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

E-CAM Deliverable 4.5
Deliverable Type: Other
Delivered in December, 2019

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	D4.5
Deliverable Nature	Other
Dissemination Level	Public
Contractual Date of Delivery	Project Month 49(October, 2019)
Actual Delivery Date	9th December, 2019
Deliverable Description	9 software modules delivered to the E-CAM repository in the area of meso– and multi–scale modelling, based on user requests and their documentation.

Document Control Information

Document	Title:	Meso– and multi–scale modelling E–CAM modules IV
	ID:	D4.5
	Version:	As of 9th December, 2019
	Status:	Accepted by WP leader
	Available at:	https://www.e-cam2020.eu/deliverables
	Document history:	Internal Project Management Link
Review	Review Status:	Reviewed
Authorship	Written by:	Jony Castagna(Science and Technology Facilities Council, United Kingdom)
	Contributors:	Jony Castagna (Science and Technology Facilities Council); Alan Ó Cais (JSC).
	Reviewed by:	Godehard Sutmann (JSC)
	Approved by:	Burkhard Dünweg (Max Planck Institute for Polymer Research)

Document Keywords

Keywords:	E-CAM, CECAM, Module, DL_MESO_DPD, GROMACS, GC–AdResS, GPU, DPD, coarse–graining, electrostatics, load–balancing

9th December, 2019

[1] jony.castagna@stfc.ac.uk

Contents

Executive Summary

In this report for Deliverable 4.5 of E-CAM, nine software modules in meso– and multi–scale modelling are presented.

The modules represented efforts integrated into larger software packages and thus extend their applicability. The first package is DL_MESO_DPD [1, 2] that provides a toolbox to do Dissipative Particle Dynamics (DPD) simulations, routinely used in an industrial context to find out the static and dynamic behaviour of soft-matter systems.

The DL_MESO packages has previously (in [3]) been identified as one of the important codes for mesoscale simulations within E-CAM, and the single- and multi-GPU version of DL_MESO_DPD have already been introduced in deliverables D4.2[4] and D4.3[5], respectively, and recently tested up to 4096 GPUs with results presented in D4.4[6]. In the current deliverable, we present the following 4 modules that relate to DL_MESO_DPD's extension, performance and optimisation on multi-GPU architectures:

- Surface boundary conditions
- Many-body DPD model on single GPU
- Long Integer on DL_MESO_DPD multi-GPU
- Load balancing for multi-GPU DL_MESO.

The second software package covered in this report is A Load-balancing Library (ALL), which is a dynamic load balancing library that is both efficient and widely applicable. Load-balancing is a complex topic, particularly when considering multi-scale approaches, due to heterogeneity in both computational approaches and target architectures. Therefore, it has has not yet been implemented in a number of important codes of the E-CAM multi-scale community, e.g. DL_MESO, DL_POLY, Espresso, Espresso++, to name a few. Other codes (e.g. LAMMPS) have implemented somewhat simpler schemes, which might however turn out to lack sufficient flexibility to accommodate all important cases. Therefore, the ALL library was created in the context of an Extended Software Development Workshop (ESDW) within E-CAM, where code developers of CECAM community codes were invited together with E-CAM postdocs, to work on the implementation of load balancing strategies. The goal of this activity was to increase the scalability of these applications to a larger number of cores on HPC systems, for spatially inhomogeneous systems, and thus to reduce the time-to-solution of the applications.

ALL aims to provide an easy way to include dynamic domain-based load balancing into particle based simulation codes. The library is developed in the Simulation Laboratory Molecular Systems of the Juelich Supercomputing Centre at Forschungszentrum Juelich. It includes several load-balancing schemes, with additional approaches currently being added. The following list gives an overview about the currently included schemes, which each represent a module presented in this report:

- Tensor method
- Staggered Grid method
- Unstructured Mesh scheme
- Voronoi Mesh scheme
- Histogram method.

A short description is written for each module, followed by a link to the respective Merge-Request on the GitLab service of E-CAM. These merge requests contain detailed information about the code development, testing and documentation of the modules.

