Development of communication mechanism for the fully coupled multiscale model of 3D compression test

Adam Legwand¹ · Konrad Perzyński¹ · Łukasz Madej¹

Received: 3 December 2019 / Accepted: 4 July 2020 / Published online: 16 July 2020
© The Author(s) 2020

Abstract

The implementation of a dedicated communication mechanism between micro- and macroscale models of the fully coupled multiscale numerical approach is the topic of the paper. The presented solution is based on the System V IPC message queue that is part of modern Linux-based operating systems. Developed and implemented approach is validated with a selected case study of a 3D compression test with the microscale model based on an implicit Johnson–Cook model, which is solved as a separate external process. The communication mechanism can be the basis of the efficient fully coupled FE² model.

Keywords Multiscale modelling · Inter-process communication · Fully coupled model

1 Introduction

The modern metal forming industry must process advanced multiphase materials with precise control of their microstructure evolution as this is the only way to obtain a final product with integrated excellent mechanical and chemical properties. This task presents a great challenge for classical numerical approaches used in the metal forming community, such as the Finite Element models, which usually treat the material with its microstructure as a homogenous medium [1]. In such material models, microstructure features like grain size or volume fraction of particular phases are considered in an average form and are incorporated into the analytical or differential equations used to replicate the flow stress curve [2]. However, as local interactions between particular microstructure constituents within the multiphase materials are responsible for mentioned elevated in-use properties, considering them in an average form is no longer satisfactory. Therefore, explicit microstructure representation approaches are being developed. In this case, each

microstructure feature is directly replicated in the numerical model, which allows taking into account all its local interactions. The microstructure morphology can be replicated based on the experimental data (e.g., computed tomography) or generated numerically. In the latter case, these kinds of models are called synthetic microstructure models [3–6]. Finally, microscale models are used to calculate the flow stress values for the macroscale model, and multiscale modelling framework is established [7, 8].

Nowadays, some simple multiscale modelling techniques can already be found in modern commercial FE software. For example, Simulia Abaqus provides submodelling capabilities, where the macroscopic model is used to provide boundary conditions for the model at a lower length scale. Such a concept is referenced as one-way coupling as data is exchanged only in one direction [9]. Although this feature is useful for investigation of lower scale phenomena, it does not improve the quality of macroscale calculations. In order to improve the quality of predictions at both scales fully coupled data exchange mechanisms have to be used. However, at present, none of

✉ Adam Legwand, alegwand@agh.edu.pl | ¹Faculty of Metals Engineering and Industrial Computer Science, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Kraków, Poland.

the widely used commercial numerical packages provide such feature. Differences between the described communication mechanisms are presented in Fig. 1.

In the case of multiscale models instead of an application of a single numerical method to tackle investigated problems, a set of methods, such as molecular dynamics, cellular automata, Monte Carlo, finite element method etc. can be applied to describe material behavior at different scales. However, as mentioned, the crucial aspect of the multiscale modelling concept is the development of appropriate communication mechanisms between such heterogeneous computing methods. Therefore, the concept of the universal data communication mechanism that allows for a full coupling, i.e. fully synchronized data exchange between e.g. two FEM-based models is the main subject of this paper.

The paper continues with the presentation of the role of communication interface in a fully coupled multiscale model. Selected operating system level techniques are briefly presented, including their possible incorporation into the commercial FE software. Then, the implementation of the system prototype based on Simulia Abaqus software is described. Finally, the results of a simple test case scenario, based on the Johnson–Cook model, are presented.

2 Communication interface

The communication mechanism is a crucial element in the fully coupled multiscale model as it provides transfer of data required by all models during the simulation, i.e., boundary conditions for micro model and material response for a macro model. Because communication is realized during calculations, implemented tools have to be able to access and update data that is generated directly by a numerical solver in each time increment. Additionally, as the data exchange is performed in each increment, time overhead is also an issue that needs to be addressed in order to ensure realistic computing times.

