Documentation for UMCP continuum modeling software

May 6, 2019

Contents

This document is meant to be a practical guide to running the continuum model software, as well as a guide to the underlying theory. If a command or option does not work as expected or if usage is unclear, please let us know.

1 Introduction

The `hd` program models bilayers using the subdivision limit surface algorithm, previously applied to lipid bilayers by Klug [1].

[1]

2 Running dynamics: `hd`

Dynamics are computed using the program `hd`, short for Hamiltonian dynamics.

HAMILTONIAN DYNAMICS

Hamiltonian dynamics is a convenient formalism for propagating Newtonian dynamics with generalized variables (equivalent to constraints). Constraints and generalized variables (i.e., not necessary particle coordinate variables) are an essential feature of continuum membrane simulations, because the mesh is represented by control points and particles are frequently constrained to be on the membrane. The theory of Hamiltonian dynamics is discussed in section 4.

All arguments to `hd` are optional. The first argument can be the name of an input file; this is the standard usage. Subsequent arguments override simple input file directives:

```
> hd run.inp nsteps=10                                          (C1)
```

2.1 Dynamics options and parameters

The choice of timestep is critical for computing proper ensembles. If the timestep is too large, the system will be unstable and program execution will halt. The option `timestep_analysis=yes` provides a rough estimate of a proper timescale for dynamics.

2.1.1 Loading a simulation state

Dynamics and minimization can be restarted by loading a save file. The input syntax is `load`:

```
> hd run.inp load=min.save                                      (C2)
```

This command can be put in the input file as well. Save files are generated at the end of minimization and dynamics. At the end of minimization, the file `min.save` is generated. At the end of a dynamics simulation, the file `jobName.save` is created, where `jobName` is the overall job name given in the input file. The default for this name is `default`, so the default for the save file is `default.save`.

3 Setting the bilayer lipid composition

The bilayer lipid composition is set in the input file; unlike other input commands it cannot be overridden with command-line options. Lipid composition commands begin with `lipid` and then are followed by sub-commands. For example, to set the inner leaflet to be DOPC, include

```
lipid inner DOPC 100                                                    (I 1)
```

in the input file. The leaflet is selected with either `inner` or `outer`. The fourth argument is the amount of lipid, in parts. To an outer leaflet with 50% DOPC and 50% DOPE, include, for example

```
lipid inner DOPC 100
lipid inner DOPE 100                                                    (I 2)
```

or

```
lipid inner DOPC 50
lipid inner DOPE 50                                                     (I 3)
```

where with the parts mechanism the total amount of lipid always sums to 100%.

To add a new lipid to the library, use the sub-command `library`:

```
lipid library SAPC 80.0 -0.01                                           (I 4)
```

where `SAPC` is the name of the lipid, `80.0` is the area-per-lipid, and `-0.01` is the spontaneous curvature. This lipid can now be used in a compositional command:

```
lipid inner POPC 70
lipid inner SAPC 30                                                     (I 5)
```

The input file is parsed iteratively, beginning with `library` sub-commands, so the order in the input file is not significant.

4 Hamiltonian and Lagrangian dynamics

4.1 Lagrangian mechanics

The Lagrangian L is defined as:

$$L = T - V \tag{1}$$

with V the potential energy.[1] The kinetic energy

$$T = \frac{1}{2}\sum m_k \nu_k^2 \tag{2}$$

is defined in terms of the real mass variables (ν_k and r_k). It is computed for the generalized coordinates q as:

$$\boldsymbol{\nu}_k = \sum_j \frac{\partial \boldsymbol{r}_k}{\partial q_j}\dot{q}_j + \frac{\partial \boldsymbol{r}_k}{\partial t} \tag{3}$$

Here $\boldsymbol{\nu}_k$ is the velocity of particle k and m_k is its mass. In the case of the membrane itself the generalized coordinate is the control point. The mass variables are sections of the membrane corresponding to points on the surface. The quantities $\frac{\partial \boldsymbol{r}_k}{\partial q_j}$ are determined by the spline coefficients of the face, and each point is a linear combination of the (for regular faces) twelve control points. The quantity $\frac{\partial L}{\partial q_j}$ corresponds to the force, $\frac{\partial V}{\partial x}$. The differential equation of motion is

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_j}\right) = \frac{\partial L}{\partial q_j} \tag{4}$$

[1] This section is adapted from the English language Wikipedia, beginning around 2019.

