Meso– and multi–scale modelling E–CAM modules V

E-CAM Deliverable 4.6
Deliverable Type: Other
Delivered in March, 2021

E-CAM
The European Centre of Excellence for
Software, Training and Consultancy
in Simulation and Modelling

Funded by the European Union under grant agreement 676531

Project and Deliverable Information

Project Title	E-CAM: An e-infrastructure for software, training and discussion in simulation and modelling
Project Ref.	Grant Agreement 676531
Project Website	https://www.e-cam2020.eu
EC Project Officer	Juan PELEGRIN
Deliverable ID	D4.6
Deliverable Nature	Other
Dissemination Level	Public
Contractual Date of Delivery	Project Month 62(November, 2020)
Actual Delivery Date	12th March, 2021
Deliverable Description	Final Software modules delivered to the E-CAM repository in the area of meso and mulit-scale modelling responding to requests of users, and their documentation.

Document Control Information

Document	Title:	Meso– and multi–scale modelling E–CAM modules V
	ID:	D4.6
	Version:	As of 12th March, 2021
	Status:	Accepted by WP leader
	Available at:	https://www.e-cam2020.eu/deliverables
	Document history:	Internal Project Management Link
Review	Review Status:	Reviwed
Authorship	Written by:	Jurij Sablić(University of Barcelona)
	Contributors:	Jony Castagna (STFC), Alan O´Cais (JSC), Ana Catarina Mendonça (EPFL), Rene Halver (JSC), Stephan Schulz (JSC)
	Reviewed by:	Burkhard Dünweg (Max Planck Institute for Polymer Research), Godehard Sutmann (JSC)
	Approved by:	Burkhard Dünweg (Max Planck Institute for Polymer Research)

Document Keywords

Keywords:	E-CAM, CECAM, Module, DL_MESO_DPD, GPU, DPD, coarse–graining, load–balancing, Lattice-Boltzmann

12th March, 2021

***Disclaimer**:This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the Project and to the extent foreseen in such agreements.*

[1] jurij.sablic@gmail.com

Contents

Executive Summary

In this report for Deliverable 4.6 of E-CAM, we present nine software modules in meso– and multi–scale modelling. The modules are divided into three groups. The first one contains three modules, incorporated within the DL_MESO_DPD simulation code [1, 2]:

- Many-body DPD model on multi-GPU
- Incorporation of Kokkos into the first integration step of the Velocity Verlet algorithm
- Incorporation of Kokkos into the second integration step of the Velocity Verlet algorithm.

With the latter two we improve the portability and computational efficiency of the code by offloading both steps of the Velocity Verlet integration scheme, while with the first module, we proceed with the development and optimisation of multi-GPU architectures of the code. The ongoing development of the code within the scope of E-CAM has previously been reported in five deliverables on meso– and multi–scale modelling [3, 4, 5, 6, 7].

The second group of modules concerns further development and application of A Load-balancing Library (ALL), which was first reported in D4.5 [7]. It consists of four modules:

- ALL C++ interface
- ALL Fortran interface
- ALL library in Material Point Method
- ALL library in HemeLB workflow

In the first two we present interfaces that enable the usage of the library in C/C++ and Fortran-based codes. In the last two, on the other hand, we report on the incorporation of the library into two simulation packages, where ALL contributes to the computational optimisation in studies of deformations of continuous media as well as of blood flow.

The last group of modules provides additional features in the Lattice–Boltzmann code Ludwig [8]:

- Externally imposed chemical potential gradient for binary fluid mixture
- Implementation of crystalline capillaries.

The first one is used in the studies of diffusiophoretic and diffusioosmotic phenomena in binary fluid flows through different capillaries. The crystalline ones are introduced into Ludwig by the last module of this group.

For each module, we provide an explanation about its implementation, usage, and the motivation for its development. The description is accompanied by a link to the respective Merge-Request on the GitLab repository of E-CAM, where we provide further information about the code development, testing and documentation of the modules.

As this deliverable is the last one of its series, a section on the overall impact of the results achieved within the Work Package (WP) meso– and multi–scale modelling is also included, with an overview of the results achieved so far, how these have been disseminated in scientific publications, the industrial impact of the software developed, the training given in the WP which lead to many of the software outputs, and the societal impact of the WP.

1 Introduction

WP 4 is concerned with simulations at the mesoscopic scale. As the name indicates, the aim of these methods is the description of physical phenomena whose relevant time and space scales are somewhere in between the microscopic and macroscopic ones. The simulation methods used to simulate the former, such as Molecular Dynamics, often become computationally too demanding for the description of the mesoscale phenomena, while the macroscopic methods and theories, such as continuum (fluid) mechanics, are too coarse to capture them. Consequently, there have been many efforts invested into the development and optimization of simulation techniques which would provide a better insight into the behavior of matter on the mesoscale.

In this deliverable, we present nine modules. Three of them introduce new features into the DL_MESO_DPD simulation package [1, 2]; four of them describe developments and adoptions by applications of the load balancing library ALL; and two of them are incorporated in the Lattice–Boltzmann code Ludwig [8, 9].

1.1 Overall scope of the module set

The modules presented in this report can be divided into three groups. In the the first one, we incorporate new features into the DL_MESO_DPD [1, 2] simulation package, which provides a toolbox to do Dissipative Particle Dynamics (DPD) simulations. DPD is in principle Molecular Dynamics, where atoms are replaced by effective beads that represent many microscopic degrees of freedom. The force exerted on any of these beads has three contributions. The first one is due to the soft interaction between the beads, the second one is a pairwise dissipative force, and the third one is a pairwise random force with thermal Langevin noise, which represents the effect of those degrees of freedom that are not explicitly taken into account. The DL_MESO packages has previously (in [3]) been identified as one of the important codes for mesoscale simulations within E-CAM. The modules that are being presented here (in Section 2) have been developed by the E-CAM programmer Jony Castagna.

