UMMAP: a statistical analysis software package for molecular modelling

David J. Bray, Annalaura Del Regno and Richard L. Anderson

Published version information

Citation: DJ Bray, A Del Regno and RL Anderson. "UMMAP: a statistical analysis software package for molecular modelling." Molecular Simulation, vol. 46, no. 4 (2019): 308-322

DOI: 10.1080/08927022.2019.1699656

This is an Accepted Manuscript of an article published by Taylor & Francis in Molecular Simulation on 16 December 2019, available online: http://www.tandfonline.com/10.1080/08927022.2019.1699656

UMMAP: A statistical analysis software package for molecular modelling

David J. Bray[a, *], Annalaura Del Regno[a,†] and Richard L. Anderson[a]

[a] The Hartree Centre, STFC Daresbury Laboratory, Warrington, UK

ARTICLE HISTORY
Compiled January 15, 2020

ABSTRACT
The Universal Molecular Modelling Analysis Package (UMMAP) is a software package designed to provide a command line driven analysis suite for particle based simulation codes. It is developed to work closely alongside the mesoscale dissipative particle dynamics software embedded in DL_MESO, and the molecular dynamics codes NAMD and LAMMPS. UMMAP is designed to be extendable to other modelling software. UMMAP is targeted at users working in the computational soft matter field and those looking for an 'all-in-one' analytical software package. It offers a unique mix of analytical tools, from widely used geometrical analyses (e.g., bond and angles distributions, structure factor, radial distributions and density profiles), to specific solutions for the soft matter community (e.g., polymer statistics, critical micelle concentration and cluster/micelle shape analyses). It provides the possibility, for a more expert user, to build in additional bespoke analyses, whilst benefiting from UMMAP 's core functionality. It is very flexible in the way it reads and analyses the trajectory, allowing for several tools to be used at the same time *via* a single command line input. A rational set of software flags and detailed error messages render UMMAP ergonomic and user friendly while output integrity is ensured by storing manifest files to prevent the storage of unrelated material.

KEYWORDS
Analytics; Simulation; Dissipative Particle Dynamics; Molecular Dynamics; Software

1. Introduction

Molecular simulation methods such as molecular dynamics (MD), [1] coarse-grained molecular dynamics (CGMD) [2] and dissipative particle dynamics (DPD) [3,4], are established computational techniques widely used in the study of materials chemistry and soft matter. In these techniques, molecules are represented as interacting particles which have discrete positions and move in discrete time-steps; often contained within a finite bounded system, they can give a discrete local representation of the medium's microstructure.

Over the years, several high performance simulation engines have been developed to help standardise the execution of molecular simulations, including GROMACS [5], NAMD [6], LAMMPS [7], and DL_MESO [8], among others. A key output produced by

∗ David J. Bray. Email: david.bray@stfc.ac.uk
† Present address: BASF SE, Materials Molecular Modeling, Carl Bosch Str. 38, 67056, Ludwigshafen, Germany

these engines is a trajectory file which stores information on all the particles (such as their positions or velocities) at regular time intervals. Statistical analysis of trajectory files is a key part of extracting useful information from simulations; these can range considerably depending on the nature of the problem being studied and often require specific tools to be written. While the biochemistry community has been well supplied with several high-level bespoke analytical tools, such as the GROMACS utility tools [5] or Visual Molecular Dynamics (VMD) [9], other fields (e.g., surfactant chemistry) often require either the modification of existing packages, or the development of in-house analytical codes. This can limit the use of computational techniques, especially as part of an industrial discovery process, where time dedicated to code development may be limited. An additional level of complexity derives from the existence of multiple trajectory formats, more often than not, associated with specific simulation engines; these include, but not limited to, *XYZ* (LAMMPS), *PDB*, *TRR* (GROMACS), *DCD* (NAMD, LAMMPS) and *HISTORY* (DL_MESO). Moving from one engine, thus one trajectory format, to another may require additional changes to the analytical tools.

The Universal Molecular Modelling Analysis Package (UMMAP) is designed to work with a large number of trajectory file formats. It is easy to operate and presents a high degree of functionality providing the user with valuable insights in a wide range of application areas. Additionally, its flexibility allows for new modules to be easily implemented, while relying on its core functionality. UMMAP was conceived in 2014 as part of a research program exploring computer aided formulation for industry (InnovateUK project 101712). At the time there was no specific analysis software dedicated to the study of micelle properties or phase behaviour of surfactants aggregates. One possibility could have been to create a series of analytic codes with specific functions. However, this solution would not have been feasible in the long term as different simulations packages, versions and modelling methods (e.g., DPD and MD) were used. This would have created a menu of different analytic code options that could easily become difficult to manage and maintain. Furthermore, each independent analysis code would have required the same core functionalities, for example, a trajectory reader and particle selector. It was therefore decided that a general analysis code, with a trajectory reader decoupled from analytical tools, was needed. UMMAP was created by implementing a front-end interface on top of the separate modules for trajectory reading and analysis.

UMMAP is an attempt to fulfil four key criteria: (a) to provide a wide range of analytical tools while allowing for easy inclusion of new functionality; (b) to provide systematic output clearly organised; (c) to provide a detail record of the analysis performed, for easy identification at a later date; (d) to provide high levels of support for the user to maximise usability. The latter three criteria are critical in insuring the data remains useful well into the future. Design considerations for UMMAP are given in Table 1.

The current version of UMMAP has been used extensively in the study of micelle formation [10–12], octanol-water partition coefficients [13,14], DPD model parametrisation [11,13,14], analysis of liquid phase mixing and separation, and the study of crystallisation and polymer melts.

This article aims to provide a general introduction of the UMMAP software. It is organised as follows. In Section 2 we present the software implementation, giving the reader an introduction to UMMAP 's technical structure, tasks and core routines. In Section 3 a detailed guide to the code and its functionality is presented. Section 4 covers the outputs from UMMAP. In Section 5 a collection of case studies are provided. Additional discussion, conclusion and an overview over future developments

Table 1. Design considerations of UMMAP.

1.	Should contain a range of analysis, supplementary and post analysis tools with functionality that is extendable and can be customised;
2.	Should consist of independent modules where the handling of trajectories is separated from the analysis tools;
3.	Should be provided with a command line based text interface, sensibly organised with a supplementary parameter input file for setting infrequently changed properties;
4.	Should allow for easy trajectory and topology inquires;
5.	Should provide the ability of selecting specific time frames within a given trajectory;
6.	Should grant for a sophisticated group selection criteria with two types of grouping available: simple list of particles or linked-list of particles;
7.	Should provide spatial modelling facilities for testing analysis and reference models;
8.	Should provide a structured output directory, detailed manifest of analysis performed and clearly organised log file;
9.	Should enable fast generation of plots and images from selected data outputs;
10.	Should provide intelligible error messages to guide towards solutions;
11.	Should be suitable for use on both personal desktop and high performance computing (HPC) facilities.

are available in Section 6.

UMMAP is available *via* restricted license terms from the Science and Technology Facilities Council and is free for academic use.

2. Implementation

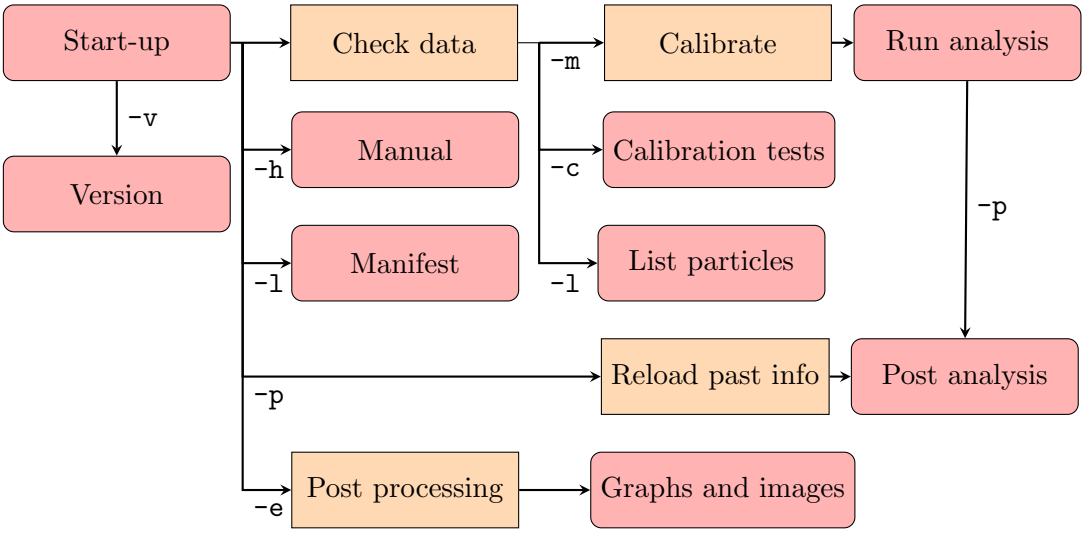
UMMAP 's source code is written in C, with readers for DL_MESO written in Fortran 95. It is developed using Git version control [15], tested under continuous integration and stored in a GitLab repository [16]. UMMAP is compiled with CMake [17] using standard C and Fortran compilers, such as those provided by gnu, Intel or IBM. The coding style is highly modular, i.e., analytic tools, trajectory reading, core routines and library utilities are all developed as independent modules. C data structures are used to transfer complex information within UMMAP; each data structure is designed to hold information on a single concept. UMMAP 's main concepts are outlined in Table 2, while its general organisation is shown in Figure 1. The modular and data structure design of UMMAP are key to enabling later expansion of functionality by either developers or users.

