Meso– and multi–scale modelling E-CAM modules I

E-CAM Deliverable 4.2
Deliverable Type: Other
Delivered in October, 2017

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	D4.2
Deliverable Nature	Other
Dissemination Level	Public
Contractual Date of Delivery	Project Month 21(June2017)
Actual Delivery Date	30.10.2017
Deliverable Description	9 software modules delivered to the E-CAM repository in the area of meso– and multi–scale modelling, and their documentation.

Document Control Information

Document	Title:	Meso– and multi–scale modelling E-CAM modules I
	ID:	D4.2
	Version:	As of Monday 30th October, 2017
	Status:	Accepted by WP leader
	Available at:	https://www.e-cam2020.eu/deliverables
	Document history:	Internal Project Management Link
Review	Review Status:	Reviewed
	Action Requested:	Submit
Authorship	Written by:	Burkhard Dünweg (DE-SMSM, Max Planck Institute for Polymer Research)
	Contributors:	Jony Castagna and Silvia Chiacchiera, Science and Technology Facilities Council; Hideki Kobayashi, Max Planck Institute for Polymer Research
	Reviewed by:	Ignacio Pagonabarraga and Ana Catarina Mendonça, EPFL
	Approved by:	Ignacio Pagonabarraga, EPFL

Document Keywords

Keywords:	E-CAM, CECAM, Module, DL_MESO_DPD, ESPResSo++, GPU, DPD, dipole moments, polymer melts, equilibration, coarse–graining

Monday 30th October, 2017

[1] duenweg@mpip-mainz.mpg.de

Contents

List of Figures

Executive Summary

In this report for Deliverable 4.2 of E-CAM, nine software modules in meso– and multi–scale modelling are presented. Five of the modules have been implemented in DL_MESO_DPD:

- GPU implementation of DL_MESO_DPD
- Analysis of dipole moments
- Autocorrelation function of dipole moments
- Autocorrelation function of individual dipole moments
- Formatting the HISTORY files of DL_MESO_DPD.

Furthermore, there are four modules that have been implemented in ESPResSo++. Together they form an implementation of a novel hierarchical strategy to generate equilibrium structures of polymer melts:

- Md-Softblob
- Minimize Energy
- Fixed-Local-Tuple
- Constrain-COM

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 characterized by a description of the physics "somewhere in between" atomistic Molecular Dynamics and engineering–style continuum mechanics. Many mesoscale 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.

The software modules of the present report are all integrated into big software packages and thus extend their applicability. The first package is DL_MESO_DPD [2, 5] that provides a toolbox to do DPD simulations. DPD ("dissipative particle dynamics") is essentially Molecular Dynamics, however not for atoms but for effective particles that represent many microscopic degrees of freedom. These effective particles 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 second package is ESPResSo++ [3, 4], which is based upon Molecular Dynamics but adds lots of mesoscale aspects like DPD, coupling to a Lattice Boltzmann fluid, and, in particular, is well suited for systematic coarse–graining applications. Both packages have previously (in deliverable 4.1 [1]) been identified as important codes for mesoscale simulations within E-CAM. The modules that are being presented here have been developed by the E-CAM programmer Jony Castagna (Sec. 2.1), the E-CAM postdoc Silvia Chiacchiera (Sec. 2.2), and the E-CAM postdoc Hideki Kobayashi (Sec. 3). The postdocs worked within the framework of pilot projects which implied a strong interaction with industry (S. Chiacchiera: Unilever; H. Kobayashi: Michelin).