In the second group, we present modules based on the load balancing library ALL. Particle simulations are usually parallelized by the domain-decomposition technique, which means that every processor administrates a different spatial region of the simulation domain and keeps track of those particles that are located in it. The workload of each processor is thus mostly determined by the number of particles its domain contains and the number of interactions that are evaluated in it. In some physical systems (e.g. condensing fluids), where during the course of simulation particles accumulate in one part of the simulation box, this may lead to the loss of scalability. If the part of the computational domain where the particles accumulate is for instance located on one processor, the latter will have a higher computational load and will be slower in comparison with the others. Since the overall run time of a simulation is usually determined by the slowest process, due to the synchronization between processes, such an inhomogeneity in load distribution may lead to the loss of scalability, which is especially pronounced in strong scaling behavior. The solution to this problem is the inclusion of dynamic load balancing techniques, which redistribute the workload among the processors, by lowering the load of the most busy cores and enhancing the load of the most idle ones. To this end, we have developed A Load-balancing Library (ALL) within the scope of the E-CAM project, that started in the context of an Extended Software Development Workshop (ESDW). In [7], we reported the first modules on this topic and here we present ALL's further development and application.

The final group of modules introduces new features into the Ludwig code. This is a code for the Lattice–Boltzmann method, which enables simulation of fluids by solving the Lattice–Boltzmann equation. Despite it being a mesoscopic method, its main purpose is usually the simulation of hydrodynamics. Beyond simple fluids, complex systems such as liquid crystals, active and binary fluids may be simulated as well; this is typically facilitated by a coupling to a free-energy based model.

1.1.1 Extension and improvements of the DL_MESO_DPD code

The first three modules are based on the DPD code DL_MESO_DPD. The first module (Sec. 2.1) expands the usage of many-body DPD from a single-GPU version, presented in [7], to multiple GPUs. The many-body version of DPD overcomes the limitation of studying systems with quadratic equations of state. Here, the forces are based upon a local density, which in turn is based on the interaction of all particles within a cut-off radius. The scheme's extension to multiple GPUs enables simulations of larger systems on longer timescales that were unreachable before.

The second and third module, presented in Secs. 2.2 and 2.3, respectively, both incorporate the Kokkos library into the DL_MESO_DPD code. The library is used to offload the steps of the Velocity Verlet (VV) algorithm, which is executed in every step of the DPD simulation and whose computational optimization greatly affects the performance of the code. The second and third module of this group optimizes the first and the second integration step of the VV algorithm, respectively.

1.1.2 Improving usability of the ALL library and application in simulation software

In the next set of modules within this deliverable, we improve the usability of the ALL load-balancing library, which is developed in the Simulation Laboratory Molecular Systems of Jülich Supercomputing Centre at Forschungszentrum Jülich, in four directions.

The first module, described in Sec. 3.1, introduces a C++ interface into the library. This is an essential feature in order to incorporate the library in any C or C++ based simulation codes.

The second module (Sec. 3.2) provides an interface that enables the ALL library to be used in Fortran codes. This enhances the usability of the library, as many codes used in scientific computing are written in Fortran.

The third module, presented in Sec. 3.3, incorporates ALL in a Material Point Method (MPM) simulation code, thus enhancing its computational performance and broadening the usability of the library onto continuum spatiotemporal scales.

The final module in this group results from the collaboration between the ALL library developers and the CompBioMed HPC Center of Excellence. The module is described in Sec. 3.4 and connects ALL to HemeLB, which is a highly scalable Lattice–Boltzmann code, optimized for simulations of blood flow. This results in a better theoretical load distribution of HemeLB and expands the applicability of the ALL library to the Lattice–Boltzmann simulation method.

1.1.3 New features in Ludwig

The last two modules presented in this deliverable extend the usability of the Lattice–Boltzmann code Ludwig.

The first module, described in Sec. 4.1, introduces the possibility of performing the simulations under constant external chemical potential gradient, which is an essential feature in studies of diffusioosmotic and diffusiophoretic phenomena.

In the second module, presented in Sec. 4.2, we incorporate into the code new crystalline geometries, which represent porous media.

1.2 General applications and possible exploitation of the codes

DPD is regularly used in industry in order to describe the static and dynamic behaviour of soft-matter systems, such as colloidal dispersions, emulsions and other amphiphilic systems, polymer solutions, etc. The understanding of properties of such materials is important in biomedicine as well as in cosmetic, food, pharmaceutical and other industries. Developing the code for GPUs (Sec. 2.1) and computationally optimizing the fundamental parts of the algorithm (Secs. 2.2 and 2.3) are thus of substantial importance for industry as they make computations cheaper and enable simulation on longer spatial and temporal scales. The ultimate intention of these three modules for DL_MESO_DPD is to simulate a large drop of water between two surfaces. The simulation is in fact of high interest in the ink injection industry and represents a common challenge in many manufacturing processes.

In this deliverable, we also continue extending the applicability of the ALL library. The C++ interface, described in Sec. 3.1, besides being an essential feature of ALL also makes it potentially useful in increasing the computational efficiency in any HPC environment, where C++ is a popular programming language. Next, the Fortran interface, presented in Sec. 3.2, extends the usability of the library to Fortran, which is still very frequently used in the scientific computing community. Besides that, in Secs 3.3 and 3.4 we extend the usability of ALL from particle-based methods to grid-based ones. We incorporate it into the MPM code GMPM-PoC (Sec. 3.3) and into the Lattice–Boltzmann code HemeLB (Sec. 3.4). In the former, ALL optimizes the performance of the code in macroscopic studies of deformation phenomena, while in the latter it may potentially lead to an optimization of the simulation of blood flow through geometries which mimic the human vascular system.

The Lattice–Boltzmann method, as a hydrodynamics solver, has a huge range of applications in earth and energy sciences, bio-medicine, soft matter theory, and countless more. The modules based on Ludwig, presented in Secs. 4.1 and 4.2, are used to study the phenomena of diffusioosmosis and diffusiophoresis. Diffusioosmosis occurs, for instance, in microfluidic devices, while diffusiophoresis has a potential application in, for example, membraneless water filtration, which has recently been extensively studied. Furthermore, understanding the behavior of chemically driven binary fluid flows through porous media, especially near the fluid-fluid critical point, may also have potential technological applications. The modules presented in this deliverable open up the possibility to study such systems, and interest in this work has been demonstrated by Unilever.

