



SimGen: A General Simulation Method for Large Systems

William R. Taylor

Francis Crick Institute, 1 Midland Road, London NW1 1AT, UK

Correspondence to William R. Taylor: william.taylor@crick.ac.uk

<http://dx.doi.org/10.1016/j.jmb.2016.10.011>

Edited by Michael Sternberg

Abstract

SimGen is a stand-alone computer program that reads a script of commands to represent complex macromolecules, including proteins and nucleic acids, in a structural hierarchy that can then be viewed using an integral graphical viewer or animated through a high-level application programming interface in C++.

Structural levels in the hierarchy range from α -carbon or phosphate backbones through secondary structure to domains, molecules, and multimers with each level represented in an identical data structure that can be manipulated using the application programming interface. Unlike most coarse-grained simulation approaches, the higher-level objects represented in SimGen can be soft, allowing the lower-level objects that they contain to interact directly.

The default motion simulated by SimGen is a Brownian-like diffusion that can be set to occur across all levels of representation in the hierarchy. Links can also be defined between objects, which, when combined with large high-level random movements, result in an effective search strategy for constraint satisfaction, including structure prediction from predicted pairwise distances.

The implementation of SimGen makes use of the hierarchic data structure to avoid unnecessary calculation, especially for collision detection, allowing it to be simultaneously run and viewed on a laptop computer while simulating large systems of over 20,000 objects. It has been used previously to model complex molecular interactions including the motion of a myosin-V dimer “walking” on an actin fibre, RNA stem-loop packing, and the simulation of cell motion and aggregation. Several extensions to this original functionality are described.

© 2016 The Francis Crick Institute. Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Introduction

Macromolecular modelling is increasingly being directed towards ever larger molecules and assemblies of molecules, including both nucleic acids and proteins—often in a lipid environment. This trend is driven by a rich harvest of structures, including individual components solved not only by X-ray diffraction and NMR methods but also by new high-resolution electron microscopy. Most importantly, modelling methods are needed to allow this structural information to be extended as widely as possible to interpret the even larger flood of sequence data that has been generated from new rapid sequencing technologies.

Given these data, the range of modelling activity is wide—including the packing of known component protein structures into a complex under the restraint

of predicted or experimental pairwise interactions or into a low-resolution envelope provided by electron microscopy or X-ray (see Ref. [1] for a wide-ranging survey). Having obtained a satisfactory model for a complex, the task is often to envisage or predict how it might function, for example, in a molecular motor. Reaching this stage using conventional modelling methods, such as docking and molecular dynamics, can be very time consuming for the investigator and also in the required computer time.

Over a period of many years, methods have been developed to try to reduce both the human and computational burden of constructing and simulating large molecular systems. The most widely adopted approach has been to combine components into groups that then move as a rigid body. This coarse-grained (CG) principle can be applied at any level: a few atoms can be treated as one, a whole amino acid

side-chain or nucleotide base can be reduced to a sphere, an alpha-helix or RNA stem loop can be treated as a tube and so on, up to treating a whole domain or protein as a sphere or ellipsoid (for reviews, see Refs. [2–5]). Ultimately, the greatest saving can be made by treating the whole system as a single sphere, but it should be clear that this results in an unacceptable loss of detail and that any simulation at this level is unlikely to provide biological insight. This tension between resolution and tractability exists to varying degrees across the full range of CG modelling and in an attempt to circumvent it, multi-scale (or hybrid) modelling methods have been developed in which, as the name suggests, multiple levels of representation can co-exist [6–11]. The SimGen method described here is much simpler than these methods yet incorporates a very general way to include up to several multiple levels of representation without loss of detail at the lowest level.

Overview

SimGen was originally developed just to represent and construct protein molecules as a hierarchy of different structural levels in which each object in the hierarchy can be placed, copied, moved, and linked to construct a complex assembly of one or multiple chains. From this, it was extended to add interaction between objects based on an efficient collision detection algorithm [12,13]. However, it is not a molecular dynamics program and contains no potentials, forces, or momentum, and its default mode of operation is to do nothing. However, provision is made for a user-supplied routine, called the “driver”, through which every element in the structural hierarchy can be animated using a simple set of geometric operations.

