Thermo-mechanical homogenization of composite materials

Shadi Alameddin[1], Felix Fritzen[2]

[1] Data Analytics in Engineering, Institute of Applied Mechanics, University of Stuttgart
[2] Data Analytics in Engineering, Institute of Applied Mechanics, University of Stuttgart

Abstract

This work presents a multiscale simulation framework that will be used for the simulation and experimental validation of eigenstresses in composite materials generated via laser-dispersion. These materials are obtained by adding tungsten carbide particles into the melt pool of a base metal to generate surface coatings. Such coatings are used to boost wear-resistance, more precisely to protect metallic surfaces against abrasion, erosion or corrosion. The coating significantly extends the part's lifetime due to the outstanding material characteristics of the locally produced metal matrix composite (MMC). Eigenstresses, which are the residual stresses left in the MMC material after the coating process, shall be investigated and predicted within the framework of this project and their effect on the lifetime shall be estimated.

Computational homogenization is employed to predict the thermo-mechanical response of heterogeneous structures. In such schemes, the constitutive response of every integration point in the macro scale is obtained by solving a boundary value problem on a representative volume element (RVE) with boundary conditions from the corresponding integration point. Direct numerical simulation (DNS) of RVE response, which can be seen in FE^2 context, is computationally demanding. In contrast, reduced order models (ROM) and data-driven surrogate models provide an appealing and efficient alternative to DNS. In this work, an interface to LS-DYNA is developed to allow for:

1 implementation possibilities of user-defined material routines (UMATs) in Python and C++
2 easy integration of temperature-dependent thermo-mechanical material parameter in LS-DYNA
3 direct integration of data-driven models and ROM written in Python and C++
4 the usage LS-DYNA or self-written software as the DNS solver to compute RVE responses

Simple examples and an open-source code are provided to allow a straightforward deployment of UMATs and two-scale simulations in LS-DYNA.

1 Introduction

Composite materials are widely used due to their improved mechanical and thermal properties and innovative manufacturing possibilities in comparison to their individual phases. However, processes that ensure reliable designs such as finite element analysis (FEA) should account for materials' microstructural characteristics as they drive the overall structural response. These characteristics include the phase volume fraction and the spatial orientation of the phases, to name but a few. In order to account for the heterogeneity, the computational homogenization method (e.g., FE^2 [1,2]) has shown an accurate prediction of the actual nonlinear behavior of composites. Nevertheless, the drawback of this technique is its computational cost driven by two nested high-fidelity models: micro and macro ones, as depicted in Fig. 1. This means that the effective macro strain $\overline{\boldsymbol{\varepsilon}}$ is computed for every material point, or integration point in discrete settings, of this model and used to set the boundary conditions for the representative volume element (RVE) model which in turn updates the effective stress $\overline{\boldsymbol{\sigma}}$ and the algorithmic stiffness $\overline{\mathbb{C}}$. At this level, model order reduction techniques and surrogate models can be introduced to decrease the computational demands for solving the homogenization problem, resulting, e.g. in the FE^{2R} method [3] or in hybrid ROM+data-driven surrogates [4].

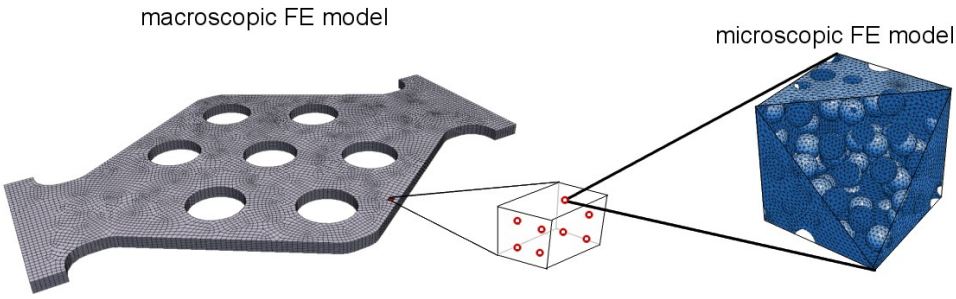

Fig 1. Illustration of the implementation of the FE² framework

The objective of this work is to propose an efficient scheme and implementation to ease testing of multiscale models within LS-DYNA. An overview of computational homogenization is provided in section 2. Then details on the implementation into LS-DYNA are discussed in section 3. A set of practical examples is illustrated in section 4.