1.3 How to read this report

For each module, we give a short overview, followed by links to the `Merge-Request` and the `Documentation` on the GitLab service of E-CAM, which shows detailed information about code development, testing and documentation. The documentation shows how to use the modules in practice, while possible practical exploitation and industrial applications have been outlined above (partly, with some additional remarks added below).

As this deliverable is the last one of its series, a section on the overall impact of the results achieved within the WP is also included, with an overview of the results achieved so far, how these have been disseminated in scientific publications, the industrial impact of the software developed, the training given in the WP which lead to many of the software outputs, and the societal impact of the WP.

2 Modules based on DL_MESO_DPD

The base code for the following three modules is DL_MESO_DPD [1, 2], the DPD code from the mesoscopic simulation package DL_MESO, developed by M. Seaton at Daresbury Laboratory. This open source code is available from Science and Technology Facilities Council (STFC) under both academic (free) and commercial (paid) licenses.

The single- and multi-GPU versions of DL_MESO_DPD have been developed by Jony Castagna and reported as modules in D4.2 [4], D4.3 [5], D4.4 [6], and D4.5 [7]. The tests of its performance on up to 4096 GPUs were presented in D4.4 [6].

2.1 Many-body DPD model on multi-GPU

This module describes the implementation of the many-body DPD scheme in the multiple GPU version of the simulation code DL_MESO_DPD.

2.1.1 Module description

This module consists of the implementation of many-body DPD on the multi GPU version of DL_MESO_DPD. The new feature overcomes the difficulty of simulating systems with non quadratic equation of state and enables simulations of complex phenomena.

For good performance, we simultaneously execute the communication of particle positions and of local densities on the one hand, and parts of the force calculation on the other. This is achieved via splitting the force calculation in two parts: f' the contributions due to the particles in the internal domain cells and f'' the contribution due to the particles in the boundary and halo cells. During the calculation of f' the local densities and the particle positions are swapped between GPUs, while f'' is calculated subsequently.

2.1.2 Motivation and exploitation

One of the main weak points of the standard DPD model is its equation of state. For example, compressible gas or vapor liquid mixtures are difficult, if not impossible, to be simulated correctly. The many-body DPD approach allows us to overcome this limitation by extending the range of interaction to a large cut-off radius. With this modification we are able to simulate complex systems like liquid drops and multiphase flows.

Merge Request	Merge Request for Many-body DPD scheme on multi-GPU
Direct Documentation Link	Many-body DPD scheme for DL_MESO_DPD on multi-GPU

2.2 Incorporation of Kokkos into the first integration step of the Velocity Verlet algorithm

In this module, we present the first version of DL_MESO_DPD with the Kokkos library, which offloads one of the main steps of the time marching scheme.

2.2.1 Module description

The Kokkos library offloads the first step of the Velocity Verlet (VV) scheme. The implementation of the library into the code enables the execution of DL_MESO_DPD on NVidia GPUs as well as on other GPUs or architectures (many-core hardware such as Intel Knights Landing). The goal of this implementation is to achieve portability among accelerators and CPU architectures without sacrificing performance. This also helps us to implement a conceptual separation between computational science and the HPC infrastructure where the code is executed.

Kokkos is a C++ library, while DL_MESO_DPD is written in Fortran 90. The current implementation requires a transfer from Fortran to C++, due to the use of Fortran pointers that do not conform to the ISO_C_BINDING standard.

2.2.2 Motivation and exploitation

In DL_MESO_DPD the coordinates and momenta of particles are propagated in time by the VV scheme, which consists of 3 steps: 1) first velocity integration by $\Delta t/2$ and particle positions integration by Δt, 2) force calculation and 3) second velocity integration by $\Delta t/2$. The performance of this part of the algorithm significantly affects the performance of the whole code. To this end, we use Kokkos, which offloads the workload of the first part of the VV algorithm.

Merge Request	Merge Request for DL_MESO (DPD) on Kokkos: Verlet Velocity step 1
Direct Documentation Link	DL_MESO (DPD) on Kokkos: Verlet Velocity step 1

2.3 Incorporation of Kokkos into the second integration step of the Velocity Verlet algorithm

This module is related to the previous one (Sec. 2.2). The Kokkos library is used to offload the second loop of the VV algorithm. It is implemented in DL_MESO_DPD in order to achieve portability combined with high performance.

2.3.1 Module description

Kokkos library offloads the second integration step of the VV scheme. The main difference in comparison with Sec. 2.2 is that we are using a *parallel_reduce* loop rather than *parallel_for* as we also need to calculate the stress tensor via reduction operations.

2.3.2 Motivation and exploitation

The motivation for this module is identical to that of Sec. 2.2.

Merge Request	Merge Request for Incorporation of Kokkos into the 2nd step of the VV algorithm
Direct Documentation Link	DL_MESO (DPD) on Kokkos: Verlet Velocity step 2

3 Modules based on the ALL library

In this section we describe the modules that broaden the usability and applicability of the ALL library, whose main aim is to provide an efficient way of dynamic domain-based load balancing in particle-based simulation packages. The library has been developed in the Simulation Laboratory Molecular Systems of the Jülich Supercomputing Centre at Forschungszentrum Jülich. All additional information about the library and the source code are available at http://slms.pages.jsc.fz-juelich.de/websites/all-website/.

3.1 ALL C++ interface

The module presents the C++ interface in the ALL library. This is a necessary feature for all users to couple their codes, based on C or C++, to the library.

3.1.1 Module description

The interface is part of the ALL library and can be found at https://gitlab.version.fz-juelich.de/SLMS/loadbalancing in the `include` subdirectory and is contained in `ALL.hpp` (which uses C++ template metaprogramming). As the C++ interface is the default interface of the ALL library, there is no need to explicitly enable it. The interface provides functions to create an object, which handles the computations of new boundaries based on the provided sets of domain borders and work loads. In addition, if the library is compiled with VTK support (requires setting the CMake variable `CM_ALL_VTK_OUTPUT`), functionality to create VTK descriptions of the domain structure is provided (for all methods working on orthogonal domains).