The lowest level object in the hierarchy is a sphere that is usually equated with the α -carbon of the protein chain or nucleotide phosphate. An entry in a user-provided table specifies its properties, such as radius, bond length, and how far two atoms will be pushed apart if their spheres overlap. An equivalent set of properties can be specified for each higher level but includes also the option to be a sphere, ellipsoid, or tube. A parameter is also provided that specifies how much a low-level object (child) should be moved inwards if it lies outside the envelope of its higher-level object (parent). A full description of the various options and parameters can be found in the user manual included in the Supplementary Material.

The simplest use of the program would be to represent an all- α -type protein as a string of tubes using their α -carbon coordinate positions from a known PDB structure. As stated above, if the α -carbon radius and the α -helix tube radius are set so that nothing bumps and if all α -carbon atoms are inside their assigned tube, then the model will remain static. However, a default level of random motion can

also be specified in the parameter table, which allows the model to explore its local conformational space as a constrained random walk. This motion requires some connection between the levels as it would not make sense to have a parent object wander independently of the children it contains. To do this, the parent is continually moved to the centroid of its children, and conversely, any move to the parent is applied to all its children, and recursively, all their children in turn.

The only other feature of the program that is controlled through a command script (without any need for user programming) is to specify additional links and bonds. This is particularly useful to maintain the local conformation of secondary structure elements, and as it would be tedious to have to specify every virtual hydrogen-bond in a structure, the program automatically assigns virtual bonds between relative α -carbon positions i to $i+3$ and i to $i+4$ in an α -helix, although the non-local bonds that bind a β -sheet must be specified individually (with the help of a utility program). As links can also be specified between any pair of objects at the same level, a set of restraints can be created, and with the molecule allowed to shake, motions will be directed towards conformations that satisfy the restraints.

In this mode of operation, SimGen sounds much like any other constraint satisfaction program. However, the way in which it treats collisions (bumps) between objects is more unique. At the lowest atomic level, this is simply a hard shell repulsion, but at each higher level, two repulsion parameters can be specified, called *hard* and *soft*. When *soft* is zero, objects are repelled using a step size specified by *hard*, but when *hard* is zero, objects are repelled by a step size that is a function of the value of *soft* and the number of children that are in collision between the two parent objects. In other words, objects can either be hard shells or insubstantial containers, or if both parameters are given a value, any combination between—giving the object a “jelly”-like consistency. This is an important feature of SimGen as it means that however crude the CG objects are, the details of their contents can still be allowed to interact.

Another unique aspect of SimGen that is worth describing briefly is the way in which the different behaviours are implemented. Because there are no potentials or momentum, different actions can be calculated asynchronously, allowing each to be implemented by an independent parallel process (or thread). These have names like *bumper*, *shaker*, *linker*, and *keeper*, all of which can easily be associated with the actions outlined above. Full details of these routines can be found in the user manual included in the Supplementary Material and also in Ref. [14]. Finally, there is also the *viewer* that provides real-time visualisation and the *driver* that must be written by the user if a more purposeful behaviour is required rather than just random (Brownian) movement.

Some details of the method and the way in which it can be animated using a `driver` application are best illustrated by a series of examples, which will also introduce something of the flavour of the command script used to set up complex models. More details of the algorithms with a particular focus on the implementation of collision detection can be found in Ref. [14].

Applications

Actin: a multidomain globular protein

Previously, *SimGen* has been used to simulate the “walking” motion of myosin-V on an actin fibre [15,16]. As an example of a globular protein hierarchic organisation, the actin molecule from this study will be described to give a flavour of the command language.

Levels of structure follow the classic hierarchic organisation of a globular protein as: residue (α -carbon sphere), secondary structure (tube segment), domain (ellipsoid), protein, and multimer. For all bar the residue level, each object in a level is introduced by the command word “GROUP” which can simply be inserted into a standard PDB file. The type of secondary structure is specified by a number 1, 2 for α -helix, β -strands, and 0 for anything else. A tube of an appropriate diameter and length is automatically assigned depending on the structure type, but this can be overridden by additional values on the GROUP command line. Each GROUPed segment at the next higher (domain) level must contain all the α -carbon atoms that belong to the domain whether they are contiguous in sequence or not. If this level has been defined as ellipsoidal, the numbers on the GROUP command can now be used to specify eccentricity and the major axis direction. Both the secondary structure definitions and the domain ellipsoid parameters can be calculated using separate programs based on previous methods [17–19]. (See Fig. 1)

Additional details of this application with a fuller description of the command files can be found in Ref. [20], and the user manual is included in the Supplementary Material.