Applied to the control points, the equation of motion is

$$M_{ij}\ddot{q}_j = \frac{\partial V}{\partial q_i} \tag{5}$$

where

$$M_{ij} = \sum_k m_k \frac{\partial \boldsymbol{r}_k}{\partial q_j} \cdot \frac{\partial \boldsymbol{r}_k}{\partial q_i} \tag{6}$$

This matrix is inverted to put the equations in form suitable for integration:

$$\ddot{q}_j = M_{ij}^{-1}\frac{\partial V}{\partial q_i} \tag{7}$$

4.2 Hamiltonian dynamics

In Hamiltonian dynamics a new variable is introduced, the momentum p:

$$p_i = \frac{\partial L}{\partial \dot{q}_i} \tag{8}$$

This equation is then solved for $\dot{q}_i$ and substituted into the Hamiltonian H:

$$H = \sum \dot{q}_i p_i - L \tag{9}$$

For the control points p is to related to the set q through:

$$p_i = \sum_j M_{ij}\dot{q}_j \tag{10}$$

$$\dot{q}_j = \sum_i M_{ij}^{-1} p_i \tag{11}$$

The equations of motion are then:

$$\dot{p} = -\frac{\partial H}{\partial q} \tag{12}$$

$$\dot{q} = +\frac{\partial H}{\partial p} \tag{13}$$

4.3 Particle on the membrane surface

The coordinates of a particle on the membrane's surface are determined by their internal coordinates u and v as well as the control points of the surrounding mesh:

$$\boldsymbol{r}_i = \sum_l \boldsymbol{q}_l s_l u^{a_l} v^{b_l} \tag{14}$$

where here the generalized variable q is shown in vector form. The powers a_l and b_l are non-negative integers less than five, and s_l is the spline coefficient. To calculate T the velocity $\boldsymbol{\nu}$ must be expressed in terms of the generalized coordinates and their time derivatives:

$$\boldsymbol{\nu} = \sum_l \frac{\partial \boldsymbol{r}}{\partial q_l}\dot{q}_l + \frac{\partial \boldsymbol{r}}{\partial u}\dot{u} + \frac{\partial \boldsymbol{r}}{\partial v}\dot{v} \tag{15}$$

The particle's attachment modifies the dynamics of the membrane as its mass adds to that of the membrane. The control point momenta are now determined as:

$$p_i = M_{ij}\dot{q}_j + m_p \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \frac{\partial \boldsymbol{r}}{\partial q_k} \dot{q}_k + m_p \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \frac{\partial \boldsymbol{r}}{\partial u} \dot{u} + m_p \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \frac{\partial \boldsymbol{r}}{\partial v} \dot{v} \tag{16}$$

where the sums for the attached particle are only over control points that are included in the spline.

The attached particle's velocity depends on $\dot{q}_i$, $\dot{u}$ and $\dot{v}$, so all cross-terms in T must be accounted for in the calculation of, for example, $\frac{\partial T}{\partial \dot{u}}$.

$$p_u = m_p [\frac{\partial \boldsymbol{r}}{\partial q_k} \cdot \boldsymbol{r}_u \dot{q}_k + \boldsymbol{r}_u \cdot \boldsymbol{r}_u \dot{u} + \boldsymbol{r}_u \cdot \boldsymbol{r}_v \dot{v}] \tag{17}$$

and

$$p_v = m_[[\frac{\partial \boldsymbol{r}}{\partial q_k} \cdot \boldsymbol{r}_v \dot{q}_k + \boldsymbol{r}_v \cdot \boldsymbol{r}_u \dot{u} + \boldsymbol{r}_v \cdot \boldsymbol{r}_v \dot{v}] \tag{18}$$

The routines to evaluate these terms are `surface::ru` ($\boldsymbol{r}_u$), `surface::rv` ($\boldsymbol{r}_v$), and `surface::get_pt_coeffs` ($\frac{\partial \boldsymbol{r}}{\partial q_k}$), available in `uv_map.C`.

