equal to the stress power at relevant point on the macro scale. For small strain, the following equation holds,

$$\langle \boldsymbol{\sigma} \cdot \dot{\boldsymbol{\varepsilon}} \rangle = \langle \boldsymbol{\sigma} \rangle \cdot \langle \dot{\boldsymbol{\varepsilon}} \rangle \,, \tag{2.24}$$

where $\langle \cdot \rangle$ means averaging of the considered variable. In simulations certain boundary conditions are devised to fulfil Hill-Mandel condition automatically.

3 Finite Element Framework FEniCS

FEniCS project was started at University of Chicago and Charmers University of Technology in 2003. The name "FEniCS" consists "FE", "Finite Element" and "CS", "Computational Software". "ni" makes the name sound like "phenix". It is a collection of packages that make the best of each package to realize Computational Mathematical and Modelling. The main interface language of FEniCS is Python, which is a fast language for prototyping, while the most of codes and libraries are implemented in C++, whose efficiency in computing stands out. Various linear algebra backends and parallelization allow FEniCS boost in speed. As for modelling, UFL (Unified Form Language) and FFC (FEniCS Form Compiler) allow the direct translation of mathematical formulation such as linear form and bilinear form into symbolic codes. It can be regarded as the language of modelling, which simplifies the modelling process to a large extent [7].

This part concentrates on the introduction of FEniCS. Firstly the work flow and components of FEniCS will be presented. Then the most simple example is given in order to clarify its efficiency in modelling. At last some key features of FEniCS is listed and discussed, which provides the flexibility in simulations.

3.1 The Structure of FEniCS

The work flow of FEniCS can be described with the following diagram. It is seen that the whole process is decomposed into three major parts. Each part represents one or several functionalities. For form compiler UFL language is involved. With the help of form compiler (FFC), C++ code of the whole model is obtained. Till this point the mathematical derivation, which is the weak form of a PDE in most cases, is converted into pure symbolic efficient C++ codes. The interface between symbolic codes and numerical codes is called UFC (Unified Form-assembly Code). It works as a fixed interface that achieve efficient assembling of finite elements [9].

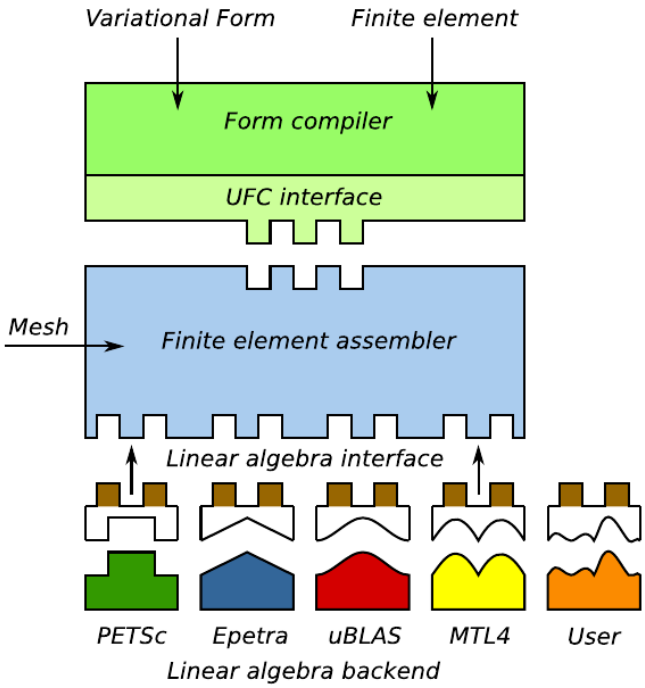

Figure 3.1: Building Blocks of FEniCS [9]

Numerical algorithms enter after the UFC interface. The main package that carries the numerical algorithm is DOLFIN. DOLFIN's task is to wrap functionalities of each components and their interactions. The user interface of DOLFIN is Python, where object oriented programming can be followed in a easy fashion.

Other building blocks such as mshr for geometrical modelling and mesh generation, and FIAT (FInite element Automatic Tabulator) for automatic finite element generation for arbitrary orders. Linear algebra backends work

as add-ons for FEniCS, which provides the possibility of extension. The visualization package in use is VTK, a substitute of the old one, Viper. The output files could be in various formats. Therefore software like ParaView could apply in terms of visualization. The full structure of FEniCS can be viewed in the following diagram.

