CHAPTER 2 - MANY-PARTICLE MOTION

The classical motion of particles in a many-body system is well-described by alternative formulations of mechanics developed by Lagrange and Hamilton, both of which incorporate the mechanics developed by Newton in the late 1600s. Our purpose here is to review a few brief elements of the formalisms, and then discuss their implementation in computer simulations. The fundamental descriptions of particle motion in Lagrange's approach are (generalized) positions and velocities - a useful approach when part of a particle's interactions with its surroundings is velocity-dependent, as with drag forces. In Hamilton's approach, the motion is described in terms of positions and momenta, useful when the interaction is momentum-dependent.

2.1 Lagrangian formalism

In the formalism developed by Lagrange, particles are characterized by a set of generalized coordinates and their time derivatives, conventionally represented by the symbols q_i and $\dot{q}_i$. In the Cartesian coordinate system, positions $\boldsymbol{x}$ and velocities $\boldsymbol{v}$ could be used for the generalized coordinates and their derivatives. The time evolution of q_i and $\dot{q}_i$ is governed by the Lagrangian function $L(q_i , \dot{q}_i)$, which is related to the kinetic energy K and potential energy V via

$$L = K - V. \tag{2.1}$$

In Newtonian mechanics, the potential energy is used to generate a set of forces and accelerations, which can be integrated to give velocities and hence the time evolution of positions. In Lagrangian mechanics, the fundamental equation of motion is

$$\frac{d}{dt}(\partial L / \partial \dot{q}_i) - (\partial L / \partial q_i) = 0 \tag{2.2}$$

One can see how Eq. (2.2) contains the elements of Newtonian mechanics by considering a Cartesian system:

•the first term involves the time derivative of $\dot{q}_i$, and hence generates an acceleration

•the second term involves a derivative of a potential energy with respect to a position, which corresponds to a force in the Newtonian approach.

Hence, Eq. (2.2) generates the same relation between acceleration and the derivative of the potential energy as does Newton's law, but without the explicit appearance of a force.

2.2 Hamiltonian formalism

In Hamilton's approach, the motion of particles is described by a set of generalized coordinates q_i and their corresponding momenta p_i. The time evolution of q_i and p_i are then given by two first-order equations

$$\dot{q}_i = \partial H / \partial p_i$$
$$\dot{p}_i = - \partial H / \partial q_i. \qquad (2.3)$$

The function H that appears in Eq. (2.3) is the Hamiltonian of the system, and is equal to the sum of the kinetic and potential energies

$$H = K + V. \qquad (2.4)$$

The Hamiltonian is a function of q_i and p_i.

2.3 Computer-friendly potentials

The equations of motion in both the Hamiltonian and Lagrangian formalisms contain derivatives of a potential energy. Although the derivatives are evaluated analytically before insertion into a computer simulation, they should be chosen so that they execute rapidly. A few common choices are given below.

Square well

The square well potential is used extensively in our studies of polymers. An attractive square well has a step function form

$$\begin{aligned} V(r) &= \infty && \text{for } r < a \\ V(r) &= -V_0 && \text{for } a \le r \le \sqrt{2}\,a \\ V(r) &= 0 && \text{for } r > \sqrt{2}\,a, \end{aligned} \qquad (2.5)$$

where r is the interparticle distance and a is a parameter. The square well interaction is excellent for Monte Carlo simulations (see Chap. 4), but deserves special attention if used in dynamics studies, as pointed out in Sec. 2.3.

Lennard-Jones potential

Although originally used for studies of fluids, this potential has widespread application. It has the functional form

$$V(r) = 4\varepsilon[(\rho / r)^{12} - (\rho / r)^{6}] \qquad (2.6)$$

where the parameters ε and ρ set the energy and length scales of the potential, respectively. The r^{-12}-term is repulsive and dominates at small values of ρ/r, providing a short-range steric repulsion between particles. However, the r^{-6}-term becomes increasingly important at modest ρ / r, and results in an attractive force for $r > 2^{1/6}\rho$. Both terms decrease with increasing r, so that the potential ultimately vanishes at large r.

