博文

A 100-line matlab code for molecular dynamics (slow version)

已有 21526 次阅读 2016-4-8 04:54 | 个人分类:计算物理基础 | 系统分类:科研笔记 | 关键词:学者 | 分子动力学模拟

用matlab写一个100行的分子动力学模拟程序

樊哲勇

<brucenju(at)gmail.com><zheyongfan(at)163.com>

最后修改时间：2016年4月24日

（感谢由琪同学：帮忙更正了一个注释中的笔误）

（上传了全部代码）

find_force.m

樊哲勇

加为好友 给我留言

打个招呼 发消息

find_neighbor.m

initialize_position.m

initialize_velocity.m

md.m

test_md.m

一、引言

分子动力学模拟(molecular dynamics simulation)是一个重要的计算方法，在物理、材料、化学以及生物等学科中都有广泛应用。本文介绍如何用matlab写出一个完整的分子动力学模拟程序。我先对分子动力学模拟的各个必要部分进行简要介绍，并同时给出相关的matlab函数，然后再给出调用这些函数的主函数，最后简要介绍如何验证程序的正确性。整个代码包括注释都只有100行左右。本文的最佳读者为对分子动力学模拟感兴趣的研究生。我了解到有些运用分子动力学模拟做研究的学生太过依赖于商业或者自由软件，自己并不理解分子动力学模

拟的细节。我认为这种现象是不令人满意的。本文的主要目的是帮助这些学生通过自己动手写代码来更好地掌握该方法。但要提醒的是：我假定读者对matlab编程相当地熟悉；在很多情况下，我认为代码比公式或者语言更能直接地表达我的思想。

二、分子动力学模拟的大意

1、分子动力学模拟的定义

要给分子动力学模拟下一个完美的定义是很难的；不同的人可能有不同的看法。如果非要我给出一个定义，那么我将给出下面的定义：

分子动力学模拟是一种数值计算方法，在这种方法中，我们对一个具有一定初始条件和边界条件的具有相互作用的多粒子系统的运动方程进行数值积分，得到系统在相空间中的一条离散的轨迹，并用统计力学的方法从这条相轨迹中提取出有用的物理结果。

如果你是初学者，那么你肯定不能完全理解这个定义。希望这样的读者在读完本文后能大致明白这个定义所涉及到的主要方面。

2、分子动力学模拟的计算流程

根据上述定义，我们可以设想一个典型的简单的分子动力学模拟有如下大致的计算流程:

1）设置系统的初始化条件

a)初始化各个粒子的位置矢量

b)初始化各个粒子的速度矢量

2）根据系统中的粒子所满足的相互作用规律，利用牛顿第二定律对所有粒子的运动方程进行数值积分，即不断地更新每个粒子的坐标和速度。最终,我们将得到一系列离散的时刻系统在相空间中的位置，即一条离散的相轨迹。

3）用统计力学的方法分析相轨迹所蕴含的物理规律。

三、初始化

初始化指的就是给定一个初始的相空间点，这包括各个粒子初始的坐标和速度。在分子动力学模拟中，我们需要对3*N（N是粒子数目）个二阶常微分方程进行数值积分。因为每一个二阶常微分方程的求解都需要有两个初始条件，所以我们需要确定6*N个初始条件：3*N个初始坐标分量和3*N个初始速度分量。

1、坐标（位置）的初始化

位置的初始化指的是为系统中的每个粒子选定一个初始的位置坐标。分子动力学模拟中如何初始化位置主要取决于所要模拟的体系。例如，如果要模拟固态氩，就得让各个氩原子的位置按面心立方结构排列。如果要模拟的是液态氩或者气态氩，依然可以将各个氩原子的位置按面心立方结构排列，因为随着系统的演化，各个原子的坐标会自动地偏离该初始结构，如果各个原子有一个初始的随机速度的话。

本文以具有面心立方结构的固态氩为例子作进行讨论。下面是一个固态氩系统坐标初始化的matlab函数:

```
function [r]=initialize_position(N,D,n0,nxyz,a) % FCC crystal
% D: dimension, which should be 3
% n0: number of atoms in a cubic unit cell, which is 4 in this case
% nxyz(1,3): nxyz(d) is the number of unit cells in the d-th direction
% N: number of atoms in the system, which is n0*nxyz(1)*nxyz(2)*nxyz(3)
```

```
% a(1,3): a(d) is the lattice constant in the d-th direction
% r(N,3): r(i,d) is the position of atom i along the d-th direction
r0=[0,0,0;0,0.5,0.5;0.5,0,0.5;0.5,0.5,0];r=zeros(N,D);n=0;
for nx=0:nxyz(1)-1
  for ny=0:nxyz(2)-1
    for nz=0:nxyz(3)-1
      for m=1:n0
        n=n+1;r(n,:)=a.*([nx,ny,nz]+r0(m,:));
      end
    end
  end
end
```

