upscale

Upscaling **P**roduct development **S**imulation **C**apabilities exploiting **A**rtificial inte**L**ligence for **E**lectrified vehicles

D5.7 Requirements for setting up an AI model to improve parallelization of the solver

M. Andres (VW), Ch. Breitfuss (VIF), F. Daim (ESI), A. Dumon (ESI), Alain Trameçon (ESI), N. Hascoët (ENSAM), F. Chinesta (ENSAM), E. Cortelletti (CRF), C. Jiménez (IDIADA), S. Shaik (IDIADA)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 824306.

PROJECT TITLE	Upscaling product development simulation capabilities exploiting artificial intelligence for electrified vehicles
PROJECT ACRONYM	Upscale
GRANT AGREEMENT NUMBER	824306
INSTRUMENT	RIA
CALL	LC-GV-2018
STARTING DATE OF THE PROJECT	November, 1ST 2018
PROJECT DURATION	42 Months

The UpScale Project

UPSCALE is the first EU project with the specific goal of integrating Artificial Intelligence (AI) with traditional physics-based Computer Aided Engineering to reduce the development time and increase the performance of electric vehicles (EVs).
Nowadays High-Performance Computing (HPC) and Computer Aided Engineering (CAE) play a decisive role in vehicle development processes, thus the two most HPC and CAE intensive parts of the development, which are vehicle aero-thermal and vehicle crash performance, have been chosen as use cases for the endeavour.
Through the combined effort of universities, research laboratories, European automotive OEMs, software companies and an AI-SME specialized in Machine Learning (ML), the UPSCALE, project will provide a unique and effective environment to produce novel AI-based CAE-software solutions to improve the competitiveness of the automotive industry.

The UpScale Consortium

PARTICIPANT Nº	PARTICIPANT ORGANISATION NAME	COUNTRY
1 (Coordinator)	**IDIADA AUTOMOTIVE TECHNOLOGY SA (IDIADA),**	Spain
2	VOLVO PERSONVAGNAR AB (Volvo Cars)	Sweden
3	VOLKSWAGEN AG (VW)	Germany
4	CENTRO RICERCHE FIAT SCPA (CRF)	Italy
5	ESI GROUP (ESI GROUP)	France
6	ENGYS LTD (ENGYS LTD)	United Kingdom
7	VIRTUAL VEHICLE RESEARCH GmbH (VIF)	Austria
8	VRIJE UNIVERSITEIT BRUSSEL (VUB)	Belgium
9	ECOLE NATIONALE SUPERIEURE DES ARTS ET METIERS (ENSAM PARISTECH)	France
10	ALGORITHMICA TECHNOLOGIES GMBH (ALGORITHMICA)	Germany
11	F INICIATIVAS I MAS D MAS I SL (F-INICIATIVAS)	Spain

Document Details

DELIVERABLE TYPE	Other
DELIVERABLE Nº	D5.7
DELIVERABLE TITLE	Requirements for setting up an AI model to improve parallelization of the solver
NAME OF LEAD PARTNERS FOR THIS DELIVERABLE	ESI
VERSION	3
CONTRACTUAL DELIVERY DATE	Month 36
ACTUAL DELIVERY DATE	Month 39
DISSEMINATION LEVEL	Public

ABSTRACT

The present deliverable describes the requirements for efficient deployment of short circuit and stiffness ROM in a full vehicle model in terms of computing efficiency and scalability. Several ROM methods are computed and compared.

Revision History

The following table describes the main changes done in the document since it was created

REVISION	DATE	DESCRIPTION	AUTHOR (ORGANIZATION)
1	10.12.2021	First draft	A Dumon (ESI)
2	16.12.2021	Section 3 added	F Daim (ESI)
3	19.12.2021	Final version	A Dumon (ESI)

Abbreviations

COG	Center of Gravity
DOE	Design of Experiment
DUT	Device under Test
FEA	Finite Element Analysis
FEM	Finite Element Method
LHS	Latin Hypercube Sampling
ROM	Reduced Order Model
RVE	Representative Volume Elements
SC	Short Circuit
VPS	Virtual Performance Solution
API	Application Programming Interface
IDMD	Incremental Dynamic Mode Decomposition
TANN	Thermodynamics-based Artificial Neural Networks

Table of Contents

1 Executive Summary

The present deliverable analyzes the computing efficiency for the various methodologies of reduced order models (ROM), which deployment was described in D3.3. This entails optimization of computing efficiency, time step, mesh gauges and scalability in DMP on full vehicle models.

The short circuit ROM was evaluated as an embedded user material at a macro scale with an embedded short circuit criterion on a full vehicle pole impact simulation. Various subcycling parameters between the ROM and FEA solver were investigated with an optimum found to produce similar computing costs as the standard honeycomb material. Coarser mesh gauges were tested to evaluate if the computation could be further sped up. Last, scaling up to 144 processors was confirmed.

The cell stiffness ROM integration in a crash simulation required the development of a new user generalized element API in VPS to enable the hybrid computation of reduced order model and finite element solver. First, the generic API calls within the general FEA code is detailed. Then, the deployment of the Incremental Dynamic Mode Decomposition (IDMD) ROM in the generic user element is presented. Last, the challenges and improvements to the coupling are mentioned.

An alternative method for the cell stiffness ROM is presented by leveraging ESI neural network material model as used for honeycomb or composite structures. The method is evaluated at cell and vehicle level in terms of computing efficiency and mesh gauge dependency. The TANN approach can be combined easily with the short circuit ROM as they are both based on strain tensor inputs.

The deliverable objectives according to the proposal could be fulfilled. The time amendment enabled design iterations on a coming general API in ESI VPS which enables coupling of the cell stiffness ROM with FEA. To limit time deviation on the final deliverables, a neural network-based approach was developed and evaluated in terms of computing efficiency at the vehicle level. Additionally, this will allow evaluation with both cell stiffness and internal short circuit ROMs.

This deliverable is uploaded with 3 months delay due to an alignment of the generalized user API for FEA/ROM coupling in development version of VPS2022.0 with the modular material (MMAT) user API in the same version. This delay doesn’t affect any project activities, deliverables or results.

2 Introduction and Overview

In the UPSCALE project, several ROMs were trained in D3.2 and deployed in D3.3 to be able to capture the various phenomenon with improved computational costs:

- A short circuit ROM trained to predict the local short circuit phenomenon from a very detailed cell model incompatible with vehicle model
- A stiffness ROM to predict the stiffness of a cell based on envelope nodal displacements and other variables

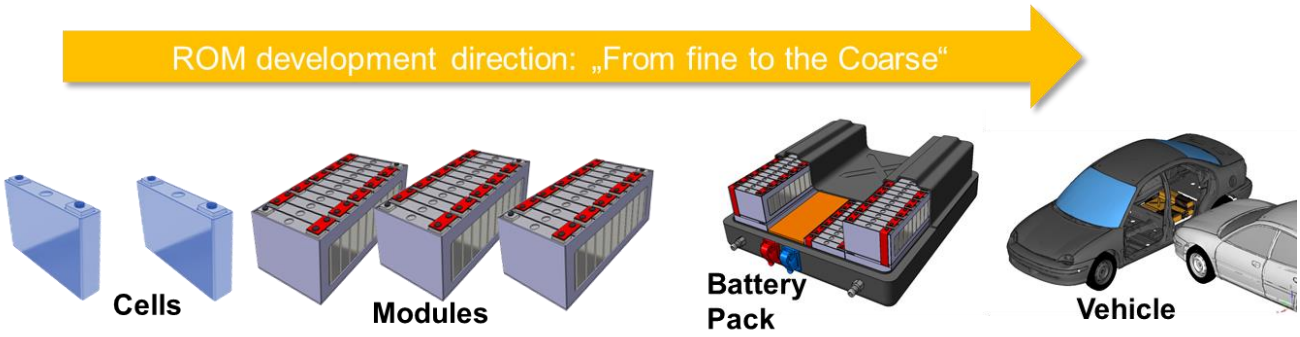

Figure 1: Methodology for use of ROMs in UPSCALE project

The current deliverable aims at quantifying the computing efficiency of vehicle models in DMP with these various ROMs to gain some understanding and guidelines.