1.1 Overall scope of the module set

The modules presented in this report were generated within three projects. Firstly, the DPD code DL_Meso_DPD has been ported to graphics cards (GPUs). Secondly, there are DL_Meso_DPD modules that deal with the static and dynamic dielectric response of materials, i. e. with their behavior in external electric fields. Thirdly, there are ESPResSo++ modules that deal with dense polymer systems represented by atomistic models or simplified bead–spring models. For such systems, it is exceedingly difficult to computationally generate samples that are in proper thermal equilibrium, where the chains take random coil conformations and are nicely packed with respect to each other. This problem becomes worse and worse with increasing molecular weight and has plagued the polymer simulation community for decades. It cannot be solved by simply adding more computer power, because in a straightforward Molecular Dynamics simulation it is not due to insufficient sample sizes but rather insufficient observation time. Recently, there was a breakthrough made by Guojie Zhang et al. [6], where a system of high molecular weight is mapped onto an equivalent one with lower molecular weight (i. e. a more coarse-grained system). The latter system can be equilibrated more easily. After generating such a conformation, the more detailed description is re-introduced. This procedure can be done recursively, such that one obtains a method that is in spirit quite similar to multigrid methods. The modules provided here aim at making this method available to a wide audience of academic and industrial users.

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 behavior 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. 2.1) is thus inherently useful in order to provide cheaper calculations.

The modules for dielectric response are part of a more general strategy to take into account both solvent flow and the response to external electric fields. This has been used (to just mention one example for practical applications) in experiments that translocate charged DNA molecules through membranes, which in turn allows an efficient way of genome sequencing.

Concerning the modules in ESPResSo++, it should be mentioned that there is direct interest in the complex mechanical (or viscoelastic) behavior of dense polymer systems e. g. in industries like oil, plastics, and rubber (last not least car tires). The modules have been developed in direct collaboration with the tire industry (Michelin). Equilibrated samples can be subjected to externally imposed deformations, such that the (linear and nonlinear) viscoelastic response (i. e. the time-dependent stresses in the material) can be determined. This opens the perspective of computer-assisted material design, where the influence of molecular weight, molecular architecture and molecular chemistry can be studied. There is particular interest in the behavior of block copolymers, and the next software development steps will aim at the ability to study such more complex architectures.

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 exploitations and industrial applications have been outlined above.

2 Modules based on DL_MESO_DPD

The base code for the following five modules is DL_MESO_DPD [2, 5], the Dissipative Particle Dynamics (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.

2.1 First GPU version of the DL_MESO_DPD code

In order to accelerate the DL_MESO_DPD code on the latest and future exascale hardware, a first version for NVidia GPUs has been developed. This is only a starting point, it does not yet cover all the possible cases and it does not yet support multiple GPUs. However, it represents an HPC milestone for the application, complementing the already present parallel versions developed for shared and distributed memory (MPI/OpenMP). The GPU version to CPU version performance analysis can be found in the module documentation and in deliverable D7.2.:E-CAM software porting and benchmarking data I (submitted to the EU in September 2017 and available from the project website here).

In this version, the full computational workload is offloaded to the GPUs with the "H_MAINLOOP" call present in the "dlmesodpd.f90" file. In this way, the initialising IO operations are left unaltered and fully compatible with the serial version. A major change compared to the serial version is in the algorithm used to find the particle-particle interaction forces: in order to guarantee better coalescent access for the CUDA-threads, the algorithm has been re-adapted to the GPU architecture reordering the cell-linked list arrays.

The FORTRAN files (mainly unaltered from the serial version) are saved in the "src" folder, while the CUDA files with their corresponding headers files are in stored in the "src_cuda" folder and "headers", respectively. The folder "bin" contains the Makefile. This arrangement of the folders allows to use the hidden Eclipse IDE project files (.cproject, .project, .settings).

Future plan involves extending the GPU version to other type of forces (bond, angle, dihedral, etc.), Ewald summation methods for systems with charged particles and multiple GPUs on the same node and across nodes.

Merge Request	Merge Request for first version on GPU files
Direct Documentation Link	Documentation for first version on GPU files

2.2 Analysis tools for polarizable systems in DL_MESO_DPD

The next four modules, available under BSD license, are post-processing utilities to be used with DL_MESO in its last released version, version 2.6 (dating November 2015). They have been developed in the context of the pilot project 1 of WP 4, which concerns the derivation of a realistic polarizable model of water to be used in DPD simulations. This project involves a collaboration between computational scientists (STFC Daresbury), academia (University of Manchester), and industry (Unilever).