UMMAP's core routine can be summarised as follows: (i) initiation; (ii) analysis selection; (iii) analysis calculation; (iv) clean up and close. Figure 2 shows a simplified diagram of UMMAP's core routines and how these interconnect: green boxes indicate decision points; red boxes are the start of key routines; the purple box indicates user input; orange boxes give the break down of main analysis routines; and yellow boxes

Table 2. UMMAP's main concepts.

File System	Holds data on directories paths;
Parameters	Stores data on input parameters defined from command line and parameter input file;
System Data	Holds information on trajectory and topology of particles and properties of the simulation;
Particle Groups	Stores information on simple group selection;
Advance Particle Groups	Stores information on complex group selection;
Regions	Defines localised regions of space;
Measurements	Holds data for each analysis tool by type and instance.

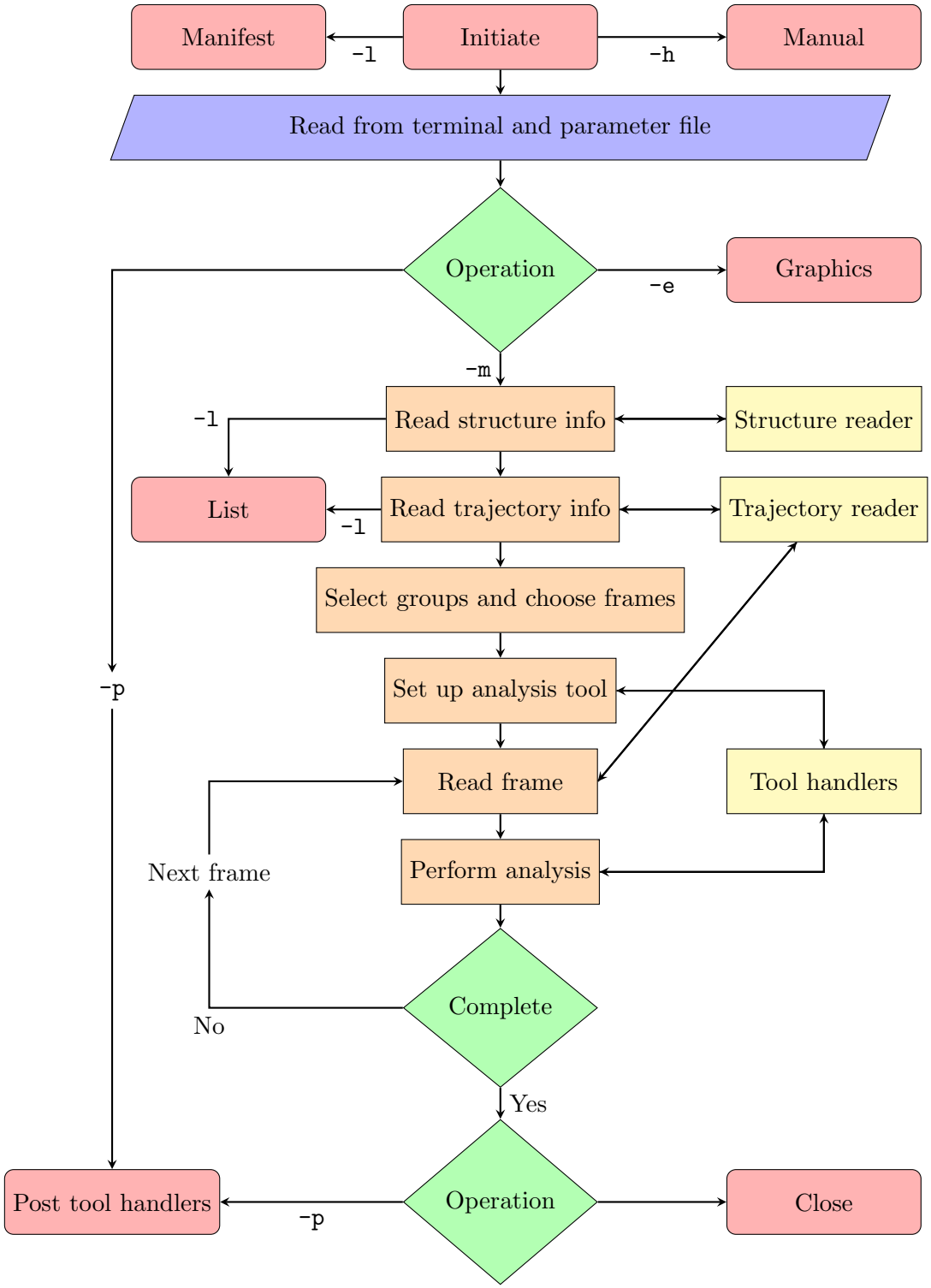
Figure 1. High level flow diagram of UMMAP 's tasks. Flag names (software toggles) are shown next to the corresponding principle operation.



indicates the self-contained non-core modules which extend UMMAP 's functionality. Where a decision point occurs an indication is displayed by way of the feature flag used (software toggles) of the task that will enact the specific path. In UMMAP these flags are specified by the input of specific text, such as `-m`, on the command line. For example, if the main analysis is called (`-m` flag) then UMMAP will (i) read information from the structure and trajectory files, (ii) setup groups, masses and analysis tools, (iii) read the required trajectory and run through the various analysis called by the user and finally (iv) clean up and close down. More information on the flags used in UMMAP is presented in Section 3.1.

The flow diagram in Figure 2 provides insight into the code's modular design where each specific analysis, type of structure and trajectory reader has its own set of template sub-routines called from a common master set of functions. Modular design is also present within each task, where the main analysis (`-m`), post analysis (`-p`) and post graphics (`-e`) follow their own functional flow. Further standardisation and

Figure 2. Simplified flow diagram of UMMAP 's core routines. `-x` indicates routes activated using the corresponding `-x` terminal flag.



modularisation is achieved by packaging frequently used routines, such as common mathematical functions, compilation and printing of histograms, handling of periodic boundary conditions, opening/writing and closing of files, for example.

3. Using UMMAP

3.1. The UMMAP *Interface*

The UMMAP command line interface is flag-based and it is designed to allow for a variety of user abilities. A minimum set of flags are required from an unfamiliar user to perform simple tasks, while a series of sub-flags allow advanced users to fine-tune different analyses. As many flags can be used simultaneously, the interface has been specifically designed to help the user navigate through the different options.

UMMAP input is very flexible. For example, flags can be expressed in any combination, there is no case sensitivity and the absence of spaces is handled internally, improving the user experience. Additional protections have been implemented against duplicate flag calling, out of range inputs and invalid command combinations, which will result in termination and an error message being displayed in standard output on the console. Error messages specifically highlight the offending flag/text within the console. When a keyword is expected, a dictionary of valid keywords will be shown. If the user specified a value that is not acceptable then UMMAP will notify the user of the acceptable ranges or options.

UMMAP 's flags fall into four categories; tasks, mandatory parameters, optional parameters and tool customisation flags. The most important flags define the tasks. UMMAP has eight principal tasks (see Table 3) which can be invoked individually, such as `-h` or in combination, such as `-m` with `-p`.

Table 3. UMMAP's principal tasks' flags.

Flag	Task action	Example use
-v	version	`ummap -v`
-h	manual	`ummap -h`
-l	list information on manifest	`ummap -l -o` *directory*
-l	list information on simulation	`ummap -l -f` *trajectory*
-m	frame analysis	`ummap ··· -m C`
-p	post analysis	`ummap ··· -p CMC`
-c	tool calibration	`ummap ··· -c C`
-e	graphical post processing	`ummap -e` *directory*

The next important set of flags are the key parameters flags (See Table 4). These provide the mandatory information that UMMAP needs to perform the selected task (Table 3) and are specific to that operation. For example, the main analysis task (`-m`) requires information regarding the name of the trajectory file (specified with the flag `-f` *file*), additional topological information provided by structure file (through flag `-s` *file*, if required) and the group selection (specified with the flags `-g/n` *selections*) to perform the analysis. UMMAP accepts several popular trajectory formats, such as *HISTORY*, *DCD*, *PDB*, *VTF*, *XYZ* and structure files, such as *FIELD*, *PDB*, *GRO*,

LAMMPS input file, and it is designed to recognise the specific format automatically from the naming conventions. Additional optional flags (See Table 4) can be used to specify the name and location of the output directory (`-o` *directory*) or select specific time frames from the trajectory file (`-t` *selection*).

The syntax for frame selection is designed to be intelligible and valid keywords combinations include: `FIRST` X, `LAST` X, `AFTER` X, `BETWEEN` X Y and `SINGLE` X. For example, `-t BETWEEN` X Y means read between frame number X and Y and `-t AFTER` X means read after frame X. UMMAP is designed to allow a high degree of customisation including providing new topological information that overrides or complements those given by the trajectory/topology files. Among other things, users can define or re-define the system box dimensions, boundary conditions and particle masses and charges.

Table 4. Examples of UMMAP's mandatory parameter and optional flags.