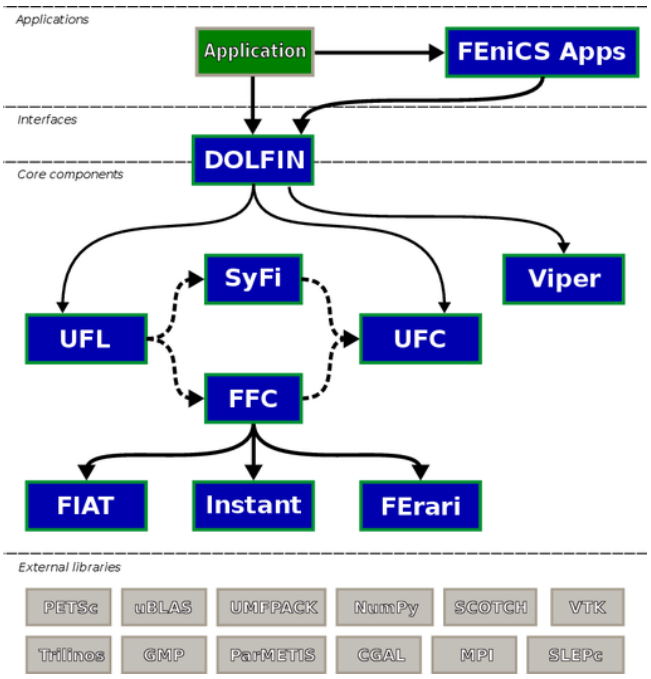

Figure 3.2: Components of FEniCS [9]

3.2 Simple Poisson Problem

An implementation of simple Poisson problem is presented in the official tutorial of FEniCS. This example is reproduced here to give a quick overview of the usage and convince its simplicity in modelling [9].

```
from dolfin import *

# Create mesh and define function space
mesh = UnitSquareMesh(6, 4)
V = FunctionSpace(mesh, 'Lagrange', 1)

# Define boundary conditions
# Value function for boundary condition x[0]->x, x[1]->y
u0 = Expression('1 + x[0]*x[0] + 2*x[1]*x[1]')
# Mark boundary
def u0_boundary(x, on_boundary):
    return on_boundary
# Add Dirichlet boundary condition
bc = DirichletBC(V, u0, u0_boundary)

# Define variational problem
u = TrialFunction(V)
v = TestFunction(V)
f = Constant(-6.0)
# Bilinear Form
a = inner(nabla_grad(u), nabla_grad(v))*dx
# Linear Form
L = f*v*dx

# Compute solution
u = Function(V)
solve(a == L, u, bc)

# Plot solution
plot(u, interactive = True)

# Dump solution to file in VTK format
file = File('poisson.pvd')
file << u
```

As can be immediately seen in the code, the encapsulation of almost every functionality is well preserved in FEniCS. Moreover, the name of functions and classes are in good consistent with the mathematical language. The modelling always begins with defining its geometry and function space. This setting is similar with solving a PDE in weak form, where its domain and function spaces are usually given in condition. Geometry and mesh can also be imported using external files. Various formats are supported, such as `.msh` generated by `gmsh`. One thing to notice is that element type is defined when defining function space. As shown in the example `'Lagrange'`. The order of element is simply 1.

It follows with defining the boundary conditions such as Dirichlet boundary conditions. An `Expression` is a class holding the codes of creating a mathematical function in the domain. It is extensible with C++. Here we use it to define the boundary values. As for imposing this boundary condition, one need to mark the boundary first. Marking other boundary entities such as points, lines, and facets is the same with the provided code structure. `on_boundary` is only a predefined marker that will return `True` when on the boundary.

The mathematical model of this problem is

$$\begin{cases} -\Delta u(\mathbf{x}) = 0 & \text{in } \Omega \\ u(\mathbf{x}) = u_0 & \text{on } \partial\Omega. \end{cases} \tag{3.1}$$

The according weak form can be written as

$$a(u, v) = L(v), \ \forall v \in \hat{V} \tag{3.2}$$

with the bilinear form and linear form defined as

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\mathbf{x}, \ L(v) = \int_\Omega f v \, \mathrm{d}\mathbf{x}.$$

The sought after function is u in function space $V = \left\{v \in H^1(\Omega) : v = u_0 \text{ on } \partial\Omega\right\}$ and the according $\hat{V}$ is defined as $\hat{V} = \left\{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\right\}$. In the code the above mathematical model is conveniently expressed with `a = inner(nabla_grad(u), nabla_grad(v))*dx` and `L = f*v*dx`.

Next is the solving step, where one should redefine a new `Function` to hold the solution. Here overriding the above defined `u` is implemented. Post processing such as visualization of the result follows with `plot`. Output of the result is also standard with the listed commands.

The presentation of the simple Poisson problem serves as an introduction to the functionalities of FEniCS. To sum up, the implementation is in accordance with mathematical notation. Pre and post processing of the computation can be realized using handy functions supported by FEniCS. More useful features are discussed in the following section.

3.3 Relevant Key Features

When the problem becomes more complicated, there are more advanced and handy usage come into play. Here list only a few that appear in the material modelling and multi field problems.