2、速度的初始化

我们知道，任何经典热力学系统在平衡时各个粒子的速度满足麦克斯韦分布。然而，作为初始条件，我们并不一定要求粒子的速度满足麦克斯韦分布。最简单的速度初始化方法是产生3*N个在某个区间均匀分布的随机速度值，再通过如下两个基本条件对速度值进行修正。

第一个条件是让系统的总动量为零。也就是说，我们不希望系统的质心在模拟的过程中跑动。

第二个条件是系统的总动能应该与所选定的温度对应。我们知道，在经典统计力学中，能量均分定理成立，即粒子的哈密顿量中每一个具有平方形式的能量项的统计平均值都等于k_B*T/2。其中，k_B是玻尔兹曼常数，T是系统的绝对温度。所以，在将质心的动量取为零之后就可以对每个粒子的速度进行一个标度变换，使得系统的初始温度与所设定的温度一致。

下面是一个速度初始化的matlab函数：

```
function [v]=initialize_velocity(K_B,N,D,T,m)
% K_B: Boltzmann's constant
% N: number of atoms in the system
% D: dimension, which is 3
% T: temperature prescribed
% m(N,1): m(i) is the mass of atom i
% v(N,3): v(i,d) is the velocity of atom i in the d-th direction
v=rand(N,3)-0.5;
momentum_average=zeros(1,D);
for d=1:D
  momentum_average(d)=sum(v(:,d).*m)/N;
end
for n=1:N
  v(n,:)=v(n,:)-momentum_average/m(n);
end
v=v*sqrt(T*D*K_B*N/sum(m.*sum(v.^2,2))); % scale velocity
```

四、力的计算。

1、Lennard-Jones势

宏观物质的性质在很大程度上是由微观粒子之间的相互作用力决定的。所以，对粒子间相互作用力的计算在分子动力学模拟中是至关重要的。粒子间有何种相互作用不是分子动力学模拟本身所能回答的；它本质上是一个量子力学的问题。在经典分子动力学模拟中，分子间的相互作用力是由一个经验势函数描述的。已发展出很多这样的势函数，其中最简单同时又最有用的势函数之一是Lennard-Jones势：

U(d)=4*epsilon*(sigma^12/d^12–sigma^6/d^6).

其中，epsilon和sigma是常量(分别具有能量和长度的量纲)，d是两个粒子间的距离，U(d) 是两个粒子之间的相互作用势能。这个函数形式是数学家John Lennard-Jones于1931年提出的，非常适合描述固态氩系统：

维基百科上对John Lennard-Jones的介绍：

https://en.wikipedia.org/wiki/John_Lennard-Jones#cite_note-paper4-7。

能够用相互作用势能表达的力是保守力，它等于势能的负梯度。根据势能推导相互作用力是本科《理论力学》水平的问题，故在此不作推导。

2、边界条件

在我们的定义中，除了初始条件，还提到了边界条件。边界条件对常微分方程的求解并不是必要的，但在分子动力学模拟中通常会根据所模拟的物理体系选取合适的边界条件，以期得到更合理的结果。边界条件的选取对力的计算也是有影响的。常用的边界条件有好几种，但我们这里只讨论其中的一种：周期性边界条件(periodic boundary conditions)。在计算机模拟中，模拟的系统的尺寸一定是有限的，通常比实验上对应的体系的尺寸小很多，选取周期性边界条件通常可以让模拟的体系更加接近于实际的情形，因为原本有边界的系统在应用了周期性边界条件之后，“似乎”没有边界了。当然，并不能说应用了周期性边界条件的系统就等价于无限大的系统，只能说周期性边界条件的应用可以部分地消除边界效应，让所模拟的系统的性质更加接近于无限大的系统的性质。