1 Introduction

Work package 4 deals with "mesoscale" simulations that are characterised by a description of the physics "somewhere in between" atomistic Molecular Dynamics and engineering–style continuum mechanics. Many meso-scale methods have been developed, and this is still a highly active field. Of particular importance is the development of systematic "coarse–graining" approaches, where the same physics is being described on two or more differing length and time scales, and relations are established between these in a quantitative fashion.

1.1 Overall scope of the module set

The modules presented in this report represent efforts integrated into larger software packages and thus extend their applicability.

The first package is DL_MESO_DPD [1, 2] that provides a toolbox to do DPD simulations. DPD is essentially Molecular Dynamics, however not for atoms but for effective beads that represent many microscopic degrees of freedom. These effective beads interact via soft potentials, and pairwise friction augmented with thermal Langevin noise represents the effect of those degrees of freedom 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 (Sec. 1.1.1) have been developed by the E-CAM programmer Jony Castagna.

The second software package covered in this report is ALL. It is a dynamic load balancing library that is both efficient and widely applicable. Load-balancing it a complex topic, particularly when considering multi-scale approaches, due to heterogeneity in both computational approaches and target architectures. Therefore, it has has not yet been implemented in a number of important codes of the E-CAM multi-scale community, e.g. DL_Meso, DL_Poly, Espresso, Espresso++, to name a few. Other codes (e.g. LAMMPS) have implemented somewhat simpler schemes, which might however turn out to lack sufficient flexibility to accommodate all important cases. Therefore, the ALL library was created in the context of an ESDW within E-CAM, where code developers of CECAM community codes were invited together with E-CAM postdocs, to work on the implementation of load balancing strategies. The goal of this activity was to increase the scalability of these applications to a larger number of cores on HPC systems, for spatially inhomogeneous systems, and thus to reduce the time-to-solution of the applications.

1.1.1 Extension and improvements of the DL_MESO_DPD code on multi-GPU systems

Based on the DPD code DL_MESO_DPD, we present 4 modules in this deliverable that are related to its performance, optimisation and extension on multi-GPU architectures.

The first module adds a new type of boundary conditions in the multi-GPU version. The current version supports only fully periodic boundary conditions, while this module allows one to define wall-frozen boundaries in each direction. This will allow runing more complex geometries like flows in micro-channels.

The second module implements the many-body DPD scheme on the single-GPU version. A primary constraint of the DPD model is due to the fact that the possible equations of state (quadratic only) cannot be changed due to the soft potential model. The many-body DPD scheme allows us to overcome this limitation by adding a local density based on the interaction between all particles within a cut-off radius. This allows the simulation of complex phenomena like surface drop and vapour liquid interfaces. This module is a preliminary one for the multi-GPU version.

The third module integrates, in the multi-GPU version of DL_MESO_DPD, ALL developed in the Simulation Laboratory Molecular Systems of the Juelich Supercomputing Centre at Forschungszentrum Juelich. The library automates the re-partition of work load between GPUs allowing better scaling when unbalanced systems, such as large proteins in water, or polymers, are simulated. The ALL library provides many possible load balancing schemes (see Section 1.1.2 for details).

The final module allows us to exceed the current limit on the number of beads of 2^{31} (2 billion). This constraint is due to the 32-bit integer declaration of the arrays used across the full code. The switch from 32-bit to 64-bit allows to run up to 2^{63} beads (usually the system memory will limit to a lower value), well beyond current memory capability on any supercomputer. In a previous deliverable [7], we presented the strong and weak scaling of the current version up to 2048 GPUs. Despite the good efficiency achieved ($> 85\%$), the number of beads per device when using 2048 GPUs was less than 1 million, and larger simulations could not be run due to the above limit.

1.1.2 Adding schemes to ALL

A Load-balancing Library (ALL) aims to provide an easy way to include dynamic domain-based load balancing into particle based simulation codes. The library is developed in the Simulation Laboratory Molecular Systems of the Juelich Supercomputing Centre at Forschungszentrum Juelich.