The first three modules are strongly connected to the pilot project and are meant to analyze polarizable solvents (e. g., compute their electric permittivity). They involve quantities related to charge-dipole moments, and can be applied to systems including molecules with a generic charge structure, as long as each molecule is neutral (otherwise the charge dipole moment would be frame-dependent), and no free charges are present. The fourth module is a generic formatting tool for the trajectory files.

These modules have been (and are being) used within final year projects of Master Engineering students at Manchester University, under the supervision of Prof. Andrew Masters. Three projects, considering polarizable fluids made of dimers and trimers, were completed in May-June 2017, by Susan Shujun Liu, Jamie Sukumaran, and Michael Casey.

2.2.1 Analysis of charge dipole moments

The module `gen_dipole.f90` is a generalization of the `dipole.f90` post-processing utility of DL_MESO_DPD.

It processes the trajectory (HISTORY) files to obtain the charge dipole moments of all the (neutral) molecules in the system. It produces files `dipole_*` containing the time evolution of relevant quantities (e. g., the system total dipole moment). In the case of a single molecular species, it also prints to the standard output the Kirkwood number g_k and the relative electric permittivity ϵ_r for this species (computed in linear response theory), together with an estimate for their errors (standard error).

The module can be used when designing models with a target value of ϵ_r. The Kirkwood factor g_k measures the orientational correlations between the dipoles ($g_k \simeq 1$ in absence of correlations). Its value is useful to confirm the applicability of theoretical approximations, often assuming $g_k = 1$. Possible uses of the output files are: monitoring the polarization in response to an external electric field, measuring the fluctuations in molecular/total charge dipole moments.

Merge Request	Merge Request for gen_dipole.f90
Direct Documentation Link	Documentation of gen_dipole.f90

2.2.2 Autocorrelation functions of charge dipole moments

The module `gen_dipoleaf.f90` is a generalization of the `dipoleaf.f90` post-processing utility of DL_MESO _DPD. It processes the trajectory (HISTORY) files to obtain the charge dipole moments of all the (neutral) molecules in the system, and subsequently computes the Dipole Autocorrelation Functions (DAFs) for each molecule type. As an output, it produces a file `DIPAFDAT` containing both the un-normalized and normalized DAFs, and, optionally, a file `DIPAFFFT` containing the Fourier transform of the latter.

The output file `DIPAFDAT` can be analyzed to obtain the decay time of autocorrelations, which can be used to define an efficient sampling of the simulation. It can be compared with the analogous microscopic one obtained for individual molecules (see next module).

Merge Request	Merge Request for gen_dipoleaf.f90
Direct Documentation Link	Documentation of gen_dipoleaf.f90

2.2.3 Autocorrelation functions of individual charge dipole moments

The module `gen_moldipaf.f90` is similar is purpose to `gen_dipoleaf.f90`, but applies to microscopic rather than macroscopic charge dipole moments. As an output, it produces a file `MDIPAFDAT` containing both the un-normalized and normalized DAFs, and, optionally, a file `MDIPAFFFT` containing the Fourier transform of the latter.

Also `MDIPAFDAT` can be analyzed to obtain the decay time of autocorrelations.

Merge Request	Merge Request for gen_moldipaf.f90
Direct Documentation Link	Documentation of gen_moldipaf.f90

2.2.4 Formatting the HISTORY files

The module `format_history.f90` converts the trajectory (HISTORY) files from unformatted to a human readable form, (optionally) including explicative comments about all the quantities.

This module is mainly for learning/checking purposes. The first aim is to help the user to check that the system was prepared as intended (e. g., showing all the bead properties and initial positions, all the bonds etc). The idea is to use it on small systems when familiarizing with the structure of input files needed for the simulation. Secondly, it can be used as a starting point for a user-defined analysis of trajectories.

Merge Request	Merge Request for format_history.f90
Direct Documentation Link	Documentation of format_history.f90

3 Modules based on ESPResSo++: Hierarchical strategy for simple one-component polymer melts

ESPResSo++ [3, 4] ("Extensible Software Package for Research in Soft Matter based upon C++") is a general-purpose simulation package for soft-matter research, mainly developed at the Max Planck Institute for Polymer Research Mainz. It is freely available under the GNU Public License.

