

A language for molecular dynamics

David Fincham

SERC Daresbury Laboratory and Department of Physics, Keele University

Email: D.Fincham @ UK.AC.KEELE

Introduction

Most programs for molecular dynamics developed in the academic environment, such as those in the CCP5 library, are comparatively simple, since they are generally written to simulate a particular system or a restricted class of systems, e.g. molecular liquids, ionic solids. Setting up a simulation job is usually a matter of editing an input file which specifies such things as the parameters of the interaction potential, and control variables such as number of steps in the run and frequency of r.d.f. calculation.

Commercially-produced software capable of performing simulations has been available for some time in the bio-molecular field and is becoming available in the materials modelling area. Such programs are more complex as they need to be able to handle almost any system in which the user may be interested, and typically they are driven by an interactive front end running on a workstation. The program is controlled by menu selection and graphical model building. Such sophisticated software can be very useful in some academic research, but is unsuitable for the researcher who wants to extend the use of simulation into new areas or develop new techniques, since the source code is necessarily complex and hard to modify, and in any event not normally accessible by users.

A third method of "driving" a simulation program, intermediate between the "parameter file" and "graphical user interface" is the use of a command language. The purpose of this article is to describe the command language Dynamo on which I now base all my simulation programs, and to commend this approach as being extremely suitable for the development of programs in the academic research environment. The language interpreter is written in Fortran and forms part of my programs Moliq-Dynamo (molecular liquids) and Shell-Dynamo (ionic materials with shell-model option) which are being included in the CCP5 program library.

Advantages

The use of a command language has numerous advantages over the "parameter file" approach for academically-developed programs.

(a) *Ease of development.* Programs are normally developed in stages: typically one gets the basic algorithm going for a simple system; then it is extended to more complex systems; and then various options and analysis features are added. If a parameter file is being used, more and more parameters need to be added as the program evolves, so the

format of the file is continually changing. Old files then cease to work, and to recreate the original, simple, test runs may involve extensive editing of what is, by now, quite a complex file. Using a command language one develops the program by adding new commands; but the old command files still work, and can be rerun to test that no errors have been introduced. The use of a command language encourages a modular and incremental approach to program development which minimises errors and speeds progress.

(b) *Ease and flexibility of use.* Editing a parameter file to set up a particular simulation can be a tedious and error-prone task, particularly if it involves remembering the values and purposes of a large number of control switches. Use of a command language is much easier since commands can have meaningful names and only the commands and parameters actually required need to be specified. More importantly, the use of a command language can give great flexibility in the simulation protocol adopted. Most simulation programs controlled by parameter files have only one option: perform N steps of equilibration followed by M steps of measurement. Using the Dynamo language the user has complete freedom to change algorithms or parameters at any stage of the simulation, switch on or off any analysis, restart averaging or print selected outputs. Within one execution of the program it is quite possible to simulate several state points, change a mixture composition, or even simulate completely different systems.

An illustration

The general form of a command in Dynamo is very simple: it consists of a command name, followed by none or more sub-commands, followed by none or more string parameters, followed by none or more numerical parameters. Commands and sub-commands may be abbreviated to four characters, and are case-insensitive.

I will illustrate Dynamo by describing a file of commands which repeats one of the early simulations of Verlet, using Moliq-Dynamo (see Figure 1).

The command MOLECULE introduces the definition of a molecule type and gives it a name. The molecule definition is terminated by the ENDMOLECULE command. In this case it is necessary to specify only one interaction site using the LJSITE command. It is possible to have multiple interaction sites, massless sites and masses without interactions. The BOUNDARY PSC command specifies the boundary as periodic-simple-cubic: this is actually the default and this command could have been omitted, but other options are available. The NUMBER command is used to specify the number of each type of molecule included in the simulation. The command STATE is used to specify the desired state point, here with subcommands TEMPER (required temperature in K) and MOLARV (molar volume in cm^3). Other subcommands enable a density rather than volume to be specified, and in addition a pressure for controlled pressure simulations.

The VALUE command is used to specify various numerical parameters: here the timestep and cutoffs for the interaction and the neighbour list. All quantities have defaults: for example the default value for the interaction cutoff is half the box length.

FILLBOX sets up the initial lattice, and SETVELS sets the initial velocities, according to the required temperature. NEIGHBOUR APLIST sets the neighbour search algorithm, in this case an all-pairs neighbour list; the usual options are available such as LINK (link-cells) and LCLIST (neighbour list created by link-cell search).

The command OUTPUT THERMO gives an interval in steps between output of thermodynamic quantities, and like many commands it controls what happens during subsequent RUNMD commands. RUNMD is the only command which actually performs a simulation, and has a sub-command which specifies the type of dynamics to be used. EQUIL (equilibration dynamics) involves temperature windowing in combination with a Berendsen heat-bath. Other options include LEAP (standard energy-conserving leapfrog) and TPBATH (Berendsen temperature and pressure control) among others.

After equilibration in the example a restart file (coordinates and velocities) is written using the command SAVE RESTART. The converse operation is LOAD RESTART. If SAVE CONTINUE and LOAD CONTINUE are used instead the file contains in addition all accumulators so that a run can be continued exactly from the point at which it was terminated.

The command ZERO is particularly important in Dynamo. Any type of measurement may be used in conjunction with any type of dynamics, and thermodynamic quantities are always calculated and incrementally averaged. ZERO resets to zero all the accumulators, of correlation functions as well as of thermodynamic quantities, and so restarts all averaging and analysis. The commands RDF, ACF and MSD switch on radial distribution function, time auto-correlation function and mean square displacement analyses for the next RUNMD call. They have a few sub-commands and numerical parameters which it is not necessary to describe here. OUTPUT CORREL will print out all non-zero correlation functions.

Other features

The Dynamo language has too many features to be described fully here, and there are necessarily differences between Moliq-Dynamo and Shell-Dynamo because of the different forms for the interactions. A few points are worth noting.

Commands are read from standard input so the Dynamo programs can be run interactively, which is useful during testing. A command OBEY enables a file of commands to be read and obeyed. It is useful to establish a library of such files describing various model systems. There is also a looping feature in the language. An ECHO command enables the output file to be annotated freely. Numerical parameters are read using Fortran list-directed reads. This enables built-in facilities for such reads to be used: for example, values may be omitted and will then take their default values.

Conclusion

I hope this article has given a flavour of the Dynamo language and convinced you that it can make life easier for the simulator. The structure of the Dynamo programs is very modular and well-documented. If you are planning to develop a new simulation program one of them might be a good starting point. I would be happy to assist with advice. Let me know how you get on!

Figure

Command file for an argon simulation

```
MOLECULE Argon
!      x y z sig eps q m
LJSITE Arg 0.0 0.0 0.0 0.3405 119.8 0.0 40
ENDMOLECULE
BOUNDARY PSC
NUMBER Argon 864
! State point corresponds to T*=1.036 and rho*=0.65
STATE TEMPER 124.11
STATE MOLARV 36.58
VALUE DT 0.025
VALUE CUTOFF 0.86
VALUE OUTERC 0.96
FILLBOX FCC
SETVELS
NEIGHBOUR APLIST
OUTPUT THERMO 50
! Equilibration
RUNMD EQUIL 250
FIXCM
SAVE RESTART argon.864
! restart averaging: switch on correlation functions
! then run for 10 ps using leapfrog dynamics
ZERO
RDF LIMITS 0.2 1.0 0.02
RDF CC 20
ACF VELOCS 1 50
MSD 4 50
RUNMD LEAP 400
OUTPUT CORREL
```