队伍编号	21230050015
题号	B

基于智能优化算法的金和硼团簇结构预测研究

摘 要

对原子团簇基态结构的预测是研究团簇的核心问题之一，其结果直接影响了对团簇性质的讨论。本文对Au**团簇**的基态结构预测使用$Sutton-Chen$**势能函数**，对B**团簇**的基态结构预测使用**反演势能函数**来描述团簇中原子间的相互作用，并通过求解势能函数的最小值来预测团簇的基本结构。但团簇结构优化问题已经证实为NP难问题，目前还没有多项式时间的精确算法可以解决此类问题。目前求解此问题的重点是如何创造出**高效的启发式算法**。

针对于此，本文的主要工作是运用启发式算法对SC**势能函数**描述的Au和**反演势能函数**描述的B团簇进行基态结构的预测。得到如下成果:

问题一的模型介绍了SC势能函数，并将最优结构与能量联系起来，预测出Au_{20}的全局最优结构为**正四面体状结构**；问题二的模型通过**改进的**BH**算法**，动态预测出Au_{32}的全局最优结构为**类似“足球”形状**，由12块五边形，20块六边形组成，其内部各键键长接近定值，该结构具有**一定的对称性和稳定性**；问题三的模型选取适当的势能模型描述团簇原子间的作用力，于是引入**多体势函数**，并对**偏差进行了修正**；预测团簇的基态结构也是一个**全局优化问题**，问题四的模型利用**智能优化算法**中的**差分进化算法**对B原子团簇进行结构优化，预测出B_{40}^-的结构为**近球形**，且该结构具有**高度对称性和较强的稳定性**。

本文的所获得的结论对今后原子团簇结构的实验研究具有一定的参考意义。文中运用的启发式算法以及全局优化算法也有望成为复杂团簇结构研究的有效方法。

关键词：团簇结构优化、结构预测、全局优化、势能函数、启发式算法

一、 问题背景

团簇是介于原子/分子与宏观物质之间的，具有确定的原子组成和化学结构与性质的亚微观结构，是关联宏观性质和物质微观结构较为理想的模型，是认识微观物质世界的重要桥梁，对深刻认识和理解物质转化的规律具有重大意义。

原子团簇具有与原子和分子截然不同的性质，研究团簇如何由原子一步步发展而成，即预测团簇的最优结构，对发现新型团簇材料的结构和性能具有重要意义。而原子团簇基态结构的预测既是一个连续优化问题，同时也是一个NP难问题，故对于本问题的研究同时也对求解优化问题以及NP难问题有新的思考。

二、 模型假设

1. 不考虑相对论效应；
2. 不考虑微观条件下的粒子不确定性。

三、 符号及软件说明

符号	符号解释说明
$E(x)$	势能
N	原子数
r_{ij}	两个原子X_i和X_j间的欧氏距离
$\vec{F}_{j->i}$	原子X_j对X_i的作用力
∇f	目标函数f的梯度向量
$\nabla^2 f$	目标函数f的海森矩阵
$V_r(i)$	对能
$V_d(i)$	原子内嵌势能
ε	能量值

软件说明：建模中使用了$MATLAB\ 2020$，$PyCharm$，$DMol3$，$Code\ Blocks$等软件。

名词解释：

势函数：描述原子（分子）间相互作用的函数；

对势：通常使用的原子间相互作用势为对势，又称为两体势；

多体势：通常的多体势只包含二体势和三体势；

内部算子操作：将团簇中高能量的原子移动至团簇的内部；

表面算子操作：将团簇中高能量的原子移动到表面某合适的位置。

四、 问题分析

4.1 问题一分析

对于研究对象单原子团簇(即只有一种物质的原子团簇)Au_{20}，基于势能函数的金属纳米团簇结构优化的衡量指标是体系的能量值，可以通过求解原子团簇势能函数的最小值预测团簇的基态结构。

可以通过分析 1000 个数据，找到能量最小的一组模型，将其空间模型绘制出来，观察并总结模型特征，并得到势能函数的参数，进而预测出Au_{20}能量最小时的形状，即可以找到全局最优结构。