Flag	**Task action**	**Example use**
-f	specify trajectory file	`ummap ··· -f` *`filename`*
-s	specify structure file	`ummap ··· -s` *`filename`*
-g	group selection	`ummap ··· -g {C6} resname HEXANE`
-n	linked-list group selection	`ummap ··· -n {vector} A1 A2`
-t	select time frames	`ummap ··· -t LAST 5`
-o	specify output directory	`ummap ··· -o` *`directory`*
-setpbc	set periodic boundary conditions	`ummap ··· -setpbc XYZ`
-setbox	set system box dimensions	`ummap ··· -setbox 5.5 5.5 10.5`
-setmass	adjust masses of particle types	`ummap ··· -setmass` *`filename`*
-setcharge	adjust charges of particle types	`ummap ··· -setcharge` *`filename`*

The final type of flag is the tool customisation flags which control how the actual analysis is performed and are specific to the analysis being performed. These are always found as part of the individual tool specification given in the input for the `-m`, `-p` or `-c` task flag and are a set of positive signed flag statements placed directly after each tool declaration within square braces, e.g. for frame analysis these take the form: `-m` *`Tool`* `[+`*`option`*`]`. The brackets are a visual aid used to subjugate these tool options from the overall interface. Example tool customisation flags include `+t` which specifies the number of time frames to average over, and `+l` which is used to define the point spacing of a histogram. Further detail on the use of tool customisation flags is covered in Section 3.3 on performing analysis and actual examples are given in the case studies of Section 5.

3.2. Finding Information

As well as the full text manual, UMMAP is provided with an in-built manual on the functions available and how to access them. Specific examples are provided to aid the user further. This manual can be accessed by typing the flag `-h` into the command line, i.e.

```
ummap -h
```

An example of a section of the manual is shown in Figure 3 (a).

One feature of UMMAP 's is its ability for the user to query simulation information such as system topology, simulation box size, trajectory length and also previously performed analyses. This information is displayed directly onto the screen and stored in the log file (more details can be found in Section 4).

After reading in the trajectory, UMMAP produces tables which show the labelling available for group selection; an example is provided in Figure 3 (b). Information is given on the naming, number of particles of each name, residue names and their weight fractions, known masses and charges. A breakdown of particles in each residue (molecule) type is also displayed. Similarly to VMD [9], in UMMAP particles can be classified by residue name (keyword: `resname`), atom/particle type (keyword: `type`), atom/particle name (keyword: `name`), atom/particle number (keyword: `id`), residue id number (keyword: `resid`) and segment name (keyword: `segname`). Uniquely, UMMAP allows users to select indices based locally on their position within a molecule template using the keyword `resatomid`. For example, the fifth particle listed in a molecule is `resatomid 5`. This feature can also be used to implicitly identify different molecule types based solely on the combination of order and number of atoms present in the molecule template. Information on trajectory files can be obtained using a combination of file and list flags,

```
ummap -f traject.vtf -s FIELD -l
```

The example here shows a *VTF* formatted trajectory file and FIELD topology file as produced by DL_MESO. Executing this command will cause UMMAP to print to screen all information about the topology contained within the FIELD file and information on the trajectory file such as number of (complete) time frames, boundary conditions and system box size. This command is particularly useful for understanding trajectory files quickly without opening them in a text editor and when file names are non-descriptive, such as those used in DL_MESO.

Users are able to inquire about the contents of any pre-existing UMMAP output directories, by using the list flag in combination solely with the output file flag,

```
ummap -l -o directory
```

This produces a summary screen on the output directory, compiled from the UMMAP manifest files stored in the directory, and outlines details on all past UMMAP runs carried out in this directory. Details are given regarding the associated trajectory files sampled in previous runs, all analysis performed and the names of all the particle groups defined by the user.

3.3. Performing Analysis

There are three types of analysis contained within UMMAP; primary frame analysis, supplementary frame analysis and post analysis. Examples of each type of analysis can be found in Table 5. Some of UMMAP 's tools are collected together as themed packages consisting of a primary analysis and set of supplementary and/or post analyses, for example the cluster package. Primary analysis (`-m` flag) is performed directly on the trajectory data. Examples include radial distribution, cluster analysis and density analysis. Supplementary frame analysis tools (also called with `-m` flag) extend the primary analysis and pull information from both the trajectory file and the primary analysis tool output, for example cluster shape tool provides additional analytics to the cluster tool. Post analysis (`-p` flag) enables additional analysis types to be carried

out at a later date without the need to read from trajectory data, for example the critical micelle concentration (CMC) tool uses the stored data generated by the cluster analysis tool.

3.3.1. Frame Analysis

At startup UMMAP stores all the topology information about the system (i.e., particle names, particle types, residue names) and these descriptions are assumed to remain unchanged during the simulation. UMMAP next generates particle groups selections.

The user specifies groups using the `-g` or `-n` flags and a combination of logic and keyword statements. Two types of groups can be specified: a simple list of particle indices via flag `-g`, where each entry is evaluated separately (such as needed for calculating means); and a list of sets of N particle indices which can be evaluated together (such as defining vectors or angles) *via* flag `-n`.

For `-g`, complex group selection can be set up by using a combination of lists, logic keywords `AND`, `OR`, `NOT` and `[ ]` bracket notation. Groups can also be named using the { } brackets, rendering a collection of beads/atoms more recognisable to the user. Multiple groups can be specified at the same time using a colon separator. For example,

```
-g {head} [ALK_1 ALK_2] AND resname OCTANE : {water} H2O
```

specifies two particle groups called `head` and `water`.

For `-n`, groups are selected as a set of particle `names` or `resatomids` found within one or more `resname` type. To be valid UMMAP will expect the number of occurrences of each `name` listed to be the same. To limit selection to a particular `resname` the logic keyword `OF` can be tacked on the end followed by either keyword `resname` *word* or `resid` *index*. For example,

```
-n {vector} ALK_1 ALK_2 OF resname OCTANE
```

specifies a list of all particle pairs consisting of particle types `ALK_1 ALK_2` found in each `resid` of `resname OCTANE` and names the group `vector`.

After group generation, UMMAP initiates each analysis tool specified. Analysis of a trajectory file is performed as follows:

```
ummap -f filename -g group selection -m Tool [+options] ···
```

Uniquely, UMMAP can simultaneously open different analysis tools, as requested, and run multiple instances of each.

A specific group is associated with each tool at this time; this is accomplished by either (i) using the groups specified by the user with the flag `-g` or `-n` or (ii) by selecting some of the groups within the tool using the `+g/+g2/+g3` tool customisation options. This gives the user complete control over what is analysed and means that for the cost of a single read through the trajectory file, several analyse can be performed independently on the specific groups required. This has the benefit of reducing computational time significantly when compared to running each analysis separately. For example:

```
ummap -f traject.vtf -g ALK : {water} H2O -m GR [+g ALK] SNAP [+g
                                water]
```

calculates the radial distribution for all particles called `ALK` and takes snapshots of the group called `water` at each time frame sampled. The customisation options for the analysis (in this case `+g` which defines the group to use) are placed within `[ ]` brackets

just to the right of their associated tool (in this case `GR` and `SNAP`).

During frame analysis, each frame is read individually into memory, and then overwritten when the next frame is read based on the frame selection defined using `-t` (by default UMMAP will read and analyse all frames). This means that UMMAP can analyse very large trajectories (i.e., several million particles, or thousands of frames), of several terabytes in size using only the memory required to process a single frame. It is also possible to read from a sequence of trajectory files which means that UMMAP can analyse simulations that have been restarted several times even if the trajectory is not contained within one file.

A special feature of UMMAP is the ability of restarting or extending any analyses from the point they were stopped. This is very beneficial, especially when dealing with large trajectories, as it allows analyses 'on-the-fly' without the need of re-running the calculations once the simulation is completed. This also provides an advantage when dealing with unstable HPC systems and remote accesses.

3.3.2. Post Analysis

Post analysis tools build upon the frame analysis tool using the output data produced instead of interacting directly with the raw trajectory data. To do this UMMAP's manifest log-files (Section 4.3) is used to determine which tools have been run and which files need to be reanalysed; this means that they can be setup without having to store data from the main tool in memory and allows them to be run at a later date without the trajectory data being present.

Post analysis can be done in two ways; along side the main analysis,

`ummap -f` *`filename`* `-m` *`Tool`* `[+`*`options`*`] -p` *`Post Tool`* `[+`*`options`*`]`

or after the main analysis.

`ummap -p` *`Post Tool`* `[+`*`options`*`] -o` *directory*

Thus the following single command: `ummap -f traject.vtf -g ALK -m C -p CMC`, can be carried out in two stages. First, `ummap -f traject.vtf -g ALK -m C -o` *`directory`*; followed by, `ummap -f traject.vtf -g ALK -p CMC -o` *`directory`*.

Table 5. Some of UMMAP's analysis types specified with the `-m` or `-p` command. Primary, supplementary and post analysis indicated by [1], [2] and [3] respectively.