2 Computational homogenization

This study is confined to a first-order homogenisation of thermo-mechanical problems in an infinitesimal strain framework. This class of homogenisation techniques assume that the micro-structural length scale l is separated from the structural (macro-structural) length scale L, i.e. $l \ll L$. The aforementioned assumption implies that the fluctuations of the macroscopic fields are negligible with respect to the RVE. The two scales are coupled via the scale transition relations

$$\overline{\boldsymbol{\varepsilon}} = \frac{1}{|\Omega|}\int_{\partial\Omega} \mathrm{sym}(\boldsymbol{u}\otimes\boldsymbol{n})\mathrm{d}A = \langle\boldsymbol{\varepsilon}\rangle, \quad \overline{\boldsymbol{\sigma}} = \frac{1}{|\Omega|}\int_{\partial\Omega} \mathrm{sym}(\boldsymbol{t}\otimes\boldsymbol{x})\mathrm{d}A = \langle\boldsymbol{\sigma}\rangle \quad \text{(Eq 2.1)}$$

where the macroscopic fields are overlined and the volume averaging operator $\langle\bullet\rangle$ is defined via

$$\overline{\bullet} = \langle\bullet\rangle = \langle\bullet\rangle_{\Omega} = \frac{1}{|\Omega|}\int_{\Omega} \bullet \, \mathrm{d}V. \quad \text{(Eq 2.2)}$$

Here $\boldsymbol{u}$ is the displacement field, $\boldsymbol{n}$ represents the outward unit normal and the traction vector is $\boldsymbol{t} = \boldsymbol{\sigma}\cdot\boldsymbol{n}$. In the structural scale, balance of linear and angular momentum should hold

$$\begin{aligned} \mathrm{div}(\overline{\boldsymbol{\sigma}}) + \overline{\boldsymbol{b}} = \boldsymbol{0}, \quad \overline{\boldsymbol{\sigma}} = \overline{\boldsymbol{\sigma}}^{\top} \quad & \text{in } \overline{\Omega} \\ \overline{\boldsymbol{u}} = \overline{\boldsymbol{u}}_* \quad & \text{on } \partial\overline{\Omega}_u \\ \overline{\boldsymbol{t}} = \overline{\boldsymbol{t}}_* \quad & \text{on } \partial\overline{\Omega}_t = \partial\overline{\Omega}\backslash\partial\overline{\Omega}_u \end{aligned} \quad \text{(Eq 2.3)}$$

with $\overline{\boldsymbol{b}}$ denoting the volumetric force density and $\overline{\boldsymbol{u}}_*$, $\overline{\boldsymbol{t}}_*$ specifying the displacement and traction boundary conditions, respectively. The macroscopic stress $\overline{\boldsymbol{\sigma}}$ depends on the microscopic stress field $\boldsymbol{\sigma}$ through the averaging relation (2.2). The field $\boldsymbol{\sigma}$ is the solution to the microscopic boundary value problem

$$\mathrm{div}(\boldsymbol{\sigma}) = \boldsymbol{0}, \quad \boldsymbol{\sigma} = \boldsymbol{\sigma}^{\top}, \quad \langle\boldsymbol{\varepsilon}\rangle = \overline{\boldsymbol{\varepsilon}}. \quad \text{(Eq 2.4)}$$

The boundary conditions of this microscopic boundary value problem are taken in accordance with the Hill-Mandel condition [5]: the total mechanical power should be identical on both length scales, i.e.

$$\langle\boldsymbol{\sigma}\cdot\dot{\boldsymbol{\varepsilon}}\rangle = \langle\boldsymbol{\sigma}\rangle\cdot\langle\dot{\boldsymbol{\varepsilon}}\rangle = \overline{\boldsymbol{\sigma}}\cdot\dot{\overline{\boldsymbol{\varepsilon}}}. \quad \text{(Eq 2.5)}$$

