Implementing Ab Initio Molecular Dynamics

Anthony Froehlich and Eric D. Glendening
Department of Chemistry and Physics, Indiana State University, Terre Haute, IN 47809

Introduction

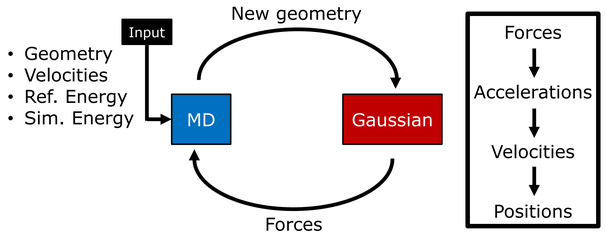
Molecular dynamics (MD) is defined as the computational study of atomic motion over a period of time. Most MD programs use predetermined potentials or force fields in order to determine the forces on, and by extension the kinematics of, the atoms in the system. Because this method requires little computationally heavy work to determine forces, accelerations, and velocities, it is possible to observe very large systems comprised of thousands of atoms. The downside of this method is the fact that chemical changes cannot be properly observed if the electronic structure of the system is not taken into consideration. A solution to this problem is to use ab initio molecular dynamics (AIMD). While classical MD uses predetermined values to assign forces to atoms, AIMD uses electronic structure calculations to find the energy of a system as well as the atomic forces at each step of a calculation, giving a much more accurate simulation at the cost of time and system size. Because the electronic structure is considered, chemical bonds and electron interactions can be tracked throughout the course of a simulation, leading to a more complete picture than one would get otherwise.

Implementation

Our AIMD program begins with an input file containing information relevant to the calculation. This includes, but is not limited to, the initial geometry of the system, the initial velocities of all atoms, a reference energy, and a simulation energy. The reference energy should be the equilibrium energy of the system found using the same level of theory as used throughout the simulation. The simulation energy is a sort of energy threshold above the reference energy that the system will never cross. As the potential energy (U) of the system approaches the sum of the reference and simulation energy, the kinetic energy (T) of the system will decrease so that the total energy will always equal the aforementioned sum.

$$E_{tot} = E_{ref} + E_{sim} = T + U$$

This information is passed to the body of the our program, which is called MD. Using the velocities and positions given in the input file, a simple extrapolation is used to predict the next geometry. This new geometry is passed to Gaussian, the computational chemistry packaged used to give the energy of each geometry as well as the forces on each atom. From these forces, the accelerations of each atom are derived, and a more accurate extrapolation method is used to determine the next geometry. This is repeated until the simulation is ended.



Newton's Equations of Motion

For any molecular dynamics simulation to work, Newtonian mechanics of motion must be taken into consideration at some point. While classical mechanics may not be used to derive forces, they are required to propagate the motion of the atoms or molecules being studied. In every case, accelerations are derived from the forces, new velocities are derived from current velocities and accelerations, and new positions are derived from current positions and velocities. The propagation method, however, can change drastically from program to program, and even within programs. In MD, there are currently two methods of propagation that are used. The first is a simple extrapolation used to find the velocities in the first step of the calculation, when limited information is available.

$$v_1 = v_0 + a_0 \cdot dt$$

In later steps, when more information is available, a different extrapolation that utilizes the accelerations at both the previous and current position is used to find velocities.

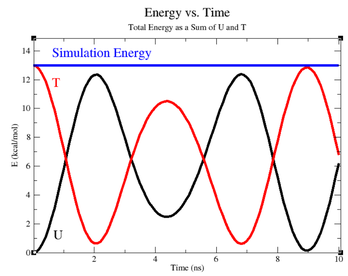
$$v_1 = v_0 + (2a_0 + 5a_{-1}) \cdot dt/6$$

After the velocities are found, another simple extrapolation is used to find the next geometry.

```
c  Else if the second configuration (scaling velocities to give
c  simulation energy)...

      else if(nstep.eq.1) then
        do i = 1,natoms
          do j = 1,3
            dr(j,i) = drm1(j,i) + d2rm1(j,i) * tstep
          enddo
        enddo
        call delerot(r,drm1,xmass,natoms)
        call vscale(dr,xmass,esim,eref,ev,natoms)

c  Else...

      else
        do i = 1,natoms
          do j = 1,3
            dr(j,i) = drm1(j,i) + (two * d2r(j,i) + five * d2rm1(j,i)
     +                                - d2rm2(j,i)) * tstep / six
          enddo
        enddo
      end if
```

Energy Conservation

In our simulations, the system being observed is treated as being completely isolated, meaning there is no way for energy to enter or exit the system. Because of this, it is essential that energy be conserved to ensure accurate results. This is done by scaling the velocities given in the input file once at the beginning of the simulation to raise or lower the kinetic energy so that the simulation energy is reached.