Flag	Tool name	Description
		General Tools
`B`[1]	Bond Distance	Calculates distributions and statistics of vectors;
`ANG`[1]	Angle	Calculates distributions and statistics of planar angles;
`DIH`[1]	Dihedral	Calculates distributions and statistics of dihedral angles;
`GR`[1]	Radial Distribution	Calculates a variety of radial distribution;
`SF`[1]	Structure Factor	Calculates variations of the structure factor;
`SEP`[1]	Separation Distance	Calculates data on the minimum distance between groups;
`POLY`[1]	Polymer	Calculates common statistics associated with polymers;
`G`[1]	General Properties	Gives global properties about the system such as density, volume, kinetic energy;
`MSD`[1]	Mean Square Displacement	Calculates the MSD for a group;
`DO`[1]	Distance Orientation	Calculates properties using planar angle and distance criteria between two vectors;
`SNAP`[1]	Snapshot	Take snapshots of the system at particular frames.
		Cluster Package
`C`[1]	Cluster Analysis	Distance base aggregation identification tool that collects composition and statistics on clusters and micelles;
`S`[2]	Cluster Shape	Calculates topological properties of clusters;
`CMC`[3]	Critical Micelle Concentration	Calculates measures for the CMC using the micelle distribution;
`CRD`[2]	Cluster Radial Distribution	Collection of radial distributions centred around core of micelles that can be used to understand preferences of chemical groups;
		Density Package
`D`[1]	Local Density	Calculates the three dimensional density profile and reduced dimensional forms;
`PC`[3]	Liquid Phases and Partition Coefficient	Identifies interface and molecular composition of liquid bulk phases.

3.4. Generating Images with UMMAP

It is possible to automatically generate figures from certain output files or images from snapshot data with UMMAP by invoking the command `ummap -e` *`directory`*. This is currently facilitated by UMMAP setting up (by generating input files) and executing the external graphic programs gnuplot (for graphs) and VMD (for snapshot images of the system). This feature requires the availability of the relevant packages.

3.5. Spatial Modelling Function

One of UMMAP abilities is that it can be used to produce reference data from standard models that can be used to compare trajectory data against. For example, comparing a calculated radial distribution function (Section 5.1) to a randomly packed solution. This enables the user to evaluate their finding against known configurations. UMMAP does this by first, generating a 'virtual' frame of particles who's positions are distributed based on a chosen spatial model (such as particles distributed at random, in clusters or on a lattice grid) and then analysing the frame in the same way a real trajectory would be analysed. By sampling a sufficient number of frames the statistical behaviour of the reference system can be generated. Reference model types currently available are; complete spatial random model (`CSR`); uniform lattice distributed models (`SPP`); model based on probability distribution (`PP`); model based on random walk (`RW`). Modelling can be invoked using the following command format,

`ummap -model` *`model`* `[`*`+options`*`] -m` *`Tool`*

For example, invoking `ummap -model CSR [+n 500 +l 10 10 10] -t FIRST 100` will generate 100 frames of 500 particles distributed at random in the system of dimension $10 \times 10 \times 10$.

4. Data Output Structure

UMMAP can be executed from any location in the file system by specifying trajectory and output desired location. If the output directory file already exists, a check is performed to assess if the trajectory file is the same as in a manifest file (Section 4.3), reducing the chances of contaminating the output directory with different simulation data. UMMAP will also prevent the user from overwriting an existing particle group with a different selection or from reassigning masses, radii or charges.

4.1. On Screen and Log-file Output

During execution UMMAP prints a formatted output to the screen (Figure 3(c)). Different colours are used to help draw the users attention to important parts of the output. For example, errors are highlight with blinking red keywords while the topology of the simulation is contained within orange walled text boxes. If errors with the command line input are present, UMMAP will give an indication to the location of the error and guide the user to the solution (Figure 3(d)). Improper set-up of tools, input files reading problems, discrepancies between manifest record and current set-up are also highlighted by UMMAP. In all cases, the presence of an error will cause UMMAP to terminate.

A human readable log-file is generated each time UMMAP is executed and contains detail on the tools used, their set up, details on the trajectory and frames analysed (Figure 3(b)), for example. Information on previous runs of UMMAP can also be found in the backup log-files, labelled in the output directory by prefixing a hash symbol onto the log-file and attaching a post-fixed bracketed number ordered such that the latest previous run has the highest number.

4.2. Output Directory and File Formats

Each time UMMAP is executed the results are sent to an output directory (defined *via* `-o` *directory*), which is hierarchically organised by being divided into sub-directories called *Data*, *Snaps*, *Images*, *Graphs*, *Reports* and *Utilities*. An example of the directory organisation is shown in Figure 4. All Statistics and data generated by UMMAP are stored into the *Data* sub-directory categorised by the analytic tool name which generated it. Similarly, snapshots of the trajectory will be stored into the *Snaps* sub-directory. The *Utilities* directory holds all manifest files stored by UMMAP (such as particle groups, tools run, masses and charges of particles, for example).

UMMAP produces several types of output, including raw data gathered per frame, statistics, means and time-averaged histograms and annotated frame snapshots of the particles that can be used to visualise properties in relation to the system of particles.

All UMMAP output files follow a common naming convention, beginning with a keyword indicating type of statistics (such as `TIME`, `STAT`, `MEAN`, `HIST`), followed by a meaningful description specific to the tool, then bracketed group names which identify the groups used. Histograms have additional naming criteria including, an indication of the frame number produced at and the number of frames averaged over. For example, `TIME_Cluster_stats_[G1].dat` is the time series statistics of the average cluster as obtained from the cluster analysis tool using group `G1` and `HIST_Gr-INTER_[CHCH]-[CHCH]_BT00000100_FT00000099.dat` is a block time averaged histogram over 100 frames for the radial distribution of group `CHCH` taken at frame 99.

All statistics files (*Data* folder) are text files in *CSV* format, using spaces between entries. Each file comes with a header where information on the column format and a descriptive title of the file is reported as comment lines (each started with a hash). These files are suitable for loading in graphing software such as gnuplot or xmgrace.

Frame snapshots (*Snaps* folder) provide information on particle coordinates, and can be outputted in the following common formats such as *PDB*, *VTF*, *XYZ*. The naming convention is similar to statistic files with the frame number given at the end. When using *PDB* or *VTF* the occupancy and beta column is used to store descriptive properties unique to the tool used to generate the output. For example with the cluster shape tool the occupancy and beta column is used to store the size and id of the cluster that the particle belongs to, respectively.

4.3. Manifest Files

One of the key criteria around which UMMAP was designed is data integrity. Preventing the user from mixing analysis from different sources is achieved in UMMAP through the manifest files, stored in the *Utilities* sub-directory. UMMAP uses the manifest files to store information regarding the input files read, past particle groups selected, and tools that were used. At start-up UMMAP will check if the manifest files already exist

(a)

```
        G or GENERAL : General tool. Provides infomation on the basic properties of the sytem

        Command:


                ... -m G [ option]

        Options are:

             +o :(TYPE (TYPE ...)) Set type options for output (BOX, VOL, COM, GROUP etc.). User
                                   can specify a list of as many options as required

        Example Uses:


                ummap -f traject.vtf -g ALK -m G

                ummap -f traject.vtf -g ALK -m G [+o BOX VOL]
Type any other key to go back to full list of tools
Type q to quit
```

(b)

```
-------------------------------------------------------------------------------------------
|                          PARTICLE TYPES (KEYWORD: ATOMTYPE)                             |
-------------------------------------------------------------------------------------------
|                         Total number of particle types is 4                             |
| ------------------------------------------------------------------------------------  |
|Index Name                Particle Count (particle % | wf%)     Mass   Charge   Radius |
| ------------------------------------------------------------------------------------  |
|[  0] CH3OH                     4863 (  48.620% |  48.620% )     1.00     0.00     1.00 |
|[  1] H2O                       2883 (  28.824% |  28.824% )     1.00     0.00     1.00 |
|[  2] CHCH                      1692 (  16.917% |  16.917% )     1.00     0.00     1.00 |
|[  3] CH3                        564 (   5.639% |   5.639% )     1.00     0.00     1.00 |
-------------------------------------------------------------------------------------------
```

(c)

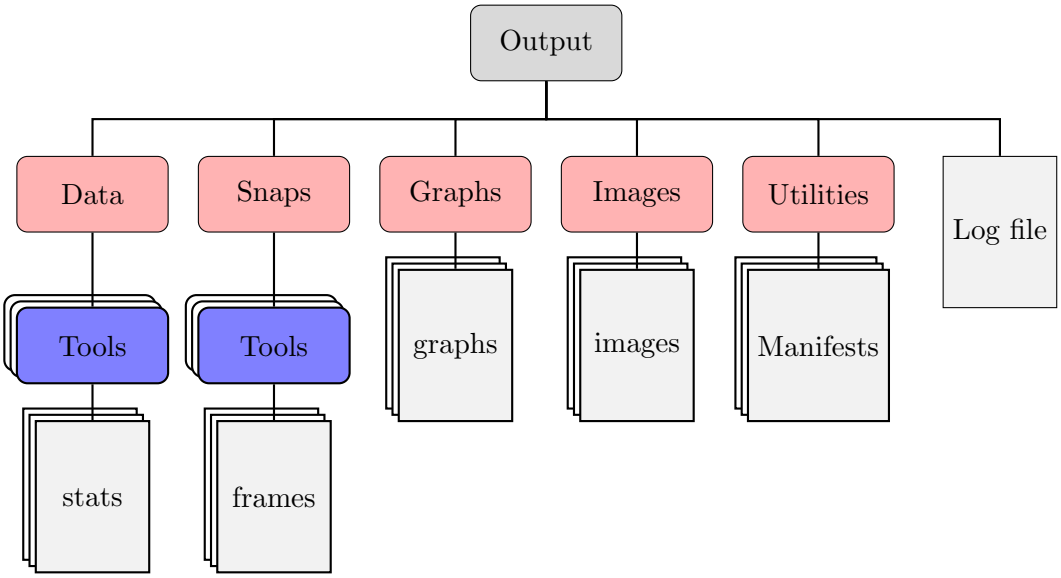
```
###############################################################################################
#                                                                                             #
#                                        Welcome to UMMAP                                     #
#                                                                                             #
###############################################################################################

     This is UMMAP - A post analysis tool for molecular simulations
     Developed at STFC Daresbury Laboratory, UK ((C) 2014-2018)
     Author is  D. J. Bray

###############################################################################################
#                                                                                             #
#                                          STARTUP                                            #
#                                                                                             #
###############################################################################################

STARTUP             : You are running UMMAP Program version  2.752900
STARTUP             : Date performed Wed Nov 28 23:35:23 2018

STARTUP             : This is the stdout output
STARTUP             : Initiation log file UMMAP_STARTUP_19327.log
```