These conditions are, for instance, satisfied by periodic displacement fluctuations of the form

$$\boldsymbol{u}(\boldsymbol{x}) = \overline{\boldsymbol{u}}(\overline{\boldsymbol{x}}) + \overline{\boldsymbol{\varepsilon}}\boldsymbol{x} + \widetilde{\boldsymbol{u}}(\boldsymbol{x}) \quad \text{(Eq 2.6)}$$

with $\boldsymbol{x}\in\Omega, \overline{\boldsymbol{x}}\in\overline{\Omega}$ and the periodic fluctuation field $\widetilde{\boldsymbol{u}}(\boldsymbol{x}_+) = \widetilde{\boldsymbol{u}}(\boldsymbol{x}_-)$, i.e. points ($\boldsymbol{x}_+$ and $\boldsymbol{x}_-$)on opposing faces of the RVE have the same displacement fluctuations. The quantity $\overline{\boldsymbol{u}}$ is constant over the RVE and the symmetric gradient of $\widetilde{\boldsymbol{u}}$ vanishes in average. Hence, by introducing strain fluctuations $\widetilde{\boldsymbol{\varepsilon}} = \nabla^{\mathrm{sym}}\widetilde{\boldsymbol{u}}$, the strain field can be additively split into an average strain $\boldsymbol{\varepsilon}$ and the fluctuations $\widetilde{\boldsymbol{\varepsilon}}$ via

$$\boldsymbol{\varepsilon}(\boldsymbol{x}) = \overline{\boldsymbol{\varepsilon}} + \widetilde{\boldsymbol{\varepsilon}}(\boldsymbol{x}) \quad \text{(Eq 2.7)}$$

without violating the boundary conditions.

The thermal problem is similar to the mechanical one but the solution variable is temperature θ instead of displacement $\boldsymbol{u}$ and consequently, the heat flux $\boldsymbol{q}$ which is linearly related to the temperature gradient $\boldsymbol{g} = \nabla\theta$ by Fourier's law $\boldsymbol{q} = -\boldsymbol{\kappa}\boldsymbol{g}$. Thermal conductivity is denoted by $\boldsymbol{\kappa}$.With the introduction of temperature fluctuations $\widetilde{\boldsymbol{g}} = \nabla\widetilde{\theta}$, temperature gradient may be written as

$$\boldsymbol{g}(\boldsymbol{x}) = \overline{\boldsymbol{g}} + \widetilde{\boldsymbol{g}}(\boldsymbol{x}). \quad \text{(Eq 2.8)}$$

If thermo-elastic coupling is not taken into account, the effective constitutive response is written as

$$\overline{\boldsymbol{\sigma}} = \overline{\mathbb{C}}\,\overline{\boldsymbol{\varepsilon}}, \overline{\boldsymbol{q}} = -\overline{\boldsymbol{\kappa}}\,\overline{\boldsymbol{g}}. \quad \text{(Eq 2.9)}$$

However, for coupled analysis, the effective stress response changes to

$$\overline{\boldsymbol{\sigma}} = \overline{\mathbb{C}}\,\left(\overline{\boldsymbol{\varepsilon}} - \overline{\boldsymbol{\varepsilon}}_{\mathrm{th}}(\theta)\right), \overline{\boldsymbol{\varepsilon}}_{\mathrm{th}}(\theta) = \int_{\theta_0}^{\theta} \overline{\boldsymbol{\alpha}}(\hat{\theta})\,\mathrm{d}\hat{\theta} \quad \text{(Eq 2.10)}$$

where θ_0 is the initial temperature and $\overline{\boldsymbol{\alpha}}$ is the effective thermal expansion coefficient. Further details on how to obtain effective thermo-elastic properties are provided in [6,7]. For completeness, the effective heat capacity is also obtained via (2.2).

3 Implementation

This section includes an overview of the provided extensions and illustrates how the workflow is realized.

In context of thermo-elastic homogenization in LS-DYNA, the following remarks were summarized:
- Temperature dependent material properties can be defined by specifying a minimum of two and a maximum of eight data points, i.e. material properties are defined in a tabulated format with maximum of eight data points.
- RVE mechanical response can be computed with the help of **`*RVE_ANALYSIS_FEM`** keyword but to the best of the authors knowledge, RVE thermal response is not realizable yet in LS-DYNA.
- There is no straightforward way to interact with machine learning models.

After an evaluation of the remarks above, an extension to LS-DYNA object version was proposed with the following objectives:
- Implement temperature dependent material parameters as functions of temperature leading to accurate capture of these parameters without loss of accuracy due to discretization.
- Compute thermo-mechanical RVE response in LS-DYNA.
- Provide a user-friendly interface to link machine learning models to LS-DYNA.
- Tutorial oriented repository to allow for further extensions by the community.