Eliminating Translation

All energies that are external to the system must be eliminated to ensure that all kinetic energy is applied to the normal modes of the system. To accomplish this, the translational and rotational energies of the system must be removed at the beginning of the simulation. For translation, this means finding the average velocity of the system, and correcting each atomic velocity by that amount.

$$v_{ave} = \sum_{i=1}^{N} v_i m_i / m_{tot}$$

$$v_i = v_o - v_{ave}$$

```
c  Calculate center of mass for r coordinates:

      tmass = zero
      do j = 1,3
        rshft(j) = zero
      enddo

      do i = 1,natoms
        tmass = tmass + xmass(i)
        do j = 1,3
          rshft(j) = rshft(j) + r(j,i) * xmass(i)
        enddo
      enddo

c  Shift r coordinates to place center of mass at origin:

      do i = 1,natoms
        do j = 1,3
          r(j,i) = r(j,i) - rshft(j) / tmass
        enddo
      enddo

      return
      end
```

Eliminating Rotation

Eliminating the rotation is slightly more complicated that translation, as angular momentum is conserved about the principal axes of rotation, not the Cartesian axes. Consequently, the principal moments of inertia and principal axes of rotation must be found before rotation may be removed. After this is done, rotation is removed as follows.

$$\mathbf{L} = \sum_{i=1}^{N} m_i (r_i \times v_i)$$

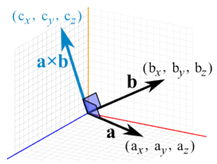
$$L_\alpha = \mathbf{L} \cdot \mathbf{u}_\alpha$$

$$\omega_\alpha = L_\alpha / I_\alpha$$

$$\boldsymbol{\omega} = \boldsymbol{\omega}' \mathbf{u}$$

$$\mathbf{v'}_i = \mathbf{v}_i - (\boldsymbol{\omega}_i \times \mathbf{r}_i)$$

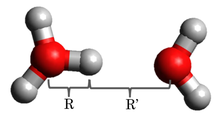
$$E_{rot} = \frac{1}{2}\left(\frac{L_\alpha^2}{I_\alpha}\frac{L_\beta^2}{I_\beta}\frac{L_\gamma^2}{I_\gamma}\right)$$




```
c  Find the principal moment of inertia tensor for the geometry r

      call findpmoi(r,xmass,natoms,pmoi)

c  Diagonalize the principal moment of inertial tensor

      call jacobi(pmoi,moments,axes,3,3,0,ierr)

      if(ierr.eq.1) call halt('JACOBI: Failed to converge')

c  Find the angular velocity vector of the system

      do i = 1,3
        L(i) = zero
      enddo

      do i = 1,natoms
        call vx(r(1,i),dr(1,i),rxv)

        do j = 1,3
          L(j) = L(j) + rxv(j) * xmass(i)
        enddo

      enddo


      do i = 1,3
        call vd(L,axes(1,i),Lpa(i))
      enddo

      do i = 1,3
        omegapa(i) = Lpa(i) / moments(i)
      enddo

      do i = 1,3
        omega(i) = omegapa(1) * axes(i,1) + omegapa(2) * axes(i,2) +
     +             omegapa(3) * axes(i,3)
      enddo

c  Subtract the corresponding tangential velocity from each atom

      do i = 1,natoms
        call vx(omega,r(1,i),wxr)
        do j = 1,3
          dr(j,i) = dr(j,i) - wxr(j)
        enddo
      enddo
```

Implementing Constraints

The next goal for the MD program is to implement constrained dynamics. This involves adding more components to the program that can constrain reaction coordinates and change them throughout a simulation.



With constrained dynamics implemented, we can study more aspects of a system than before. This includes the mean potential energy surface for a reaction, found by making small steps along one or more reaction coordinates and running a simulation at each step. Large amounts of data will be found at each step, meaning that it is very likely that more information can be ascertained from a constrained dynamics simulation than for a regular molecular dynamics simulation.

Continuing Effort

Going forward, more progress will be made towards implementing constrained dynamics capabilities, as well as making other changes to the program. Elimination of translational and rotational energy will be made optional, and more extrapolation methods will be made available. These and many more user options will be made available to allow flexibility in the program and more specific simulations to be performed.

Acknowledgements

The authors gratefully acknowledge the financial support of the ISU Department of Chemistry and Physics and the Center for Student Research and Creativity.

- Ahmad Ahmad
- Bryan Walker
- Paul Wells
- Drew Ratliff
- Dr. Joseph West
- Indiana State University SURE 2019

References

Gaussian 16, Revision C.01, Frisch, M. J. et al., Gaussian, Inc., Wallingford CT, 2016.

Kräutler, V. et al. (2001), A fast SHAKE algorithm to solve distance constraint equations for small molecules in molecular dynamics simulations. J. Comput. Chem., 22: 501-508.

Pierce, Rod. "Cross Product" Math Is Fun. Ed. Rod Pierce. 8 Feb 2019. 31 Jul 2019 <http://www.mathsisfun.com/algebra/vectors-cross-product.html>

Indiana State University