First we explore the function space in FEniCS. There are several kinds of `FunctionSpace` in FEniCS. If a vector or tensor is used, `VectorFunctionSpace` or `TensorFunctionSpace` should be defined. More powerful usage is the `MixedFunctionSpace`, which receives a list of `FunctionSpace` and compose them together into a merged one. It is especially useful when multiple fields are accounted. When concerning about constructing function spaces, various function space families are available. Thanks to the general representation of function spaces arbitrary order of elements and a lot many types are possible. For very specific function space which requires periodicity `constrained_type` keyword is set to be a periodic mapping. This mapping is derived from `SubDomain`. Much caution needs to be devoted into the definition of mapping, which can be observed in the corresponding code for material modelling.

```
# Useful properties and variables for FunctionSpace definition
FS = FunctionSpace(mesh=some_mesh, family='Lagrange', degree=2, constrained_domain=
                                                    some_defined_periodic_mapping)

# Vector and tensor function spaces
VFS = VectorFunctionSpace(mesh, 'Lagrange', 1)
TFS = TensorFunctionSpace(mesh, 'Hermite', 1)

# Merged function Space
MFS = MixedFunctionSpace([FS, VFS, TFS])
```

When it is concerned with multi field modelling, not only merged function but also split functions are required, since in the formulation different terms are calculated with different functions. A mixed function is generated

with a `MixedFunctionSpace` and the corresponding component functions are obtained through `split` without losing dependency to the merged function. In the problem formulation it is often the case that inner product, outer product, derivative, and integral etc. are built. This is achieved easily with the help of UFL operators. Some of the operators such as `nabla_grad()` and `inner()` are already mentioned above. One strength of FEniCS is that this formulation is not restricted in the vector and matrix, but applicable on tensors of higher order. The index notation representation is also valid in the formulation, namely `i,j,k,l`, obeying Einstein's summation convention. After the derivation a stiffness matrix is assembled with `assemble`. The usages are as follows.

```
# Merged function space
MFS = MixedFunctionSpace([FS, VFS, TFS])

# Merged function and split functions
merged_f = Function(MFS)
(f, vf, tf) = split(merged_f)

# Integral
L = f*v*dx

# Derivative and its alternative
L_derivative = derivative(L, f, df)
L_diff = diff(L, f)

# Index notation
i, j, k, l = indices(4)
Energy = 0.5*F[i,j]*C_i_[i,j,k,l]*F[k,l]*dx

# Stiffness matrix, note that test function and trial function should be initiated first
f_test = TestFunction(FS)
f_trial = TrialFunction(FS)
a = f_test*C*t_trial*dx
K = assemble(a)
```

The last usage is particularly useful in composite modelling, i.e. defining different domains. The definition is also not complicated. Subdomains are defining through subclassing of `SubDomain`. Overriding its member method `inside()` gives the condition to distinguish whether it is in the corresponding subdomain or not. In integral the differentials are set accordingly to the subdomain.

```
# Inclusion definition
class Inclusion(SubDomain):
    def inside(self, x, on_boundary):
        d = sqrt((x[0] - 0.5)**2 + (x[1] - 0.5)**2)
        return d<0.25 or near(d,0.25)

# Initiate an instance
circle = Inclusion()

# Set cell domains for integral
domains = CellFunction("size_t", mesh)
domains.set_all(0)
# Mark inclusion
circle.mark(domains,1)

# Integrate accordingly
dx = Measure('dx', domain=mesh, subdomain_data=domains)
Pi = psi_m*dx(0) + psi_i*dx(1)
```

Bibliography

[1] Assyr Abdulle. Numerical homogenization methods. *Springer, Encyclopedia of Applied and Computational Mathematics*, 2013.

[2] Doina Cioranescu and Patrizia Donato. Introduction to homogenization. 2000.

[3] Yalchin Efendiev and Thomas Y Hou. *Multiscale finite element methods: theory and applications*, volume 4. Springer Science & Business Media, 2009.

[4] R Glüge, M Weber, and A Bertram. Comparison of spherical and cubical statistical volume elements with respect to convergence, anisotropy, and localization behavior. *Computational Materials Science*, 63:91–104, 2012.

[5] Behrooz Hassani and Ernest Hinton. A review of homogenization and topology optimization i — homogenization theory for media with periodic structure. *Computers & Structures*, 69(6):707–717, 1998.

[6] Scott J Hollister and Noboru Kikuchi. A comparison of homogenization and standard mechanics analyses for periodic porous composites. *Computational Mechanics*, 10(2):73–95, 1992.

[7] Robert C Kirby and Anders Logg. A compiler for variational forms. *ACM Transactions on Mathematical Software (TOMS)*, 32(3):417–444, 2006.

[8] E Weinan, Bjorn Engquist, Xiantao Li, Weiqing Ren, and Eric Vanden-Eijnden. Heterogeneous multiscale methods: a review. *Commun. Comput. Phys*, 2(3):367–450, 2007.

[9] Garth Wells, Kent-Andre Mardal, and Anders Logg. *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Springer, 2012.