The proposed extension is designed to be flexible and extendable. Hence, it is comprises a combination of Fortran, C++ and Python scripts. An overview of the workflow is provided below while use cases and examples are presnted in the next section.

In order for interested readers to have a smooth start in using the provided code and extending it, all manipulations of LS-DYNA object version files are stored as patch files. Hence, it is straightforward to see which places have been edited and results can be replicated on different machines or different operation systems even though the usage of different operation systems have not been tested in the scope of this work. Note that it is expected that the user has access to LS-DYNA object version, i.e. the version to which user material routines may be linked. More details of the code usage are provided in the `readme` file and in the header of each file.

1.1 Usage of LS-DYNA to compute the effective thermo-mechanical RVE response

In periodic homogenization framework, the microscopic boundary value problem is solved for given boundary conditions from the macroscopic scale as in (2.6). Then after obtaining the stess and heat

flux fields, they are averaged via the discrete version of (2.2), i.e. by taking the corresponding discrete values at integration point level and multiplying by respective integration weights.

The actual implementation is called from `dyn21umat.f` and included in `rve_2d.f` and `rve_3d.f`. The keyword "**`*RVE_ANALYSIS_FEM`**" from [8] was used in the input file to to apply periodic displacement boundary conditions to the RVE. These boundary conditions are intrinsically realized with the help of the multipoint constrained keyword "**`*CONSTRAINED_MULTIPLE_GLOBAL`**". Note that it is also possible to use "**`*INCLUDE_UNITCELL`**" to apply periodic boundary conditions.

Note that to the best of the authors knowledge, periodic thermal fluctuation boundary conditions are not realizable in LS-DYNA, yet. Hence, in the thermal simulation, zero temperature fluctuations are assumed which is equivalent to a Voigt-like assumption for the thermal microscopic problem. This is known to overestimate the actual conductivity and will be replaced by a computational homogenization procedure incorporating periodic temperature fluctuations at a later stage.

1.2 Implementation of new user material subroutines

The native implmentation of user materials within LS-DYNA is usually realized in Fortran 77. However, Python and C/C++ code is often easier to develop. This holds particularly true in the context of linking external tools and packages, e.g., machine learning frameworks like Google's TensorFlow or Facebook's pyTorch.

Python was chosen at first due to its code readability that can enhance code development cycle and its wide usage for machine learning tasks. As a side outcome, C++ extension was developed due to the fact that enriching Fortran with Python was achieved with the help of Python/C API.

Below are two examples/prototypes of how to call C++ from Fortran then how to call Python from C++. For the sake of brevity, functions are presented in a general framework without any relation to FEM or LS-DYNA. However, scripts that are actually called from LS-DYNA can be found in `umat` directory.

In the first example, a C++ function is defined. This function has to be defined with a linkage-specification, `extern "C"`, for example as in code block 1.

```
#include <iostream>

extern "C" {
void cpp_function_(const double* argument_1, double* argument_2, double& argument_3);
}

void cpp_function_(const double* argument_1, double* argument_2, double& argument_3)
{
   std::cout << "cpp_function was called" << std::endl;
}
```

Code block 1.

With `extern "C"`, a function-name in C++ is given C linkage, i.e. there is no overloading and C++ compiler does not add argument/parameter type information to the name used for linkage. In other words, function name is the only thing needed to link to such function. Then the fucntion is directly callable from Fortran following code block 2.

```
call cpp_function(argument_1, argument_2, argument_3)
```

Code block 2.

Calling Python from C++ is a more challenging task, it requires the usage of Python/C API. An exemplary implementation of embedding Python in a C++ code can be realized via code block 3.

```
#include <iostream>
```

```cpp
#include <Python.h>

extern "C" {
void cpp_call_python_(const double* argument_1, double* argument_2, double& argument_3);
}

void cpp_call_python_(const double* argument_1, double* argument_2, double& argument_3)
{
    Py_Initialize();

    PyObject* py_module = PyImport_ImportModule("file_name");

    PyObject* py_function = PyObject_GetAttrString(py_module, "py_function");

    PyObject* output = PyObject_CallFunctionObjArgs(py_function,
                                        PyFloat_FromDouble(argument_1[0]),
                                        PyFloat_FromDouble(argument_2[0]),
                                        PyFloat_FromDouble(argument_3), NULL);

    std::cout << "output #0 "<< PyFloat_AsDouble(PyList_GetItem(output,0)) << std::endl;
    std::cout << "output #1 "<< PyFloat_AsDouble(PyList_GetItem(output,1)) << std::endl;

    Py_Finalize();
}
```

Code block 3.