4.2 问题二分析

基于问题一的算法以及得到的结果，计划利用改进的$Basin-hopping$(简称BH)算法以及**拟物拟人算法**的实现步骤（如格局的不断变化）来找到Au_{32}的不同异构体。在问题一的启发下，分析得到的异构体的$3D$构型的特点，找到能量较小时对应的模型变化规律，把异构体数据（主要包括x,y,z空间坐标）进行分组，并通过**拟牛顿法**下的$L-BFGS$算法得到每组数据的局部最优结构，再对每组局部最结构进行能量的比较，进而得到Au_{32}的全局最优结构。

4.3 问题三分析

问题三与四相较于问题一与二，其研究对象发生变化(由原子变成带电离子)，虽然总目标和所需解决的问题相似，但需要的算法并不相同。只考虑对势可能误差较大，需在原子受力时进一步考虑两个（三个，甚至多个）原子的合力对单原子的影响。基于此得到总势能函数，找到函数的最值点，分析能量最小的几组构型，进而预测B_{45}^-中全局最优结构。

4.4 问题四分析

考虑到B_{40}^-的结构较为复杂，而差分进化法可以用来解决比较复杂的一些优化问题，故计划利用**差分进化算法**优化B_{40}^-的结构。

五、 模型建立与问题求解

5.1 问题一模型建立与求解

问题二是问题一的发展和优化，两者有相似的解决思路，将在问题一和问题二的解题思路总结如下：

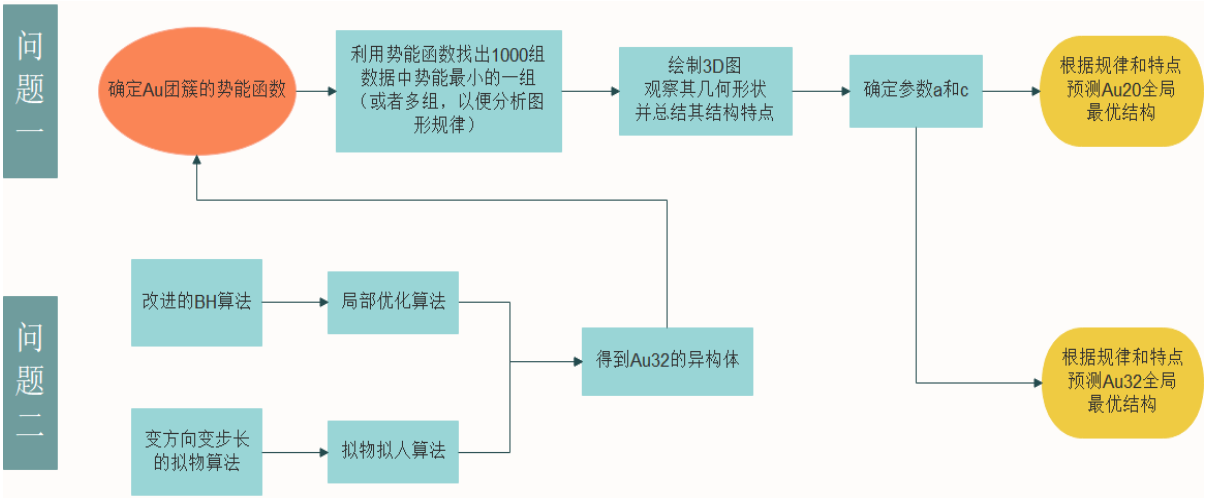


图 1 问题一与二的模型思路

5.1.1 模型建立

1. 势能函数

Sutton－Chen 势能函数（以下简称*SC*势能函数）是一种多体势函数，用来描述金属团簇结构，例如本文问题一的 Au_{20} 团簇[1]。*SC*势能函数描述如下：

$$E(x)=E(X_1,...,X_N)=\sum_{i=1}^{N}[V_r(i)+V_d(i)] \tag{1}$$

$$V_r(i)=\frac{1}{2}\sum_{j=1,j\neq i}^{N}\left(\frac{a}{r_{ij}}\right)^n, V_d(j)=-c\sqrt{\sum_{j=1,j\neq i}^{N}\left(\frac{a}{r_{ij}}\right)^m} \tag{2}$$

其中，N 表示原子数，$V_r(i)$ 为对能，$V_d(i)$ 为原子内嵌势能，$X_i\in R^3$ 表示原子的坐标，r_{ij} 表示 X_i 和 X_j 之间的欧氏距离，a 为距离控制参数，c 为量纲参数，n,m 均为参数，且 $n>m$。对于本题研究对象 Au_{20} 团簇，其相应的参数取值如下[2]：$n=10, m=8, c=34.408, a=3$。