3 Battery risk evaluation with short circuit ROM

In D3.3, we presented the first results with the full vehicle model including a coarse cell modelling with a short circuit risk obtained with AI techniques. An important aspect of the computing time response of the full vehicle model is the number of solid elements in the macro cell model with short circuit ROM.

The homogenized cell model for stiffness was generated in D1.6 based on a detailed cell model built in D3.2 and then complemented with an AI short circuit risk. The reference mesh gauge was 3 mm.

3.1 Short circuit ROM input and training

As a reminder, the methodology for battery risk evaluation with short circuit ROM is described in detail in D3.2. The short circuit ROM building and deployment tools are also described in detail in D3.3.

3.2 Computing setup

To perform a meaningful comparison, the models are run on ESI Bruyeres HPC center in France. The standard run options on the cluster server are the following:

Platform	Linux-3.10.0-1127.10.1.el7.x86_64
CPU	Intel(R) Xeon(R) Gold 6150 CPU @ 2.70GHz
RAM	376Gb
Solver mode	DMP-sp
Precision	Single
Parallelism	Hybrid
MPI	Intel MPI 5.1.3.258

Table 1 : Parameters of the computation runs

The models are run with a varying number of processors from 72 to 144 processors. As D1.10 did not provide significant improvement of the domain decomposition, the standard option for crash application was used, Volumetric Coordinate Bisection (VCB).

3.3 Short circuit ROM performance

In D3.3, a first optimization of the computing efficiency with the user material with short circuit ROM was performed with optimization of data reading and subcycling of the AI calculation of the risk with respect to the mechanical and solver time steps. A subcycling value of 10 was deemed suitable to ensure good computing efficiency on a unit cell model.

The same methodology is applied to a modified e-Golf pole impact case with coarse cell models including the AI calculation of the short circuit risk for the cell models as provided in D3.2 and D1.6. The detail of the e-Golf load case is provided in D5.5.

Case	Model	Gauge	Time 72 skylake	Time 144 skylake
1	Baseline	3 mm	8.5 hrs	5.5 hrs
2	AI short circuit	3 mm	8.5 hrs	5.5 hrs

Table 2: Computing time for the modified e-Golf with homogenized cell model with AI short circuit on 72 and 144 cpus skylake

With the proposed subcycling, the computing efficiency of the material enriched with AI for the short circuit evaluation seem close to the baseline of the standard honeycomb material: VPS material type 42.

ESI Group has developed an internal tool to generate a consolidated performance assessment of either one simulation or for a series of simulations based on result and log files. For more detail, see D1.10 Using this performance tool to analyze the DMP and scaling as presented in D1.10, we reach the following results.

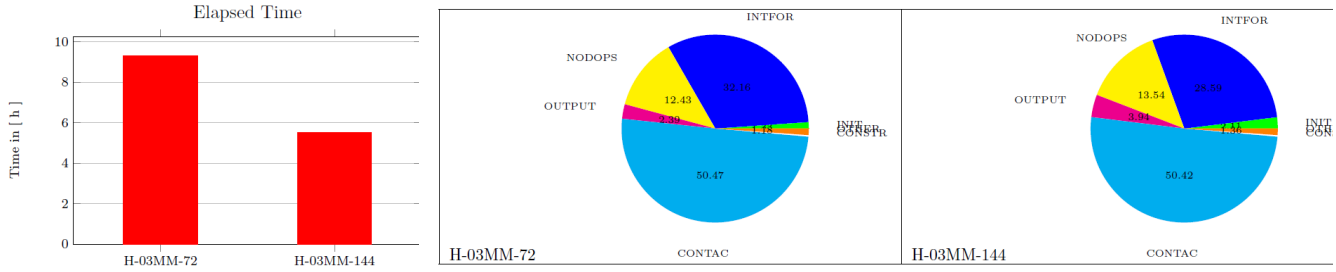

Figure 2: Computing times comparison and share of various solver operations for the modified e-Golf with homogenized cell model with AI short circuit on 72 and 144 cpus skylake, run1 and run2 respectively

The scaling ratio between 72 and 144 processors with the honeycomb enriched with short circuit ROM is 1.7, which is reasonable for a model with user materials and a high number of solid elements. The share of nodal operations and contact does not change significantly with higher number of processors, showing the good handling of domain decomposition at higher cpu count.

3.4 Importance of cell mesh gauge

An important aspect of the computing time response of the full vehicle model is the number of solid elements in the macro cell model with short circuit ROM.

The battery pack element count is around 1 million added solid elements and with a reference mesh gauge of 3 mm. Mesh gauges of 3 to 10mm are evaluated to assess the effect on performance. As a note, the 10 mm mesh gauge only ensures one element in cell thickness, whereas the 5mm mesh gauge ensures the standard 3 solids in the cell thickness.

	Nsolids thousands	Min length mm	Nshells thousands	Min length mm
MACRO cell	6.6	3	12.8	1.7
Full pack	1340	3	2598	1.7

	Nsolids thousands	Min length mm	Nshells thousands	Min length mm	Nelt thousands
MACRO cell	3.1	5	2.2	5	5
Full pack	629	5	447	5	1076

	Nsolids thousands	Min length mm	Nshells thousands	Min length mm	Nelt thousands
MACRO cell	0.2	10	0.7	10	1
Full pack	41	10	142	10	183

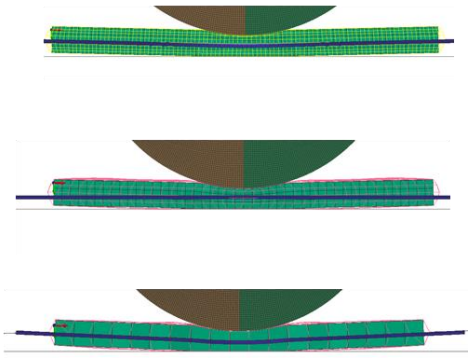

Figure 3: Element count and visualization for the homogenized cell with different mesh gauges

A first analysis on the cell response is performed to ensure the various mesh gauges perform with a similar mechanical response as the 3mm mesh gauge homogenized cell model. The force displacement behavior is similar for the unit case, except for the folding case where all the mesh gauges only approximate the buckling visible on the detailed model.