We here present a suite of programs which implement the recursive equilibration strategy of Guojie Zhang et al. [6] within ESPResSo++. In our opinion, the strategy is so far the best method to efficiently find well-equilibrated polymer configurations. It is hoped that the port into a freely available package will make this novel method usable and popular within the community of polymer simulators at large, both in an academic and an industrial context. Figure 1 illustrates the hierarchical equilibration strategy used, which is further explained below.

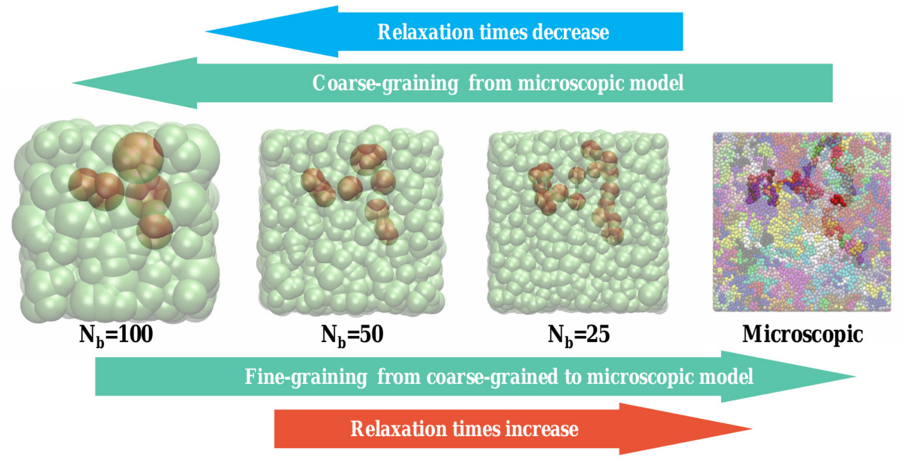
Relaxation times decrease
Coarse-graining from microscopic model
N_b=100
N_b=50
N_b=25
Microscopic
Fine-graining from coarse-grained to microscopic model
Relaxation times increase

Figure 1: Schematic illustration of the hierarchical equilibration strategy.

To decrease the relaxation time, microscopic monomers are coarse-grained (CG) by mapping each subchain with N_b monomers onto a soft "blob". The CG system is then characterized by a much lower molecular weight and thus is equilibrated quickly. One thus obtain a configuration that is equilibrated on large scales but does not provide information about the structure on smaller (i. e. more fine-grained (FG)) scales.

To obtain the latter, the resolution is increased by applying a fine-graining procedure to the previous (more coarse-grained) level. In such a fine-graining step, each CG polymer chain is replaced with a more fine-grained chain, by dividing a CG blob into several FG blobs.

Both coarse-graining and fine-graining are done recursively such that the lowest level is a polymer melt with full atomistic details, while the top level is a melt that contains chains with only a very low degree of polymerization (in the extreme case, just $N = 1$), composed of monomers that are very soft blobs.

The FG blobs that result from the fine-graining step are set up in such a way that their conformation is consistent with the conformation at the more coarse-grained level. After this setup, the local FG conformation is relaxed into a local equilibrium, again consistent with the (fixed) CG blobs.

3.1 Md-Softblob

The present module is the central part of the program suite. It implements the actual coarse-graining step, which is therefore described in more detail:

A polymer chain, originally consisting of N monomers, is replaced by a coarse-grained (CG) chain consisting of N/N_b soft blobs linked by a harmonic bond potential, $V_{bond} = 3k_B T d^2/2b^2_{CG}$, and an angular bond-bending potential $V_{bend} = k_B T k_{bend}(1+\cos(\theta))/2$. Here d is the distance and θ is the angle between consecutive bonds. The interactions between non-bonded soft blobs are taken into account by a repulsive pair potential $V_{nb} = k_B T \epsilon U_G(r_{ij})$. Here r_{ij} is the center-to-center distance between the two blobs, $U_G(r_{ij})$ is a Gaussian function with variance $\overline{\sigma}^2 = \sigma_i^2 + \sigma_j^2$ and σ_i is the gyration radius of blob number i. The gyration radius σ is in turn fluctuating. This fluctuation is controlled by the potential $V_{sphere} = k_B T(a_1 N_b^3 \sigma^{-6} + a_2 N_b^{-1}\sigma^2 + a_3\sigma^{-3})$. Equilibrated configurations of soft blobs are generated by Molecular Dynamics (MD) simulations based on the above model.