In theory, the $\dot{q}_i$, $\dot{u}$, and $\dot{v}$ are then substituted into Eq. 9 to determine the new H. Setting aside briefly the momentum of the underlying mesh, $\dot{\boldsymbol{q}}$, the surface coordinates of the particle momentum can be cast as:

$$\begin{bmatrix} p_u \\ p_v \end{bmatrix} = m_p \sum_{kl} \begin{bmatrix} \boldsymbol{r}_u \cdot \boldsymbol{r}_u & \boldsymbol{r}_u \cdot \boldsymbol{r}_v \\ \boldsymbol{r}_v \cdot \boldsymbol{r}_u & \boldsymbol{r}_v \cdot \boldsymbol{r}_v \end{bmatrix} \cdot \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} \tag{19}$$

or

$$p_\alpha = m_p g_{\alpha\beta} \dot{\beta} \tag{20}$$

where α, β are u or v. Conceptually, the factors act to slow down $\dot{u}$ as it approaches regions with high metrics. It does not do this with a force, rather, $\dot{u}$ is computed instantaneously from p, which is the variable being propagated.

Unlike g_{uv}, the control point mass matrix M_{ij} is determined only by network topology, if the masses represented by the mesh are not chosen to change. Thus, in the absence of bound particles, its inverse does not need to be calculated at each dynamics step to evaluate $\dot{\boldsymbol{q}}$ from $\boldsymbol{p}$ using Eq. 11. Eqs. 16, 17, and 18 introduce state-dependent ($\boldsymbol{q}$, u, and v) coupling between $\dot{\boldsymbol{q}}$ and both the mesh and embedded particles.

To solve for the complete set of coordinate time derivatives, including both control point momenta and particle momenta, with the eventual goal of their elimination requires solving:

$$\begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{p}_j \\ \vdots \\ p_u \\ p_v \end{bmatrix} = \left(\begin{bmatrix} M_{ii} & M_{ij} & \cdots & 0 & 0 \\ M_{ji} & M_{jj} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & & m_p g_{uu} & m_p g_{uv} \\ 0 & 0 & & m_p g_{vu} & m_p g_{vv} \end{bmatrix} \right.$$

$$\left. + m_p \begin{bmatrix} 0 & \cdots & \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \boldsymbol{r}_u & \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \boldsymbol{r}_v \\ \vdots & \ddots & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \boldsymbol{r}_u & \cdots & 0 & 0 \\ \frac{\partial \boldsymbol{r}}{\partial q_i} \cdot \boldsymbol{r}_v & \cdots & 0 & 0 \end{bmatrix} \right) \cdot \begin{bmatrix} \dot{\boldsymbol{q}}_i \\ \dot{\boldsymbol{q}}_j \\ \vdots \\ \dot{u} \\ \dot{v} \end{bmatrix}. \tag{21}$$

The inverse of a slightly perturbed matrix $A_0 + \delta A$ can be computed with the truncated series

$$(A_0 + \delta A)^{-1} = A_0^{-1} - A_0^{-1} \delta A A_0^{-1} + \mathcal{O}[\delta A^2] \tag{22}$$

because

$$(A_0^{-1} - A_0^{-1}\delta A A_0^{-1}) \cdot (A_0 + \delta A) = I - A_0^{-1}\delta A A_0^{-1}\delta A \tag{23}$$

The kinetic energy T now has position-dependent cross-terms between the particle and membrane that contribute to the time derivative of the momenta.

4.4 Propagating particle momenta p

Particle momenta p are propagated using the force, e.g., $\frac{\partial H}{\partial u}$. The kinetic energy now depends on u through $\frac{\partial T}{\partial u}$ and $\frac{\partial T}{\partial v}$.