It is often convenient to break up the potential into separate attractive and repulsive pieces. One convention is

$$V_{rep}(r) = \begin{cases} 4\varepsilon[(\rho / r)^{12} - (\rho / r)^{6}] + \varepsilon & \text{for } r < 2^{1/6}\rho \\ 0 & \text{for } r \ge 2^{1/6}\rho \end{cases} \qquad (2.7a)$$

$$V_{att}(r) = \begin{cases} -\varepsilon & \text{for } r < 2^{1/6}\rho \\ 4\varepsilon[(\rho / r)^{12} - (\rho / r)^{6}] & \text{for } 2^{1/6}\rho \le r \end{cases} \qquad (2.7b)$$

The constant ε has been added to the individual terms of the potential so that v_{rep} always has a derivative greater than or equal to zero (*i.e.*, is never attractive), while v_{att} always has a derivative less than or equal to zero (*i.e.*, is never repulsive).

Polymer bonds

An alternative to the square well interaction for elements of a polymer chain is a continuous function developed by Bishop, Kalos and Frisch. Their approach uses the repulsive component of the Lennard-Jones interaction, Eq. (2.7a),

$$V_{\text{rep}}(r) = \begin{cases} 4\varepsilon[(\rho / r)^{12} - (\rho / r)^6] + \varepsilon & \text{for } r < 2^{1/6}\rho \\ 0 & \text{for } r \geq 2^{1/6}\rho \end{cases}$$

(normalized so as to vanish at $r = 2^{1/6}\rho$) and adds a tethering term

$$\begin{aligned} V(r) &= -0.5\, kR_o^2 \ln(1 - (r / R_o)^2) && \text{for } 0 \leq r \leq R_o \\ V(r) &= \infty && \text{for } r > R_o \end{aligned} \qquad (2.8)$$

that restricts r to be less than R_o. A common choice for the parameters of Eq. (2.8) is

$$R_o = 1.5\rho \qquad k\rho^2 / \varepsilon = 30. \qquad (2.9)$$

When differentiated, Eq. (2.8) gives a force that is an even polynomial in r, and is therefore well-suited for dynamical studies. However, the presence of the logarithm, which is slow to execute, makes Eq. (2.8) a poor choice for Monte Carlo work.

2.4 - Molecular dynamics

The integration of Newtonian and Hamiltonian equations of motion is frequently used in simulations of physical systems. In situations where the time evolution of a system is not of interest, such as many systems in equilibrium statistical mechanics, then the Metropolis Monte Carlo method is often appropriate (see Chap. 4). However, where time dependence is of interest (diffusion, for example), then it may be appropriate to solve the equations of motion and obtain the full dynamics of the system. Simulations based upon the dynamical equations of individual particles are conventionally referred to as Molecular Dynamics, or MD.

Here, we assemble many elements of simulation techniques to form the basis of the Molecular Dynamics (MD) method. Some programming issues are specific to MD, and these form the bulk of this section.

Initialization

There are several issues to be considered in initializing a system for a Molecular Dynamics simulation. There is the need to respect constraints imposed by the interaction potential between particles, or by the boundary conditions. For example, if there are hard core interactions between particles, then the initial state must not violate the minimum interparticle distances imposed by the hard cores. Similarly, if the system is confined to a box (even a periodic one), then the particles in the initial state must be within the box.

Because MD is a dynamical simulation, then initial values must be chosen for the velocities or momenta. If the physics of the system is such that the total energy of the system is conserved, then the initial system must have the desired energy. This can be achieved by several different procedures:

•Evaluate the total potential energy once the initial positions have been chosen; assign the difference between the potential and total energy to the particles' kinetic energy; distribute the kinetic energy among the particles according to a selected procedure.

•Assign velocities to particles, then rescale the velocities in an iterative manner until the system attains the desired total energy.