It includes several load-balancing schemes, with additional approaches currently being added. The following list gives an overview about the currently included schemes, which each represent a module presented in this report, and some short explanations.

- The first module is the Tensor-Product scheme. The work on all processes is reduced (in the sense of functional programming) over the Cartesian planes in the systems. This work is then equalised among processors by adjusting the borders of the Cartesian planes.
- The second module adds the Staggered-grid scheme which uses a 3-step hierarchical approach. Work over the Cartesian planes is reduced before the borders of these planes are adjusted; in each of the Cartesian planes the work is reduced for each Cartesian column, these columns are then adjusted to each other to homogenise the work in each column; the work between neighbouring domains in each column is adjusted. Each adjustment is done locally with the neighbouring planes, columns or domains by adjusting the adjacent boundaries.
- The third module is the Topological Mesh scheme. In contrast to the previous methods, this method adjusts domains not by moving boundaries but vertices, i.e. corner points, of domains. For each vertex a force, based on the differences in work of the neighbouring domains, is computed and the vertex is shifted in a way to equalise the work between these neighbouring domains.
- The fourth module uses a Voronoi mesh scheme. Similar to the topological mesh method, this method computes a force, based on work differences. In contrast to the topological mesh method, the force acts on a Voronoi point rather than a vertex, i.e. a point defining a Voronoi cell, which describes the domain. Consequently, the number of neighbours is not a conserved quantity, i.e. the topology may change over time.
- The final module uses a histogram-based staggered-grid approach. Resulting in the same grid as the staggered-grid scheme, this scheme uses the cumulative work function in each of the three Cartesian directions in order to generate this grid. Using histograms and the previously defined distribution of process domains in a Cartesian grid, this scheme generates in three steps a staggered-grid result, in which the work is distributed as evenly as the resolution of the underlying histogram allows. In contrast to the previously mentioned schemes this scheme depends on a global exchange of work between processes.

1.2 General applications and possible exploitation of the codes

DPD is routinely used in an industrial context to find out the static and dynamic behaviour of soft-matter systems. Examples include colloidal dispersions, emulsions and other amphiphilic systems, polymer solutions, etc. Such materials are being produced or processed in industries like cosmetics, food, pharmaceutics, biomedicine, etc. Porting the method to GPUs (Sec. 1.1.1) is thus inherently useful in order to provide cheaper calculations. The ultimate intention of these four 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.

The potential of the library ALL for simulation codes in general, but in particular in multi-scaling modelling, is quite significant. In this report we already include a module for DL_MESO_DPD that highlights it's ability to quickly and easily improve application performance. A number of other simulation codes are already integrating ALL, with another HPC CoE CompBioMed being enthusiastic adopters (a joint paper is currently in preparation).

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).

2 Modules based on DL_MESO_DPD

The base code for the following four 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 version of DL_MESO_DPD, developed by Jony Castagna, has been introduced already in D4.2 [8], D4.3 [9] and recently tested up to 4096 GPUs with results presented in D4.4 [6].

2.1 Surface boundary conditions

This module adds frozen-wall boundary conditions, allow the simulation of complex geometries like channel flows and surface liquid interactions.

2.1.1 Module description

The current multi-GPU version supports only fully periodic boundary conditions. This is due to its simplicity and symmetry which allows a easy implementation of the communication between GPUs. This module extend the option of boundary conditions to frozen-wall surfaces, represented in DPD as frozen beads supported by an appropriate force calculation. The main effort is in the correction of the potential calculation and eventual implementation of a dedicated subroutine to calculate the bounce-back effect of the free particle moving towards the wall.

The module allows to simulate wall surfaces in all three directions and retain similar efficiency in performance and scaling when compared to the fully periodic conditions. A Poiseuille flow, i.e. a laminar flow within parallel walls, is presented as a test case.

2.1.2 Motivation and exploitation

Real cases often involve complex geometries and require the implementation of solid walls as boundary conditions. A typical example is the flow in microchannels used for example in the production of Graphene. The interaction between fluid and surface create a different profile of velocities which has a strong impact on the fluid dynamics, especially in case of non-Newtonian fluids (i.e. where the shear stress is a non linear function of the velocity gradient) like shampoo and other body-care products.