Within the module, the following classes have been implemented or modified:

- A *VSpherePair* class for calculating $V_{nb} = k_B T \epsilon U_G(r_{ij})$
- A *VSphereSelf* class for calculating $V_{sphere} = k_B T(a_1 N_b^3 \sigma^{-6} + a_2 N_b^{-1}\sigma^2 + a_3\sigma^{-3})$
- A *Particle* class for comunicating the property "radius" between different cores
- A *LangevinThermostatOnRadius* class for simulating the fluctuations of the radii of the blobs

Merge Request	Merge Request for MD-Softblob
Direct Documentation Link	Documentation for MD-Softblob

3.2 Minimize Energy

When the microscopic monomers are re-inserted into the soft blobs, the polymer configurations should satisfy a local energy minimum to avoid overlap between particles. This module provides a steepest-descent method, which is a typical energy minimization method.

This module is a modification of the already existing class *espressopp.integrator.MinimizeEnergy*. The modifications are as follows:

- Corrected the procedure of particle redistribution to cells.
- A variable relaxation of the energy per step γ is implemented. In this case, the position of particles is updated following the equation:

  $$p_{i+1} = p_i + (d_{max}/f_{max})F_i$$

  where d_{max} is the maximum update of particle coordinates in a single steepest-descent step and γ is adjusted via $\gamma = d_{max}/f_{max}$ where f_{max} is the maximum force in a single step.

These modifications significantly stabilize the procedure of redistributing particles to cells, and any value d_{max} less than half the skin parameter of the Verlet list can be used.

Merge Request	Merge Request for Minimize Energy
Direct Documentation Link	Documentation for Minimize Energy

3.3 Fixed-Local-Tuple

For re-inserting the microscopic details, the set of FG blobs is set up in such a way that its conformation is consistent with the conformation at the more coarse–grained level. After this setup, the local FG conformation is relaxed into a local equilibrium, again consistent with the (fixed) CG blobs.

This procedure needs a data structure where tuples of particles are stored in lists. In contrast to simple Molecular Dynamics, which is based only on pairs of particles, we here need to allow tuples of arbitrary size. The present module provides this more general data structure. The list can contain both real and ghost particles.

Merge Request	Merge Request for Fixed Local Tuple
Direct Documentation Link	Documentation for Fixed Local Tuple

3.4 Constrain-COM

A re-inserted set of FG blobs must be set up in such a way that its conformation is consistent with the conformation at the more coarse-grained level. Therefore the center of mass (COM) of the set of FG blobs must coincide with the center of the corresponding CG blob, during an initial period which is long enough that nearly perfect local equilibrium is reached. After that, the constraint is lifted.

The present module provides the C++ class for applying a suitable constraint that conserves the position of the COM of N FG blobs.

Merge Request	Merge Request for Constrain-COM
Direct Documentation Link	Documentation for Constrain-COM

4 Outlook

The deliverable contains nine modules that are all integrated in two big software packages (DL_MESO_DPD and ESPResSo++) that focus on meso– and multi–scale modelling. All of them have been accepted into the E-CAM software library.

In the future, we plan to develop modules for simulating one system by using two different degrees of coarse–graining within two different parts of one simulation box (the so–called AdResS, “adaptive resolution scheme”, method). Furthermore, the hierarchical strategy for dense polymer systems shall be extended to polymer mixtures and block copolymers. The GPU implementation of DL_MESO_DPD shall be extended further.