Interactions between particles, through collisions or smooth potentials, will drive a given initial distribution towards its equilibrium value.

The relaxation to a thermal distribution of velocities can be speeded somewhat by choosing the initial velocities from a thermal distribution. Classical Maxwell-Boltzmann particles of mass m have a one-dimensional velocity distribution of the form

$$P(v_x)dv_x = (m / 2\pi k_B T)^{1/2} \exp(-mv_x^2/2k_B T)dv_x \qquad (2.10)$$

where T is the temperature and $P(v_x)dv_x$ is the probability of finding the particle with a velocity between v_x and $v_x + dv_x$. Similar distributions apply to the y and z directions. Note that the probability distribution of Eq. (2.10) is normalized to unity. Because the distribution is a normal or Gaussian distribution, velocities can be selected from it according to the algorithm in Chap. 3.

Another issue to be concerned about in MD is conservation of angular momentum. The equations of motion respect conservation of angular momentum, so that the system will continue to carry any initial angular momentum given to it, within the accuracy of the integration procedure. How important this effect is depends upon the specific situation, but systems with free boundaries are susceptible to unanticipated side effects from angular momentum conservation. For example, if velocities are randomly assigned to vertices on a linear chain, then the chain probably

has a non-zero angular momentum. As the motion of the vertices evolves, the chain will start to spin, perhaps much more rapidly than anticipated. Hence, it is important to set the initial angular momentum to its desired value.

Propagation

The starting point for simulating most physical systems is either a Hamiltonian or Lagrangian function, depending upon the kinematic variables in the problem. Interactions which are velocity-dependent, such as drag, generally lead one towards the Lagrangian formulation, while interactions which are momentum-dependent are more appropriate to the Hamiltonian formulation. In fact, a large amount of simulation work in physics involves potential energies that depend only on spatial coordinates, for which one can go straight to Newton's equations.

Many simple systems have a potential energy that depends only upon the difference in particle positions, as in

$$V(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3 \ldots \boldsymbol{r}_N) = V(\boldsymbol{r}_{12}, \boldsymbol{r}_{13}, \boldsymbol{r}_{ij} \ldots \boldsymbol{r}_{N-1,N}) \tag{2.11}$$

where $\boldsymbol{r}_i$ is the position vector of the i^{th} particle and $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$. A further simplification that occurs for most model studies is the absence of three-body interactions, so that the potential can be written as a sum of two-body terms

$$V(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4, \ldots \boldsymbol{r}_N) = \sum_{i=2}^{N} \sum_{j=1}^{i} v_{ij}(\boldsymbol{r}_{ij}) = \sum_{i>j} v_{ij}(\boldsymbol{r}_{ij}) \tag{2.12}$$

The force $\boldsymbol{f}_i$ on particle *i* from all other particles in the system can be computed from the derivative of the total potential

$$\boldsymbol{f}_i = -dV(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4 \ldots \boldsymbol{r}_N) / d\boldsymbol{r}_i. \tag{2.13}$$

Where the potential separates into two-body terms as in Eq. (2.12), then the force $\boldsymbol{f}_i$ can be written as a sum of two-body forces $\boldsymbol{f}_{ij}$

$$\boldsymbol{f}_i(\boldsymbol{r}_i) = \sum_{j \neq i} \boldsymbol{f}_{ij}(\boldsymbol{r}_{ij}) \tag{2.14}$$

each of which can be obtained from

$$\boldsymbol{f}_{ij} = -dv_{ij}(\boldsymbol{r}_{ij}) / d\boldsymbol{r}_{ij}. \qquad (2.15)$$

The force $\boldsymbol{f}_{ij}$ is directed along $\boldsymbol{r}_{ij}$. One must be careful to keep track of the indices, since both $\boldsymbol{r}_{ij}$ and $\boldsymbol{f}_{ij}$ are *odd* in i and j.

