What is Molecular Dynamics?

A molecule in Molecular Dynamics is a general term for a particle with mass, position and velocity, which is subject to forces and moves according to Newton’s equations of motion.

Molecular Dynamics (MD) is widely used to simulate many-particle systems ranging from solids, liquids, gases, and biomolecules on Earth, to the motion of stars and galaxies in the Universe. Important examples include

- Single molecules subject to external forces, like the van der Pol Oscillator

\[
\frac{d^2 x}{dt^2} - \mu (1 - x^2) \frac{dx}{dt} + x = 0
\]

- The Fermi-Pasta-Ulam Oscillator Chain
- The discretized Korteweg de Vries equation with soliton solutions studied by Zabusky and Kruskal
- Molecular dynamics of Argon atoms studied by Rahman

Newton’s equations of motion for the system are integrated numerically. If the system is in equilibrium, static properties such as temperature and pressure are measured as averages over time. Dynamical properties such as heat transport, or relaxation of systems far from equilibrium, can also be studied.
Equipartition and the Fermi-Pasta-Ulam Problem

- The MANIAC I computer was designed by N. Metropolis at Los Alamos National Laboratory and built in 1952 – some specifications
- von Neumann and Fermi suggested using it to prove the Boltzmann equipartition theorem computationally
  - A chain of linearly (harmonically) coupled oscillators cannot thermalize because the normal modes are non-interacting
  - So make the restoring forces nonlinear – interactions between normal modes should cause the system to thermalize with kinetic energy distributed equally between the modes!
- Fermi, Pasta and Ulam did the simulation and found that the system does not thermalize – it is not Ergodic
- Their paper Studies of Nonlinear Problems is a must read for computational physicists!
- Later, Zabusky and Kruskal, Phys. Rev. Lett. 15, 240 (1965) discretized the Korteweg-deVries equation in a form very similar to the Fermi-Pasta-Ulam problem with a quadratic force and discovered K-deV solitons!
Molecular Dynamics Simulation of Argon Atoms

• Consider simulating a very large number of particles or molecules, e.g., one mole \((6.02 \times 10^{23}\) molecules) of monoatomic argon in thermal equilibrium.
  
  ◦ One is certainly not interested in the positions and velocities of every one of these atoms as functions of time.
  
  ◦ One would like to study certain average properties of the sample such as its temperature \(T\), pressure \(P\), or heat capacity \(C_V\) at constant volume \(V\).
  
  ◦ If the system is in thermal equilibrium at a sufficiently high temperature, then the average kinetic energy of an argon atom is related to the temperature by the \textit{equipartition theorem}

\[
\langle K \rangle = \left\langle \frac{1}{2}mv^2 \right\rangle = \frac{3}{2}kT ,
\]

where \(k = 1.38 \times 10^{-23}\) J/K is Boltzmann's constant.

  ◦ To apply this formula, we need to generate a sample of atoms in thermal equilibrium, measure their velocities and compute \(\langle K \rangle\).

• The fundamental work on this problem was done by A. Rahman,\textit{ Phys. Rev.} \textbf{136}, A405 (1964).

• It was extended in many important ways by L. Verlet, \textit{Phys. Rev.} \textbf{159}, 98 (1967), who introduced the Verlet algorithm and the use of a neighbor list to speed up the calculation.
Solving Newton’s Equations of Motion

Newton’s equations of motion for particles are ordinary differential equations for the positions and velocities as functions of time. Solving such equations using Runge-Kutta methods was covered in Computational Physics 1 Topic 5 last semester.

The van der Pol Equation

The position $x(t)$ of the oscillator is determined by

$$\frac{d^2 x}{dt^2} - \mu(1 - x^2) \frac{dx}{dt} + x = 0$$

Specifying initial conditions $x(0), \dot{x}(0)$ determines the solution uniquely.

First order form

Any system of coupled ordinary differential equations can be written as a first order ODE in the form

$$\frac{dy}{dt} = f(t, y, \ldots)$$

where $y(t)$ is the solution vector, $t$ is the independent variable and $f$ is the flow vector or derivative vector, and the dots represent constant parameters independent of $t$.