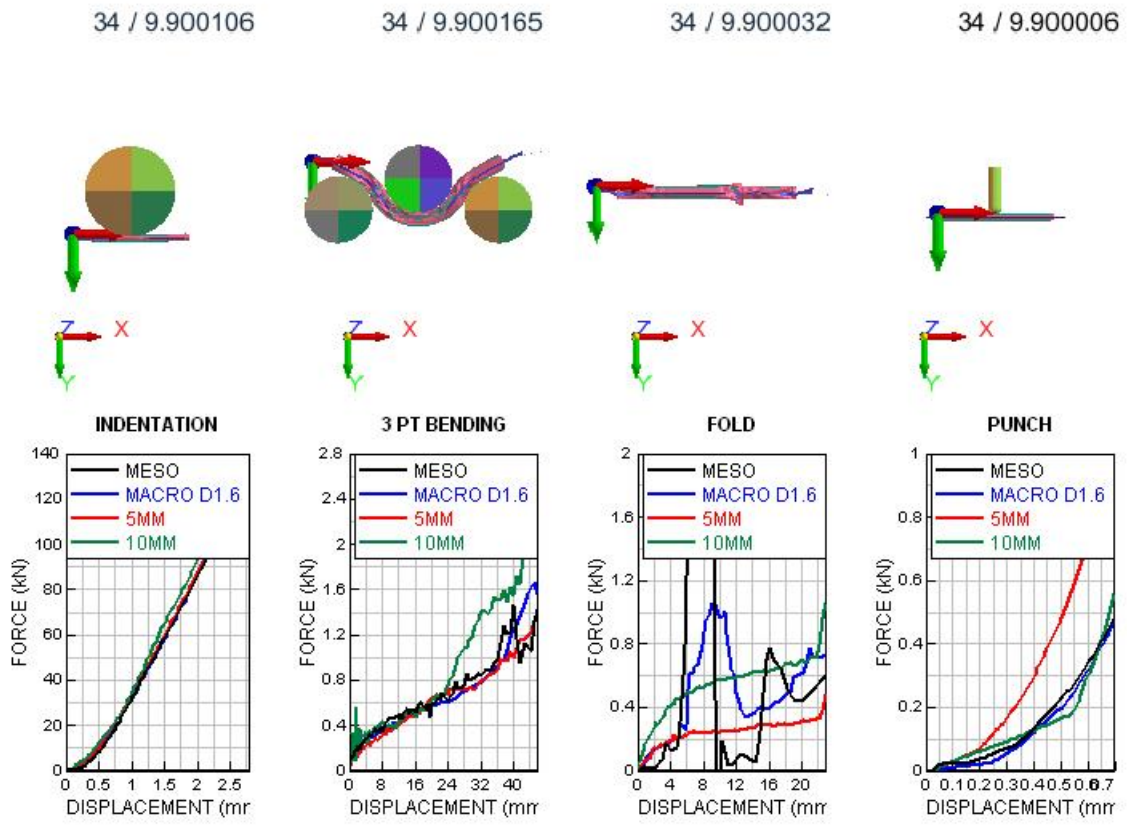

Figure 4: (a) Visualization of cell with internal short circuit ROM for various mesh gauge and (b) force-displacement response with various mesh gauges with respect to the detailed model response

After validation that the stiffness is maintained with various mesh gauges, we move to evaluation at the full vehicle level. As described in the previous section, the calculation is performed with a subcycling of 10 between the stiffness calculation with FEM and the internal short circuit risk with AI.

Table 3: Computing time for the modified e-Golf with homogenized cell model with AI short circuit on 72 cpus skylake server for various mesh gauges

Case	Model	Gauge	Time 72 skylake	Time 144 skylake
0	Baseline	3 mm	8.5 hrs	5.5 hrs
1	Baseline	5 mm	6.0 hrs	-
2	Baseline	10 mm	3.4 hrs	-
3	AI short circuit	3 mm	8.5 hrs	5.5 hrs
4	AI short circuit	5 mm	6.0 hrs	3.5 hrs
5	AI short circuit	10 mm	3.5 hrs	2.0 hrs

A takeaway is that coarsening the AI enriched elements allow to reach relatively fast computations and scaling is reasonable across the various mesh gauges. A tradeoff is to be expected in terms of deformation with a 10mm mesh gauge. A mesh gauge of 5mm could be preferred to 3mm depending on the computation time requirements.

A similar application of ESI in-house performance tool on the cases with 10 mm mesh gauge shows scaling factor of 1.74 and the share of operations depicted below. The scaling does not

impact the share of operations. A comparison with the 3 mm gauge case shows that a coarser length has a higher share of solver nodal operations and internal forces with respect to contact.

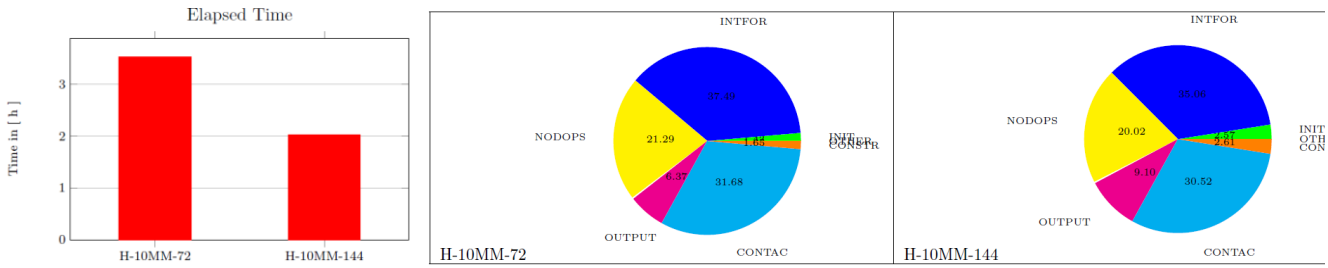

Figure 5: Computing times comparison and share of various solver operations for the modified e-Golf with homogenized cell model with AI short circuit on 72 and 144 cpus skylake with 10mm mesh gauge

4 Local-global integration: stiffness ROM with FEA analysis

In EV, short circuits occur at the battery cell level and it is thus this component that we choose as the elementary component for reduction in this project. In crash simulations, we cannot easily model this cell as a detailed FEA model due to the large number of elements needed, as well as the stringent time step requirements of explicit integration schemes in rapid dynamics. Thus, a macro model was defined on the pouch where an equivalent stiffness model is computed at each time step (See D3.3).
In this work, we will discuss the integration of our approach in the explicit scheme used in VPS via a user element plugin and which will be in the next commercial release of the product VPS 2022.0.

The user element plugin in VPS will enable us to:

1. Extract data for the training (offline) stage
2. Perform a local-global simulation where hybrid computations of a reduced-order model and the finite element solver are linked.