4.5 The effective mass matrix is banded; its inverse is short-ranged

4.6 Normal modes

The mesh control points are "generalized" coordinates in the sense that they are not point centers of mass that directly feel force. Rather, they determine the mass positions through the subdivision-limit algorithm. Compared with the complexity of having each underlying lipid or its atoms independent entities, the use of a continuum mesh is a dramatic simplification. It is often convenient to add a further layer of generalization by using so-called normal modes that are linear transformations of the mesh:

$$\boldsymbol{Q}_i = N_{ij}\boldsymbol{q}_j, \tag{24}$$

where $\boldsymbol{Q}$ is a normal mode and $\boldsymbol{q}$ is a mesh control point. Most frequently these modes are energetically decoupled at the level of the membrane elastic energy. Creation of the normal modes requires solving for N_{ij}, again considering the membrane elastic energy.

The use of normal modes for spheres and planes is invoked by specifying either the details of a single mode (`mode_x` and `mode_y`) or the lower and upper limits of modes to model (`mode_min` and `mode_max`). For a spherical mesh, it is necessary to include the input `sphere yes` to invoke spherical harmonics.

4.7 Langevin thermostatting

The generic Langevin equation[2] is

$$\dot{A}_i = \sum_j \{A_i, A_j\}\frac{\mathrm{d}H}{\mathrm{d}A_j} - \lambda_i \frac{\mathrm{d}H}{\mathrm{d}A_i} + \eta_i(t) \tag{25}$$

where here $\{\cdot,\cdot\}$ is the Poisson bracket, λ_i is the friction coefficient for degree-of-freedom A_i, and $\eta(t)$ is the random noise process with

$$\langle \eta_i(t)\eta_i(t')\rangle = 2\lambda_i\delta(t-t') \tag{26}$$

and zero mean. The units of the damping coefficients λ_i are $\frac{\mathrm{mass}}{\mathrm{time}}$. For a simple Hamiltonian like $\frac{p^2}{2m}$ the friction can be applied directly to the momentum trivially, i.e.:

$$\dot{p} = \frac{\mathrm{d}H}{\mathrm{d}q} - \lambda_p \frac{\mathrm{d}H}{\mathrm{d}p} + \eta_p(t) \tag{27}$$

$$\dot{p} = -\lambda_p \frac{p}{m} + \eta_p(t) \tag{28}$$

However, for a more complicated kinetic energy the quantity $\dot{q}$ is a convenient intermediate, see Eq. 11, with $\dot{q}$, itself computed from p, taking the place of $\frac{p}{m}$.

[2] The form of this equation was adapted from the English language Wikipedia page for the Langevin Equation on March 27th, 2019.

5 Membrane and particle dynamics

5.1 Local metric

The local metric of the membrane relates changes in a particle's surface coordinates u and v to changes in its three dimensional Cartesian vector.

5.2 Irregular mesh points

An irregular mesh point has valence not equal to six. At an irregular mesh point the metric and some related properties (curvature) diverge. Note that the tangent plane does not. Nevertheless this is a challenge for dynamics because critical quantities ($\dot{u}$, $\dot{v}$, $c(u, v)$) are changing rapidly with time. The solution currently adopted is to split timesteps as necessary, on a particle-by-particle basis, as they approach irregular vertices. Not all quantities are recomputed during the time splitting, only those that do not require interaction with other particles.

6 Timescales

6.1 Langevin dynamics

Newton's equation of motion is:

$$m\dot{\boldsymbol{v}} = \boldsymbol{f} \tag{29}$$

where $\boldsymbol{v}$ and $\boldsymbol{f}$ are time dependent quantities. With **Langevin dynamics**, a frictional drag and stochastic force are introduced. This is frequently justified as arising from implied collisions with solvent, itself in thermal equilibrium with a bath. The modified equations are

$$m\dot{\boldsymbol{v}} = -\frac{\boldsymbol{v}}{B} + \boldsymbol{f} + \tilde{\boldsymbol{f}} \tag{30}$$

or often with a friction coefficient γ replacing the mobility B:

$$m\dot{\boldsymbol{v}} = -\gamma\boldsymbol{v} + \boldsymbol{f} + \tilde{\boldsymbol{f}} \tag{31}$$

where $\tilde{\boldsymbol{f}}$ is the stochastic force.