(d)

```
?????????????????????????????????????????? Error Found ????????????????????????????????????????????

ANALYSIS INPUT ERROR (302): BAD is not an allowed keyword for +o
ANALYSIS INPUT ERROR (302): Allowed keywords are: BOX VOL DENSITY COM GROUPCOM KE GROUPKE MOM
                    GROUPMOM

        Did you mean?

                        BOX
                        VOL
                        DENSITY
                        COM
                        GROUPCOM
                        KE
                        GROUPKE
                        MOM
                        GROUPMOM

Terminal input was:

../bin/ummap -f traject.vtf -g CH3 -m  <<  G  >> [ +o  <<  BAD  >> ] -o UMMAP_OUT


????????????????????????????????????????????????????????????????????????????????????????????????
```

Figure 3. Example of (a) in-tool manual, (b) part of the log file showing topology, (c) screen output and (d) error message.

Figure 4. Hierarchical file organisation of output directory routine.



and that the source input files, i.e., trajectory file and structure file, match the record. If a discrepancy is found an error message is generated and the program is terminated. Compatibility is also checked between previously defined particle groups and correspondent naming, in order to prevent any ambiguity. The manifest files are also used for rapid file reading, when performing post analysis.

5. Case Studies

In this section, selected UMMAP 's analytic capabilities are demonstrated on four test cases. All simulations adopt DPD technique and were performed using the software package DL_MESO, however, the same analyses could easily be carried out on other MD codes listed in the Introduction.

5.1. General Simulation Statistics: Isotropic Liquid

A number of different metrics can be important in understanding behaviour of a simulated system. For example, the radial distribution function can give the probability of finding a particle at a distance, r, from a reference particle. It is useful for understanding the microstructure of a simulated system. Calculated mean squared displacement (MSD) can give details of the diffusivity of a system and can be compared to experimental observables from neutron diffraction [18] or dynamic light scattering [19]. Figure 5(a) shows an image of a system comprised of a mixture of decane and benzene. This image was generated by UMMAP *via* VMD (see Section 3.4). A schematic representation of the coarse-grained bead structure of a molecule of benzene (benzene beads carry the label `type CHCH`) and are coloured green) and decane (alkane beads carry the label `type C` and are coloured cyan) is also shown. Analyse for the two molecules produced by UMMAP are shown for the mean square displacement – MSD, (Figure 5(b)), intra-molecular radial distribution – $g(r)$ (Figure 5(c)), bond distribu-

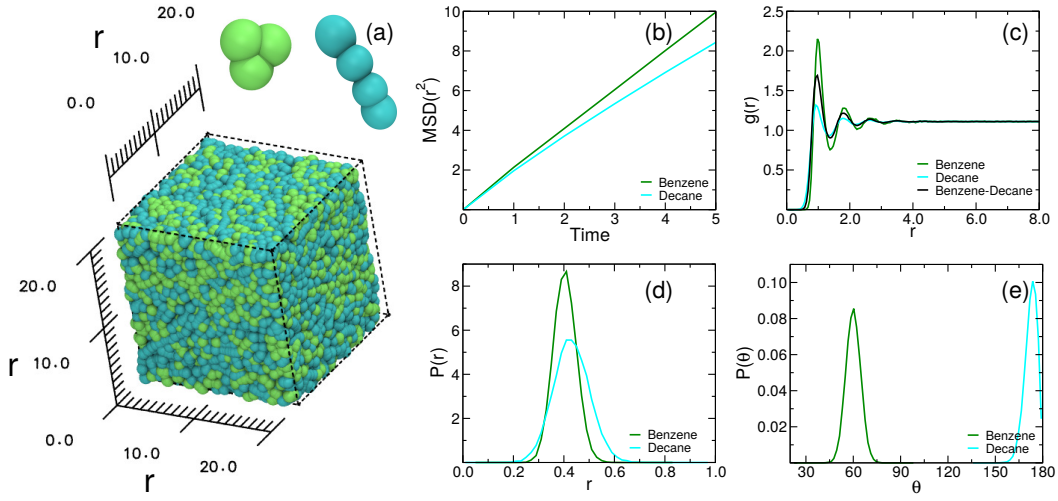


Figure 5. Examples of the (a) snapshot of the system, (b) mean square displacement, (c) radial distribution, (d) bond distribution and (e) angle distributions.

tion – $P(r)$ (Fig. 5(d)) and angle distribution – $P(\theta)$ (Figure 5(e)). Data for these images was generated by a single command as follows,

```
ummap -f trajectory -g BENZENE : DECANE -n {BENZEN2E} CHCH_0 CHCH_1
      CHCH_2 : {DECANE2} C_0 C_1 C_2 C_3 C_4 -m GR MSD B ANG
```

Here UMMAP distinguishes different beads in the molecule (in the case of DPD) by adding a underscore followed by a number to the particle type (where names are given then UMMAP will use these). The `GR` (radial distribution) and `MSD` (mean square displacement) tools use the groups defined by `-g` flag while the B (bond distribution) and `ANG` (angle distribution) tools use the groups defined by `-n`.

5.2. Micelle Properties: Surfactant-Water Mixture

Surfactants are amphiphilic molecules, consisting of hydrophilic head groups and hydrophobic tails. In aqueous environments surfactants can assemble into aggregates known as micelles as a way of minimising interactions between water molecules and their hydrophobic tails. Micelles only form when the CMC of the surfactant is exceeded. Depending upon the chemistry, composition, and concentration of a surfactant, a range of micelle configurations may be formed such as spheres, rods and worm-like structures. Micelles are ubiquitous and play a role in many important processes. For example, micelles can be can act as friction modifiers, [20,21] are responsible for the properties of cleaning products [22,23] and are used to control drug release, [24–30]. Simulations can play an important role in understanding the behaviour of surfactants and micelles.

One of UMMAP's key analytical tools is the micelle analysis package which consists of the cluster aggregate tool, the secondary tools micelle (cluster) shape and cluster radial distribution, and the post analytics CMC tool (`CMC`). In this test case, we show how UMMAP can be used to obtain data on micelles of sodium dodecyl sulphate, SDS. The coarse-grained bead representation of the SDS surfactant is $[CH_3][CH_2CH_2]_3[CH_2OSO_3^-][Na^+]$ which is present in aqueous solution (coarse-

grained bead type – H_2O). Details of the model and specific simulations for the system presented can be found in the work of Anderson *et al.* [11]

To extract micelle properties from UMMAP the core cluster aggregation tool (`C`) is called using the flag combination,

```
ummap -f trajectory -g group selection -m C [+options]
```

The cluster aggregation tool uses a distance based nearest-neighbour criteria to identify spatial networks of particles that represent cluster aggregates. The nodes of each network are defined using particle positions given by a particle list specified by the `-g` terminal flag. While edges of the network connect pairs of nodes that lie within an adaptable cut-off distance R. An initial trial value is provided for R and this can either be optimised further by the program or held fixed. If included as part of the criteria, all bonded particles are automatically considered to be in the same cluster. In atomistic models non-bonded particles are unlikely to be closer together than bonded particles thus the second criteria is not needed however, when considering coarse-grained models, it is possible for non-bonded beads to be in closer proximity to each other than a pair of bonded beads. In these cases, each bond represents a connection between beads rather than a chemical bond. This presents a complication for a clustering algorithm based on a single distance cut-off length between beads (without bond inclusion) as the appropriate cut-off distance between non-bonded beads may lead to longer distance bonded beads being treated as in separate clusters.

The cluster algorithm first determines lists of connections between particle pairs, so that every particle obtains a list of the connected neighbours. These connections are efficiently obtained from a large system by decomposing the system into a three dimensional grid of boxes with side-lengths of R and allocating to each box those particles located within their boundaries. Then particles within neighbouring boxes and within the same box are checked for connections. During this process UMMAP takes into account the orthorhombic system box boundary conditions and applies periodic boundaries conditions to the correct edges of the system box using a minimum image convention which allows connections that wrap around the box to be identified.