Another important feature of the communication tool is its versatility. It should allow for data exchange between various numerical methods such as above mentioned: finite element method, cellular automata, Monte Carlo or molecular dynamics. Also, it should allow for the data exchange between various commercial and open-source computer-aided engineering software.

To develop such a universal data exchange framework for multiscale analysis, authors decided to divide communication into two interfaces that are attached to instances of the numerical software and interact with each other using a standardized communication protocol. Each of the interfaces is attached to the proper model in order to read and write data that is then sent to the other model. Figure 2 presents a developed communication concept for the macro–micro model with highlighted tasks that are assigned to each element of the interface.

In order to read and write data, which are required during calculations, the communication interface must be based on a set of tools provided by developers of particular numerical software. Usually, the standard functionality of such commercial software can be extended by some form of custom code, such as user subroutines or macros. User subroutines allow to modify and extend the default behavior of numerical engines, while macros can be used to automatically prepare input data and read output data from result databases [10].

As the data exchange between interfaces should create limited computing overhead the standardized communication method is used in this work. Modern Linux-based operating systems provide a set of inter-process communication mechanisms (IPC), which are handled by the system kernel. With that, there is no need for additional 3rd party software. Also, provided mechanisms are compatible with most modern programming languages without the need for additional libraries. There are many different types of IPC mechanisms with various purposes that can be used for such applications. Semaphores and signals are used mainly for synchronization between processes as well as notification. Files and shared memory mechanisms can be

Fig. 1 Diagram showing differences between a one-way and a fully coupled approaches in a multiscale model

Fig. 2 Concept of the data transfer in a multiscale model

used for sharing data, however, read/write synchronization between processes is not provided [11, 12]. Finally, for the synchronized data transfer with well-defined data structures, the message queues can be used, which is the topic of the present investigation (Fig. 3).

The major advantage of the presented communication framework is its reusability as it is based on standardized and generic tools. Therefore, it can be applied to various FE software as long as they provide similar mechanisms of the subroutines used in the Simulia Abaqus. The utilization of operating system level communication mechanism allows also to use various modern programming languages and does not put constraints on the format of data that is transferred. In the present work, the data communication is compatible with both C++ and Python languages.

3 Implementation

Capabilities offered by the commercial Simulia Abaqus FE software were selected for the implementation of the communication framework.

The macro model communication interface was implemented within the VUHARD subroutine. This subroutine allows a user to implement and incorporate into the simulation a customized flow stress model using Fortran or C++ programming language. The available subroutine interface allows to access a series of data e.g. element id, calculation increment, equivalent strain, strain rate or temperature from each finite element Gauss integration point and returns calculated flow stress value for the next step iteration. However, in this work, the VUHARD is just used to extract the selected data from the mesh and send it to the microscale model interface and then receive and return the appropriate results for the next computing increment.

The developed micro model interface is divided into three parts: preprocessor, launcher, and postprocessor. The preprocessor module is responsible for loading input file that contains a description of a micro model used in a simulation and for defining boundary conditions based on data from a macro model. It is required that the input file contains a “restart” parameter definition that allows to chain following simulations. In the first time increment, the preprocessor only loads the initial input file, however

Fig. 3 Message queue diagram

in subsequent time increments “Initial State” boundary conditions are also applied. Therefore, it allows starting a simulation with a stress distribution obtained in a previous time increment. Then, the launcher module initiates the simulation. After the simulation is completed, the post-processor module performs homogenization of stress/strain data in order to prepare output information that will be sent to the macro model. The developed procedure (Fig. 4) allows for direct communication between micro and macro models, which is crucial in a multiscale concept. Code snippets in Fig. 4 are presented to highlight the most important functionalities of the implemented solution.

The described solution has been implemented in the form of a single Python script that contains a set of functions executed sequentially. In this form, the presented mechanism can be launched from a command line using the Abaqus “noGUI” command, which allows running script without a graphical user interface.