Consider modeling a particle diffusing with Langevin dynamics. In the absence of any forces, $\boldsymbol{f}$, the diffusion constant is

$$D = Bk_BT \tag{32}$$

To simulate a lipidic diffusion constant, e.g., $10^{-6}\mathrm{cm}^2/\mathrm{s}$, requires specifying B appropriately. Note that in Eq. 31 the velocity is reduced by a fraction equal to $\frac{-\Delta t}{mB}$, where Δt is the simulation timestep. This fraction must be much less than one or the particle will experience uncontrolled feedback and therefore improper integration.

6.2 Brownian dynamics

Under constant force, the velocity obeys

$$\dot{v} = -\gamma v + f/m \tag{33}$$

The solution to this simple differential equation is

$$v(t) = v_0 \exp(-\gamma t) + \frac{f}{\gamma m} \tag{34}$$

That is, the velocity decays with characteristic timescale $\tau = \frac{1}{\gamma}$. If timesteps on the order of (or larger than) τ are desired, *it makes no sense to attempt to propagate the velocity*. During the integration period the velocity will decorrelate completely from its initial value. Rather, consider an approximate dynamics that is displacement-based, that is, that attempts to model a particle's motion under thermal agitation in the presence of external forces. This is **Brownian dynamics**. Here the equation of motion is

$$\dot{\boldsymbol{r}} = -\frac{D}{k_B T}\boldsymbol{f} + \sqrt{2D}R(t) \tag{35}$$

where $R(t)$ is a so-called "Gaussian process", a mathematical object equivalent to selecting *uncorrelated* random numbers from a Gaussian distribution with zero mean and standard deviation one.

6.3 Membrane relaxation timescales

7 Pressure/tension control

Changes in the periodic boundary dimension are controlled using a vector of parameters $\boldsymbol{\alpha}$ that scales the coordinates from the original PBCs:

$$\boldsymbol{r}' = \{r_x\alpha_x, r_y\alpha_y, r_z\alpha_z\} \tag{36}$$

Naturally the periodic cell dimensions are changed by the same values of α.

Changing the value of α_x changes the shape of the underlying mesh, as well as the positions of all attached particles. Not only is the potential energy changed, but the kinetic energy as well.

The kinetic energy T is computed as

$$T = \frac{1}{2}\sum_i p_i \dot{q}_i \tag{37}$$

where p_i and q_i are the conjugate momentum and coordinate of degree-of-freedom i, which might be a single Cartesian (e.g. x) or surface coordinate (e.g. u). The quantity $\dot{q}_i$ is computed as $\hat{M}^{-1} \cdot \boldsymbol{p}$. The matrix $\hat{M}$ is the same for each Cartesian dimension. It is computed initially with $\boldsymbol{\alpha} = \{1, 1, 1\}$, scales as α^2, as it is the outer product of $\frac{\partial \boldsymbol{r}}{\partial q_i}$ with itself. Changing $\boldsymbol{\alpha}$ thus changes T for the system and so T must be considered when computing the probability p of a Monte Carlo move attempt in α:

$$p = \exp -\beta(V_{\text{new}} - V_{\text{old}} + T_{\text{new}} - T_{\text{old}}) \tag{38}$$

where β is the inverse temperature, V is the potential energy, and subscripts label the quantities before and after the attempted move.

8 Hydrodynamics with stochastic rotational dynamics (SRD)

The stochastic rotational dynamics (SRD) method is activated with input command `do_srd=yes`.

8.1 SRD particle/mesh collisions

Collisions between the SRD particle and the mesh are modeled at the lengthscale of the mesh spacing. On a *very* short timescale, a collision would change the momentum of only the individual mass point at the collision, after which the dynamics would be propagated outward, exciting various modes and eventually pushing the entire mass in the direction of the momentum exchange. In this implementation, the collision changes the momentum of only the nearest control point site. This assumption is consistent with the overall coarse-graining scheme in which very high frequency modes are not modeled. More accurate schemes could be imagined in which the trajectory following the collision more closely approximates the collision of a higher resolution mesh. However, since the SRD solvent particle and its collision are themselves coarse-grained representations of finely detailed solvents, it may not be worth the effort to more closely represent what is itself not truly physical.