在计算两个粒子，比如粒子i和粒子j的距离的时候，就要考虑周期边界条件带来的影响。举个一维的例子。采用周期边界条件时，必须将一条线段想象为一个圈。假设线圈周长为Lx=10（任意单位），第i个粒子的坐标x_i=1, 第j个粒子的坐标x_j=8。请问这两个粒子的距离是多少呢？如果忽略周期边界条件，那么答案是|x_j–x_i|=7，而且j粒子在i粒子的右边（坐标值大的一边）。但是，在采取周期边界条件时，也可认为j粒子在i粒子的左边，且坐标值可以平移至8-10=-2。这样，j与i的距离是|x_j–x_i|=3，比平移j粒子之前两个粒子之间的距离要小。

最小镜像约定(minimum image convention)规定：在计算两个粒子的距离的时候，总是取最小的可能值。设x_j-x_i=x_ij，则这个约定等价于如下规则:

如果x_ij<-Lx/2,则将x_ij换为x_ij+Lx；

如果x_ij>+Lx/2,则将x_ij换为x_ij-Lx。

最终效果就是让变换后的x_ij的绝对值小于Lx/2。相关的代码（用matlab写仅有一行）在下面构建近邻列表的函数以及求力和势能的函数中都有所体现。

3、近邻列表

所谓的近邻列表（neighbor list）指的是确定系统中各个粒子之间的拓扑关系的列表：它告诉我们每个粒子都有多少个“邻居”、分别是哪些粒子。构建近邻列表的规则是：计算每一对粒子之间的距离，若该距离小于某一个设定的截断距离（cutoff distance），则判定它们互为“邻居”。本文暂且只讨论固态系统，其近邻列表不需要更新，只需在一开始构建一次，故暂不考虑性能问题。下面是一个对大体系来说不高效但很简单的构建近邻列表的matlab函数：

```
function [NN,NL]=find_neighbor(N,L,pbc,rc,r) % slow for large systems
% N: number of atoms in the system
% L(1,3): L(d) is the box length in the d-th direction
% pbc(1,3): pbc(d)=1(0) means periodic (free) in the d-th direction
% rc: cutoff distance
% r(N,3): r(i,d) is the position of atom i in the d-th direction
% NN(N,1): NN(i) is the number of neighbors of atom i
% NL(N,:): NL(i,k) is the index of the k-th neighbor of atom i
NN=zeros(N,1);
NL=zeros(N,N-1);
L_times_pbc=L.*pbc;
rc_square=rc*rc;
```

```
for n1=1:N-1
  for n2=n1+1:N
     r12=r(n2,:)-r(n1,:);
     r12=r12-round(r12./L).*L_times_pbc; %minimum image convention
     d12_square=sum(r12.*r12);
     if d12_square<rc_square
        NN(n1)=NN(n1)+1;NL(n1,NN(n1))=n2;
        NN(n2)=NN(n2)+1;NL(n2,NN(n2))=n1;%not used now but useful later
     end
  end
end
NL=NL(:,1:max(NN)); %may save some memory
```

4、求力和势能的代码

有了近邻列表，就可以求力和势能了。废话少说，直接给代码：

```
function [f,U]=find_force(N,D,NN,NL,L,pbc,r)
% N: number of atoms in the system
% D: space-dimension, which should be 3
% NN(N,1): NN(i) is the number of neighbors of atom i
% NL(N,:): NL(i,k) is the index of the k-th neighbor of atom i
% L(1,3): L(d) is the box length in the d-th direction
% pbc(1,3): pbc(d)=1(0) means periodic (free) in the d-th direction
% r(N,3): r(i,d) is the position of atom i in the d-th direction
% f(N,3): f(i,d) is the total force on atom i in the d-th direction
% U: total potential energy of the system
EPSILON=1.032e-2; % in units of eV (only for Argon)
SIGMA=3.405; % in units of Angstrom (only for Argon)
sigma_6=SIGMA^6;sigma_12=SIGMA^12;L_times_pbc=L.*pbc;
```

```
U=0; % initialize the total potential energy of the system
f=zeros(N,D); % initialize the total forces on each atom
for n1=1:N-1 % loop over the atoms
  for m=1:NN(n1) % loop over the neighbors of atom n1
    n2=NL(n1,m); % (n1, n2) is a pair of atoms
    if n2<n1;continue;end % Use Newton's 3rd law to speed up
    r12=r(n2,:)-r(n1,:);
    r12=r12-round(r12./L).*L_times_pbc; % minimum image convention
    d12_square=sum(r12.*r12);
    d_6=d12_square^3;d_8=d12_square*d_6;d_12=d_6*d_6;d_14=d_6*d_8;
    f12=(sigma_6/d_8-2.0*sigma_12/d_14)*24.0*EPSILON;
    f(n1,:)=f(n1,:)+f12*r12;
    f(n2,:)=f(n2,:)-f12*r12; % Newton's 3rd law used here
    U=U+4.0*EPSILON*(sigma_12/d_12-sigma_6/d_6); % accumulate energy
  end
end
```