Rhodopsin: a transmembrane protein

Rhodopsin consists of seven transmembrane (TM) helices arranged in a simple up/down bundle. Each α -helix was modelled as a tube, just as described above for the globular protein, but the seven tubes were then contained in a larger tube rather than an ellipsoid. This tube had a diameter narrow enough to confine the helices in a compact packing arrangement in the plane of the membrane and long enough to allow the helices to shift to a reasonable extent up and down relative to the membrane. Because the ends of the helical tubes are not constrained to lie within their

containing tube, they are still free to tilt relative to each other as is commonly observed in such structures (see Supplementary Material Fig. 4a).

This construct for a TM protein was used with both rhodopsin and a variety of proteins of unknown structure in combination with distance restraints derived from an analysis of correlated mutations [21,22]. Starting from a number of configurations obtained from combinatorial enumeration over a hexagonal lattice [23], the helices were free to move in a random way within the limits of the constraints imposed by the model, and the predicted links between residues were continually refined towards their ideal separation. However, links that remained too long were broken to prevent the disruption of the structure caused by enforcing a combination of mutually inconsistent distances. The resulting models were then ranked on how well they had satisfied the given restraints. The best resulting models were comparable to those obtained using FILM3 or Rosetta [22].

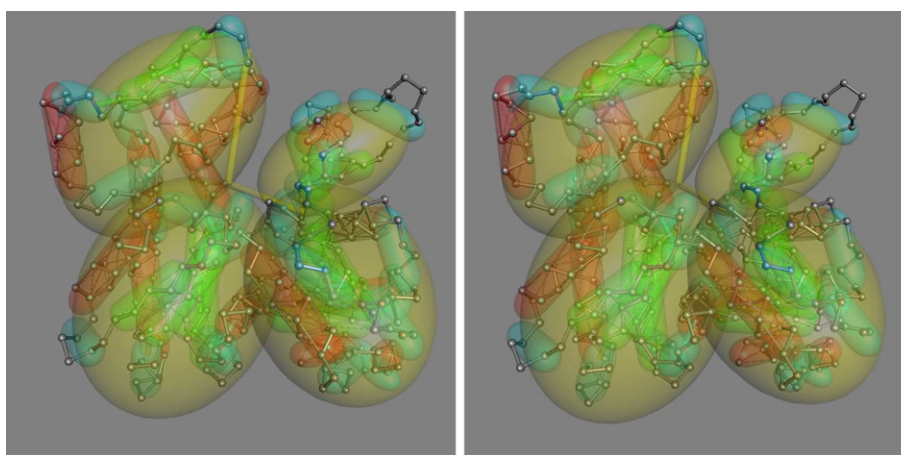
F₀-ATPase: animation of a Brownian ratchet

The applications described so far have been either inert (actin) or moving under random perturbations (rhodopsin) and neither have made use of any user code in the `driver` routine. In this example, a model is constructed for the mitochondrial F₀-ATPase, which is animated using the `driver` to simulate a Brownian ratchet.

The F₀-ATPase consists of a ring of TM helices (C-ring) that rotates past a static component (A-subunit), the structure of which has recently been solved (PDB code: 5ara) [24]. By embedding GROUP commands, the PDB structure can be divided into these components, each of which can then be manipulated directly in the `driver` routine.

The model setup in Fig. 2 consists of three components, two of which correspond directly to PDB structures and the third is a large oblate ellipsoid (flying saucer) that represents the membrane. This setup script also shows how different models (shapes, sizes, bond lengths, etc.) can be associated with different groups (although all must contain the same number of levels). Neglecting local variable declarations, the `driver` is a one-line routine that makes the C-ring rotate by random steps around the axis of the tube object that contains the C-ring. (See Fig. 2)