A more convenient way of writing Eq. (2.15), which makes the direction of the force explicit, is

$$\boldsymbol{f}_{ij} = -\frac{1}{r_{ij}} \left(\frac{dv_{ij}(r_{ij})}{dr_{ij}} \right) \boldsymbol{r}_{ij} \qquad (2.16)$$

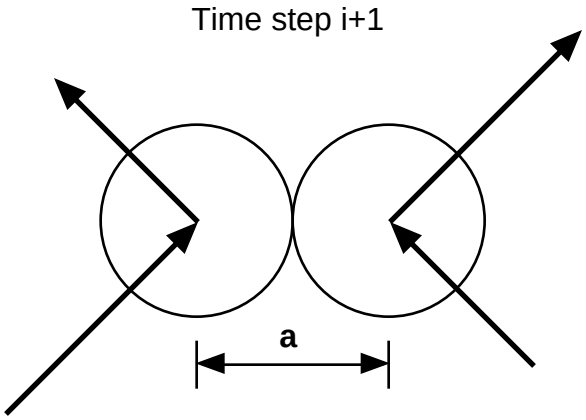
where r_{ij} is the magnitude of vector $\boldsymbol{r}_{ij}$. With some thought, functional forms can be chosen for v_{ij} such that the evaluation of the force involves only even powers of r_{ij}, and hence avoids square roots.

A number of routines for integrating ordinary differential equations are presented in Chap. 1. In general, the Euler method should be avoided as it is the least accurate. Depending on the problem, the leapfrog or Verlet algorithms may provide sufficient accuracy without excessive computing time. If high accuracy is needed, to conserve total energy of a dynamical system during many updates, for example, then more compute-intensive algorithms such as the predictor-corrector or fourth-order Runge-Kutta methods may be required.

Molecular dynamics simulations of more than 10^4 particles are commonplace, and can be handled on machines costing less than $5,000. Large-scale simulations involving 10^6 particles remain the domain of specialized machines for now. Such simulations demand that the number of calculations per time step be of order N, which is much less than the order-N^2 pairwise interactions. Means of obtaining efficient code for many-particle systems are outlined in Chap. 4. Lastly, periodic boundary conditions are used to reduce boundary effects in small to moderate-size systems, as discussed at length in Chap. 6.

Collisions

The equations of motion used in molecular dynamics simulations are most suitable for continuous differentiable functions. If the potential energies do not change rapidly with separation, then simple algorithms can be used to integrate the equations of motion. However, there are ways of handling step-function potentials.



In a strictly hard-core interaction like

$$V(r) = \infty \qquad r \le a$$
$$V(r) = 0 \qquad r > a, \tag{2.17}$$

the particles undergo uniform motion at constant speed unless they collide. The motion between collisions is certainly easy to evaluate from the MD viewpoint! The motion of a pair of particles during an elastic collision obeying the potential of Eq. (2.17) is also easy to determine. A routine that follows an elastic collision of hard spheres during an MD update is not difficult to write, and executes fairly rapidly. For dense systems, either smaller time steps must be used, or a more complex collision routine must be developed, to take into account that a given particle may undergo more than one collision during a time step.

Programming checks

Unless there are external forces or dissipative forces in the system, total energy is conserved in Molecular Dynamics. MD integration routines that are not accurate do not conserve energy on long time scales. Thus, it is important to check energy conservation during the propagation of a system, and take corrective steps to restore the energy to its initial value as needed.

The same is true of angular momentum. For systems with periodic boundaries, the effects of angular momentum conservation are not pronounced, but this is not true for isolated systems. As mentionned in Sec. 2.1, a convoluted polymer given an initial angular momentum may ultimately take on the collective shape of a spinning rod after a number of time steps if angular momentum is conserved. While the initialization procedure should set the angular momentum to zero (if desired), it should be verified that the propagation steps conserve angular momentum.

2.5 Project 2: Flight to the moon

This project is taken from Martin Siegert's *Computational Physics* course. The flight of a rocket to the Moon is a many-body problem in mechanics, and is suitable to treatment by Molecular Dynamics. We consider only the motion of the rocket and the Moon with respect to the Earth, which represents a subset of the complete many-body system.