4.1 Model input and training

To integrate the new user element plugin, the VPS input file format was adapted though the addition of a new general interface card in which the (interface) nodes are specified explicitly as shown in Figure 4.1. The order in which they are specified determines the order in which they will be received/sent by the user plugin function from/to VPS.

```
$================================================================================
$
END_CDATA
#---5---10----5---20----5---30----5---40----5---50----5---60----5---70----5---80
INCLU / pyvar.inc
INCLU / pyvar_3pts_bending.inc
INCLU / pouch_mat.inc
INCLU / aluminum_mat.inc
INCLU / copper_mat.inc
INCLU / jellyroll_mat.inc
INCLU / envelope_mesh.inc
INCLU / jellyroll_mesh.inc
INCLU / impactor.inc
INCLU / SC_Groups.inc
INCLU / idmd_output.inc
INCLU / CellPatch.inc
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                             $
$                                  CONTACTS                                   $
InputFile.pc
$ ----------------- Cell Battery patch element connecting 10k nodes -----------
$-------------16------24------32------40------48------56------64------72------80
PATCH /  9100001
                150000011500000215000003150000041500000515000006150000071500000
                150000091500001015000011150000121500001315000014150000151500001
                150000171500001815000019150000201500002115000022150000231500002
                150000251500002615000027150000281500002915000030150000311500003
                150000331500003415000035150000361500003715000038150000391500004
                150000411500004215000043150000441500004515000046150000471500004
                150000491500005015000051150000521500005315000054150000551500005
                150000571500005815000059150000601500006115000062150000631500006
                150000651500006615000067150000681500006915000070150000711500007
                150000731500007415000075150000761500007715000078150000791500008
                150000811500008215000083150000841500008515000086150000871500008
                150000891500009015000091150000921500009315000094150000951500009
                150000971500009815000099150001001500010115000102150001031500010
                150001051500010615000107150001081500010915000110150001111500011
                150001131500011415000115150001161500011715000118150001191500012
                150001211500012215000123150001241500012515000126150001271500012
CellPatch.inc
```

Figure 6: Patch element representation in development framework of general API in ESI VPS

4.2 User element plugin:

To solve the semi-discretised (spatial) dynamic equation $M\,\ddot{U}(t) = F_{int}(U(t)\,,\dot{U}(t)) + F_{ext}$ in VPS, an explicit leapfrog scheme in time is used (see Figure 4.2.) where:

- M is the mass matrix, F_{int} is the internal forces depending on the displacement U and the velocity $\dot{U}$
- $U^n, \dot{U}^n \; and \; \ddot{U}^n$ are the approximation of (respectively) displacement, velocity, and acceleration at the time step t^n. We note that in the case of element shell, U represents the transitional displacement and rotations, whereas the total force F represents the forces and the moments.

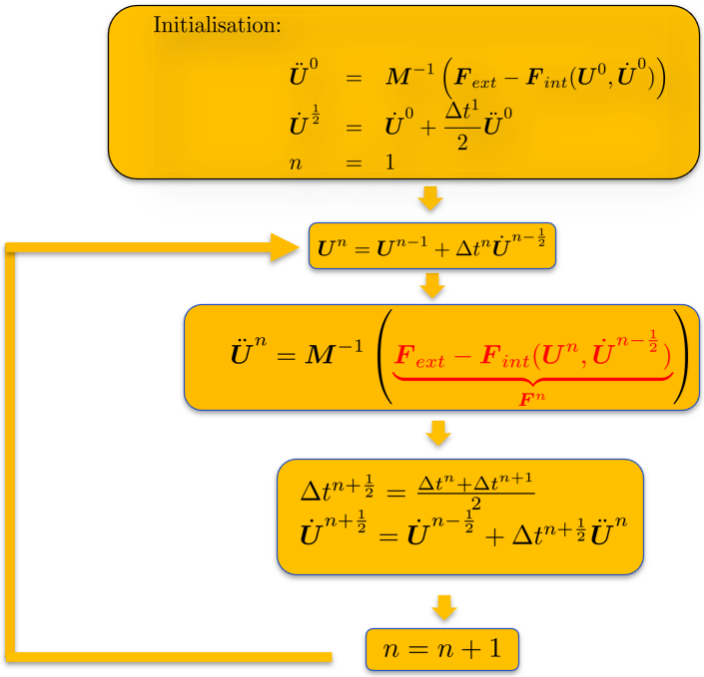

Figure 7: Explicit scheme in VPS

The iDMD is implemented as a user element plugin with the following scheme (see Figure 4.3). User element plugins for VPS can be implemented in C or Python. Here, we implemented our user plugin in python.

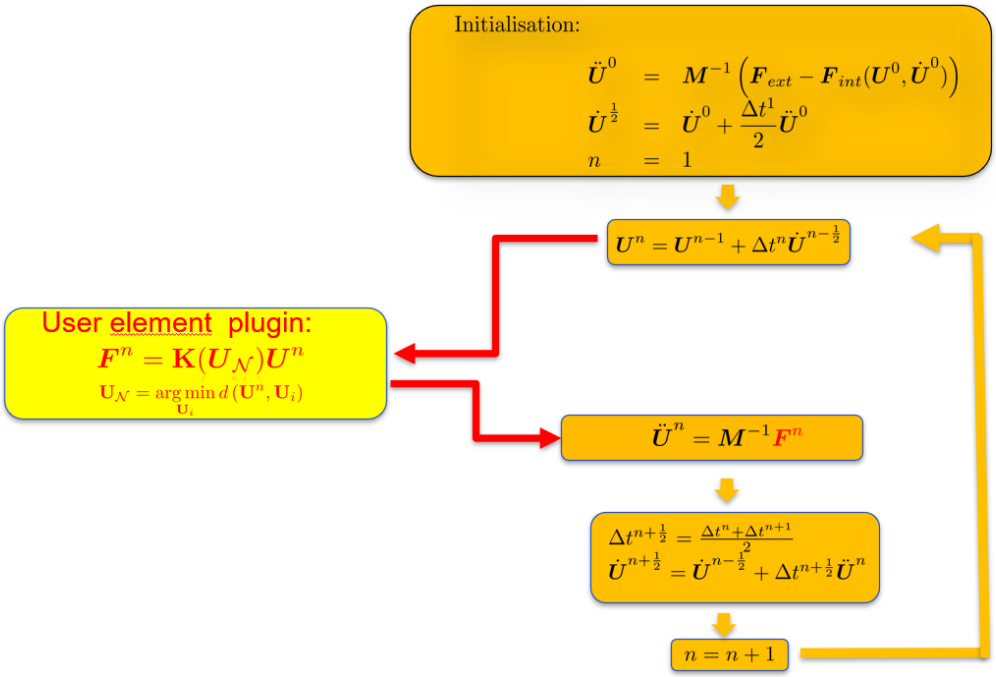

Figure 8: iDMD in the explicit scheme of VPS

At each time step, the plugin receives the displacements (translational displacements and rotations) and should send back the reaction forces (forces, moments). However, as the nodal

forces are not a standard output in VPS explicit, during the training runs that we performed we were obliged to use a trick, so-called tied elements, to extract the forces. Unfortunately, there is no equivalent way to obtain the moments in VPS. ESI Group will enable the extraction of all necessary fields in the training phase in an upcoming version of VPS. The list of planned developments is summarised below:

I. During offline (training) stage: extracts data (displacements, rotations, velocity, angular velocity, forces, moments).
II. During online stage: sets data (forces, moments)
III. Able to handle several patches: to apply the iDMD on compressed data by considering the pouch as several patches (5 patches, see D3.3).
IV. Sets local frame for fields of interests: to be able to perform the local-global integration.