With only a little extra work, small unbonded spheres (representing protons) can be added to the membrane model. These would normally be confined inside their parent container, but this can be reversed in the `driver` routine to expel them instead. Then, by extending the code in the `driver` routine, they can be given the ability to form dynamic links to polarisable residues in the protein components, which give them immunity from expulsion from the membrane.



```

PARAM actin4.model
END
GROUP 0 1
INPUT actin.dat
-----<actin.dat>

GROUP 0 4
INPUT domain1.dat
INPUT domain2.dat
INPUT domain3.dat
INPUT domain4.dat
-----<domain1.dat>

GROUP -13 19      -17.68 -1.60 -10.54      17.68 1.60 10.54
:
GROUP 0 2
ATOM      31  CA  GLY  a   31      -4.602 -5.499   9.908  1.33  1.00
ATOM      32  CA  GLY  a   32      -0.786 -5.387  10.266  1.33  1.00
GROUP 1 13      5.00 -17.10  5.84    -9.44 -16.15  8.44
ATOM      33  CA  GLY  a   33       7.109 -17.815   2.883  1.33  3.00
ATOM      34  CA  GLY  a   34       6.691 -19.585   6.304  1.33  3.00
:

```

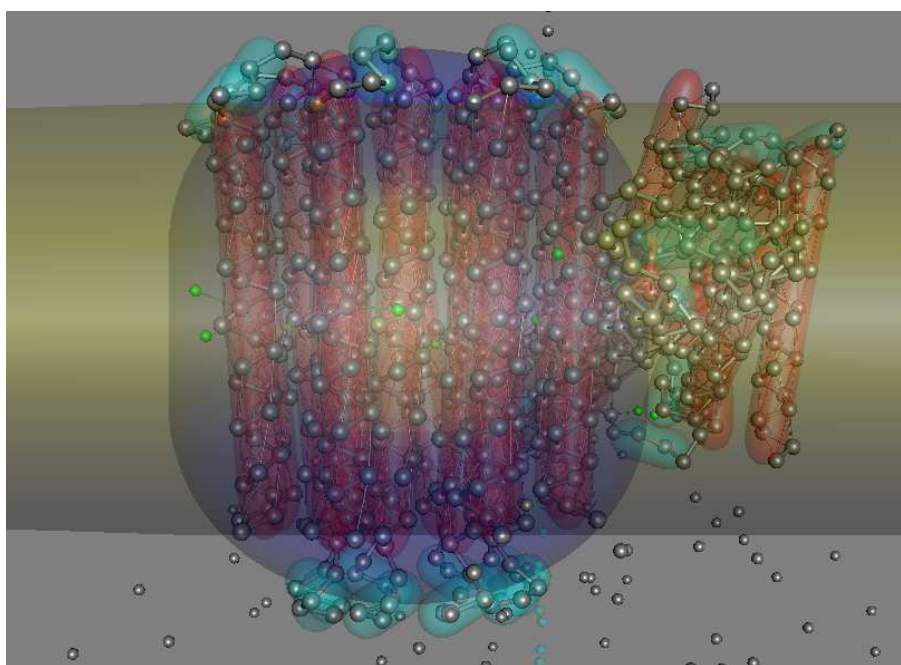
Fig. 1. Actin model. The α -carbon backbone trace for actin (PDB code: 1olg-F) is rendered (in stereo) as a ball-and-stick chain (silver) with virtual hydrogen bonds shown as thin rods. Secondary structure elements are shown as transparent tubes (with hemispherical end-caps), coloured red (α), green (β), and cyan (loops). The four domains are represented by yellow ellipsoids. The script files that created the model are shown in the boxed section, in which the domains are held in separate files that contain the PDB α -carbon records partitioned by `GROUP` commands. Separate files are divided by a dashed line with the file name in angle brackets to the right.

Adding the final condition that any C-ring rotation that takes an unprotonated residue into the membrane is forbidden produces a mechanism that can pick up protons from one side of the membrane, pass them around the C-ring, and release them on the other. If there are equal numbers of “protons” on either side of the membrane, the C-ring moves randomly both clockwise and anticlockwise, but if there is an excess on one side, then the mechanism behaves as a Brownian ratchet giving a bias towards a preferred rotation (the code for this driver routine

and a movie of the animation can be found in the Supplementary Material).