5.1.2 模型求解

根据 1000 组数据，首先利用 *Python* 编程，根据势函数找到能量最低的 8 组数据，进而利用 *MATLAB－Molecule Viewer* 进行可视化，得到如下结构图（依次按能量升序排列，即图 2 为能量最小的结构图）：

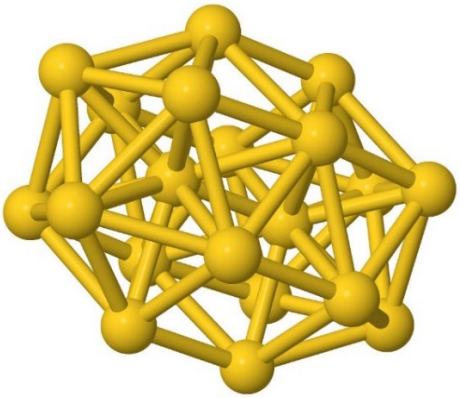


图 2 第 275 个团簇结构

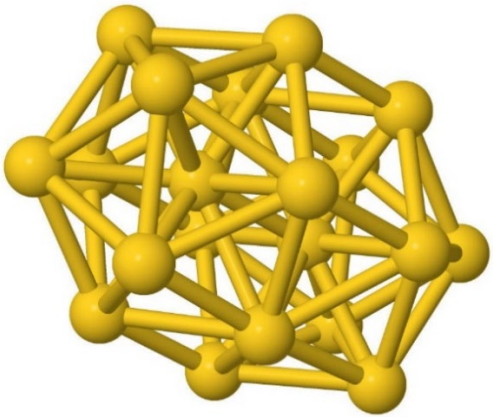


图 3 第 729 个团簇结构

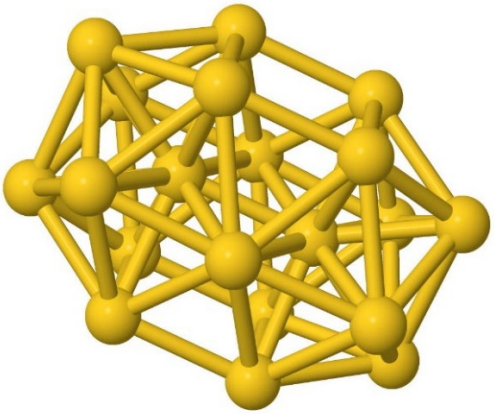


图 4 第 722 个团簇结构

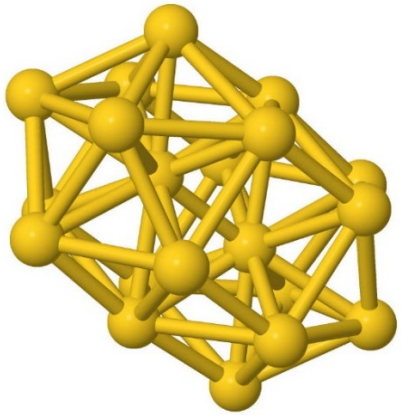


图 5 第 931 个团簇结构

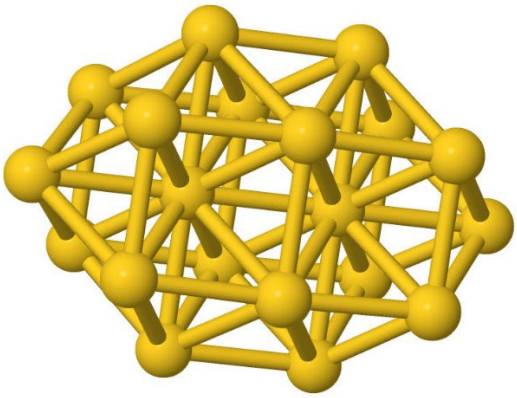


图 6 第 732 个团簇结构

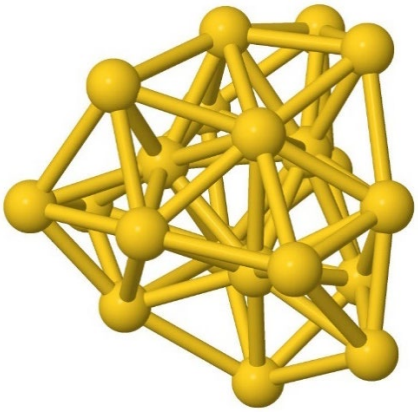


图 7 第 847 个团簇结构

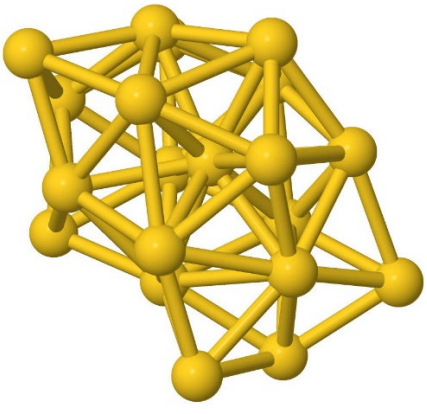


图 8 第 898 个团簇结构

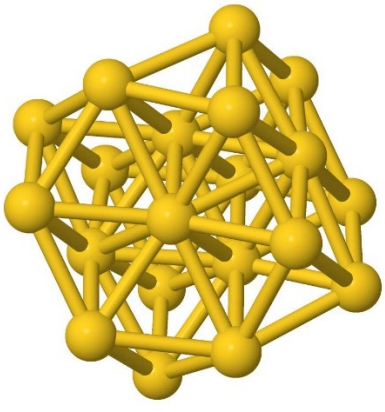


图 9 第 849 个团簇结构

可以观察到，由能量最低升序排列，能量最小的 4 个团簇（图 2-图 5）结构较为相似，通过计算其原子间的键长和键角，发现团簇结构内包含多个近似正四面体的形状，而能量排序为 5 及以后的团簇（图 6-图 9）结构较为不稳定，形状没有规律。

综上所述，本文猜测对于Au_{20}，其能量最低的结构（全局最优结构）为正四面体形状，20 个点构成一个大的正四面体，分为四层，针对每一层之间的原子，任意四个相连均构成正四面体，其具体结构如图所示：