When the pouch receives the displacements from the global structure, they should be transformed to the local frame of the pouch as this is the frame in which the training data were obtained.

Figure 9: local-global integration frame

4.2.1 User element plugin for offline stage:

During the offline (training) stage, the log_fields() function in the user element plugin is called by VPS to extract displacements (coordinates), velocities, angular velocities, forces and moments. Global parameters are made available in the plugin interface that can be used to e.g. pilot the timestep or number of cycles between each extraction. An example in python can be seen below.

```python
def log_fields():
    global logfiles, x_0, dset_t, dset_f, dset_tau, dset_u, cycle_train, NPatch, eps

    NPatch = NPatch + 1
    Dt_train  = ecu.get_parameter('Dt_train')
    NCycles  = ecu.get_parameter('NCycles')

    t     = ecu.get_time()
    dt    = ecu.get_timestep()
    cycle = ecu.get_cycle()

    element_id, module_id = ecu.get_element_user_id()
    print("element_id  ======> " , element_id )
    print("time => " , t , "cycle =>" , cycle , "dt simu  => ", dt)
    print("time => " , t , "Dt_train * cycle " , Dt_train * cycle_train , " diff  ==> ",
    #if(np.abs(t - Dt_train * cycle_train) < eps):
    if(cycle % NCycles == 0 and cycle !=0):
        x = ecu.get_position()
        v = ecu.get_velocity()
        omega = ecu.get_angular_velocity()
        f = ecu.get_assembled_force()
        tau = ecu.get_assembled_moment()

        u = x - x_0[element_id]
        u = u.transpose().reshape(1, u.size)
        f = f.transpose().reshape(1, f.size)
        tau = tau.transpose().reshape(1, f.size)
```

Collecting data on the patch

Figure 10: "log_fields" python function to collect data (displacement, velocity, force, moment) in the high-fidelity model

4.2.2 User element plugin for online stage:

During the online stage, the compute_internal_forces() function in the user element plugin is called by VPS to compute the stiffness matrix at each timestep. Nodal displacements are retrieved by the function according to their nearest neighour in the dictionary constructed during the training phase. The forces are moments are sent back via the related stiffness matrix. An example in python can be seen in the following figure.

Figure 11: "compute_internal_function" to compute the stifness matrix and send the forces and the moments

As a first test, we considered 3-point bending. For training data, we extract the displacement on the pouch interface $\boldsymbol{U} = (U_1 U_2 \cdots U_N)$ at each time step, the corresponding reaction forces $\boldsymbol{F} = (F_1 F_2 \cdots F_N)$ and the related stiffness matrices which link each force to the corresponding displacement as seen below.

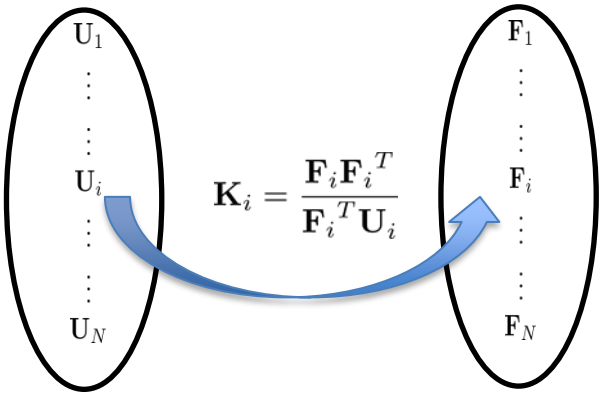

Figure 12: Stiffness matrix linking displacement to the force

So far, in our training runs for the offline stage, we obtain only translational displacements and forces (via tied elements). Nodal reaction rotations and moments are not standard outputs for the VPS solver. Hence, the computation of stiffness matrix of the pouch in the present test case

of 3-point bending is constructed only with displacement and forces. The moments are not provided back to the solver during the time integration.

This explains the large discrepancies we observe when we try to use the stiffness matrix of the pouch using the results of the high-fidelity simulation. It shows the necessity to use the whole stiffness matrix which links the displacements/rotations to the forces/moments.

Nevertheless, we notice that the time step did not drop during the coupled simulation and that the overall computation time was less than 20 minutes with a small dictionary on one processor.

5 Stiffness evaluation with Artificial Neural Networks

As a complement to the previous methodology with a coupled stiffness ROM and FEA based on an interface and a nodal displacement/force interface, the cell stiffness can be reduced with a neural network approach.

5.1 Motivations

ESI has developed for honeycomb and composite applications a neural network driven model, which enable to capture complex deformation behaviour in a computing efficient way. [2][3][6] This can be viewed as an alternative method to capture stiffness with a ROM, this time based on FEA and not on a pure ROM element. This stiffness ROM can also be combined easily with the short circuit ROM from section 2 as both models are taking FEA elements strain tensors as inputs.

5.2 Background on neural networks

Artificial neural networks (ANN) can be considered as a system of signal processing units. The units are thereby arranged in layers, where each layer l can have a distinct number of units n (l) N. Each unit is also commonly referred to as neuron. The signal processing of each layer is governed by an activation function A(l). Assuming a neural network of n L layers, the first layer l = 1 is often referred to as input layer, whereas the last layer l = nL is considered as the output layer. However, neural networks can be designed with arbitrary complexity what go beyond such classification. More details and information about neural networks and their design and implementation can be found in [1][5].

A sketch of a simple feedforward neural network is shown below. The signal at k-th neuron of the l-th layer is defined as:

$$p_k^{(l)} = A^{(l)}\left(z_k^{(l)}\right)$$

with A(l) the activation function of the corresponding layer. The function argument zk(l) is a composure of the weighted signals p (l−1) s of the precedent layer l – 1.

$$z_k^{(l)} = \sum_{s=1}^{n_N^{(l-1)}} \left(w_{ks}^{(l)} p_s^{(l-1)}\right) + b_k^{(l)}$$

$w_{ks}^{(l)}$ ... Weighting factor

$b_k^{(l)}$ ... Bias factor

with w and b the corresponding weighting and bias factors, respectively.

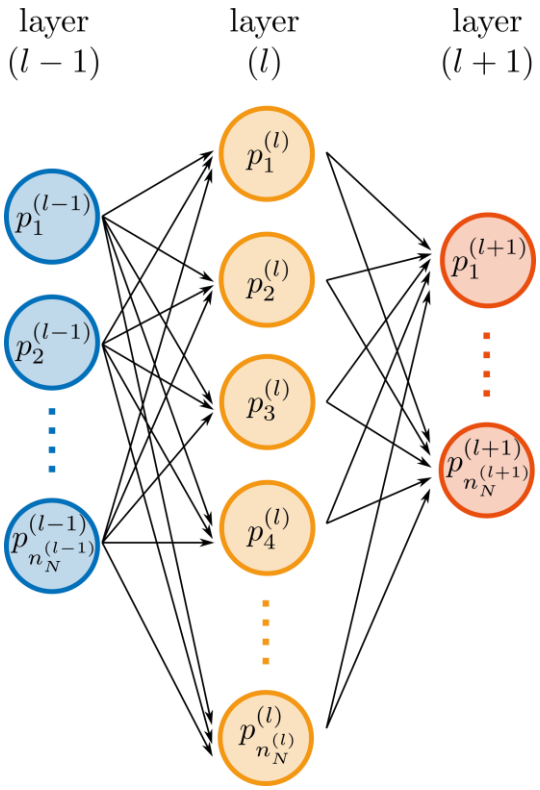

Figure 13: Feedforward artificial neural network structure

The network parameters (weight and bias factors) are identified in an iterative optimization or training process. To this end, a sufficiently large number of input and output data-sets needs to be provided as training data. In addition, the user can provide validation data to check the learning progress and the accuracy throughout the training. The progress is thereby monitored by one or several loss functions.