Recommended compiling flags when using Python 3.7 and earlier are obtained via calling `python3.X-config --cflags` while linking flags are provided via `python3.X-config --ldflags`, where X specifies which Python release is used. In the presented case `py_function` is assumed to be defined in `file_name.py`, for instance, as in code block 4.

```python
def py_function(*args, **kwargs):
    print("py_function is being called")
    print("Arguments: ", args)
    print("Keyword arguments: ", kwargs)

    return [args[0] * args[1], args[2]]
```

Code block 4.

It is worth noting that calling C++ from Fortran causes only negligible overhead. The situation is quiet different for Python though: The call to Python is a bit slower but it can still be ideal for the development process as there is, e.g., no need for multiple code compilation cycles each time a new or alerted constitutive model is investigated. Further, the code itself can be really sleek.

Mechanical and thermal material routines are visualized in Fig. 2 as a blackbox model.

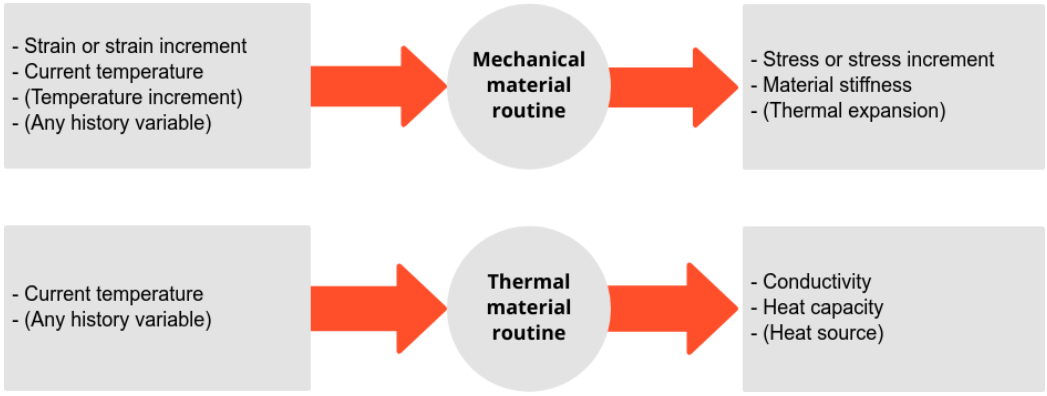

Fig 2. Illustration of how material routines are called.

Item in parentheses (●) in Fig. 2 are optional and dependent on specific use cases. Thermal expansion, for instance, is not a required output by LS-DYNA, instead it can be directly utilized in the mechanical material routine.

The above-mentioned inputs and outputs may be extended but that requires editing the main calling code in Fortran.

4 Practical examples

In this section, a set of use cases are discussed to illustrate what can be achieved with the current implementation. Corresponding files are included and each case is presented in a step by step fashion to allow for reproducibility. All cases are stored in the `examples` directory and each include a `run_example.sh` file that has all necessary commands to run such an example.

4.1 Using LS-DYNA to compute RVE response

Relevant examples are found in `examples/two_scale/2d_rve` and `examples/two_scale/3d_rve`. Given files consist of a finite element mesh file called `mesh.k` and LS-DYNA input files `input_rve.k` and `input.k`. Periodic or uniform displacement boundary conditions defined in `input_rve.k` are applied/added to a newly generated mesh file called `rve_mesh.k` by calling `lsdyna i=simple_block_mesh_1000el.k mcheck=y`.

Thermal boundary conditions are defined in `input.k` as a function of spatial nodal location as in code block 5.

```
*DEFINE_FUNCTION
1 $ function id
float f(float x, float y, float z, float vx, float vy, float vz, float time)
{
 float gradx, grady, gradz;
 gradx = 1.0; grady = 1.0; gradz = 1.0;
 return 273.0 + x*gradx + y*grady + z*gradz;
}
```

Code block 5.