This module will allow us to study such phenomena reducing the computational cost and time and scaling up to larger systems.

Merge Request	MR for Surface boundary conditions for DL_MESO_DPD on multi-GPU
Direct Documentation Link	Surface boundary conditions for DL_MESO_DPD on multi-GPU

2.2 Many-body DPD model on single GPU

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

2.2.1 Module description

A many-body DPD potential is added to the single GPU version of DL_MESO_DPD to allow simulating systems with a non quadratic equation of state. It consists mainly of calculating a local density and potential, according to a cut-off radius usually smaller than the short range force cut-off radius.

The many-body technique mainly required the porting to CUDA/C of other Fortran subroutines to account for the local density change and potential. The new implementation has been tested using a water drop on a surface and vapour liquid test cases similar to the examples described in the DL_MESO_DPD manual.

2.2.2 Motivation and exploitation

DPD has a fundamental constraint on the equation of state due to linear soft potential used. Interaction between vapour and liquid are therefore not correctly simulated unless a corrector scheme, like the many-body DPD, is used.

A typical application is the simulation of liquid-vapour interfaces occurring during distillation processes.

Another important case is the interaction between liquids and surfaces which enhance the surface tension effects like in a drop of ink in contact with a foil paper.

This module will allow us to simulate, better understand, and then optimise these industrial applications with a smaller computational cost.

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

2.3 Integrating the ALL library for load balancing

This module describes the implementation of the ALL library for load balancing in the multi-GPU version of DL_MESO_DPD.

2.3.1 Module description

Real application simulations are usually made of several, complex, multi-component species. This requires a molecular representation via several type of beads of different dimensions which lead, inevitably, to an imbalance in the distribution of the workload among the GPUs.

This module attempts to adjust the load balance using the ALL library Tensor option (see Section 3.1), i.e. adjusting the domain decomposition shifting the vertices of each partition in a such way to maintain orthogonal decomposition and without staggering the mesh.

The load balance is applied to every iteration and corresponding arrays are recalculated according to the new vertices positions.

2.3.2 Motivation and exploitation

The main reason behind this module is to improve the performance of the simulation when unbalanced systems are represented. For example, in the case of a water drop, the disparity in DPD density between the drop and the surrounding air can be very large and while most of the GPUs are idle, a few are overloaded and dictate the overall computational time.

This module show how we can achieve a performance speedup of a factor ~2 when proper load balancing is used.

Merge Request	Merge Request for ALL library for load balancing
Direct Documentation Link	Load balancing for multi-GPU DL_MESO

2.4 Extension to 64-bit integer arrays for large simulations

This module presents the extension to 64-bit of the DL_MESO_DPD multi-GPU version in order to exceed the current limit of 2 billion particles (beads in DPD) and simulate very large systems.

2.4.1 Module description

The current version of DL_MESO_DPD is limited to 2^{31} beads due to the use of 32-bit integer arrays. This module presents an extension to 64-bit, which correspond to replacing, only where needed, the *INTEGER* (in Fortran) and *int* (in CUDA/C) declarations with *INTEGER*8* and *long*, respectively.

The module has been tested using 20 billion beads and 4096 GPUs using PizDaint supercomputer at CSCS.

2.4.2 Motivation and exploitation

Typical DPD simulation use around 1 million beads. This is due not only to the coarse approximation, which drastically reduces the number required, but also to the number of time steps needed to achieve the eventual thermodynamic equilibrium. However, complex systems like very large polymers, can easily achieve and overcome the current 2 billion limit. The current usage of 32-bit integers does not allow for such simulations, despite the computational power of the GPU which could be used to run the simulation in a reasonable time.

This module will allow to simulation of large systems like macro-molecules, biological systems and polymers. However, we consider this module mainly for High Performance Computing studies, rather than industrial applications.