For the van der Pol equation

$$y = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}, \quad f(t, y, \mu) = \begin{pmatrix} \dot{x} \\ -x - \mu(x^2 - 1)\dot{x} \end{pmatrix}$$
Euler’s algorithm

Euler introduced a simple algorithm to solve

\[ \frac{dx}{dt} = f(t, x) \]

Given \( x(t) \) at any time \( t \), the value of \( x(x + dt) \) at a later time can be approximated by a Taylor expansion

\[ x(t + dt) = x(t) + \frac{dx}{dt} \bigg|_t dt + \mathcal{O}(dt^2) = x(t) + f(t, x)dt + \mathcal{O}(dt^2) . \]

Given the initial value \( x(0) \) and a step size \( dt \), the solution is determined as a sequence of points

\[ x(0), \quad x(dt) = x(0) + f(0, x(0)) \, dt, \quad x(2dt) = x(dt) + f(dt, x(dt)) \, dt, \quad \ldots \]

Euler’s algorithm is not efficient. The local error in each step is \( \mathcal{O}(dt^2) \) and the global error after \( n \) steps is of order \( \mathcal{O}(dt^2 n) \sim T \mathcal{O}(dt) \), where \( T = t_{\text{max}} - t_0 \) is the total integration time. This implies that each additional decimal place (significant figure) of accuracy requires 10 times the number of integration steps! It is also numerically unstable for certain ODEs, including many important periodic systems.

Euler’s algorithm generalizes easily to a system of coupled first ODEs:

\[ y(t + dt) = y(t) + h f(t, y, \ldots) + \mathcal{O}(h^2) , \]

where \( h = dt \) is the step size in the independent variable \( t \), and it forms the basis for more efficient and numerically stable methods.
Fourth order Runge-Kutta algorithm

Runge and Kutta showed that by combining the results of four Euler steps, the error can be reduced to $O(h^5)$. The vector form of the fourth order Runge-Kutta algorithm is

\[ \begin{align*}
    k_1 &= h f \left( t, y(t) \right) \\
    k_2 &= h f \left( t + \frac{h}{2}, y(t) + \frac{1}{2}k_1 \right) \\
    k_3 &= h f \left( t + \frac{h}{2}, y(t) + \frac{1}{2}k_2 \right) \\
    k_4 &= h f \left( t + h, y(t) + k_3 \right) \\
    y(t + h) &= y(t) + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} + O(h^5).
\end{align*} \]

Coding the Algorithm


Download and install Mathematica UBIT Software

GCC compilers and tool chain PHY 410-505 Fall Tools
The GNU Scientific Library

Download and install the GNU Scientific Library

GSL C code to solve the van der Pol equation

The GSL Manual Section 26.6 gives the following sample code

```c
/*
   GNU GSL code for the van der Pol oscillator
   gcc -o vdp-rk4 vdp-rk4.c -lgsl -lgslcblas -lm
*/

#include <stdio.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_odeiv2.h>

int func (double t, const double y[], double f[],
     void *params)
{
    double mu = *(double *)params;
    f[0] = y[1];
    f[1] = -y[0] - mu*y[1]*(y[0]*y[0] - 1);
```
return GSL_SUCCESS;
}

int
jac (double t, const double y[], double *dfdy,
     double dfdt[], void *params)
{
    double mu = *(double *)params;
    gsl_matrix_view dfdy_mat
        = gsl_matrix_view_array (dfdy, 2, 2);
    gsl_matrix * m = &dfdy_mat.matrix;
    gsl_matrix_set (m, 0, 0, 0.0);
    gsl_matrix_set (m, 0, 1, 1.0);
    gsl_matrix_set (m, 1, 0, -2.0*mu*y[0]*y[1] - 1.0);
    gsl_matrix_set (m, 1, 1, -mu*(y[0]*y[0] - 1.0));
    dfdt[0] = 0.0;
    dfdt[1] = 0.0;
    return GSL_SUCCESS;
}

int
main (void)
{
    double mu = 10;
gsl_odeiv2_system sys = { func, jac, 2, &mu };  


gsl_odeiv2_driver *d = 
gsl_odeiv2_driver_alloc_y_new (&sys, gsl_odeiv2_step_rk4, 
1e-3, 1e-8, 1e-8);  

double t = 0.0;  
double y[2] = { 1.0, 0.0 };  
int i, s;  

for (i = 0; i < 100; i++)  
{
    s = gsl_odeiv2_driver_apply_fixed_step (d, &t, 1e-3, 1000, y);  

    if (s != GSL_SUCCESS)  
    {
        printf ("error:  driver returned %d\n", s);  
        break;  
    }

    printf ("%.5e %.5e %.5e\n", t, y[0], y[1]);  
}

gsl_odeiv2_driver_free (d);
return s;
}