RNA: “folding” with restraints

To illustrate the application of the method to nucleic acid molecules, a complex RNA molecule was set-up to move randomly under a set of imposed pairwise restraints between phosphate atoms (similar to the TM protein model described above). In principle, the hierarchy of structural levels for nucleic



```

PARAM memb.model
PARAM Aunit.model
PARAM Cring.model
END
GROUP 0 3
MODEL 1
INPUT Aunit.dat
MODEL 2
INPUT Cring.dat
MODEL 0
INPUT memb.dat
-----<driver.cpp>

void driver () {
Cell *C = Cell::world->child[1]; // C-ring tube (with axis endN--endC)
    C->spin(C->endN,C->endC,0.2*randf()-0.1);
}

```

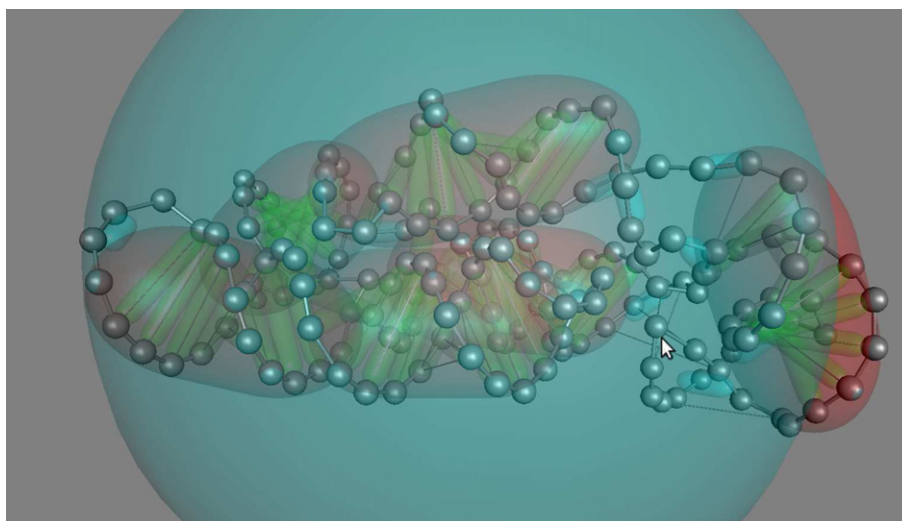
Fig. 2. F_0 -ATPase model. The two major membrane components of the mitochondrial ATPase, the A-subunit and C-ring, were set up as separate objects with the latter contained in a tube (dark blue) that can spin on its axis. Both have been located in a large oblate ellipsoid (yellow), part of which is viewed edge-on. Below this is a “swarm” of small spheres representing protons. Protons can attach to selected residues (becoming green spheres on sticks) that allow them to enter the “membrane”. The higher number of protons on one side of the membrane imposes a bias on the direction of the C-ring rotation. In the box, the top-level script for the model is shown that uses three separate parameter models for each component, and below this, a simple `driver` routine is shown that gives the random motion to the C-ring.

acids is similar to that of proteins, including secondary structures and domains, and could be constructed “by hand” using standard commands. However, nucleic acid secondary structures are double stranded, and when represented only by phosphate atoms, it is difficult to keep them in an ideal conformation using only a scaffold of links. To help maintain an ideal geometry, including chirality, the input data can include

commands to declare an RNA or DNA molecule and use a special variant of the `GROUP` command (`DOUBL`) to identify basepaired segments. (Fig. 3).

Cell adhesion and aggregation

The generality of the approach can be seen in an application to simulate cell motion and adhesion on a



```

GROUP -1 2 141.9 -2.1 -43.0 149.4 -4.2 -27.6
DOUBL 5 6
  ATOM 30 P G A 30 136.959 -7.287 -41.285 1.00 0.00
  ATOM 31 P G A 31 142.220 -9.415 -42.119 1.00 0.00
  ATOM 32 P G A 32 148.078 -10.613 -40.765 1.00 0.00
  ATOM 33 P G A 33 152.597 -10.425 -36.226 1.00 0.00
  ATOM 34 P G A 34 155.449 -10.135 -31.422 1.00 0.00
  ATOM 35 P G A 35 155.355 -10.471 -26.179 1.00 0.00
GROUP 0 4
  ATOM 36 P G A 36 152.451 -12.197 -21.671 1.00 0.00
  ATOM 37 P G A 37 146.573 -12.194 -22.567 1.00 0.00
  ATOM 38 P G A 38 141.805 -7.652 -22.765 1.00 0.00
  ATOM 39 P G A 39 140.016 -2.335 -23.535 1.00 0.00
DOUBL -5 6
  ATOM 40 P G A 40 145.310 1.614 -25.175 1.00 0.00
  ATOM 41 P G A 41 149.538 3.940 -27.476 1.00 0.00
  ATOM 42 P G A 42 152.212 4.738 -32.595 1.00 0.00
  ATOM 43 P G A 43 151.582 4.494 -38.901 1.00 0.00
  ATOM 44 P G A 44 149.744 4.292 -44.223 1.00 0.00
  ATOM 45 P G A 45 145.074 3.708 -48.596 1.00 0.00