五、运动方程的数值积分

在经典力学中，粒子的运动方程可以用牛顿第二定律表达。对运动方程进行数值积分的目的就是在给定的初始条件下找到各个粒子在一系列离散的时间点的坐标和速度值。我们假设每两个离散的时间点之间的间隔是固定的，记为dt，叫做时间步长(time step)。数值积分的方法有很多种，我们这里只介绍所谓的“速度-Verlet”积分方法。

在该方法中，第i个粒子在第m+1个离散的时刻的速度v_i(m+1)由下式给出：

v_i(m+1)=v_i(m)+a_i(m)*dt*0.5+a_i(m+1)*dt*0.5

而该粒子在第m+1个离散的时刻的坐标x_i(m+1)由下式给出：

x_i(m+1)=x_i(m)+v_i(m)*dt+a_i(m)*dt^2*0.5

由以上两式可以看出，坐标从第m个时刻到第m+1个时刻的更新可一步到位，但速度的更新不可。所以，在该积分方法中对坐标和速度的更新要按照如下方式进行：

1）先部分更新速度并完全更新坐标。

2）用更新后的坐标计算新的力（加速度）。

3）用新的力（加速度）完成速度余下部分的更新。

具体的代码将在下面的主函数中体现。

六、主函数

在matlab中并没有主函数一说。我这里说的主函数的意思是调用上面所有函数的函数。这个函数我给它起名为md，就是分子动力学（molecular dynamics）的意思。也没什么好解释的了，直接给代码（代码中有很多注释，可帮助理解）：

```
function [E]=md(D,n0,nxyz,a,pbc,Ne,Np,Ns,rc,dt,T)
% My unit system: energy-eV; length-Angstrom; mass-atomic mass unit
% D: dimension, which should be 3
% n0: number of atoms in a cubic unit cell, which is 4 in this case
% nxyz(1,3): nxyz(d) is the number of unit cells in the d-th direction
% a(1,3): a(d) is the lattice constant in the d-th direction
% pbc(1,3): pbc(d)=1(0) means periodic (free) in the d-th direction
% Ne: number of time steps in the equilibration stage
% Np: number of time steps in the production stage
```

```
% Ns: sampling interval in the production stage
% rc: cutoff distance
% dt: time step of integration in units of fs
% T: temperature prescribed in units of K
% E(:,1): potenital eneryg per atom at regular time points
% E(:,2): kinetic eneryg per atom at regular time points
% E(:,3): total eneryg per atom at regular time points
K_B=8.625e-5; % Boltzmann's constant in my unit system
TIME_UNIT_CONVERSION=10.18; % from fs to my unit system
N=n0*nxyz(1)*nxyz(2)*nxyz(3); % number of atoms
L=a.*nxyz; % box size (Angstrom)
dt=dt/TIME_UNIT_CONVERSION; % time step in my unit system
m=ones(N,1)*40; % mass for Argon atom in my unit system
r=initialize_position(N,D,n0,nxyz,a); % intial positions
v=initialize_velocity(K_B,N,D,T,m); % initial velocities
[NN,NL]=find_neighbor(N,L,pbc,rc,r); % fixed neighbor list
[f]=find_force(N,D,NN,NL,L,pbc,r); % initial forces
E=zeros(Np/Ns,3); % energy data to be computed
for step=1:(Ne+Np) % time-evolution started
  for d=1:D % step 1 of Velocity-Verlet
     v(:,d)=v(:,d)+(f(:,d)./m)*(dt*0.5);
     r(:,d)=r(:,d)+v(:,d)*dt;
  end
  [f,U]=find_force(N,D,NN,NL,L,pbc,r); % update forces
  for d=1:D % step 2 of Velocity-Verlet
     v(:,d)=v(:,d)+(f(:,d)./m)*(dt*0.5);
  end
  if step<=Ne; % control temperature in the equilibration stage
     v=v*sqrt(T*D*K_B*N/sum(m.*sum(v.^2,2))); % scale velocity
  elseif mod(step,Ns)==0 % measure in the production stage
     E((step-Ne)/Ns,1)=U/N; % potential (per atom)
     E((step-Ne)/Ns,2)=0.5*sum(m.*sum(v.^2,2))/N; % kinetic energy
  end
end % time-evolution completed
```

```
E(:,3)=E(:,1)+E(:,2); % total enegy (per atom)
```