Physical system

The interaction between the objects in the system is their mutual gravitational attraction

$$\boldsymbol{F} = -\boldsymbol{r} G m_1 m_2 / r^3, \qquad (2.18)$$

where the force has been written in a vector form to emphasize that the direction of the force is opposite to the displacement vector between the objects. The masses of the objects are m_1 and m_2, while the gravitational constant G has the value 6.67×10^{-11} Nm^2/kg^2. In this project, the gravitational effect of the Sun is neglected, as are drag and tidal forces. The masses of the Earth and Moon are $M_E = 5.98 \times 10^{24}$ kg and $M_M = 7.35 \times 10^{22}$ kg; the mass of the rocket does not enter into the problem (why?).

Simulation parameters

Fix the center of the Earth at the coordinate origin, and take the coordinates of the Moon at time $t = 0$ to be $(D \cos\alpha, D \sin\alpha)$, where $D = 3.84 \times 10^8$ m is the distance between the Earth and the Moon. Choose the coordinate system such that the Moon rotates counter-clockwise. If the Moon has a circular orbit, then its period T is given by $T^2 = 4\pi^2 D^3 / G M_E$, and its angular frequency of rotation is $\omega^2 = G M_E / D^3$. Find T and ω consistent with the given value of D.

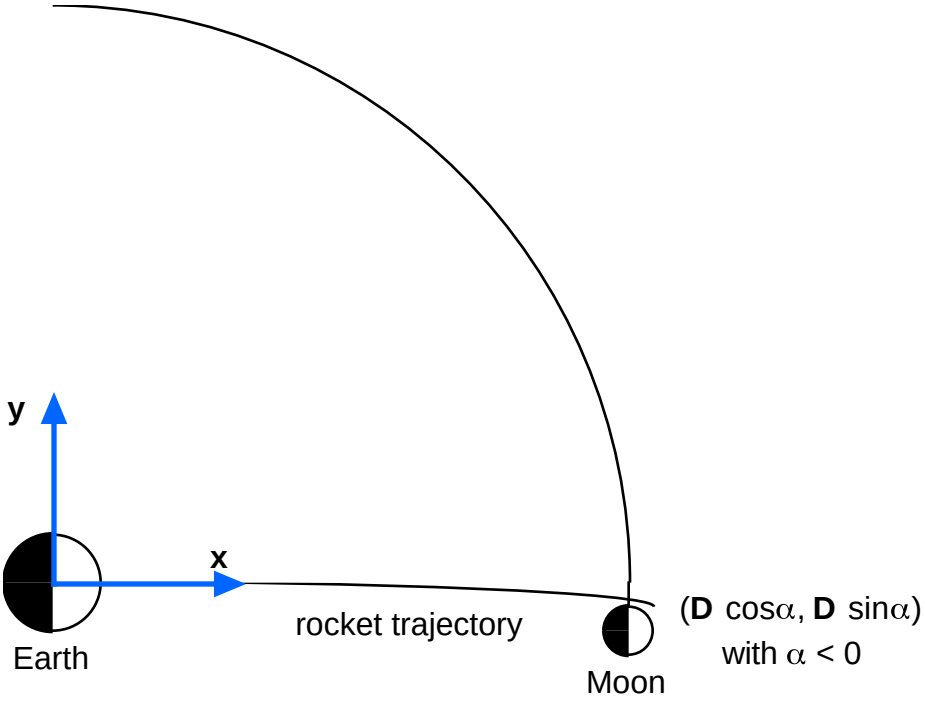


Fig. 2.1 Initial configuration of the Earth-Moon-rocket system for a negative value of the lunar angle α.

The system is illustrated in Fig. 2.1 for a negative value of α. The rocket is launched from a position (R_E , 0) on the x-axis, where R_E is the radius of the Earth = 6.38 x 10^6 m (the radius of the Moon, not drawn to scale in Fig. 2.1, is smaller at R_M = 1.74 x 10^6 m). The initial velocity of the rocket lies along the x- axis at (v_0 , 0).