Once all particle lists are obtained the clusters can be identified by working through the network formed from all interconnected connections. To identify a new cluster an unassigned test particle is chosen and all connecting particles assigned to the same cluster. Next, each of these particles are used to identify the following set of particles that are directly connected along the network of connections. Paths that lead to particles already assigned in the network are ignored. This process is repeated until only terminal particles are found (i.e., those only connected to already assigned particles) and the full network of the cluster is mapped. Then a new unassigned particle is found and the process repeated until there are no unassigned particles left and all clusters are identified. The advantage of this method is that particles are only assigned once.

The centre of mass of the cluster can be determined during this process of cluster assignment by unwrapping the particle positions ($\mathbf{x}_i$) on the fly as the cluster network is grown. Here if a connection crosses over a boundary then the position of the as yet unplaced particle is mapped to minimise the separation distance between it and the already placed connected particle. This causes the cluster to fully unwrap across as many boundaries as required which is useful when the cluster extents are many times the box size. Better representation of cluster packing within the system can be obtained by translating the whole cluster so that the centre of mass lies within the boundaries of the system box.

Figure 6(d) shows an image snapshot of the simulation system generated by UMMAP

using the cluster tool. Here only the surfactant is shown and each individual aggregate is highlighted by a single colour.

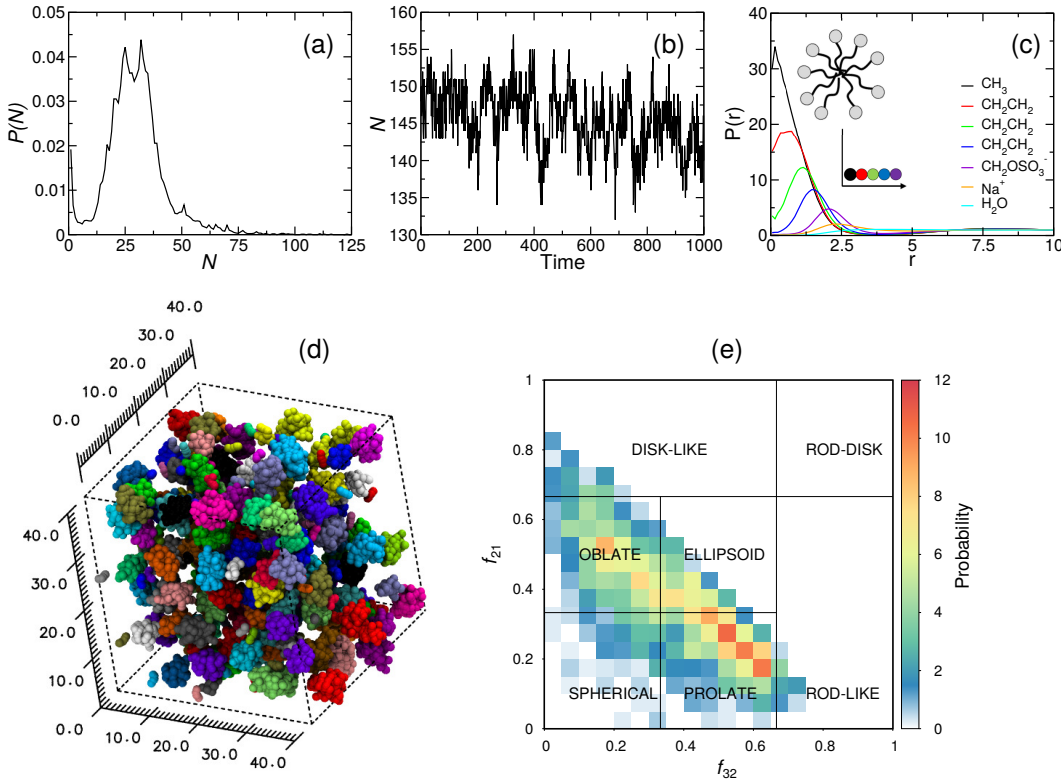


Figure 6. Examples of the (a) cluster size distribution, (b) number of aggregates as a function of frame time, (c) radial distribution between centre of a micelle and a particle group, (d) snapshot of clusters and (e) micelle shape classification.

With aggregates identified, the cluster tool can be used to evaluate a number of statistical properties, such as the time averaged distribution of cluster sizes (Figure 6(a)), and obtain time series statistics, such as the mean aggregate size given in molecules (Figure 6(b)).

The micelle shape tool extends the capability of the cluster tool in two ways; firstly, by classifying clusters into either micelles (large aggregates) or monomers (small aggregates); secondly by characterising the shape of the cluster. The tool is called using the command,

`ummap -f` *`trajectory`* `-g` *`group selection`* `-m C [`*`+options`*`] S [`*`+options`*`]`

Cluster type is classified as micelle (rather than monomers/pre-micellar) by the following user-defined criteria. Clusters must be of a sufficient size, defined by a fixed number of molecules (i.e., $N < N_m$); clusters must on average exceed a minimum number of contacts per particle node. The shape of a micelle (or additionally aggregates) is determined by principle component analysis. A gyration tensor is calculated for each micelle cluster using,

$$\mathbf{G} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \tag{1}$$

The eigenvalues and eigenvectors are determined from,

$$\mathbf{G}\mathbf{P}_j = (\lambda_j)^2\mathbf{P}_j, j = 1, 2, 3 \tag{2}$$

where $\mathbf{P}_j$ gives the principle axes of the cluster. λ_j gives the gyration length in each principle direction. From λ_j a variety of shape metrics are determined including, maximum extents ($\mathbf{E}$), radius of gyration, asphericity, acylindricity and the flatness measures $f_{32} = (\lambda_3 - \lambda_2)/\lambda_3$, about major-mid principle axes, and $f_{21} = (\lambda_2 - \lambda_1)/\lambda_3$ scaled flatness, about mid-minor principle axes. These flatnesses are used as part of the criteria for shape classification when combined with user-defined size cut-offs; N_m, N_{rod}, N_{disc}, and flatness cut-offs; $\epsilon, \epsilon_{rod}, \epsilon_{disc}$. Thus currently the cluster shape tool has eight finite shape classifications (See Table 6 for details); monomer, spherical, oblate, prolate, ellipsoid, rod, disc, rod-disc, and three extended states; worm-like, lamellar /slab-like and gel-like. For a shape to be considered extended it must cross at least two periodic boundaries.

Figure 6(e) shows how classification can give indications of the types of clusters present in the simulation. The plot shows the time-averaged probability of a surfactant molecule (contained in micelles only) being in a cluster with particular flatness properties (f_{21}, f_{32}). Overlaid on top is the corresponding shape classifications; the plot shows that the micelles are mainly prolate. These types of diagrams can be used for for identifying the micellar phase state of the system.

For near-spherical micelles the cluster radial distribution tool can add additional information describing where chemical species tend to lie with respect to the core of a micelle. Figure 6(c) shows radial distributions between particular chemical groups and the micelle centre; in this case it is shown that the core of the micelle consists of CH_3 and CH_2CH_2 groups which are the tail of the surfactant, whereas the surfactant head group ($CH_2OSO_3^-$) lies further out and water (H_2O) and counter ions (Na^+) still further out. This type of plot can be used to provide information on micelle organisation and the degree of penetration and competition between various salts/ions,

Table 6. Shape classifications for the cluster shape tool. Here L_x, L_y, L_z are the system box dimensions, E_1, E_2, E_3 are the maximum extents of the target cluster, along the three principle directions (shortest to longest respectively), N is the size of the cluster, in molecules, and f_{21}, f_{32} its flatness measures. $N_m, N_{disc}, N_{rod}, \epsilon, \epsilon_{rod}$ and ϵ_{disc} are user defined cut-offs.

monomer	$N < N_m$
spherical	$f_{32} \leq \epsilon$ and $f_{12} \leq \epsilon$
oblate	$f_{32} \leq \epsilon$ and $f_{21} > \epsilon$
prolate	$f_{32} > \epsilon$ and $f_{21} \leq \epsilon$
ellipsoid	$f_{32} > \epsilon$ and $f_{21} > \epsilon$
rod	$N > N_{rod}$, $f_{32} > \epsilon_{rod}$ and $f_{21} \leq \epsilon_{disc}$
disc	$N > N_{disc}$, $f_{32} \leq \epsilon_{rod}$ and $f_{21} > \epsilon_{disc}$
rod-disc	$N > N_{disc}$, $N > N_{rod}$, $f_{32} > \epsilon_{rod}$ and $f_{21} > \epsilon_{disc}$
worm-like	spans the system box in one direction: $E_1 > \min(L_x, L_y, L_z)$.
lamellar	spans the system box in two direction: $E_1, E_2 > \min(L_x, L_y, L_z)$
gel-like	spans the system box in three directions: $E_1, E_2, E_3 > \min(L_x, L_y, L_z)$

solvents and additives.

5.3. Phase Separated Systems

Performing simulations of phase separated systems is a useful method for the determination of, for example, interfacial tension [31], solubilites or partition coefficients [13,32] (LogP). These simulations are usually set up such that the two phase separating components each occupy half of the simulation cell (Figure 7(a)). In the case of LogP and interfacial tension an additional component or molecule type is distributed throughout the overall cell and will reach equilibrium location based on the interaction parameters driving the system.