七、验证程序正确性的判据之一：能量守恒

上面已经给出了全部的源代码。虽然我觉得我已经很小心地写了，但其中很有可能有bug。程序中的bug主要分两种，一种是简单的语法错误，一种是隐藏的逻辑错误。前者在编译（或者尝试运行）时就能被发现并修正，后者却很难完全消除。写好了一个代码，在用它做科研中的计算之前要尽可能全面地对其进行检验。检验分子动力学模拟程序的判据有很多，但其中最关键的一个恐怕要数能量守恒了（动量守恒也类似）。牛顿第三定律从理论上保证了系统的总能量是守恒的，但由于我们在作仅有有限精度的数值计算，故能量在系统的演化过程中还是会有涨落。下面用前面的matlab代码来验证能量守恒被满足的程度。

将上述各个函数的代码拷贝至同名文件（注意后缀名为.m）并放置于同一文件夹内，启动matlab，进入该文件夹，在命令行（命令窗口）敲入如下命令并回车：

```
tic;[E]=md(3,4,[4,4,4],5.45*[1,1,1],[1,1,1],1000,1000,20,10,5,80);toc
```

简要说明一下这个命令中的参数所描述的模拟对象：

一个含有256个氩原子，晶格常数为5.45 angstrom，温度为80 kelvin 的固态氩系统。三个方向都应用了周期边界条件。积分步长为5 femtosecond，平衡（控制温度）过程用1000步（5 picosecond），产出（不控制温度）过程也用1000步（5 picosecond），但每隔20步（0.1 picosecond）输出一次能量值。截断距离为10 angstrom（每个原子有86个“邻居”）。

该计算在我的笔记本电脑上需要63.758959秒。完成该计算之后，变量E中便保存了一些能量值（包括平均每个原子的动能、势能、以及总能）。下图显示了动能、势能、以及总能在产出过程（没有控制温度，即没有对速度进行标度变换）中随时间的变化情况：

用mean(E)命令可以算出各个能量的时间平均值，结果如下：

势能平均值：-0.0721 eV

动能平均值： 0.0103 eV

总能平均值：-0.0618 eV

其中，我们对势能的大小没什么感觉，不查资料的话并不知道它应该是多少才合理，但我们可以很快验证动能大小的正确性。根据能量均分定理，平均每个原子的动能的大小应该等于玻尔兹曼常数乘以温度的3/2倍。因为我们所取的温度为80开尔文，故动能应该为1.5*8.625e-5*80电子伏特，用matlab算一下可知，这正好等于上面的动能平均值：0.0103 eV。

用std(E)命令可以算出各个能量的标准偏差（standard deviation），结果如下：

势能标准偏差：3.738*1.0e-4 eV

动能标准偏差：3.736*1.0e-4 eV

总能标准偏差：2.435*1.0e-7 eV

可以看出，动能和势能的标准偏差很接近。从上图可以看出，动能和势能是此消彼长的关系；这正反映了能量守恒。具体地说，总能的标准偏差比动能和势能的标准偏差小 3 个数量级；这是比较合理的结果。另外，如果计算总能的相对误差，会发现其在1/100000以内。通常，如果总能的相对误差在1/10000之内都是比较正常的。

好了，关于能量守恒的验证就到这里为止。有兴趣的读者可以自己以上述代码为基础作更多测试。

读到这里，有的读者可能会认为我并没有像引言里说的那样，“介绍如何用matlab写出一个完整的分子动力学模拟程序”，而只是硬生生地给出了程序。确实如此。我原本想一步一步介绍如何将原理和公式变成代码的，但发现那样会很繁琐。我暂时没有精力去做那样的事情。但是我也很欣赏Linus Torvalds的那句简写为五个字母RTFSC的粗话：“Read the fucking source code.”。这句话是对搞计算机的人的最大的忠告之一；我认为它也适用于搞计算物理的人。

八、总结

本文通过一个具体的matlab代码简要介绍了分子动力学模拟的基本技术。该matlab代码除去注释之后不到100行，是一个“麻雀虽小，五脏俱全”的分子动力学模拟代码，可以作为初学者学习分子动力学模拟的出发点。该代码在性能上还有优化的空间（用一些技巧可将计算速度提高好几倍），详情留给以后的博文。最后，欢迎读者指出本文和代码中的不足，在此先表示感谢！

下一篇：A 100-line matlab code for molecular dynamics (fast version)