Thermodynamics based artificial neural networks (TANN) links internal work E with local strain ε which is a standard output of FEM solvers and a basis for neural network approximation. Stresses
obtained from 1st order network gradients and tangent stiffness is obtained from 2nd order network gradients. This enables training of the neural network based on mechanical response as described in the following section.

$$E = \mathrm{NN}_E\left(\overline{\varepsilon}\right)$$

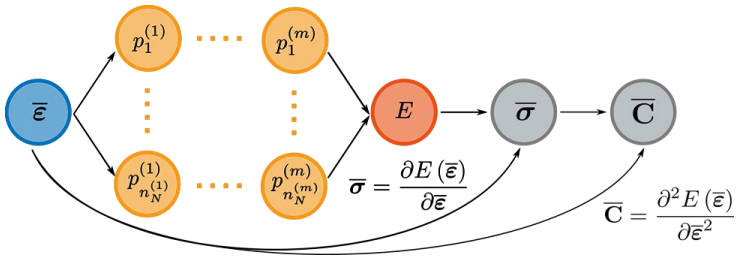

Figure 14: Network activation functions for stiffness ROM with TANN

5.3 TANN model methodology

The neural network is trained by the mechanical material behavior.

A representative volume is extracted from the detailed cell model from D3.2. The detailed representative volume element is then loaded in different directions, including multi-direction loads to generate an array of elastic-plastic responses. These responses enable the training of the TANN stiffness ROM in an offline phase to identify some network parameters, such as layers, neurons, weights, and bias factors. In the offline phase, the trained and homogenized TANN material model is included as a material law in the software code for each macro element: the neural network links strain and stress tensors. This TANN model can be complemented with a short circuit risk from D3.2 based on invariant strains and support vector machine (SVM).

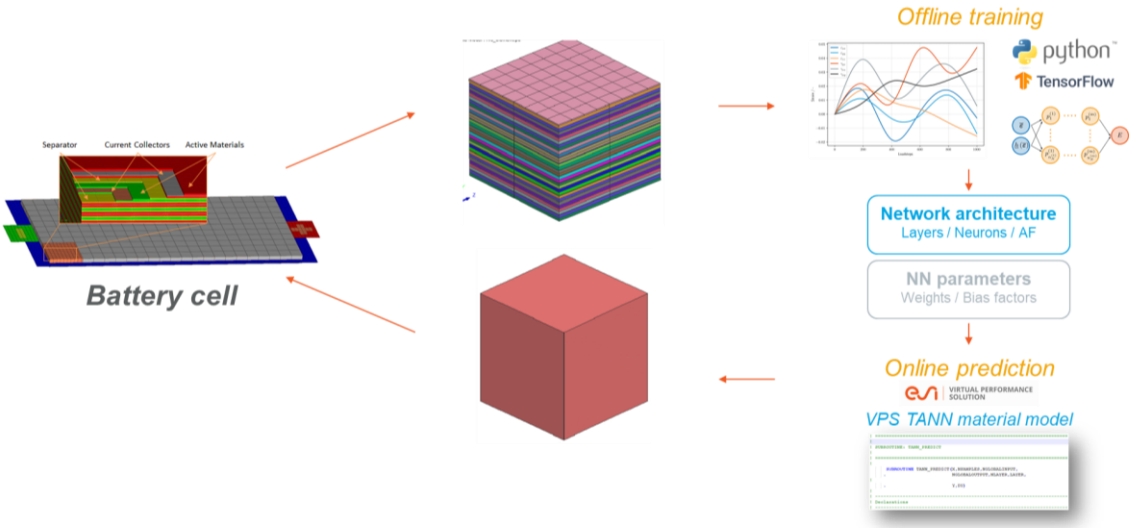

Figure 15:TANN methodology overview

5.4 Training on RVE cell and homogenization

A model is built from D3.2 with a representative volume of 3mm edge length. The generated volume includes the same material and component dimensions as the detailed cell, anode, cathode, separator, and includes approximately 1000 solids. To homogenize the stiffness trained on the extracted volume, periodic boundary conditions are defined on all edges.

(a) (b)

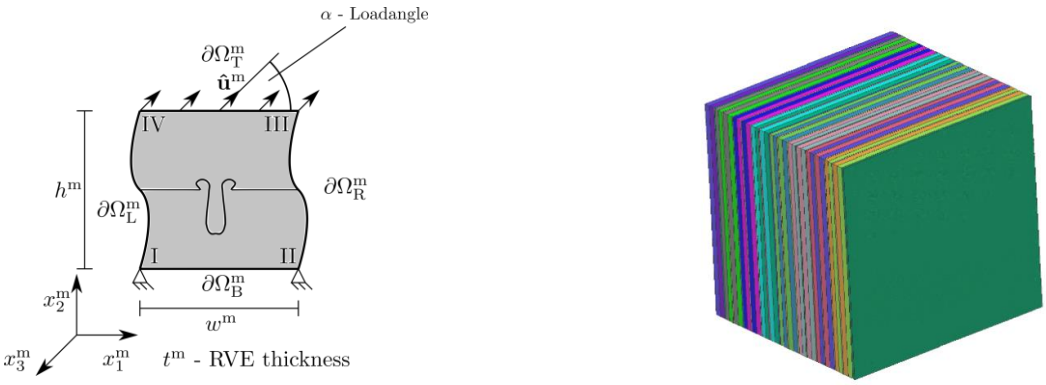

Figure 16: (a) Scheme of periodic boundary conditions and (b) Representative volume element model

The scale bridging between the micro- and the macro scale is accomplished using the HILL averaging principle. It relates the macroscopic virtual work density with the volume average of the total virtual work on the microscale.

The testing of the representative cell elements and training of the TANN phase is performed with an in-house ESI tool which also plots the goodness of the TANN stiffness model for all cases in terms of stress versus strains.