The main objective of the micro interface is to wait for data from the macro model interface, run a simulation of the micro model and then process output and send results to the macro model. In the presented prototype, instead of

FEM-based simulation at the microscale, a Johnson–Cook equation is used as a case study to calculate flow stress value for a given equivalent strain value. Such a simple case study is used to validate the developed mechanisms and to check if they do not introduce any unphysical artifacts.

For communication between interfaces, two message queues are used. The first is called REQUEST_MQ, which is used to transfer data from the macro model to the micro model. The second is called RESPONSE_MQ, which transfers data back from the micro to the macro model. This naming convention imitates client-server architecture, where requests are sent by a client and responses by a server (Fig. 5).

In order to transfer data between interfaces, it must be agglomerated in the form of dedicated messages. Although C++ library allows defining the structure of the message, in the Python by default the message is cast to a string array. Because of that, in the developed communication framework prototype, the data that is transferred is always translated into a string array before putting it into the queue.

Fig. 4 Diagram of the micro model interface

Fig. 5 Diagram showing message queue

4 Test case

As mentioned, in order to validate the implemented communication interface, the simple case study was proposed that consists of three variations of the microscale flow stress model as seen in Fig. 6.

The macroscale numerical model of the uniaxial 3D cube compression test was prepared as the basis for the investigation. Boundary conditions were applied to the bottom and top surfaces, as presented in Fig. 7. Displacement is represented as BC_DISPL and restraint of all degrees of freedom is represented as DC_LOCK. The model was discretized by 1000 8-noded hexahedrons.

All three variations of the microscale flow stress model are based on the Johnson–Cook (JC) approach. In the first variant, the built-in JC model provided by Abaqus software was used to describe the flow behavior of the investigated ferritic steel. In the second variant, a conventional

Fig. 7 Simple compression test used for the test case of the implemented communication solution

JC model implementation within the VUHARD subroutine was used. Finally, in the last variant, the JC model was implemented as a separate instance of Abaqus. In this

Fig. 6 Diagram showing performed test cases

Fig. 8 Stress distribution in the deformed cubic sample computed with Johnson–Cook model **a** built-in model, **b** VUHARD subroutine, **c** Python macro executed at separate processes

case, the developed communication mechanisms were used. The VUHARD based interface was attached to the described macroscopic model of the compression test and the Python-based interface was attached to the separate Abaqus process with the Johnson–Cook equation used to calculate flow stress based on the data received from the message queue.

During these tests, stress distribution, homogenized flow curves and execution times were compared between the three different variants to evaluate the impact of developed communication interface on results and computation times. Examples of obtained results are gathered in Fig. 8.

As demonstrated in Fig. 8, there is no impact on stress distribution after the compression, which proves that the developed communication mechanisms do not affect the data consistency in any artificial way. Similar behavior is visible in Fig. 9, where despite the data type conversion there is no significant influence on predicted homogenized flow stress curves from the three investigated variants.

However, the developed communication framework has a significant influence on computational times as shown in Table 1.

The total computational time of simulation which is based on the message queues is approx. 30 times greater in comparison to a standard, built-in JC model within Abaqus software. Therefore, after initial investigations, the authors decided to improve the prototype of the communication interface by changing the procedure of data packing into a message. The initial method, which was based on string operations was replaced by the byte arrays. With that, after repeating all three test scenarios, the total time was reduced by half compared to the initial implementation of the queue-based method.

It should also be mentioned that computation times can be further reduced by parallelization mechanisms that are implemented in Simulia Abaqus. Both macro- and microscale models can be parallelized by adjusting the

Fig. 9 Comparison of homogenized flow stress curves obtained from three tested JC model variants

Table 1 Total computational times of FE compression test obtained from three investigated test cases

Method	Relative time (%)
Abaqus Johnson–Cook model	100
Subroutine based method	109
Message queue-based method	3253
Improved message queue-based method	1718

setup options of a solver. Additionally, the use of the IPC mechanism allows for a seamless transition from single-thread computations to multi-threaded computations by creating more message queues or by using identifiers to distinguish threads. However, the presented solution is not designed to be applied for simple implicit models such as Johnson–Cook, which was used here just as a validation case study. Its purpose is to allow to use of fully coupled complex explicit microstructure models that presently cannot be used in the commercial software. Therefore,

computing times are not the major priority at the current stage of the work.