Then thermo-elastic response is written to the terminal after calling `lsdynaumat i=input.k`. Note that `lsdynaumat` command here, refers to the newly compiled version of LS-DYNA. Compilation steps are include in the `readme` file.

For a chosen set of material parameters, boundary conditions and 2D mesh as in Fig. 3, the corresponding stress σ_x distribution is depicted in Fig. 3.

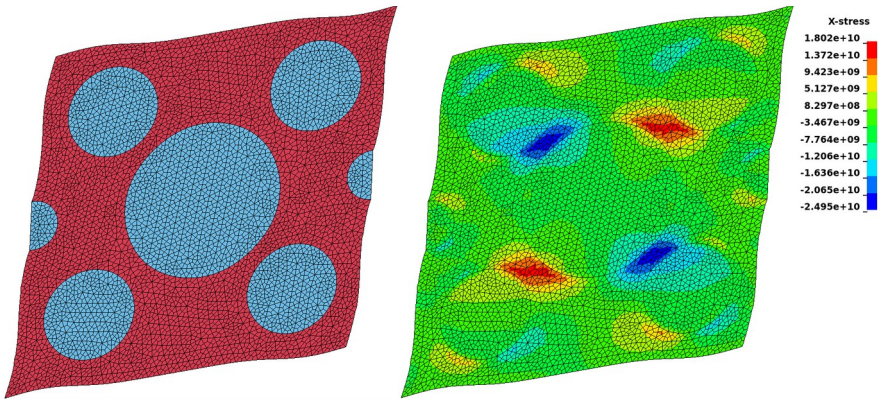

Fig 3. Exemplary 2D RVE and its corresponding response.

Assuming that an RVE has unit length, the user can easily modify its boundary conditions by altering values at the end of `rve_mesh.k` file under **`*BOUNDARY_PRESCRIBED_MOTION_NODE`** card. Further, this step can be automated as done in `rve_change_pbc.sh`.

Note that in case of thermoelasticity it is enough to load a 3D RVE in six orthogonal strain directions, three orthogonal temperature gradients and solve one eigenstress problem to get all effective properties at a given temperature [7]. The number of simulations is reduced in case of 2D microstructure to three mechanical simulations, two thermal ones and one eigenstress problem.

4.2 Analytical expressions for material properties

As a proof of concept case, temperature dependent material parameters are considered here. Given scattered material paramaters over a wide temperature range it is possible to fit these parameters using a polynomial function with low degree (one, two or three). Then such polynomails can be automatically written as lambda functions, i.e. usually one liner small anonymous function that can take any number of arguments but can only have one expression.

For example, heat capacity and elastic modulus of oxygen-free high conductivity (OFHC) copper are illustrated, after the fitting process, in Fig. 4.

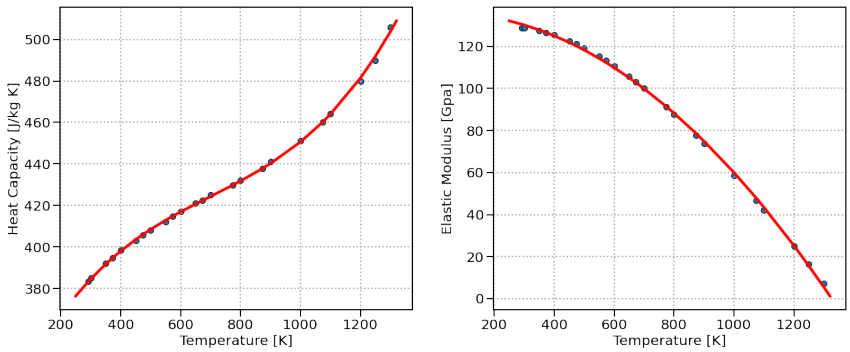

Fig 4. Temperature dependent material parameters of OFHC.

As an output of the fitting procedure, the functions in code block 6 are defined and thier handles can be directly passed to a user defined material routine.

```
heat_capacity_cu = lambda x: 3.162e+02 * x ** 0 + 3.178e-01 * x ** 1 - 3.497e-04 * x ** 2 + 1.663e-07 * x ** 3
elastic_modulus_cu = lambda x: 1.357e+02 * x ** 0 + 5.857e-03 * x ** 1 - 8.161e-05 * x ** 2
```

Code block 6.

In other words the implementation of a UMAT does not change when changing material parameters, this is particularly useful for automation purposes or when fitting experemental data.