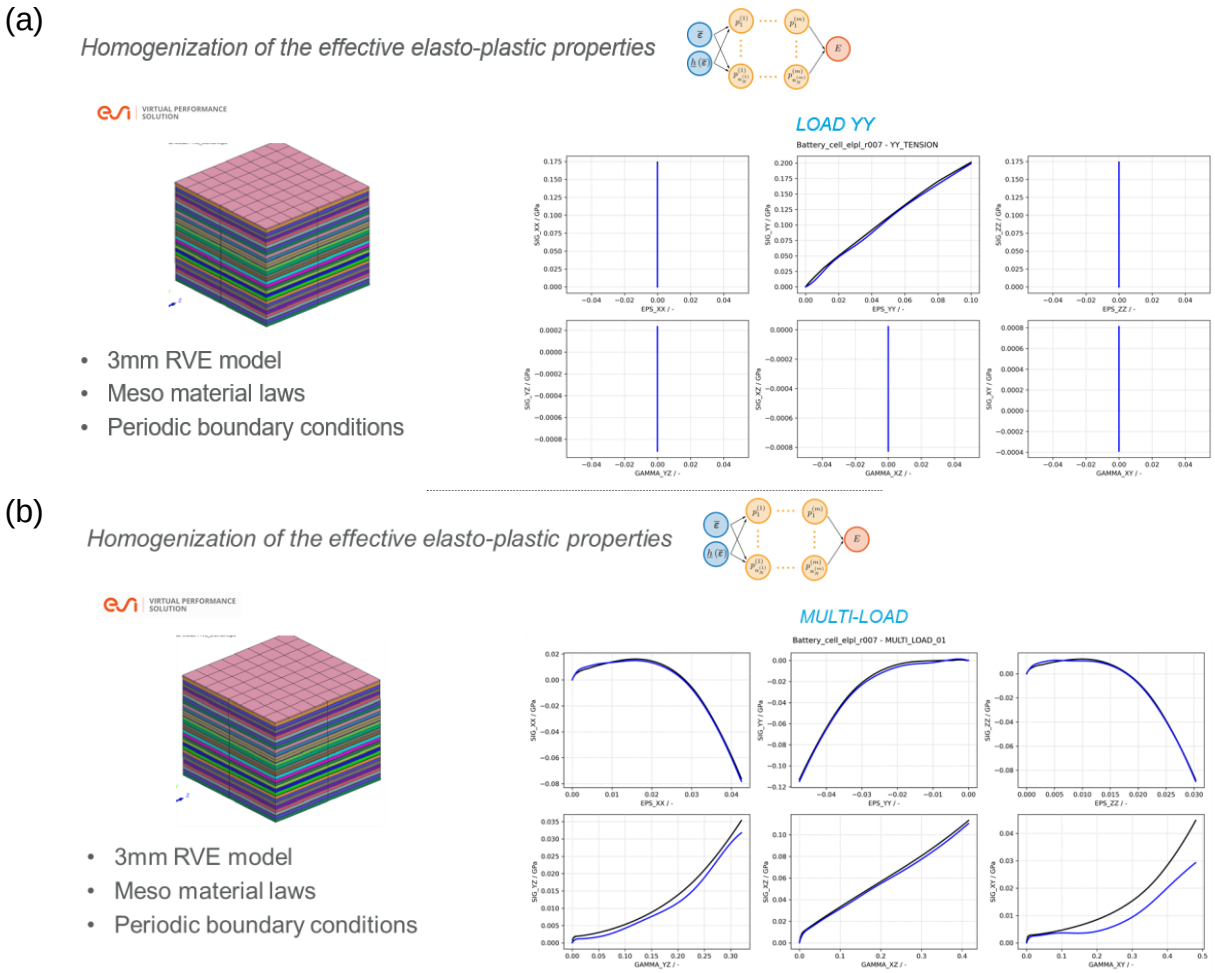

Figure 17: Comparison of RVE and TANN model responses for (a) a unidirectional load and (b) multi loads

The TANN model parameters are then exported directly to Fortran code which can be compiled and deployed as a user material, see next section.

5.5 Model use and deployment

This model, as the honeycomb material with short circuit risk ROM from D3.3 is based on the ESI VPS plugin user material framework which enables the user to program his own material model as Fortran code, and use the compiled library during a run, eg. on the full vehicle simulation. The TANN model for stiffness ROM, possibly with short circuit risk, is embedded in a compiled material library that can be deployed on a HPC cluster.

The plugin library is identified with a library name indicated by the user in the cell material card. At each computational time step, the user routine gets each solid strains, calculate the stiffness and the short circuit from the models described in the previous section and in D3.3, respectively.

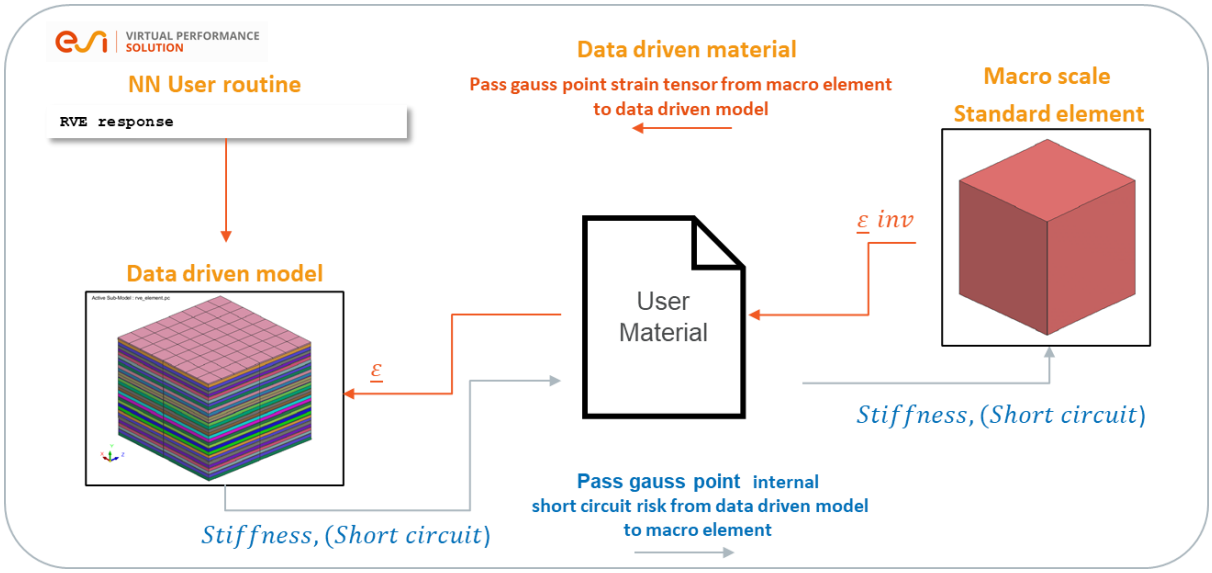

Figure 18: Scheme of resolution with TANN model for stiffness

5.6 VPS material card

The TANN model for the stiffness ROM embeds the TANN parameters directly in the library The 2 parameters that can be accessed by the user are the CXX field to set up some user imposed stiffness parameter to drive the time step of the model, instead of the stiffness value obtained by the TANN model. In addition, the cycling parameter SCYCLE enables the user to tune the stiffness ROM calculation with respect to the full vehicle crash simulation. Typical stiffness calculation cycling are between 1 and 50 and higher values make the computation of the stiffness ROM faster, at the expense of accuracy.