References

Acronyms Used

DPD	Dissipative Particle Dynamics
STFC	Science and Technology Facilities Council
DAFs	Dipole Autocorrelation Functions

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/158
duenweg@mpip-mainz.mpg.de . . . mailto:duenweg@mpip-mainz.mpg.de
http://creativecommons.org/licenses/by/4.0 . . . http://creativecommons.org/licenses/by/4.0

Page 3

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

Page 4

DL_MESO . . . https://www.scd.stfc.ac.uk/Pages/DL_MESO.aspx
here . . . https://www.e-cam2020.eu/deliverables/
Merge Request for first version on GPU files . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-M
merge_requests/18
Documentation for first version on GPU files . . . http://e-cam-meso-and-multi-scale-modelling-modules.readthedocs.io/en/latest/modules/DL_MESO_DPD_onGPU/add_gpu_version/readme.html
pilot project 1 of WP 4 . . . https://www.e-cam2020.eu/pilot-project-unilever/

Page 5

Merge Request for gen_dipole.f90 . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modules/
merge_requests/10
Documentation of gen_dipole.f90 . . . http://e-cam-meso-and-multi-scale-modelling-modules.readthedocs.io/en/latest/modules/DL_MESO_DPD/dipole_dlmeso_dpd/readme.html
Merge Request for gen_dipoleaf.f90 . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modules
merge_requests/26
Documentation of gen_dipoleaf.f90 . . . http://e-cam-meso-and-multi-scale-modelling-modules.readthedocs.io/en/latest/modules/DL_MESO_DPD/dipole_af_dlmeso_dpd/readme.html
Merge Request for gen_moldipaf.f90 . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Module
merge_requests/27
Documentation of gen_moldipaf.f90 . . . http://e-cam-meso-and-multi-scale-modelling-modules.readthedocs.io/en/latest/modules/DL_MESO_DPD/moldip_af_dlmeso_dpd/readme.html
Merge Request for format_history.f90 . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modul
merge_requests/9
Documentation of format_history.f90 . . . http://e-cam-meso-and-multi-scale-modelling-modules.readthedocs.io/en/latest/modules/DL_MESO_DPD/format_dlmeso_dpd/readme.html

Page 6

Merge Request for MD-Softblob . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modules/
merge_requests/28
Documentation for MD-Softblob . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modules/
blob/master/modules/hierarchical-strategy/components/md-softblob/readme.rst

Page 7

Merge Request for Minimize Energy . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Module
merge_requests/23
Documentation for Minimize Energy . . . https://gitlab.e-cam2020.eu:10443/e-cam/Meso-Multi-Scale-Modelling-
blob/master/modules/hierarchical-strategy/components/minimize-energy/readme.rst
Merge Request for Fixed Local Tuple . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Module
merge_requests/29
Documentation for Fixed Local Tuple . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modul

edit/master/modules/hierarchical-strategy/components/fixed-local-tuple/readme.rst
Merge Request for Constrain-COM . . . https://gitlab.e-cam2020.eu/e-cam/Meso-Multi-Scale-Modelling-Modules
merge_requests/30
Documentation for Constrain-COM . . . https://gitlab.e-cam2020.eu:10443/e-cam/Meso-Multi-Scale-Modelling-
edit/master/modules/hierarchical-strategy/components/constrain-com/readme.rst

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

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

[3] http://www.espresso-pp.de/.

[4] Jonathan D. Halverson, Thomas Brandes, Olaf Lenz, Axel Arnold, Stas Bevc, Vitaliy Starchenko, Kurt Kremer, Torsten Stuehn, and Dirk Reith. ESPResSo++: A modern multiscale simulation package for soft matter systems. *Computer Physics Communications*, 184(4):1129–1149, April 2013.

[5] Michael A. Seaton, Richard L. Anderson, Sebastian Metz, and William Smith. DL_meso: highly scalable mesoscale simulations. *Molecular Simulation*, 39(10):796–821, September 2013.

[6] Guojie Zhang, Livia A. Moreira, Torsten Stuehn, Kostas Ch. Daoulas, and Kurt Kremer. Equilibration of High Molecular Weight Polymer Melts: A Hierarchical Strategy. *ACS Macro Letters*, 3(2):198–203, February 2014.