Such implementations are possible in Python or C++. Hence, relevant examples are provied in `examples/analytical_mat_parameter`, where material parameters have to be defined for a macrostructure with two different material phases that is illustrated in Fig. 5. Analytical expressions for material parameters are already defined in `umat/umat.py` and `umat/umat.cpp`. However, a compilation step is still needed to activate either the Python interface or the C++ one, this is done by setting `python_interface` or `cpp_interface` to `.true.` in `umat_elastic_44_14.F90` and compiling following the instructions in `readme`. Under sufficient boundary conditions, a thermomechanical response is visualized in Fig. 5.

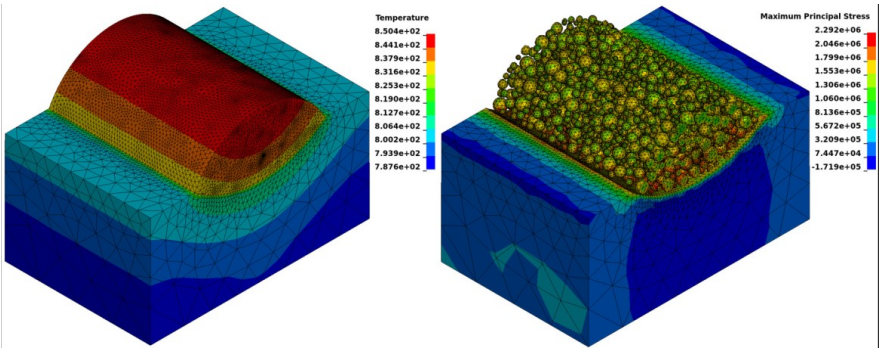

Fig 5. Exemplary results of a two phase structure using UMAT.

4.3 Realization of two scale simulation scheme in LS-DYNA

This last showcase illustrates the possibilities of incorporating high-end multiscale methodologies within commercial software. In such scenario, academic codes can be used to feed ROM data and commercial codes for the actual deployment in a trial to meet industry needs. Here, an efficient in-house Fourier-Accelerated Nodal Solver (FANS) [10] is used to generate effective thermo-elastic response of a 3D RVE. Even if the data was generated from any surrogate model, the way how the code is used will not be altered.

In the example in `examples/two_scale/homogeneous_single_track`, an RVE effective response under different load temperatures is assumed to exist and stored in a tabulated format in an HDF5 file. For the sake of brevity, linear interpolation is used to evaluate effective properties at current temperature given the stored response at one higher and one lower temperatures.

The heterogeneous structure in Fig. 5 is replaced buy a homogeneous one as in Fig. 6 and the corresponding user material input card is updated as in code block 7.

```
*MAT_THERMAL_USER_DEFINED
$#     tmid        ro        mt       lmc       nhv      aopt     iortho      ihve
         2       1.0        14         8        13       0.0          0         0
$#       p1        p2        p3        p4        p5        p6        p7        p8
       0.0       0.0       0.0       0.0       0.0       0.0       0.0         2
```

Code block 7.

Here material desity is given as 1.0 [-] because the model in `rve_elastic.py` does not return heat capacity alone but multiplied by density and `p8=2` is set to match the implementation in `umat.py`, relevant part is included in code block 8.

```
class material_id(Enum):
    copper   = 0
    tungsten = 1
    rve      = 2
```

Code block 8.

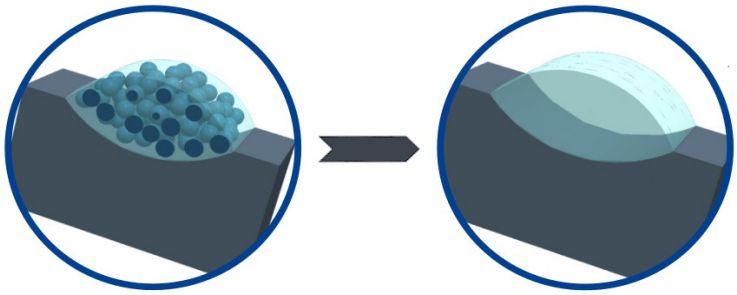

Fig 6. Replacement of heterogeneous structure by a homogeneous one.

Based on the provided input, it is possible to obtain the homogenized response of the structure in Fig. 5 as illustrated for the thermal case in Fig. 7. The choice of the RVE geometry is not discussed here as it does not influence the implementation.