The object of the simulation is to find optimal values for α and v_0 such that the rocket just passes around the Moon and returns to Earth. While it would be tempting to make v_0 large, in an effort to get to the Moon quickly, in fact the rocket would then overshoot and may not return to Earth in the lifespan of its occupants. If v_0 is too small, the rocket will not arrive at the Moon, but will be pulled back by the Earth's gravity. The minimal value of v_0 needed for the rocket to reach the Moon, neglecting the gravitational attraction of the Moon, can be found from the difference in potential energy at the surface of the Earth R_E and the orbital distance of the Moon D:

$$v_0^2 / 2 = GM_E (1/R_E - 1/D) \qquad (2.19)$$

Code

1. Start with the single-particle MD code that you wrote for Project 1. Make this into a three-body code, with forces that are, unfortunately, specific to each pair of particles because of the masses involved. Given that there are only three particles, neighbor lists are unnecessary; given that the force is long-ranged, they are inappropropriate.

2. With the Earth fixed at the coordinate origin, initialize the system as described in the section on *Simulation parameters*.

3. Assume uniform circular motion for the Moon. Evolve the motion of the rocket using one of the integration procedures from Chap. 1. Try both Verlet and leapfrog algorithms. Small step sizes will be required to follow the motion of the rocket as it passes the Moon.

4. Follow the motion of the rocket back to the Earth, and make sure that it hits the Earth (better yet, enters tangentially).

Analysis

1. Follow the trajectories for different combinations of v_0 and α.

•Fix α, then scan through a range of values of v_0 centered around Eq. (2.19). Use steps of 10 m/s in v_0.

•Choose a number of values for α between -0.6 and -1.0 radians.

2. Find the minimal value of v_0 that takes the rocket to the Moon and back, passing close to the surface of the Moon.

3. Find the time required to reach the Moon and for the return trip.

4. Change the step size to gauge the accuracy of your calculations

Report

Your report should include the following items:
•a few paragraphs stating the problem, and the algorithm used to solve it
•an outline of your code
•a table of values to illustrate the trajectory; display the distances to the Moon and the Earth, rather than just *x,y* coordinates

•a table showing the minimum distance as a function of v_0 and α; you don't need to do this in too fine steps, but 10-20 entries would be appropriate
•your "best" values for v_0 and α
•a discussion of the numerical accuracy of your code, given the number of steps required for the integration. Refer back to results from project 1. As you change the step size, by how much do your results change (show the minimum distance as a function of step size for your best value of v_0 and α)? Compare the numerical accuracy that you want (~radius of the Moon) with the distance over which you are integrating.
•a copy of your code.

Demonstration code

The demonstration code uses the Verlet algorithm to integrate the equations of motion of the rocket. The position of the Moon is advanced in a circular orbit with a period corresponding to $D = 3.84 \times 10^8$ m. As the code runs, the elapsed time in seconds is displayed in the upper left-hand corner of the screen. The rocket is drawn in red, the Moon in yellow, and the stationary Earth in green. The opening page of the demo allows you to select from three initial lunar angles α, and three initial rocket velocities [near the value of v_0 in Eq. (2.19)].

1. Choose $\alpha = 0$. As is immediately obvious, this case is unrealistic: the Moon has long since left the scene by the time that the rocket reaches its furthest distance from Earth. However, one can see that small changes in the initial velocity of the rocket result in large changes in the maximum distance from Earth that it can achieve. The values of v_0 available on the menu differ by only a few percent, but the range of the rocket changes by more than a factor of two.

2. Explore other values of α. The initial value of α must be in the range -0.6 to -0.8 radians in order for the Moon to be in the vicinity of the rocket when it attains the Moon's orbital radius. By trying several combinations of v_0 and α, you will see how sensitive the optimal trajectory is to the launch conditions.