Merge Request	Merge Request for Extension to 64-bit integer arrays
Direct Documentation Link	Long Integers for DL_MESO_DPD multi-GPU

3 Modules based on the ALL library

The ALL library aims to provide an easy and portable way to include dynamic domain-based load balancing into particle based simulation codes. The library is developed in the Simulation Laboratory Molecular Systems of the Juelich Supercomputing Centre at Forschungszentrum Juelich.

The modules listed below provide additional methods to the ALL library, up-to-date descriptions of the methods in the library can be found in the ALL README file.

3.1 ALL Tensor method

3.1.1 Module description

For the Tensor-Product method, the work on all processes (subdomains) is reduced (in the sense of functional programming) over the Cartesian planes in the systems. This work is then equalised among processors by adjusting the borders of the Cartesian planes.

Merge Request	ALL Tensor method
Direct Documentation Link	Documentation for ALL Tensor method

3.2 ALL Staggered Grid method

3.2.1 Module description

In the staggered-grid scheme, a 3-step hierarchical approach is applied, where:

- work over the Cartesian planes is reduced, before the borders of these planes are adjusted;
- in each of the Cartesian planes the work is reduced for each Cartesian column. These columns are then adjusted to each other to homogenise the work in each column;
- the work between neighbouring domains in each column is adjusted, each adjustment is done locally with the neighbouring planes, columns or domains by adjusting the adjacent boundaries.

Merge Request	ALL Staggered Grid method
Direct Documentation Link	Documentation for ALL Staggered Grid method

3.3 ALL Unstructured Mesh method

3.3.1 Module description

In contrast to *tensor* and *staggered* options, the unstructured mesh method adjusts domains not by moving boundaries but vertices, i.e. corner points, of domains. For each vertex a force, based on the differences in work of the neighbouring domains, is computed and the vertex is shifted in a way to equalise the work between these neighbouring domains.

Merge Request	ALL Unstructured Mesh method
Direct Documentation Link	Documentation for ALL Unstructured Mesh method

3.4 ALL Voronoi Mesh method

3.4.1 Module description

Similar to the topological mesh method (Section 3.3), the Voronoi mesh method computes a force, based on work differences. In contrast to the topological mesh method, the force acts on a Voronoi point rather than a vertex, i.e. a point defining a Voronoi cell, which describes the domain. Consequently, the number of neighbours is not a conserved quantity, i.e. the topology may change over time. ALL uses the Voro++ library published by the Lawrance Berkeley Laboratory for the generation of the Voronoi mesh.

Merge Request	ALL Voronoi Voronoi Mesh method
Direct Documentation Link	Documentation for ALL Voronoi Mesh method

3.5 ALL Histogram method

3.5.1 Module description

The histogram-based staggered-grid scheme results in the same grid as the staggered-grid scheme (Section 3.2), this scheme uses the cumulative work function in each of the three Cartesian directions in order to generate this grid. Using histograms and the previously defined distribution of process domains in a Cartesian grid, this scheme generates in three steps a staggered-grid result, in which the work is distributed as evenly as the resolution of the underlying histogram allows. In contrast to the other schemes this scheme depends on a global exchange of work between processes.

Merge Request	ALL Histogram method
Direct Documentation Link	Documentation for ALL Histogram method

3.6 Motivation and exploitation

All of the previously described methods give the user options to help provide reasonable load balancing given the constraints of their application. Which method is best suited to a particular problem is best chosen by the user, and there are a number of considerations that can affect their choice:

- Some methods cannot *guarantee* very good balancing but give measurable improvement for codes which cannot easily be adjusted for a topology change in the computational mesh;
- How expensive it is to compute the work functions and move particles - this affects whether they would use the load balancing dynamically, periodically, or just once;
- depending on the amount of times the load balancing routines will be called, they may wish to choose schemes that are less expensive to compute;
- topological mesh schemes are typically suited to only particular codes that can handle irregular grids.

Even within this deliverable the ALL library has shown its impact (see Section 2.3). During the ESDW related to ALL, a number of other codes have shown their interest in the capabilities of the library. In particular, it is being used for the flagship application HemeLB of the HPC CoE CompBioMed.