5 Summary

Presented research has proven that it is possible to implement a universal communication mechanism that is based on the commercial numerical finite element software. Available tools, such as subroutines and scripting interfaces, allow implementing a communication interface to extract required data outside the solver engine during the simulation. Using IPC mechanisms, that are implemented at the operating system level, allows limiting data and time overhead although it requires efficient interface implementations. In the current state, general performance is reduced mainly by some coding shortcuts. However, some initial work in the area of datacasting showed that the final performance of the presented solution can be improved.

The future work is focused on the application of the developed communication framework to the fully coupled FE² model, where the microscale model will also be the finite element model based on the representative volume element concept.

Funding The work was realized as a part of fundamental research financed by the Ministry of Science and Higher Education, Grant No. 16.16.110.663. FEM calculations were realized at the AGH ACK CYFRONET computing center.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Werner E, Wesenjak R, Fillafer A, Meier F, Krempaszky C (2016) Microstructure-based modelling of multiphase materials and complex structures. *Contin Mech Thermodyn* 28:1325–1346. <https://doi.org/10.1007/s00161-015-0477-7>
2. Szyndler J, Imran M, Afzal MJ, Bambach M (2018) Numerical analysis of damage during hot forming. *Comput Methods Mater Sci* 18:107–114
3. Perzynski K, Wang J, Radwanski K, Muszka K, Madej L (2019) Identification of critical strains for the random cellular automata finite element failure model based on in situ tensile test. *Mech Mater* 133:154–164. <https://doi.org/10.1016/j.mechmat.2019.03.013>
4. Madej L, Sitko M, Legwand A, Perzynski K, Michalik K (2018) Development and evaluation of data transfer protocols in the fully coupled random cellular automata finite element model of dynamic recrystallization. *J Comput Sci* 26:66–77. <https://doi.org/10.1016/j.jocs.2018.03.007>
5. Langston P, Kennedy A (2014) Discrete element modelling of the packing of spheres and its application to the structure of porous metals made by infiltration of packed beds of NaCl beads. *Powder Technol* 268:210–218. <https://doi.org/10.1016/j.powtec.2014.08.018>
6. Doroszko M, Seweryn A (2020) Pore-scale numerical modelling of large deformation behaviour of sintered porous metals under compression using computed microtomography. *Mech Mater* 141:103259. <https://doi.org/10.1016/j.mechmat.2019.103259>
7. Horstemeyer MF (2009) Multiscale modeling: a review. In: Leszczynski J, Shukla MK (eds) *Practical aspects of computational chemistry: methods, concepts and applications*. Springer, Dordrecht, pp 87–135
8. Horstemeyer MF (2012) *Integrated computational materials engineering (ICME) for metals*. Wiley, Hoboken
9. Pietrzyk M, Madej L, Rauch L, Szeliga D (eds) (2015) Increase model predictive capabilities by multiscale modelling. In: *Computational materials engineering, achieving high accuracy and efficiency in metals processing simulations*, Elsevier, pp 209–253
10. Sarkar S, Singh IV, Mishra BK, Shedbale AS, Poh LH (2019) A comparative study and ABAQUS implementation of conventional and localizing gradient enhanced damage models. *Finite Elem Anal Des* 160:1–31. <https://doi.org/10.1016/j.finel.2019.04.001>
11. Silberschatz A, Galvin PB, Gagne G (2011) *Processes*. In: Golub B (ed) *Operating system concepts*, 9th edn. Wiley, New York, pp 105–136
12. Wolf M (ed) (2017) *Processes and operating systems*. In: *Computers as components: principles of embedded computing system design*, 4th edn. Morgan Kaufmann, pp 321–390

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.