The additional fields set up the risk ROM parameters as explained in D3.2 and D3.3. The AI short circuit tool from D3.2 generates the short circuit ROM as a series of files storing the required classification and regression algorithms. The tool is extended to store the model information in data readable by the user plugin in VPS: table entities. The risk ROM can then predict an internal risk based on the strain field of the stiffness ROM.

```
$#       IDMAT    MATYP          RHO    ISINT    ISHG  ISTRAT   IFROZ
MATER /      1       85  2.55915186E-6        0       0       0       0
$# BLANK                                                     QVM      IDMPD
                                                              1.         0
NAME UPSCALE_Battery_Cell_TANN
$#                                                                 PLUGNAME
ULIB                                                          mat85_battery
$       CXX     SCYCLE     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK
        1.        50
$#                                                                   Card4
$#ILUTSVC1 ILUTSVC2  ILUTSVM1  ILUTSVM2
         1         2        11        12
$#                                                                   Card5
$     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK

$#                                                                   Card6
$     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK

$#                                                                   Card7
$     BLANK     BLANK     BLANK     BLANK     BLANK     BLANK       KSI        F0
                                                                    0.0
$#                                                                   Card8
$     BLANK     BLANK     BLANK     BLANK        Q1        Q2        Q3     BLANK

$---5---10----5---20----5---30----5---40----5---50----5---60----5---70----5---80
```

Figure 19: VPS material card for user library including TANN stiffness ROM and AI short circuit ROM

5.6.1.1 User material library use

A technical description of the user material card is provided above.

Similar as the short circuit ROM, user material plugin for VPS are usually identified by a name “ULIB” and different for simple and double precision launch of the solver. For example, a material called “mat85_short_circuit” for simple and double precision will be named the following:

mat85_short_circuit_dp.so
mat85_short_circuit_sp.so

The plugin material can be selected at launch by setting the *PAMSHARE* variable prior to launch of the VPS solver. This can be performed on a Linux OS HPC by the following:

```
export PAMSHARE=”user/path_to_plugin”
pamcrash model.pc
```

5.7 Stiffness ROM material performance

The TANN user material library is first assessed on a unit cell case for performance reason and computing cost of the model. This enables tuning of the subcycling and of time step controls which are the most critical parameters for cpu performance.

The internal stiffness estimation within the stiffness ROM may impose some lower time step than imposed by the element gauge in the coarse cell model (0.02µs versus 0.05 µs). This can significantly slow the calculation with the stiffness ROM. Stiffness and mass driven time step controls are added to ensure a time step close to the time step of the vehicle of 0.05 µs. This significantly improves the performance.

Another important parameter is the cycling of stiffness update with the neural network with respect to the time step of the solver. Values of 1 to 50 are tested and accuracy is compared. A value between 10 and 50 in terms of subcycling seem the good optimum for accuracy and cpu

efficiency and reach a similar computing time as the pure physical model. The various cases are summarized below.

Case	Time step	Subcycling	Time 10 cpu ind tend=10ms
Reference mat42	**Natural (0.04us)**	**-**	**2 min**
1	Natural	1	33 min
2	Vehicle 0.05us	1	8 min
3	Vehicle 0.05us	10	3 min
4	Vehicle 0.05us	50	2.5min

Table 4: Computing time for the cell model with stiffness ROM and short circuit ROM for various values of subcyling for the stiffness calculation update on 10 processors

The same stiffness ROM model is assessed on the modified e-Golf in a pole impact scenario to assess computing efficiency and scalability with the number of processors (DMP) with a stiffness update every 50 cycles. The model is slightly more costly than the standard honeycomb material model with a 20% increase in CPU time. This added cost could be further reduced with an increased subcycling rate over 50 but the effect on mechanical accuracy should be evaluated.

Model	Gauge	Time 72 skylake
Baseline	3 mm	8.5 hrs
TANN v3	3 mm	10.2 hrs

Table 5: Computing time for the cell model with stiffness ROM and short circuit ROM for various values of subcyling for the stiffness calculation update on 10 processors

5.8 Effect of mesh gauge on performance

Similarly as for the short circuit ROM, the neural net driven model is estimated for various mesh gauges and number of processors to estimate its DMP scaling and computing efficiency. This is assessed on the same modified e-Golf in a pole impact scenario as in the previous section.

Case	Model	Gauge	Time 72 skylake	Time 144 skylake
0	Baseline	3 mm	8.5 hrs	5.5 hrs
1	Baseline	5 mm	6.0 hrs	-
2	Baseline	10 mm	3.4 hrs	-
3	AI short circuit	3 mm	10.2 hrs	-
4	AI short circuit	5 mm	6.1 hrs	4.5 hrs
5	AI short circuit	10 mm	4.0 hrs	2.2 hrs

Table 6: Computing time for the modified e-Golf model with the TANN stiffness ROM approach on 72 skylake processors with final time 80ms for various mesh gauges

The TANN model scales well with number of processors and reducing the mesh gauge significantly decreases the computing time of the model. The computing time is increased slightly as compared to the short circuit ROM but of the same order of magnitude with the correct subcycling. As a note, the TANN stiffness ROM could possibly accommodate coarser mesh

gauge than the honeycomb material law due to having more degrees of freedom within its training.

6 Possible risks and recommendations

The stiffness ROM based on IDMD method as described in section 4 has several attached risks which may prevent a successful deployment on the full vehicle in a timely manner:

- The ROM model deployment in the general interface API is still ongoing improvement at the unit cell level
- The general interface API will be available in the next VPS version which will be available to partners early 2022, which will leave little time to partners to test and iterate
- The stiffness ROM does not yet include the short circuit risk from D3.2 and the method would need to be adapted from the current approach

To mitigate this risk, the approach described in section 5 is proposed: a stiffness ROM based on a neural network (TANN). It is already compatible with the short circuit risk from D3.2 and can be computed on vehicle model with a preliminary DMP assessment and guidelines.

7 Conclusions

An overview of computing efficiency and deployment in FEA code was provided for:

- A honeycomb material model enriched with a short circuit ROM
- A stiffness ROM based on IDMD and interfaced with FEA code leveraging a generic interface API
- A neural net material model (TANN) to represent the stiffness ROM which can be complemented with a short circuit ROM

For the various methods, the effect of mesh gauges and number of processors was investigated to produce some reference times, optimize the computing efficiency and provide some guidelines for full vehicle simulation with AI models.

8 References

[1] A. Géron and an O'Reilly Media Company Safari. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Incorporated, 2019.
[2] S Muller et al, MULTISCALE ANALYSIS OF METAL / THERMOPLASTIC COMPOSITE INTERFACES USING ESI VPS, ICCM17
[3] S Muller et al, Virtual performance characterization of 3D textile reinforced composites: multi scale modelling and efficient scale bridging, ECCOMAS19
[4] R. Hill, 1963. Elastic Properties of Reinforced Solids: Some Theoretical Principles. In: J.Mech. Phys. Solids 11, pp. 357–372.
[5] https://developers.google.com/machine-learning/glossary
[6] S Muller et al, THERMODYNAMICS BASED ARTIFICIAL NEURAL NETWORKS BASED HOMOGENIZATION OF COMPOSITE MATERIALS, ECCOMAS21
[7] Materials > Structural Materials > 3D (Solid) Materials > Usage and Rules > Material 42 in Virtual Performance Solution. Solver Reference Manual, ESI, 2020.5

9 Figures

10 Tables

11 Acknowledgement

The authors want to thank all project partners of the UPSCALE project for their inputs and the positive and constructive working atmosphere in the project.