3.1.2 Motivation and exploitation

The C++ programming language is commonly used in HPC environments. Therefore the default interface for the ALL library is written in this language. The library uses class inheritance to administrate the different load-balancing methods. Every necessary functionality to use the library is provided by the interface.

Merge Request	Merge Request for ALL C++ interface
Direct Documentation Link	ALL C++ interface

3.2 ALL Fortran interface

This module introduces a Fortran interface into the ALL library, thus extending its usability to codes based on Fortran, which is still frequently used in scientific computing.

3.2.1 Module description

The interface is implemented in the `ALL_module.F90` file inside the `src` subdirectory of the ALL library code, which is available at https://gitlab.version.fz-juelich.de/SLMS/loadbalancing. An internal C wrapper is used for the C++ class, since Fortran can only interoperate with C. The Fortran interface must be explicitly enabled when building the library. This is done by setting the "CMake" variable `CM_ALL_FORTRAN` to `ON`. The use of the `mpi_f08` module can also be enabled with `CM_ALL_USE_F08`. Then the new MPI derived types can be used directly. The requisite compilers and MPI implementations must be present. Furthermore, the ALL module must be compiled by the same Fortran compiler as the application (and MPI implementation if any MPI module is used).

3.2.2 Motivation and exploitation

There are a number of scientific applications developed in Fortran and the low entry barrier to the language makes it an easy choice for beginning scientific programmers. This module enables the usage of the features of the ALL load balancing library in Fortran codes. It basically provides the same functionality as the C++ interface. It is indispensable for any Fortran developer who is trying to use the ALL library.

Merge Request	Merge Request for ALL Fortran interface
Direct Documentation Link	ALL Fortran interface

3.3 ALL library in Material Point Method

3.3.1 Module description

In this module, we present the application of the ALL load balancing library in the Material Point Method (MPM) simulation code GMPM-PoC, which is written by Stephan Schulz from the Jülich Supercomputing Centre during his PhD thesis. MPM is used to simulate continuous matter and is especially suited for the simulation of large deformations.

Once large deformations are present, a dynamic load balancing solution is the method of choice to efficiently simulate large systems. Even if the initial work distribution is good, this is often no longer the case during the simulation run itself. The load balancing library ALL provides an easy plug-and-play solution to this problem and this module describes the details how the library is integrated. Thanks to the good load balancing provided by the library, larger systems can be simulated with less computational cost.

3.3.2 Motivation and exploitation

This development makes the simulation of continuous media (especially their deformations) computationally more feasible and extends the usability of the ALL library onto larger temporal and spatial scales. Furthermore, it demonstrates the straightforwardness of including the load balancing library into an already existing code. Depending on the simulation code, additional precautions must be taken, and those needed for the MPM simulation code are presented in this module.

The present module also shows a real-world application of the Fortran interface provided with ALL (documented in the ALL Fortran interface module in sec. 3.2). The MPM simulation code with integrated ALL is being used in the PhD thesis of Stephan Schulz at the Jülich Supercomputing Centre.

Merge Request	Merge Request for ALL library in Material Point Method
Direct Documentation Link	ALL library in Material Point Method

3.4 ALL library in HemeLB workflow

This module describes the cooperation between the ALL library and the HemeLB code, from the CompBioMed HPC Centre of Excellence. It provides a description about the work performed and the results of the cooperation.

3.4.1 Module description

As the ALL library is designed to work with particle codes, it is interesting to apply the library to a Lattice–Boltzmann solver, which is not particle-based. Therefore the mesh points of the numerical grid can be considered in analogy to particles and since each of the grid points is already assigned a workload, the sum of grid-point workloads can be used as domain workload. Since the creation and change of domain boundaries during the simulation is not feasible, an example code within ALL is used to create initial, balanced domain decompositions for various systems and the results are compared to the already existing load-balancing solution within HemeLB. Since test runs are performed on large scale resources, e.g. SuperMUC, it is necessary to also provide MPI-I/O based output for better parallel I/O efficiency.

3.4.2 Motivation and exploitation

HemeLB is a high performance Lattice–Boltzmann solver, optimized for simulating blood flow through sparse geometries, such as those found in the human vasculature [10]. The code is used within the CompBioMed HPC Centre of Excellence H2020 project [11], and is already highly optimized for HPC usage. As a consequence of the initial workshop about the ALL library hosted by JSC in the context of E-CAM, a cooperation was set up in order to analyse and test whether the use of ALL could improve the existing scalability of the code.

The current results of the cooperation indicate that the domain decomposition provided by ALL leads to a better theoretical work load distribution. Tests to check if this translates into better code performance are inconclusive yet, due to hardware-related issues on the testing platforms. These are currently under further investigation, and more definitive results about the performance of the ALL-provided domain decompositions can be expected in the near future. The results were part of a publication about HemeLB, presented in Ref. [12].

Merge Request	Merge Request for ALL library in HemeLB
Direct Documentation Link	ALL library in HemeLB

4 Modules based on Ludwig

In this section, we implement new modules to the Lattice–Boltzmann code Ludwig [8, 9]. This is an open source code available at https://github.com/ludwig-cf/ludwig.

4.1 Externally imposed chemical potential gradient for binary fluid mixtures

This module implements an externally imposed chemical potential gradient to Ludwig. The gradient is used in simulations of binary fluid mixtures and enables the investigation of the related flows in various porous materials.

4.1.1 Module description

A chemical potential gradient in a binary fluid mixture gives rise to a flow, whenever the value of the order parameter differs from 0. In Ludwig, we implement the gradient as a vectorial physical property, which can, similarly to all other physical properties, be used anywhere in the code. Furthermore, we use it in the context of binary fluid mixtures, in the subroutines that account for the time evolution of the order parameter (Cahn-Hilliard equation) and for the force which arises as a result of the non-zero gradients of chemical potential and order parameter. This module is essential for studying binary fluid flows in porous materials. A further application focuses on flows which arise due to wetting of the walls in nanochannels.

