Development of efficient data transfer protocols in the random cellular automata finite element dynamic recrystallization model

Adam Legwand, Mateusz Sitko, Kazimierz Michalik, Konrad Perzyński, Łukasz Madej*

Faculty of Metals Engineering and Industrial Computer Science, AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: alegwand@agh.edu.pl, msitko@agh.edu.pl, kamich@agh.edu.pl, kperzyns@agh.edu.pl, lmadej@agh.edu.pl

Abstract

Application of the hybrid random cellular automata finite element (RCAFE) approach to modelling dynamic recrystallization phenomenon occurring during high temperature deformation is the overall goal of the carried research. The present work is particularly focused on development of efficient data transfer mechanisms between developed in-house random cellular automata code and commercial finite element software. First, a short description of the RCAFE model concept and its advantages is presented. Then, proposed communication approaches with two different levels of complexity based on text files and sockets are discussed. Their capabilities and limitations are also evaluated within the paper.

Keywords: random cellular automata method, finite element method, dynamic recrystallization

1. Introduction

Random cellular automata finite element (RCAFE) method is a combination of two numerical approaches that work in a fully coupled manner and allow to predict kinetics as well as microstructure morphology evolution of investigated phenomena. This approach has already been successfully applied by the authors to modelling fracture initiation and propagation in dual phase steels [1].

The same concept is presently applied to predict microstructure evolution in steels under high temperature deformation. The key phenomenon in this case is the dynamic recrystallization (DRX), leading to nucleation of new grains and their subsequent growth. Thus, microstructure evolution is modelled by the random cellular automata algorithm, that can explicitly simulate changes in the grain size and recrystallized phase fraction. The finite element model on the other side predicts mechanical response of the material to local softening occurring due to progress in recrystallization.

However, the key aspect in a fully coupled approach is bilateral information exchange between two models, that have to be realized in each time step. The data exchange mechanism has to be stable, reliable and also time efficient. At the same time it has to be feasible for implementation within the framework of commercial finite element software. Two concepts of such communication mechanisms are presented and discussed in the following chapters.

2. RCAFE model for DRX

In the proposed RCAFE approach, the FE module was developed within the commercial numerical package Simulia Abaqus, while the RCA part is a proprietary software written in C++. In case of dynamic recrystallization modelling, FE module is responsible for calculation of equivalent stress and strain fields in each time step for subsequent integration points in the investigated computational domain. These data are then transferred to the RCA module, which evaluates evolution of dislocation density changes. These changes are the major driving force for evolution of microstructure morphology by nucleation and grain growth. Then, based on calculated recrystallization fraction, flow stress value is being evaluated and transferred back to each integration point in the FE module as input for the following time step. Details on the RCA model can be found in earlier work [2].

To fully couple the RCA and FE models, the user defined subroutines available in the Abaqus software and system of external text files can be used. The other solution to handle such communication, which is tested within the work is based on sockets implementation.

3. Data transfer using text files

The first developed method of data transfer between RCA and FE modules is the use of system of text files. This is the most basic and straightforward solution that offers minimal set of features required for successful communication, which are related to write/read data operations to/from text file. In order to enable such data transfer mechanism, simple functions related with file handling (open, read, write, close) are required, although those are usually provided as a part of standard built-in libraries. The approach can be easily implemented in the Abaqus user defined procedure Vumat. In this case, for each specified time step new text file is created where selected data from each integration point are saved in a simple format: one per line. After all data for that time increment is saved, file is closed and then can be used as input for the RCA module, that is implemented as a separate program outside of the Abaqus framework. As an output from the RCA part, another text file with modified flow stress values for the FE is created. In this case, each of the two moduli is checking if a specific file is already created and filled with data. Simple loop with specified timer function is used in this case. During calculations realized by the RCA, the FE model is idle, and is waiting for the appropriate file. Concept of the data transfer diagram is presented in Figure 1.

Main advantage of the presented solution is simplicity of implementation regardless of programming language and working environment that is selected. However, large batches of files are generated in this case and read/write processes limits its efficiency and extends the computing time. This may be a limiting factor especially for the 3D calculations.

*Financial assistance of project no. UMO-2014/13/B/ST8/03812 is acknowledged.