```

Fig. 3. RNA model. The structure of the lysine riboswitch (Rfam code: RF00168) has been set up as a model in which base pairs are represented as green tubes connecting pairs of phosphate atoms (silver spheres). Base pair ladders are then contained in larger red tubes, and the whole molecule is confined in a large sphere. The arrow points to a group of predicted links that form a pseudoknot that has not been defined as a secondary structure. The boxed text shows a fragment of a PDB file containing added commands to define a stem loop. The `DOUBL` command functions as a `GROUP` command but instructs the program to look for a matched segment (identified by the 5,-5 pair) to which it will be basepaired, forming a double-stranded secondary structure.

two-dimensional surface [25]. Cell bodies were modelled as a sphere with a soft (jelly-like) repulsion behaviour that contains a circular ring of hard spheres (referred to as beads rather than atoms to avoid confusion) that can move freely inside and slightly beyond the cell body, consistent with their bond length and bumping constraints (see Supplementary Material Fig. 5).

When any two objects collide (whether cells or molecules), the event is registered in the data structure

of each object and can be used in the `driver` routine to activate a user-defined behaviour, such as a change in speed or direction or the creation of a link between the objects. Different combinations of these behaviours in populations of cells can lead to unexpected “emergent” behaviours that mimic those seen in real cell populations [26] (see Supplementary Material Fig. 6).

In this application, the construction of the model is very simple, but the `driver` routine is quite extensive,

encoding a variety of behaviours that depend on who is linked to whom and how long different links have been maintained. The model remains intrinsically 3D, but the `driver` also contains a line of code to reduce every Z-coordinate by 1% every cycle. This gentle pull of “gravity” still allows cells to sometimes ride over each other or form a dense “puppy-pile” when they have strong mutual adhesion.

Implementation

The `SimGen` program is a stand-alone program written in C++ containing its own internal graphics rendering using open-GL (contained in the `viewer` routine). It is designed to be simultaneously run and viewed on any moderate laptop, but the graphics interface can be “switched-off” for multiple runs, say on a computer cluster. On a laptop, systems containing 20,000–30,000 objects lie around the limit of what can be simulated, which is equivalent to approximately 20 nucleosomes or a myosin-V dimer on a short fibre of 26 actin dimers (modelled as described above). Beyond this, real-time motion becomes “jumpy” with disjointed rendering.

The data structure that holds all the information about an object is the same whatever the level of the object in the hierarchy. This means that the program is scale-free, and every object can be manipulated by the same set of operations. Operations on the hierarchic data structure of grouped objects are processed recursively with the structure of the hierarchy being used to avoid unnecessary computation. For example, if two objects are not bumping, then no checks are made to see if their children are bumping.

Different types of operations, such as moving, bumping, bonding, etc., are performed independently and in parallel with each routine operating on a common data structure. This means that a bump will be repelled even if it results in a bond length being violated, and random moves will always be made no matter what “damage” they do. The situation is similar to a Monte Carlo simulation, in which every move is accepted rather than some being discarded if they lead to restraint violations. However, most operations maintain a list of violations ranked by severity and will try to rectify the worst first, leading quickly to the return of an acceptable conformation.

This relatively unconstrained behaviour can be very useful when exploring a wide conformational space, as a barrier can simply be overcome while the routine that enforces it is momentarily “distracted”. However, it does not mix well with any formulation that uses a potential that requires global evaluation. Whatever time point is chosen to evaluate the potential, there will always be some unresolved conflict that will result in transiently excessive energies. Rather than “freezing” the simulation while these are resolved, an alternative strategy is to have yet another parallel independent

routine (or many) that can evaluate conformations “on the fly” and dynamically return corrections to the simulation. From the molecular dynamics viewpoint, this can also be seen as using `SimGen` to enhance conformational sampling and would lend itself well to a Genetic Algorithm implementation [27].