UMMAP 's density analysis package can be invoked to explore phase separated systems. The package consists of the local density tool and post analysis tools, i.e., profile projection and slicing, liquid phase detection and partition coefficient tools. The primary analysis tool is the density analysis (Table 5). The tool first breaks the system box into a set of orthorhombic sub-regions. Next it calculates the time-averaged density of each sub-region based on the mass of the particles contained within. It is called from the command line using the flag combination:

```
ummap -f trajectory -g group selection -m D [+options]
```

To graphically visualise the density UMMAP uses a post analysis tool that will allow the user to either profile the 3D local density along one dimension, such as the z-axis, or in two dimensions such as the $x-y$ area. In both cases, an average of the density is taken across the removed dimensions, e.g., profiling along the z-axis means averaging across the x and y dimensions. The resulting data output is suitably formatted to make either a $x-y$ plot or 2D colour image, respectively. An alternative 2D representation can be obtained by slicing the 3D local density into a series of areas.

For simulations, where multiple phase separated liquids may be present (Figure 7(a)), UMMAP provides a liquid phase detection post analysis tool. To use the tool

the local concentrations of each molecule must be first profiled using the density tool. This analysis can be performed with the following command:

```
ummap -f trajectory -g resname A : resname B : resname C -m D
      [+options] -p PC [+options]
```

The tool assumes that the liquid phases have well defined inter-facial boundaries and lie effectively perpendicular to a profiling axis, such as the z-axis. The user will need to confirm this prior to running the tool.

UMMAP first profiles along the chosen axis the projected 1-dimensional local density for each molecule specified (Figure 7(b)). It then detects the position of phase interfaces, by defining it as a sharp change in density beyond a cut-off value, between each 'bulk' phase with near constant concentration. The interface cut-off value can be defined in one of three ways; either a constant value, a fraction of the maximum difference in concentration or fraction of the standard deviation in concentration. Each identified phase (region between two interfaces) is then measured and one of the following outputs can be reported; amount concentration, mass fraction, molar fraction, mass concentration or density.

Measuring the partition coefficient is a special case of the liquid phase detection tool (Figure 7(c)). Here the simulation must be set-up with only three molecule types (solvent A, solvent B and solute S), where two of the molecule types are largely insoluble (A, B) such that they form two bulk phases and the third molecule type acts as the solute (S). UMMAP will report an error if these criteria are not satisfied. The partition coefficient of an uncharged solute molecule is the ratio of the molar concentrations in a pair of coexisting bulk phases. The bulk phases themselves are typically made up from a pair of immiscible solvents. The corresponding partition coefficient is usually reported as a base 10 logarithm,

$$\mathrm{LogP} = \log_{10} \frac{[S]_A}{[S]_B} \tag{3}$$

where $[S]_A$ and $[S]_B$ are the molar concentrations of a solute molecule in the two phases, A and B. UMMAP extracts LogP values from simulation data by direct application of Equation 3, using computed values of the solute concentrations in each of the solvent phases.

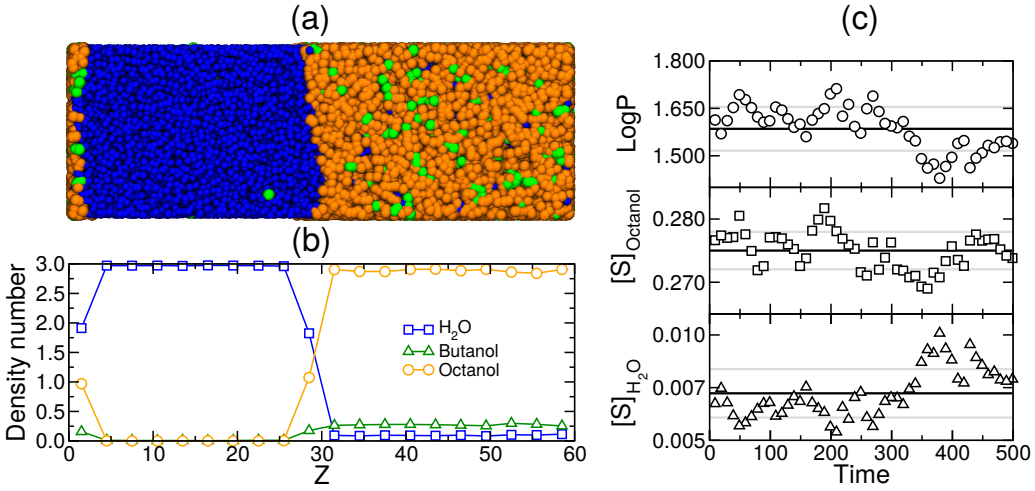


Figure 7. Examples of the (a) snapshot of the system showing the three molecule types octanol (orange), water (blue) and butanol (green), (b) projected 1D density profile across z dimension of system for the three molecule types and (c) from the top, butanol LogP, butanol in octanol concentration ($[S]_{\mathrm{Octanol}}$) and butanol in water concentration ($[S]_{\mathrm{H_2O}}$) versus time.

5.4. Identifying Order

Identifying order in simulations can provide useful information in many applications, for example in the exploration of liquid crystals [33] or in understanding wax formation and inhibition in *n*-alkane systems [34].

In this example, we show how the user can explore nematic ordering in a simulation cell containing n-alkane molecules. This type of simulation could be used to understand wax formation and/or designing wax growth inhibitors. Measures for studying nematic order are currently available as an customisation option of the bond tool (called *via* `-m B [+o ORI]`). The bond tool measures properties from a set of 'bond' vectors. Typically these vectors are defined by a pair of particles specified using the `-n` flag. When `+o ORI` option is invoked each bond vector is compared against a reference director vector defined, at each time frame, either as the global average of all bond vectors (when invoking `+director GLOBAL`) or as the local average direction of bond vectors found within a sphere of a specified radius (when invoking `+director LOCAL`). The radius is defined either as a fraction/multiple of the vector length or by a constant value. For example, the following command can be used to study global order:

```
ummap -f trajectory -n C_0 C_11 -m B [+o ORI +director GLOBAL]
```

Here `C_0 C_11` are two terminal particles in a DPD representation of n-tetracosane. The plane angle (θ) between bond vector and the director is calculated for each molecule and then used to derive the order parameter using the Legendre polynomial function given by,

$$P_2(\cos(\theta)) = \left\langle \frac{1}{2}\left(3 \times \cos(\theta)^2 - 1\right) \right\rangle \tag{4}$$

Figure 8 shows two example model systems, with samples of the local and global director as annotations, and a typical graph that may be obtained from this analysis. Highly ordered systems have an order parameter close to 1, with nematic order appearing when above 0.6 and isotropic systems around 0. Here the higher local and global order of the C22 + C24 case is clearly visible in the two order parameters.

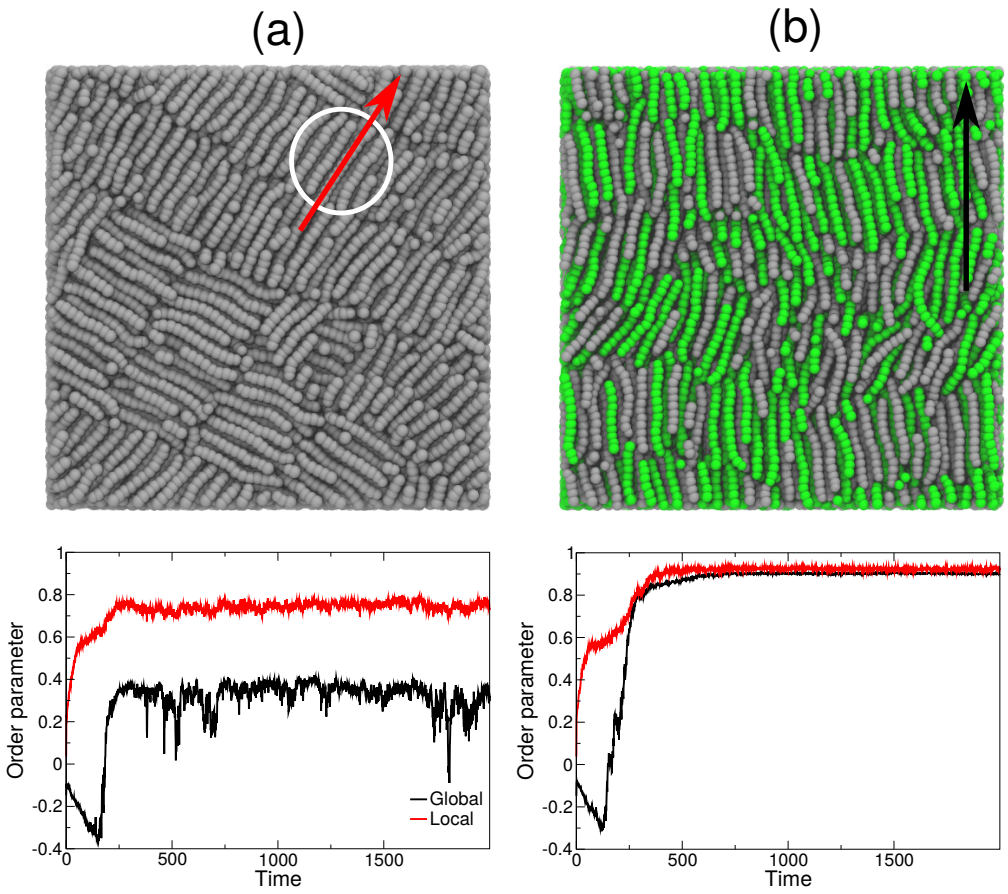


Figure 8. Example snapshots seen during simulation and corresponding plots of the order parameter as a function of simulation time obtained when specifying either local or global director for a model of (a) *n*-alkanes of carbon length C24 and (b) mixture of *n*-alkanes C22 + C24. The red arrow gives an example of the local director with the circle representing the local region which it is based on. The black arrow gives an example of the global director.