4.1.2 Motivation and exploitation

Diffusiophoresis is spontaneous motion of colloids within a solution caused by a concentration gradient of the solute within the solvent. Diffusioosmosis is a similar phenomenon, occurring in solutions in the presence of solid walls. The interaction of the fluid with the wall gives rise to bulk flow of the fluid, whenever there is a concentration gradient of the solute. This module enables the introduction of an external chemical potential gradient, which is, in the theoretical description, equivalent to the above-mentioned concentration gradient. It can therefore be used in Lattice–Boltzmann studies of diffusiophoretic and diffusioosmotic phenomena.

Merge Request	Merge Request for Externally imposed chemical potential gradient
Direct Documentation Link	Externally imposed chemical potential gradient for binary fluid mixture

4.2 Implementation of crystalline capillaries

This module implements simple cubic (SCC), body-centered cubic (BCC), and face-centered cubic (FCC) crystalline geometries as a utility to create capillaries in Ludwig. The thus created crystalline geometries are used as porous materials in Lattice–Boltzmann simulations.

4.2.1 Module description

This module enables Lattice–Botzmann simulations of flows through different crystalline geometries in Ludwig. The crystalline atoms are represented as solid nodes on the computational mesh, which give the (wetting) boundary condition to the fluid.

4.2.2 Motivation and exploitation

The phenomenon of diffusioosmosis, mentioned in Sec. 4.1, plays an important role in the flow of solutions through porous media. The crystalline structures represent various types of porous materials. By the assignment of wetting properties, we define how the solid parts of the geometries interact with the fluid. This module is essential for studies of various types of flows through such materials. In particular, we aim to focus on flows of binary fluid mixtures, driven by an externally imposed chemical potential gradient (Sec. 4.1), through porous media.

Merge Request	Merge Request for Crystalline capillaries
Direct Documentation Link	Simple cubic, body-centered cubic, and face-centered cubic crystalline capillaries

5 Overall impact of the results achieved within the Work-package

5.1 Overview of the results achieved so far

E-CAM's WP4 focuses on the development of software for meso– and multi–scale modelling simulation of systems on the mesoscopic scale. So far within this WP, **59** software modules have been developed (against a target of 43) which were certified according to the E-CAM guidelines [13], and another 8 that are work in progress (i.e. are under review to get certification). The modules developed are:

- **Modules that contribute to the porting of DL_MESO to multi-GPUs**. In collaboration with the UKRI STFC Daresbury Laboratory, E-CAM has developed a highly efficient version of DL_MESO (DPD version), a software package for mesoscale simulations developed at the UKRI STFC. This distributed GPU acceleration development is an extension of the DL_MESO package to MPI+CUDA that exploits the computational power of the latest NVIDIA cards on hybrid CPU–GPU architectures. The need to port DL_MESO to massively parallel computing platforms arose because often real systems are made of millions of particles and small clusters are usually not sufficient to obtain results in brief time. Moreover, with the advent of hybrid architectures, updating the code is becoming an important software engineering step to allow scientists to continue their work on such systems.

  The modules associated with the GPU rewrite of DL_MESO_DPD are listed on our software library here. A publication describing the work is in Ref.[14], as well as in success stories here and here.

- **Modules that develop polarizable meso-scale models**. Within a pilot project at partner UKRI STFC on Polarizable Mesoscale Models, in collaboration with Unilever, E-CAM researchers built a realistic model of water to be used in Dissipative Particle Dynamics (DPD) simulations, and developed the related utilities for the DL_MESO_DPD code. The list of modules for this purpose are listed here. A success story documenting this work is stored here.

- **Modules that help studying the rheological properties of new composite materials**. In a pilot project in collaboration with Michelin, E-CAM researchers at the MPIP Mainz implemented a hierarchical equilibration strategy for polymer melts in the ESPResSo++ software package, which can help to accurately determine and predict properties of polymer materials (e.g. rheological properties). This is highly important to researchers and industry.

  The list of modules for this purpose are listed here. A success story documenting this work is stored here.

- **Modules that extend the capabilities of the ParaDIS code**. These modules extend the ParaDIS code for discrete dislocation dynamics, with inclusion of precipitates interactions. The ParaDIS code is scalable and E-CAM work also focused on optimizing the code to run on HPC environments. The list of modules for this purpose are listed here.

- **Modules that develop the GC-AdResS scheme** . The goal of the E-CAM pilot project associated to this work, was to establish GC-AdResS (Grand Canonical Adaptive Resolution Scheme) as a standard method, and to make it easier to use with any Molecular Dynamics engine. The new implementation was based on the existing GC-AdResS code in GROMACS version 5.1.0 to 5.1.5. But it can be employed in any other MD packages such as e.g. LAMMPS and ESPResSo++. The coupling to the continuum is also being done in the code HALMD. The software developed in the context of this work is listed here. A case study describing the work is here.

- **Modules that implement the ALL load balancing library**. Scalability of parallel applications depends on a number of characteristics, among which is efficient communication, equal distribution of work or efficient data lay-out. Especially for methods based on domain decomposition, as it is standard for, e.g., molecular dynamics, dissipative particle dynamics or particle-in-cell methods, unequal load is to be expected for cases where particles are not distributed homogeneously, different costs of interaction calculations are present or heterogeneous architectures are invoked, to name a few. For these scenarios the code has to decide how to redistribute the work among processes according to a work sharing protocol or to dynamically adjust computational domains, to balance the workload. The "A Load Balancing Library" (ALL) developed within E-CAM at the Jülich Supercomputing Center aims to provide an easy and portable way to include dynamic domain-based load balancing into particle based simulation codes. It provides several schemes to find the ideal split of the workload, from the simplest orthogonal non staggered domain decomposition, to the more complex Voronoi mesh scheme.

  Library documentation is available here. The latest release is 0.9.1 and can be found under the library GitLab page. A story reporting on this work is available at this location.

- **Modules for the Mesoscale Modelling of phoretic phenomena in binary fluids**. In the context of an E-CAM pilot project described here, E-CAM researchers study diffusiophoresis and diffusioosmosis phenomena in heterogeneous fluids with the Lattice-Boltzmann method. These phenomena are of particular interest to the industrial

partner associated to this pilot project, Unilever, with whom E-CAM collaborates in the research of binary fluid flow through pores near the liquid-liquid critical point.