Conclusions

`SimGen` is primarily intended as a rapid prototyping method in which large macromolecular systems can easily be constructed using a script of commands to place, repeat, bond, and link molecules in a hierarchic structure. The default animation of the resulting system with random Brownian motion can often be all that is required to generate useful behaviour, especially when combined with additional restraints. However, the full potential of the system is only realised with the addition of application-specific code to introduce “purposeful” behaviour of the type that is often seen in molecular motors. For this, the homogeneous data structure that holds the hierarchy of objects provides an intuitive coding interface based on family relationships and high-level geometric operations. To date, the system has only been used to simulate natural molecular complexes and motors, but in the future, it may find a use in the design of artificial systems such as nanomachines.

Availability

The program, including all source code and scripts, is freely available and can be downloaded[†]. This site also contains the user manual and the examples used in this work.

Acknowledgements

Enrico Spiga is thanked for comments on the manuscript. This work was supported by the Francis Crick Institute (001179) which receives its core funding from Cancer Research UK, the UK Medical Research Council, and the Wellcome Trust.

Appendix A. Supplementary Data

Supplementary data to this article can be found online at [doi:10.1016/j.jmb.2016.10.011](https://doi.org/10.1016/j.jmb.2016.10.011).

Received 20 June 2016;

Received in revised form 28 September 2016;

Accepted 6 October 2016

Available online 19 October 2016

Keywords:

coarse-grained molecular modelling;
macromolecular simulation;
constraint satisfaction

†<https://github.com/WillieTaylor/simgen>

Abbreviations used:

CG, coarse-grained; TM, transmembrane.

