NAME

md2dII − a 2D Lennard Jone particle simulator program

SYNOPSIS

md2dII [**none**] [**<infile**] [**>outfile]**

EXAMPLE

md2dII < md2d.data > md2d.data.outfile

CONTACT PERSON

Kun-Chun Lee (klee@chem.ucla.edu)

DESCRIPTION AND HISTORY

md2dII is a simple molecular dynamics simulator program. It is designed to study 2D Lennard Jone particle system. It does not carry any sort analysis internally, which means analysis work needs to be done with independent softwares. The user is encourage to modify the code as necessary to add in the analysis routines into th main code. However, be warned, modifying the program may result in unpredictable results, and, therefor, it is the responsibility of the end user to check that everything is working properly.

The program itself has been checked for small systems. And I have tried to eliminate as many bugs as possible. If you found anything suspicously wrong, you need to contact the author as soon as possible.

A few general comments concerning the system

1. *The system is 2D.* The shape is not necessarily square, but it is periodic. The the output coordinate system is wrapped, but the internal coordinate system is not wrapped. This will come in handy if one wants to use this system to study 2D diffusion.

2. *The default system is in canonical (constant T, V, N) ensemble.* The heat bath is Andersen, although the code for velocity rescaling method is also included, but not activated because the velocity rescaling method seems to be quite inefficient at heating the system. Microcanonical ensemble can be setup by setting collision frequency in andersen thermostat to 0 (constant E,V,N).

3. *The system is assumed to composed of only Lennard-Jone particles* interacting with a cutoff potential at the minimum. The particle can have variable size and mass. Other forces may be added in the future as needed.

COMPILING MD2DII

Type command ’make’ at the shell prompt and everything should automatically recompile. By default, I have assumed that your system has both g77 and gcc, which are the GNU’s compilers. I have tested the program on *silicon* and the *O2’s* in the computer room. They have the same configuration anyway. If you do decided to try to compile the program at home on your computer, you should try first changing the names of the compilers that are stored in the *makefile* to your respective compilers. If this does not work, please contact me.

The executable file is called *runme.* You should move the file to a ’bin’ directory or a directory on your $PATH so that it can be accessed. Also remember to change the name to something easy for you.

INPUT FILE

The way *md2dII* works is that it takes in an input fi le from standard input and outputs the results to standard output. The input fi le contains sequence of command for setting up the system which will be discussed later. The output fi le is designed such that it can be used for another subsequent run without modifi -cation (ie. it has the same format as an input fi le). In theory, you should be able to pipe the output from one run directly to another run without actually writing anything to a fi le, but, for some reason, it does not work this way. I am quite puzzled myself. If anyone comes up with a solution, I would appreciate it.

OUTPUT FILE

As mentioned above, the output fi le has the same format as the input fi le. It contains addition information which are prefaced with the comment character for *md2dII*, which is '#'. There are 3 float numbers that follows the comment character in the output fi le. The fi rst one, is the time, in LJ unit. Second one is potential energy. Third is the kinetic energy. All these informations are instaneous information. There are a few quantities that the user should try to add to the program to be calculated, like pressure. We will learn how to do this in a couple of weeks.

COMMANDS

All commands can be prefi xed with one or more '@' and it will still work. '@' is used to tell the program, how many time the same run has been reinitialized. However there are a few exceptions, so do not add in extra '@' by yourself, just let the program do it for you.

The calling sequence for most commands is 'command number number . . .'. The spacing is not important, but a command can only occupy one line, which has a maximum of 80 characters, so be conservative. If you supplied wrong number of arguments the parser will detect any warn you, but it would not terminate the program unless the command is not recognized. Empty spaces are not processed.

I will use square bracket to indicate the argument type, followed by a number to indicate how many of this type. For example, *command [string]2 [number]4* means command 'command' takes 2 strings and 4 numbers. I will also use more specifi c names for the arguments when appropriate.

A few commands can be shorten to just a single letter. However this is not recommended practice for setting up the script fi le. So I will not discuss it.

This character tells the program that everything that follows should be ignored. This is equivalent to a comment line in fortran. The user should use this feature to annotate their input fi le as a note keeping process.

@! This command tells the compiler to switch over to script mode, which activates the scanner and the parser. It should be located on the fi rst line of your input fi le. The other mode, which I will call direct mode, is deprecated. If you are having trouble using the script mode, you should contact me, so that you can turn on the direct mode. However, in direct mode, you are only allow to set up the position of the atoms and nothing else; the only way to change other parameters with direct mode is to go into the code, modify, then re-compile With script mode, you can change a few parameters that may be of relevant.

seed [integer]

This command tells *md2dII* to set up another random number seed instead of the default one.

box [number]2

This command takes two argument. The fi rst argument is used to indicated the length of the box in the x direction and the second is in the y direction. Both are in units of LJ.

time [starting time] [time step] [stopping time] [output interval]

starting time, time step, and stopping time are used to tell the program that the simulation should run from time = *starting time,* which increment every cycle with *time step,* until *stopping time.* The *output interval* is optionally. It is used to indicate that the program should update the output file every *output interval* cycles. [For every output interval thermo-data (see Addition Info) is printed to standard output and, for every 8th output interval, the configurational data is also printed.]

temperature [number]

Used to set up the desired temperature.

andersen [integer]

Used to specify the collision frequency in Andersen thermostat.

atom [size] [mass] [x position] [y position]

This command is used to *add* an atom into the system. You must specify all the arguments. Be warned, you need to make sure the particles do not overlap one another. The program will automatic terminate if the overlap is significant.

iatom [atom number] [size] [mass] [x position] [y position] [x velocity] [y velocity] [x force] [y force]

This command is used to *modify* an atom that has already been added into the system (force is recalculated). The atom being modified is indicated by *atom number*

run Terminates the input process and begin the simulator.

quit Terminate the program. Note that this means that there's no output.

ADDITIONAL INFORMATION

The first few numbers following the symbol '#' are (thermo-data)

[energy/particle] [potential energy] [kinetic energy] [virial]

KNOWN BUGS

Yet to be determined. Check back later.

COPYRIGHT