The dynamics of a collision between two elastic particles conserves two quantities expressed as equations below: In Eq. 39, the momentum directed along the axis normal to the collision interface, and in Eq. 40, the overall kinetic energy:

$$\Delta \boldsymbol{p}_{\mathrm{SRD}} + \Delta \boldsymbol{p}_i = 0 \tag{39}$$

$$\frac{|\boldsymbol{p}_{\mathrm{SRD}} + \Delta \boldsymbol{p}_{\mathrm{SRD}}|^2}{2m_{\mathrm{SRD}}} + \frac{1}{2}(\boldsymbol{p}_i + \Delta \boldsymbol{p}_i) \cdot \sum_j M_{ij}^{-1}(\boldsymbol{p}_j + \Delta \boldsymbol{p}_j \delta_{ij}) = \frac{|\boldsymbol{p}_{\mathrm{SRD}}|^2}{2m_{\mathrm{SRD}}} + \frac{1}{2}\boldsymbol{p}_i \cdot \sum_j M_{ij}^{-1} \boldsymbol{p}_j = 0 \tag{40}$$

Here $\boldsymbol{p}_{\mathrm{SRD}}$ is the SRD particle momentum, $\boldsymbol{p}_i$ is the momentum of the collision vertex, m_{SRD} is the mass of the SRD particle, and M_{ij}^{-1} is the mesh effective mass inverse matrix. There are two solutions to this constraint: a trivial solution in which the particles pass through each other, and the desired solution in which finite momentum is transferred between the particles. The equation is solved by choosing scalar parameters α_{SRD} and α_i with $\Delta \boldsymbol{p}_{\mathrm{SRD}} = \alpha_{\mathrm{SRD}} \boldsymbol{n}_{\mathrm{collision}}$ and $\Delta \boldsymbol{p}_i = \alpha_{\mathrm{i}} \boldsymbol{n}_{\mathrm{collision}}$, where $\boldsymbol{n}_{\mathrm{collision}}$ is the collision axis. The equations reduce to:

$$\alpha_{\mathrm{SRD}} + \alpha_i = 0 \tag{41}$$

$$2\alpha_{\mathrm{SRD}} m_{\mathrm{SRD}}^{-1}(\boldsymbol{p}_{\mathrm{SRD}} \cdot \boldsymbol{n}_{\mathrm{collision}}) + \alpha_{\mathrm{SRD}}^2 m_{\mathrm{SRD}}^{-1} + 2\alpha_i \sum_j M_{ij}^{-1}(\boldsymbol{p}_j \cdot \boldsymbol{n}_{\mathrm{collision}}) + \alpha_i^2 M_{ii}^{-1} = 0 \tag{42}$$

Substituting in $\alpha_{\mathrm{SRD}} = -\alpha_i$ yields for Eq. 42:

$$2\alpha_i\Big(\sum_j M_{ij}^{-1} \boldsymbol{p}_j - m_{\mathrm{SRD}}^{-1} \boldsymbol{p}_{\mathrm{SRD}}\Big) \cdot \boldsymbol{n}_{\mathrm{collision}} + \alpha_i^2(m_{\mathrm{SRD}}^{-1} + M_{ii}^{-1}) = 0$$

$$\alpha_i = 2\frac{(m_{\mathrm{SRD}}^{-1} \boldsymbol{p}_{\mathrm{SRD}} - \sum_j M_{ij}^{-1} \boldsymbol{p}_j) \cdot \boldsymbol{n}_{\mathrm{collision}}}{m_{\mathrm{SRD}}^{-1} + M_{ii}^{-1}} \tag{43}$$

9 Bibliography

References

[1] Feng Feng and William S. Klug. Finite element modeling of lipid bilayer membranes. *Journal of Computational Physics*, 220(1):394–408, 2006.