4 Outlook

The deliverable contains nine modules that are integrated in two software packages: DL_MESO_DPD, and ALL, with a primary focus on meso– and multi–scale modelling. All of the modules have been accepted into the E-CAM software library. Both packages continue to be under heavy development within E-CAM, with additional capabilities being added and, in the case of ALL, the library being adopted by a number of relevant community codes. E-CAM will continue to support such feature-development, adoption by the community, and support the libraries in their usage by the community.

References

Acronyms Used

ESDW	Extended Software Development Workshop
DPD	Dissipative Particle Dynamics
STFC	Science and Technology Facilities Council
ALL	A Load-balancing Library

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/161
jony.castagna@stfc.ac.uk . . . mailto:jony.castagna@stfc.ac.uk
http://creativecommons.org/licenses/by/4.0 . . . http://creativecommons.org/licenses/by/4.0

Page 1

ALL . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/index.html#all-a-load-balancing-library

Page 2

ALL . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/index.html#all-a-load-balancing-library

Page 3

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

Page 4

DL_MESO . . . https://www.scd.stfc.ac.uk/Pages/DL_MESO.aspx
MR for Surface boundary conditions for DL_MESO_DPD on multi-GPU . . . https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/merge_requests/181
Surface boundary conditions for DL_MESO_DPD on multi-GPU . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling-Modules/modules/DL_MESO_DPD_onGPU/surfaces/readme.html

Page 5

Merge Request for Many-body DPD scheme on single GPU . . . https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/merge_requests/211
Many body DPD for DL_MESO_DPD single-GPU . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-M
modules/DL_MESO_DPD_onGPU/manybody_singleGPU/readme.html
Merge Request for ALL library for load balancing . . . https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/merge_requests/180
Load balancing for multi-GPU DL_MESO . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelli
modules/DL_MESO_DPD_onGPU/loadBalance/readme.html

Page 6

Merge Request for Extension to 64-bit integer arrays . . . https://gitlab.e-cam2020.eu:10443/e-cam/E-CAM-Library/merge_requests/183
Long Integers for DL_MESO_DPD multi-GPU . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Mode
modules/DL_MESO_DPD_onGPU/longInteger/readme.html

Page 7

ALL README . . . https://gitlab.version.fz-juelich.de/SLMS/loadbalancing/blob/master/README.md
ALL Tensor method . . . https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/merge_requests/149
Documentation for ALL Tensor method . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modelling
modules/ALL_library/tensor_method/readme.html
ALL Staggered Grid method . . . https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/merge_requests/212

Documentation for ALL Staggered Grid method . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Mod
modules/ALL_library/staggered_method/readme.html
ALL Unstructured Mesh method . . . https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/merge_requests/
214
Documentation for ALL Unstructured Mesh method . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale
modules/ALL_library/unstructured_method/readme.html
ALL Voronoi Voronoi Mesh method . . . https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/merge_requests/
215
Documentation for ALL Voronoi Mesh method . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Mod
modules/ALL_library/voronoi_method/readme.html

Page 8

ALL Histogram method . . . https://gitlab.e-cam2020.eu/e-cam/E-CAM-Library/merge_requests/213
Documentation for ALL Histogram method . . . https://e-cam.readthedocs.io/en/latest/Meso-Multi-Scale-Modell
modules/ALL_library/histogram_method/readme.html
ESDW related to ALL . . . https://www.e-cam2020.eu/event/extended-software-development-workshop-atomistic
?instance_id=77
HemeLB . . . http://www.2020science.net/software/hemelb.html
CompBioMed . . . https://www.compbiomed.eu/

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] B. Duenweg, J. Castagna, S. Chiacchiera, and H. Kobayashi, "Meso– and multi–scale modelling E-CAM modules I," Oct. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1207372

[5] B. Duenweg, J. Castagna, S. Chiacchiera, H. Kobayashi, and C. Krekeler, "Meso– and multi–scale modelling E-CAM modules II," Mar. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1210075

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

[7] A. O'Cais and J. Castagna, "E-cam software porting and benchmarking data iii," Apr. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2656216

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

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