References

- [1] A. Sali, H.M. Berman, T. Schwede, J. Trehwella, G. Kleywegt, S.K. Burley, J. Markley, H. Nakamura, P. Adams, A.M. Bonvin, W. Chiu, M.D. Peraro, F.D. Maio, T.E. Ferrin, K. Grunewald, A. Gutmanas, R. Henderson, G. Hummer, K. Iwasaki, G. Johnson, C.L. Lawson, J. Meiler, M.A. Marti-Renom, G.T. Montelione, M. Nilges, R. Nussinov, A. Patwardhan, J. Rappsilber, R.J. Read, H. Saibil, G.F. Schroder, C.D. Schwieters, C.A. Seidel, D. Svergun, M. Topf, E.L. Ulrich, S. Velankar, J.D. Westbrook, Outcome of the first wwPDB hybrid/integrative methods task force workshop, *Structure* 23 (2014) 1156–1167.
- [2] V. Tozzini, Coarse-grained models for proteins, *Curr. Opin. Struct. Biol.* 15 (2005) 144–150.
- [3] S. Riniker, J. Allison, W. van Gunsteren, On developing coarse-grained models for biomolecular simulation: a review, *Phys. Chem. Chem. Phys.* 14 (2012) 12,423–12,430, <http://dx.doi.org/10.1039/C2CP40934H>.
- [4] H. Ingolfsson, C. Lopez, J. Uusitalo, D. de Jong, S. Gopal, X. Periole, S. Marrink, The power of coarse graining in biomolecular simulations, *WIREs Comput. Mol. Sci.* 4 (2014) 225–248, <http://dx.doi.org/10.1002/wcms.1169>.
- [5] E. Spiga, M. Degiacomi, M. Peraro, New strategies for integrative dynamic modeling of macromolecular assembly, In: *Biomolecular Modelling and Simulation* (T. Karabencheva-Christova, ed.), Burlington: Academic Press, 2014, pp. 77–111, ISBN: 978-0-12-800013-7.
- [6] S. Izvekov, G.A. Voth, A multiscale coarse-graining method for biomolecular systems, *J. Phys. Chem. Lett.* 109 (2005) 2469–2473.
- [7] P. Minary, M. Levitt, Conformational optimization with natural degrees of freedom: a novel stochastic chain closure algorithm, *J. Comp. Biol.* 17 (2010) 993–1010.
- [8] A. Sim, M. Levitt, P. Minary, Modeling and design by hierarchical natural moves, *Proc. Natl. Acad. Sci. U. S. A.* 109 (2012) 2890–2895.
- [9] K. Meier, A. Choutko, J. Dolenc, A. Eichenberger, S. Riniker, W. van Gunsteren, Multi-resolution simulation of biomolecular systems: a review of methodological issues, *Angew. Chem. Int. Ed.* (2013) 2820–2834, <http://dx.doi.org/10.1002/anie.201205408>.
- [10] J. Dama, A. Sinitskiy, M. McCullagh, J. Weare, B. Roux, A. Dinner, G. Voth, The theory of ultra-coarse-graining. 1. General principles, *J. Chem. Theory Comput.* 9 (2013) 2466–2480, <http://dx.doi.org/10.1021/ct4000444>.
- [11] J. Grime, G. Voth, Highly scalable and memory efficient ultra-coarse-grained molecular dynamics simulations, *J. Chem. Theory Comput.* 10 (2014) 423–431, <http://dx.doi.org/10.1021/ct400727q>.
- [12] W.R. Taylor, Z. Katsimitsoulia, A soft collision detection algorithm for simple Brownian dynamics, *Comput. Biol. Chem.* 34 (2010) 1–10, <http://dx.doi.org/10.1016/j.compbiolchem.2009.11.003>.
- [13] Z. Katsimitsoulia, W.R. Taylor, A hierarchic algorithm for simple Brownian dynamics, *Comput. Biol. Chem.* 34 (2010) 71–79, <http://dx.doi.org/10.1016/j.compbiolchem.2010.01.001>.
- [14] W.R. Taylor, Steric exclusion and constraint satisfaction in multi-scale coarse-grained simulations, *Comput. Biol. Chem.* 64 (2016) 297–312, <http://dx.doi.org/10.1016/j.compbiolchem.2016.06.007>.
- [15] W.R. Taylor, Z. Katsimitsoulia, A coarse-grained model for actin myosin simulation, *J. Mol. Graph. Model.* 29 (2010) 266–279.
- [16] Z. Katsimitsoulia, W.R. Taylor, Coarse-grained simulation of myosin-V movement, *Comput. Math. Methods Med.* 2012 (2012) e781456, <http://dx.doi.org/10.1155/2012/781456>.
- [17] W.R. Taylor, Defining linear segments in protein structure, *J. Mol. Biol.* 310 (2001) 1135–1150.
- [18] W.R. Taylor, Protein structure domain identification, *Protein Eng.* 12 (1999) 203–216.
- [19] W.R. Taylor, J.M. Thornton, W.G. Turnell, A ellipsoidal approximation of protein shape, *J. Mol. Graph.* 1 (1983) 30–38.
- [20] W. R. Taylor, Z. Katsimitsoulia, Generalised multi-level coarse-grained molecular simulation and its application to myosin-v movement, In: *Innovations in Biomolecular Modeling and Simulations*, RSC Biomolecular Sciences, (T. Schlick, ed.), vol. 23–24, Royal Society of Chemistry (Great Britain), 2012, ISSN 1757-7152.
- [21] W.R. Taylor, An algorithm to parse segment packing in predicted protein contact maps, *Algorithms Mol. Biol.* 11 (2016) 17, <http://dx.doi.org/10.1186/s13015-016-0080-x>.
- [22] W.R. Taylor, T. Matthews-Palmer, M. Beeby, Molecular models for the core components of the flagellar type-III secretion complex, *PLoS One* (2016) <http://dx.doi.org/10.1371/journal.pone.0164047>.
- [23] W.R. Taylor, D.T. Jones, N.M. Green, A method for α -helical integral membrane protein fold prediction, *Proteins Struct. Funct. Genet.* 18 (1994) 281–294.
- [24] A. Zhou, A. Rohou, D. Schep, J. Bason, M. Montgomery, J. Walker, N. Grigorieff, J. Rubinstein, Structure and conformational states of the bovine mitochondrial ATP synthase by cryo-em, *Elife* 4 (2015) e10180, <http://dx.doi.org/10.7554/eLife.10180>.
- [25] W.R. Taylor, Z. Katsimitsoulia, A. Poliakov, Simulation of cell movement and interaction, *J. Bioinforma. Comput. Biol.* 9 (2011) 91–110.
- [26] W. Taylor, R. Morley, A. Krasavin, L. Gregory, D. Wilkinson, A. Poliakov, A mechanical model of cell segregation driven by differential adhesion, *PLoS One* 7 (2012), e43226.
- [27] K. Petersen, W.R. Taylor, Modelling zinc-binding proteins with GADGET: genetic algorithm and distance geometry for exploring topology, *J. Mol. Biol.* 325 (2003) 1039–1059.