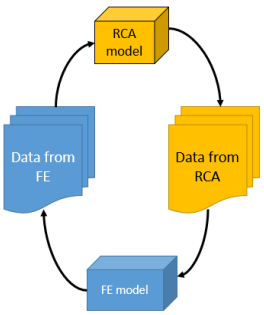

Figure 1: Concept of data transfer using text files.

4. Data transfer using sockets

An alternative for mentioned text files, is inter-process communication (IPC) sockets mechanism. It allows for data communication between processes on the same host or different hosts using data streams. Current implementations allow to use various communication protocols including network protocols such as TCP, IPv4 and IPv6 [3]. Most common implementation are Berkeley sockets, sometimes called POSIX sockets, as they are currently component of POSIX specification. Berkeley sockets allows to implement simple client-server system, which is based on the TCP protocol.

The main advantage of the socket solution in the RCAFE model is lack of text files that are generated and stored on the physical drive. Data is transferred directly between running processes namely FE solver and RCA program so when data from specific integration point is obtained it can be sent directly to the CA module and vice versa. Also, instead of waiting for all data that is required in specific increment, each application can call only for data of particular finite element. Concept of data transfer in the approach is presented in Figure 2.

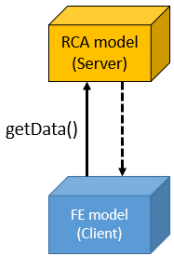

Figure 2: Concept of data transfer using sockets.

The main disadvantage is requirement of handling communication between nodes by adding additional messages that inform each side what kind of data are transferred. Also, opening and closing socket communication has to be synchronized, especially at the Abaqus subroutine side, as it is launched many times during one increment. The reason for that is that Abaqus user subroutines are working on limited number of finite elements at the same time. The last concern is that, depending on the development environment, proper socket libraries have to be used.

5. Results

Both mentioned approaches, have been successfully implemented and the fully coupled RCAFE model was created. To verify implemented solutions simple model of compression test has been prepared. Sample has been discretized using about 12300 elements. Examples of results obtained from the numerical simulation of channel die compression of micro sample case study are presented in Figure 3.

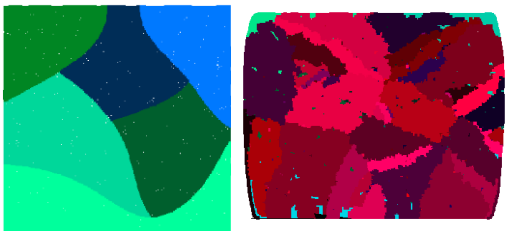

Figure 3. Initial and final microstructure morphologies of the material deformed at 900 °C.

From the present case study it seems that the computation time for solution using file-based communication was 60% faster than the same simulation but with socket-based communication. Although results clearly show that first solution is much more effective, it must be noted that performance of current version of socket-based communication is strictly limited by the way Abaqus software works. By default every call of subroutine function can handle up to 136 elements, so in order to iterate over whole finite elements space it has to be called multiple times. As a result, in every call, socket connection has to be reinitialized. For presented test case, for every time increment, socket connection has to be initialized about 90 times, while single socket initialization takes about 500ms. For presented test scenario about 60% of simulation time was related to socket initialization. Further development of this solution should be focus mainly on two issues: limiting number of socket initializations per time increment or development of additional mechanism that will allow to use only one socket instance per whole simulation.

Summary

Based on the presented results it can be concluded that:

- basic concepts of data transfer mechanism using text files has its advantages in the form of simple implementation and operating system independency. However, it generates large set of data and its efficiency is limited by read/write operations.
- socket concepts use more sophisticated solutions and offer additional functionalities however require use of more complex programming techniques. Also, because of Abaqus software limitations, sockets initialization generates large time overhead, which result in worse efficiency comparing to files-base communication solution. This will be still investigated during further research.

References

[1] Perzynski, K., Madej, L., Complex hybrid numerical model in application to failure modelling in multiphase materials, *Archives of Computational Methods in Engineering*, 2016, in press.

[2] Legwand, A. Sitko, M., Perzyński, K., and Madej, Ł., RCAFE Based Numerical Model of Dynamic Recrystallization, *Journal of Machine Engineering*, Vol. 16, No. 2, pp. 52-60, 2016.

[3] Stevens, W. R., Rago, S.A., Advanced Programming in the UNIX Environment UNIX, third ed., Addison-Wesley Professional, 2015.