6. Summary

UMMAP is a MD and DPD simulation analysis package with a particular emphasis on soft matter. Largely agnostic of the simulation engine used it contains specific routines enabling the user to extract information about micelles, phase behaviour and liquid crystal behavior of studied systems. In addition to this, UMMAP offers many standard measurements customary of standard simulation analysis packages. UMMAP provides efficient routines for the reading of large trajectory files and for subsequent analysis (and re-analysis of data). The only requirement is that a single time-frame can fit into the available memory of the computer running UMMAP. UMMAP has been developed to enable the user to add additional analysis routines to the code as modules that can draw upon the existing core functionality of the code.

UMMAP aims to do things differently to other analysis by achieving three things: (1) reducing complexity for the user in running analysis; (2) enhancing flexibility and reducing the time costs of analysis tools; (3) producing maintainable and protected data outputs.

A key drive in developing this software was to provide to the liquid-based soft matter community the equivalent capabilities that have been available for sometime to the academics in the biological communities (i.e., in the form of VMD and GROMACS tools for example). One reason for wrapping the trajectory reading, group selection and time-frame selection in one tool (unlike, for example the GROMACS utilities where the user will be require to make a group first, save the group list and then use the latter in further analysis tools) is to reduce the complexity of running analysis and to allow the same set-up to be run as required on a wide range of problems. This is particular critical if such tools are to be picked up and widely used in industrial research settings. Here in one command the user can select a group (or multiple groups) of particles and run multiple analyses, potentially freeing up a user's time to do more science. UMMAP achieves its power through internal connectivity, each tool can know about other tools which are being simultaneously run and this allows the user to build up more complex tools *via* tool chaining.

UMMAP is designed to reduce repetition of work in order to reduce time to completion, i.e., don't gather again what you should already know. This is most notable with the post analysis where the trajectory is not re-read (which eliminates time consuming stage of identifying particle topologies and reading positional data) but gathers only the required information from the current output's manifest files and stored data. Gathering information from the output directory is also used when employing UMMAP to auto-generate graphs, images and reports.

The current article lays out the framework of UMMAP (version 2.7.5) and examples are presented describing how the user can extract information on a number of different types of systems. Future work for the code will focus on extending UMMAP 's capabilities to provide insight on a wider array of properties. We also plan to explore how the archiving facilities of UMMAP can be used to reduce the start-up costs of UMMAP. A time consuming part of current version is the start up cost of reading in the topology data and defining the groups (especially for large systems in excess of 1 million particle). If the user is analyzing the same trajectory multiple times, using different analysis options, considerable gain can be made by recalling the past information on groups and topology if it is stored in the manifest log file. This has particular benefits when used as part of high throughput workflows where hundreds of different runs may be performed.

UMMAP is distributed under restricted licence from the Science and Technologies

Facilities council and is free for academic use.

Acknowledgement(s)

The authors would like to acknowledge Michael Seaton who, as the main developer of DL_MESO, was invaluable in helping to iron out issue relating to reading and interpreting the codes trajectory files. The authors thank Michael Johnston, William Swope, Maria Panoukidou and Ennio Lavagnini for their help in shaping the UMMAP code. Early UMMAP development was sponsored by Innovate UK Project No. 101712, and the authors are grateful to the other members of the UK Computer Aided Formulation (CAF) consortium project for their input and for stimulating discussions. Improvements to UMMAP functionality and the usability was supported by the STFC Hartree Centre *Innovation: Return on Research programme*, funded by the UK Department for Business, Energy & Industrial Strategy.

Disclosure statement

The authors declare no conflicts of interest.

Funding

Early UMMAP development was sponsored by Innovate UK Project No. 101712. Improvements to UMMAP functionality and the usability was supported by the STFC Hartree Centre *Innovation: Return on Research programme*, funded by the UK Department for Business, Energy & Industrial Strategy.

References

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, New York, NY, USA, 1989.

[2] S. C. Kamerlin, S. Vicatos, A. Dryga and A. Warshel, *Annual Review of Physical Chemistry*, 2011, **62**, 41–64.

[3] P. B. Groot, Robert D.Warren, *J. Chem. Phys.*, 1997, **107**, 4423–4435.

[4] P. Español and P. B. Warren, *The Journal of Chemical Physics*, 2017, **146**, 150901.

[5] D. V. D. Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark and H. J. C. Berendsen, *Journal of Computational Chemistry*, 2005, **26**, 1701–1718.

[6] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé and K. Schulten, *Journal of Computational Chemistry*, 2005, **26**, 1781–1802.

[7] S. Plimpton, *Journal of Computational Physics*, 1995, **117**, 1 – 19.

[8] M. A. Seaton, R. L. Anderson, S. Metz and W. Smith, *Molecular Simulation*, 2013, **39**, 796–821.

[9] W. Humphrey, A. Dalke and K. Schulten, *Journal of Molecular Graphics*, 1996, **14**, 33 – 38.

[10] M. A. Johnston, W. C. Swope, K. E. Jordan, P. B. Warren, M. G. Noro, D. J. Bray and R. L. Anderson, *The Journal of Physical Chemistry B*, 2016, **120**, 6337–6351.

[11] R. L. Anderson, D. J. Bray, A. Del Regno, M. A. Seaton, A. S. Ferrante and P. B. Warren, *Journal of Chemical Theory and Computation*, 2018, **14**, 2633–2643.

[12] M. Panoukidou, C. R. Wand, A. D. Regno, R. L. Anderson and P. Carbone, *Journal of Colloid and Interface Science*, 2019, **557**, 34 – 44.
[13] R. L. Anderson, D. J. Bray, A. S. Ferrante, M. G. Noro, I. P. Stott and P. B. Warren, *The Journal of Chemical Physics*, 2017, **147**, 094503.
[14] J. L. McDonagh, A. Shkurti, D. J. Bray, R. L. Anderson and E. O. Pyzer-Knapp, *Journal of Chemical Information and Modeling*, 2019, **59**, 4278 – 4288.
[15] The Git Project, *Git –fast-version-control*, `https://git-scm.com`, Accessed: 2019-03-13.
[16] GitLab Inc., *GitLab*, `https://about.gitlab.com`, Accessed: 2019-03-13.
[17] K. Martin and B. Hoffman, *IEEE Software*, 2007, **24**, 46 – 53.
[18] S. Magazù, F. Migliardo and A. Benedetto, *The Journal of Physical Chemistry B*, 2010, **114**, 9268–9274.
[19] P. A. Hassan, S. Rana and G. Verma, *Langmuir*, 2015, **31**, 3–12.
[20] R. Zheng, G. Liu, M. Devlin, K. Hux and T. chi Jao, *Tribol. Trans.*, 2009, **53**, 97–107.
[21] H. Spikes, *Tribol. Trans.*, 2015, **60**, 5.
[22] S. Skoglund, T. A. Lowe, J. Hedberg, E. Blomberg, I. O. Wallinder, S. Wold and M. Lundin, *Langmuir*, 2013, **29**, 8882–8891.
[23] X. Tang, W. Zou, P. H. Koenig, S. D. McConaughy, M. R. Weaver, D. M. Eike, M. J. Schmidt and R. G. Larson, *J. Phys. Chem. B*, 2017, **121**, 2468–2485.
[24] Y. Kakizawa and K. Kataoka, *Advanced Drug Delivery Reviews*, 2002, **54**, 203 – 222.
[25] U. Kedar, P. Phutane, S. Shidhaye and V. Kadam, *Nanomedicine: Nanotechnology, Biology and Medicine*, 2010, **6**, 714 – 729.
[26] Z. Ahmad, A. Shah, M. Siddiq and H.-B. Kraatz, *RSC Adv.*, 2014, **4**, 17028–17038.
[27] A. Mandal, R. Bisht, I. D. Rupenthal and A. K. Mitra, *J. Control. Release*, 2017, **248**, 96 – 116.
[28] S. Biswas, P. Kumari, P. M. Lakhani and B. Ghosh, *Eur. J. Pharm. Sci*, 2016, **83**, 184 – 202.
[29] M. Cagel, F. C. Tesan, E. Bernabeu, M. J. Salgueiro, M. B. Zubillaga, M. A. Moretton and D. A. Chiappetta, *Eur. J. Pharm. Biopharm.*, 2017, **113**, 211 – 228.
[30] M. Yokoyama, *J. Drug Target.*, 2014, **22**, 576–583.
[31] B. Smit, A. G. Schlijper, L. A. M. Rupert and N. M. Van Os, *The Journal of Physical Chemistry*, 1990, **94**, 6933–6935.
[32] T. Taddese and P. Carbone, *The Journal of Physical Chemistry B*, 2017, **121**, 1601–1609.
[33] Z. Sumer and A. Striolo, *Phys. Chem. Chem. Phys.*, 2018, **20**, 30514–30524.
[34] S. Shahruddin, G. Jiménez-Serratos, G. J. P. Britovsek, O. K. Matar and E. A. Müller, *Scientific Reports*, 2019, **9**, 1002.