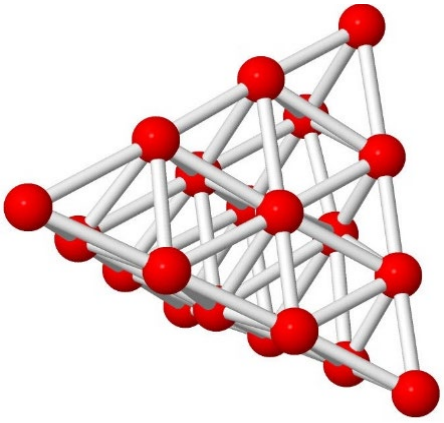


图 10 正面

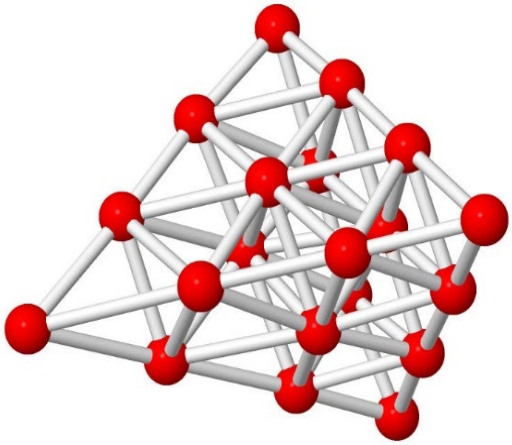


图 11 前面

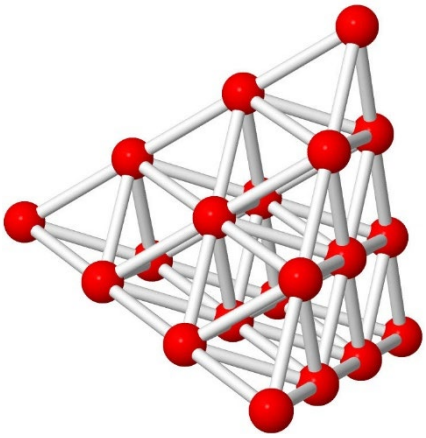


图 12 右侧

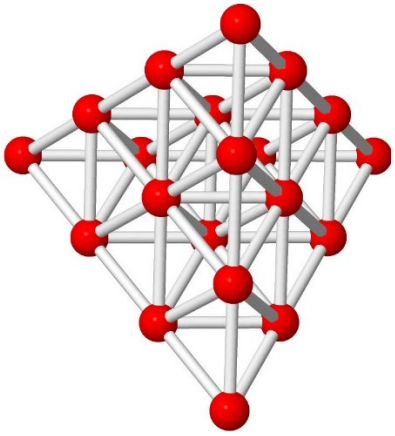


图 13 到顶层

为了验证此结构的模型是全局最优结构，本文利用势函数，计算此时结构的总能量，得到此时总能量为-5320.7394929，相比于所给 1000 组数据中能量最小（为-5193.5235756）的一组结构，本正四面体团簇结构能量值更小。

故得到Au_{20}的全局最优结构为如图 10-12 所示。形状为：正四面体，20 个原子均匀分布在正四面体的四条棱上。

5.1.3 模型总结

本模型利用“SC势能函数中的两个参数不影响势能函数最小值的求解”这条性质，在求解问题时去掉这两个参数以简化计算。得到能量最小的几组Au_{20}的构型，分析构型特点，进而预测Au_{20}的全局最优结构。

5.2 问题二模型的建立与求解

问题二基于问题一中的势函数和算法，得到异构体。进而把得到的异构体当作参考数据，分析异构体构型的特点，找出能量较小时构型的规律，再利用拟物拟人算法、BH算法和局部优化算法对Au_{32}团簇结构进行预测。

对于以上思路，建立如下模型：

5.2.1 模型建立

团簇通过物理或化学结合力组成，团簇中的每个原子都会受到其他原子的作用力，这样每一个原子都会受到来自其他原子的合力，原子将在合力作用下发生运动。当两个原子相距较远时，它们之间会在吸引力的作用下而相互靠近；当两个原子相距较远时，它们之间会在排斥力的作用下相互远离。基于上一小节的势能函数，在这里设原子i和原子j之间的作用力为：

$$\vec{F}_{j->i}=-\frac{d(V_r(r_{ij})+V_d(r_{ij}))}{dr_{ij}}\frac{\vec{X_i}-\vec{X_j}}{r_{ij}} \tag{3}$$

对于研究对象Au_{20}，可选定$n=10,m=8,c=34.408,a=1$，代入参数，此时得到：