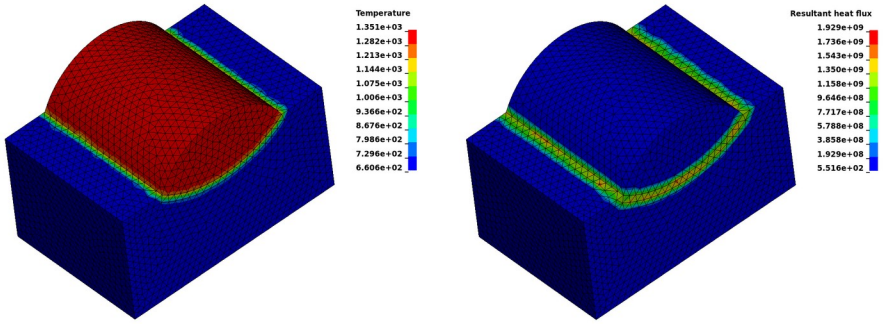

Fig 7. Homogenized response of two-phase structure.

5 Summary

An extension to LS-DYNA object version is proposed to allow the usage of LS-DYNA in two-scale simulation scenarios. At the same time, direct usage of analytical expressions for the evaluation of material parameters is realized in Python and C++.

The user-friendly extension is designed to be flexible and extendable. Some of possible use case include but not limited to:

- Reduced order model for FE^{2R} simulations
  - Use high-end multiscale methodologies within commercial software.
  - Provide "best of two worlds" experience by using academic codes for feeding the ROM and commercial codes for the actual deployment.
- Machine-learned surrogates
  These models can, for instance, replace the ROM ones by a plug & play fashion. Hence, various models may be tested to choose an optimal one for a specific objective.

All extensions are provided with an automatic build scripts. Therefore, focus can be directed toward the implementation of material routines in high-level programming language such as Python.

The source code is now available via GitHub [10].

6 Acknowledgment

The IGF-Project no.: 21.079 N / DVS-No.: 06.3341 of the “Forschungsvereinigung Schweißen und verwandte Verfahren e.V.” of the German Welding Society (DVS), Aachener Str. 172, 40223 Düsseldorf, Germany, was funded by the Federal Ministry for Economic Affairs and Energy (BMWi) via the German Federation of Industrial Research Associations (AiF) in accordance with the policy to support the Industrial Collective Research (IGF) on the orders of the German Bundestag.

7 Literature

[1] Feyel F, Chaboche JL. FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. Computer methods in applied mechanics and engineering. 2000; 183(3-4):309-30.
[2] Geers MG, Kouznetsova VG, Brekelmans WA. Multi-scale computational homogenization: Trends and challenges. Journal of computational and applied mathematics. 2010; 234(7):2175-82.
[3] Fritzen F, Hodapp M. The finite element square reduced (FE2R) method with GPU acceleration: towards three-dimensional two-scale simulations. International Journal for Numerical Methods in Engineering. 2016; 107(10):853-81.
[4] Fritzen F, Fernández M, Larsson F. On-the-fly adaptivity for nonlinear twoscale simulations using artificial neural networks and reduced order modeling. Frontiers in Materials. 2019; 6:75.
[5] Hill R. Elastic properties of reinforced solids: some theoretical principles. Journal of the Mechanics and Physics of Solids. 1963; 11(5):357-72.
[6] Wippler J, Fünfschilling S, Fritzen F, Böhlke T, Hoffmann MJ. Homogenization of the thermoelastic properties of silicon nitride. Acta materialia. 2011; 59(15):6029-38.
[7] Fritzen F, Boehlke T. Periodic three-dimensional mesh generation for particle reinforced composites with application to metal matrix composites. International Journal of Solids and Structures. 2011; 48(5):706-18.
[8] Liu Z, Wu CT, Ren B, Liu WK, Grimes R. Multiscale simulations of material with heterogeneous structures based on representative volume element techniques. 15th International LS-DYNA Users Conference. 2018; 1-10.
[9] Leuschner M, Fritzen F. Fourier-accelerated nodal solvers (FANS) for homogenization problems. Computational Mechanics. 2018; 62(3):359-92.
[10] https://gitlab.com/shadialameddin/dae_umat_2scale_lsdyna.git