The software developed in the context of this work is listed at this location, and has been reported in the present deliverable.

The full portfolio of modules developed under WP4 is accessible from the software library for WP4 at https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/.

5.2 Dissemination and exploitation

Nine scientific publications originated from the project in the area of meso– and multi–scale modelling, and at least one is work in progress.

a. Towards blood flow in the virtual human: efficient self-coupling of HemeLB, J. W. S. McCullough et al., *Interface Focus* **2020**, 11, 20190119.
DOI: 10.1098/rsfs.2019.0119.

b. Towards extreme scale dissipative particle dynamics simulations using multiple GPGPUs, J. Castagna et al., *Comp. Phys. Commun.* **2020**, 251, 107159.
DOI: 10.1016/j.cpc.2020.107159.

c. ESPResSo++ 2.0: Advanced methods for multiscale molecular simulation, Horacio V. Guzman et al., *Comp. Phys. Commun.* **2019**, 238, 66–76.
DOI: 10.1016/j.cpc.2018.12.017. Open access version here.

d. Molecular Dynamics of Open Systems: Construction of a Mean-Field Particle Reservoir, Luigi Delle Site et al., *Adv. Theory Simul.* **2019**, 1900014.
DOI: 10.1002/adts.201900014.

e. Adaptive resolution molecular dynamics technique: Down to the essential, Christian Krekeler et al., *J. Chem. Phys.* **2018**, 149, 024104.
DOI: 10.1063/1.5031206. Open access version here.

f. Ionic Liquids Treated within the Grand Canonical Adaptive Resolution Molecular Dynamics Technique, B. Shadrack Jabes, Christian Krekeler, *Computation* **2018**, 1, 23.
DOI: 10.3390/computation6010023.

g. Probing spatial locality in ionic liquids with the grand canonical adaptive resolution molecular dynamics technique, B. Shadrack Jabes et al., *J. Chem. Phys.* **2018**, 148, 193804.
DOI: 10.1063/1.5009066. Open access version here.

h. Towards Open Boundary Molecular Dynamics Simulation of Ionic Liquids, Christian Krekeler and Luigi Delle Site, *Phys. Chem. Chem. Phys.* **2017**, 19, 4701-4709.
DOI: 10.1039/C6CP07489H. Open access version here.

i. Computational efficiency and Amdahl's law for the adaptive resolution simulation technique, Christoph Junghans, Animesh Agarwal and Luigi Delle Site, *Comp. Phys. Commun.* **2017**, 215, 20-25.
DOI: 10.1016/j.cpc.2017.01.030. Open access version here.

The software produced within this WP was disseminated via the articles above, but also at conferences and workshops organized by the members of the WP; via the project website (e.g. in success stories, in the modules of the month category, on the newsletter, etc.) - for an overview of the news items on our website that are associated to this WP see here, and through six deliverables produced during the project lifetime and that are listed here (Number 4.*x* deliverables).

We would like to note that the 1st article on this list relates to a collaboration between E-CAM and the CompBioMed Centre of Excellence that started in an E-CAM ESDW on the Load Balancing Library ALL. The purpose of this collaboration was to analyse and test whether the use of ALL could improve the existing scalability of CompBioMed's flagship code HemeLB. The software module reporting this cooperation is in sec. 3.4. Further developments on this topic are expected in the near future.

The ALL library is also being exploited by other codes such a the MPM code GMPM-PoC (sec. 3.3) and the multi-GPU version of the DL_MESO_DPD package (see here), showing a high-performance redistribution of the work load across GPUs. Work is underway in E-CAM to adapt ALL to the Ludwig code.

The rewrite of the DL_MESO_DPD code reported in the 2nd article of this list allows the simulation of complex systems comprising billions of atoms on thousands of GPGPUs. This is necessary to simulate surfactants, key ingredients in personal care products, dish soaps, laundry detergents, etc., with great impact in several industries.

It is important to note that all the work done within this WP expands the capabilities of different mesoscale codes allowing them to be used for further applications. This is complemented by the ALL library that proposes to improve the scalability of many of these codes to a larger number of cores on HPC systems, and thus to reduce the time-to-solution of the applications.

5.3 Industrial impact

Industry connection within this WP happened through the two pilot project in collaboration with industry, as explained in point 2 and 3 of section 5.1.

Connection with industry arises also from the work in the DL_MESO_DPD code. Within UKRI STFC, DL_MESO_DPD is involved in projects with Unilever, Syngenta, Infineum, IBM Research Europe, and an STFC spinout company Formeric. A training event dedicated to industry is being organized on the DL_MESO_DPD code and the use of DPD simualtions to study complex systems (webpage for the event).

Furthermore, we have held four workshops in WP4 that dealt with topics of interest for industry:

a. Challenges in Multiphase Flows, 9 - 12 December 2019, Monash University Prato Center, Italy, and CECAM-DE-SMSM, Mainz, Germany. See the workshop report.

b. State-of-the-Art Workshop in Mesoscale and Multiscale Modelling, 29 May - 1 June 2017, CECAM-IRL, University College Dublin, Ireland. See the workshop report.

c. Scoping Workshop: Dissipative particle dynamics: Where do we stand on predictive application, 24 - 26 April 2018, CECAM-UK-HARTREE, United Kingdom. See the workshop report.

d. Scoping workshop: E-CAM perspectives on Simulation, Modelling and Data in Industry, 7-9 September 2016, CECAM-DE-SMSM, Mainz, Germany.

The scoping workshop (number 3 on the list) had particular success among industry and was attended by 12 industrialists from 6 large companies. The scoping workshop number 4 of the list was attended by 4 large companies and 2 SMEs.

5.4 Training

During the lifetime of the projects we organized five ESDWs in the area of meso– and multi–scale modelling:

- ESDW on Meso and multiscale modeling, 18 - 29 September 2017, CECAM-DE-MMS, Freie Universität Berlin, Berlin, Germany. Workshop report.
- ESDW on Meso and Multiscale Methods, 3 - 14 July 2017, CECAM-ES, University of Barcelona, Spain. Workshop report.
- ESDW on Load Balancing for Particle Simulations, 6-7 September 2018 (1st part) and 3-7 June 2019 (2nd part), Forschungszentrum Jülich, Jülich Supercomputing Centre, Germany. Workshop report.
- ESDW on Mesoscopic simulation models and High-Performance Computing, 14 - 18 October 2019, Aalto University & CSC IT Center for Science, Finland.
- ESDW in HPC for mesoscale simulation, 18 - 22 January 2021, Online/CECAM-UK-DARESBURY, Daresbury Laboratory, UK.

A training session on the E-CAM Load Balancing ALL took place on the 11th December 2020, in the form of a webinar. Three key lectures have been recorded and stored on E-CAM's Online training portal here.

In total, 75 people were trained at the events organized within WP4.

5.5 Societal impact

The societal benefits of the results achieved in WP4 of E-CAM are twofold. On the fundamental level, the software modules developed are expected to provide new avenues for material modeling and will make it possible to have an impact on the prediction of the properties of new materials, and in general in material science. This will be complemented by the tools developed within the project to improve performance and scalability of the codes (such as the ALL), with potential for benefiting the broader community.

On an economic level, industry will benefit from software containing efficient and easy to use simulation and analysis tools to extract observables for applications in sectors such as pharma, materials or house-hold products. These developments can provide support in applications including but not limited to:

- Drug handling: kinetics of biopolymers, proteins and membrane interactions;
- Food/dairy industry: protein aggregation, grain size in ice cream, food preservation, food stability and texture control;
- Tyre industry: development of novel composite materials, determination of rheological properties of materials;
- Materials science and daily products: surfactant kinetics, material stability, soft matter, self-assembly of nano-materais, colloids, liquid crystal based materials, liquid-surface interactions;
- Oil industry: studies of flow in porous media.

Society can benefit greatly from the development of new computational algorithms and the corresponding software that make the development of new materials cheaper and improve food quality and products in daily use.

6 Outlook

The deliverable consists of nine modules in the field of meso– and multi–scale modelling that have been developed within the scope of the E-CAM project. Three are integrated into the simulation package DL_MESO_DPD, two into Ludwig, and four extend the applicability of the ALL load balancing library. The DL_MESO_DPD group of software contributions optimize the performance of the code in terms of offloading its computationally demanding parts as well as developing novel multi-GPU architectures. The ALL modules enable the usage of the library in C/C++ as well as in Fortran-based codes and demonstrate its usefulness in simulating deformations of continua and in simulations of blood flow. Furthermore, the modules in Ludwig facilitate the studies of binary flows on the mesoscopic scale and contribute to a better understanding of the diffusiophoretic and diffusioosmotic effect in porous media. All of the modules have been accepted into the E-CAM software library or are under final review.

As this deliverable is the last one of its series in the WP4 of E-CAM, a section on the overall impact of the results achieved within the Work-package was also included.

References

Acronyms Used

ESDW Extended Software Development Workshop
DPD Dissipative Particle Dynamics
STFC Science and Technology Facilities Council
WP Work Package
ALL A Load-balancing Library
VV Velocity Verlet
MPM Material Point Method

URLs referenced

Page ii

https://www.e-cam2020.eu ... https://www.e-cam2020.eu
https://www.e-cam2020.eu/deliverables ... https://www.e-cam2020.eu/deliverables
Internal Project Management Link ... https://redmine.e-cam2020.eu/issues/162
jurij.sablic@gmail.com ... mailto:jurij.sablic@gmail.com
http://creativecommons.org/licenses/by/4.0 ... http://creativecommons.org/licenses/by/4.0

Page 4

GitLab service of E-CAM ... https://gitlab.e-cam2020.eu/

Page 5

DL_MESO ... https://www.scd.stfc.ac.uk/Pages/DL_MESO.aspx
Merge Request for Many-body DPD scheme on multi-GPU ... https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/-/merge_requests/182
Many-body DPD scheme for DL_MESO_DPD on multi-GPU ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/DL_MESO_DPD_onGPU/manybody/readme.html
Kokkos ... https://github.com/kokkos/kokkos
Kokkos ... https://github.com/kokkos/kokkos
Merge Request for DL_MESO (DPD) on Kokkos: Verlet Velocity step 1 ... https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/-/merge_requests/264
DL_MESO (DPD) on Kokkos: Verlet Velocity step 1 ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/DL_MESO_DPD_onGPU/Kokkos_VV1/readme.html

Page 6

Kokkos ... https://github.com/kokkos/kokkos
Merge Request for Incorporation of Kokkos into the 2nd step of the VV algorithm ... https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/-/merge_requests/265
DL_MESO (DPD) on Kokkos: Verlet Velocity step 2 ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/DL_MESO_DPD_onGPU/Kokkos_VV2/readme.html

Page 7

http://slms.pages.jsc.fz-juelich.de/websites/all-website/ ... http://slms.pages.jsc.fz-juelich.de/websites/all-website/
https://gitlab.version.fz-juelich.de/SLMS/loadbalancing ... https://gitlab.version.fz-juelich.de/SLMS/loadbalancing
Merge Request for ALL C++ interface ... https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/-/merge_requests/297
ALL C++ interface ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/ALL_library/cpp_interface/readme.html
https://gitlab.version.fz-juelich.de/SLMS/loadbalancing ... https://gitlab.version.fz-juelich.de/SLMS/loadbalancing
Merge Request for ALL Fortran interface ... https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/-/merge_requests/284
ALL Fortran interface ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/ALL_library/fortran_interface/readme.html

Page 8

Merge Request for ALL library in Material Point Method ... https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/-/merge_requests/285
ALL library in Material Point Method ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-M

modules/ALL_library/MPM_integration/MPMIntegration.html
CompBioMed HPC Centre of Excellence ... https://www.compbiomed.eu/
Merge Request for ALL library in HemeLB ... https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/-/merge_requests/298
ALL library in HemeLB ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/ALL_library/all_hemeLB_cooperation/hemeLBcooperation.html

Page 9

https://github.com/ludwig-cf/ludwig ... https://github.com/ludwig-cf/ludwig
Merge Request for Externally imposed chemical potential gradient ... https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/-/merge_requests/240
Externally imposed chemical potential gradient for binary fluid mixture ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/Lattice_Boltzmann/external_chemical_potential_gradient/readme.html
Merge Request for Crystalline capillaries ... https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/-/merge_requests/240
Simple cubic, body-centered cubic, and face-centered cubic crystalline capillaries ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/Lattice_Boltzmann/crystalline_capillaries/readme.html

Page 10

here ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/index.html#dl-meso-dpd
here ... https://www.e-cam2020.eu/wp-content/uploads/2019/09/DL_MESO_GPU_success_story.pdf
here ... https://www.hpccoe.eu/2021/02/05/e-cam-mesoscale-simulation-of-billion-atom-complex-sys-tems
pilot project at partner UKRI STFC on Polarizable Mesoscale Models ... https://www.e-cam2020.eu/pilot-project-unil
here ... https://www.e-cam2020.eu/pilot-project-unilever/
here ... https://www.e-cam2020.eu/mesoscale-models-for-polarisable-solvents-e-cam-case-study/
pilot project ... https://www.e-cam2020.eu/pilot-project-composite-materials/
here ... https://www.e-cam2020.eu/pilot-project-composite-materials/
here ... https://www.e-cam2020.eu/hierarchical-equilibration-strategy-for-polymer-melts-e-cam-case-s
here ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/index.html#paradis
E-CAM pilot project associated to this work ... https://www.e-cam2020.eu/pilot-project-gc-adress/
here ... https://www.e-cam2020.eu/pilot-project-gc-adress/
here ... https://www.e-cam2020.eu/e-cam-case-study-the-development-of-the-cg-adress-scheme/
here ... https://gitlab.version.fz-juelich.de/SLMS/loadbalancing
the library GitLab page ... https://gitlab.version.fz-juelich.de/SLMS/loadbalancing/-/releases
this location ... https://www.e-cam2020.eu/the-all-load-balancing-library/
here ... https://www.e-cam2020.eu/pilot-project-on-mesoscale-modelling-of-phoretic-phenomena-in-bina

Page 11

at this location ... https://www.e-cam2020.eu/pilot-project-on-mesoscale-modelling-of-phoretic-phenomena-
https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/ ... https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/index.html
here ... https://arxiv.org/pdf/1806.10841.pdf
here ... https://zenodo.org/record/1312743#.W7tYshMzY61
here ... https://zenodo.org/record/1163670#.WnH-AUtG2WY
here ... https://zenodo.org/record/1193355#.YDvKHJNKi3I
here ... https://zenodo.org/record/321807#.YDvKYJNKi3J
here ... https://www.e-cam2020.eu/tag/mesoscale-simulation/
here ... https://www.e-cam2020.eu/deliverables/
CompBioMed Centre of Excellence ... https://www.compbiomed.eu/
HemeLB ... http://hemelb.org/#about
here ... https://www.e-cam2020.eu/december-module-of-the-month-load-balancing-for-multi-gpu-dl_meso/

Page 12

webpage for the event ... https://www.cecam.org/workshop-details/1074
workshop report ... https://www.e-cam2020.eu/wp-content/uploads/2020/05/SAW-Challenges-in-Multiphase-F
pdf
workshop report ... https://www.e-cam2020.eu/wp-content/uploads/2018/04/ECAM-SAW-WP4-Report_vf.pdf

workshop report . . . https://www.e-cam2020.eu/wp-content/uploads/2018/09/SCOW_Dissipative-particle-dynar
REPORT.pdf
Workshop report . . . https://www.e-cam2020.eu/wp-content/uploads/2018/06/ESDW8_Meso-_and_Multiscale_
Modelling_REPORT.pdf
Workshop report . . . https://www.e-cam2020.eu/wp-content/uploads/2019/07/ESDW4_Mesoscale_Report.
pdf
Workshop report . . . https://www.e-cam2020.eu/wp-content/uploads/2021/03/ESDW_WP4_LB_Juelich_
2018_2019_ScientificReport.pdf
training session on the E-CAM Load Balancing ALL . . . https://www.cecam.org/workshop-details/1021
here . . . https://training.e-cam2020.eu/datasets/5fd74115e4b0e4ec0f94eb27

Citations

[1] http://www.cse.clrc.ac.uk/ccg/software/DL_MESO/.

[2] M. A. Seaton, R. L. Anderson, S. Metz, and W. Smith, "DL_MESO: highly scalable mesoscale simulations," *Molecular Simulation*, vol. 39, no. 10, pp. 796–821, Sep. 2013.

[3] https://zenodo.org/record/841696.

[4] https://zenodo.org/record/1207372.

[5] https://zenodo.org/record/1210075.

[6] https://zenodo.org/record/2555012.

[7] https://zenodo.org/record/3568175.

[8] J.-C. Desplat, I. Pagonabarraga, and P. Bladon, "Ludwig: A parallel lattice-boltzmann code for complex fluids," *Comput. Phys. Commun.*, vol. 134, no. 3, pp. 273 – 290, 2001.

[9] https://ludwig.epcc.ed.ac.uk/.

[10] http://hemelb.org/#about.

[11] https://www.compbiomed.eu/.

[12] J. W. S. McCullough, R. A. Richardson, A. Patronis, R. Halver, R. Marshall, M. Ruefenacht, B. J. N. Wylie, T. Odaker, M. Wiedemann, B. Lloyd, E. Neufeld, G. Sutmann, A. Skjellum, D. Kranzlmüller, and P. V. Coveney, "Towards blood flow in the virtual human: efficient self-coupling of hemelb," *Interface Focus*, vol. 11, no. 1, p. 20190119, 2021.

[13] A. O. Cais, "Esdw technical software guidelines i," Mar. 2016. [Online]. Available: https://doi.org/10.5281/zenodo.841735

[14] J. Castagna, X. Guo, M. Seaton, and A. O'Cais, "Towards extreme scale dissipative particle dynamics simulations using multiple gpgpus," *Computer Physics Communications*, vol. 251, p. 107159, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465520300199