$$\begin{aligned}\vec{F}_{j->i}&=(5a^{10}r_{ij}^{-12}-4ca^4r_{ij}^{-6})\left(\vec{X_i}-\vec{X_j}\right)\\&=(5r_{ij}^{-12}-137.632r_{ij}^{-6})\left(\vec{X_i}-\vec{X_j}\right)\end{aligned} \tag{4}$$

5.2.1.1 变方向变步长拟物算法

算法具体过程：

输入：原子个数N，最大时刻$t_{\max}$

Step1：$step_{\max}=0.24,step_{\min}=0.01$，算法截止条件$\varepsilon=1.0e-6$，当前时刻$t=0$，初始化步长$nowstep=step_{\max}$，利用限制团簇区域大小的方法产生初始格局$(X_1(t),X_2(t),\cdots,X_N(t))$，（$X_i(t)$为第$i$个原子$t$时刻的坐标），计算此时的函数势能值$U_0$.

Step2：计算原子在t时刻受到的合力为$(F_1(t),F_2(t),\cdots,F_N(t))$（$loop$为第$loop$执行$Step2$，若$F_{total}(t)<\varepsilon$或$t>t_{max}$，转$Step4$，否则先让第 1 个原子在合力的作用下运动：

$$(X_1(t+1),X_2(t+1),\cdots,X_N(t+1)) = (X_1(t),X_2(t),\cdots,X_N(t)) + \left(1-\frac{1}{t+2}\right)*nowstep*(F_1(t),1,\cdots,1) \tag{5}$$

再根据最新位置重新计算合力.

再让第 2 个原子在合力的作用下运动：

$$(X_1(t+1),X_2(t+1),\cdots,X_N(t+1)) = (X_1(t),X_2(t),\cdots,X_N(t)) + \left(1-\frac{1}{t+2}\right)*nowstep*(1,F_2(t),\cdots,1) \tag{6}$$

再根据最新位置重新计算合力.

不断重复上述步骤，直至让第N个原子在合力的作用下运动：

$$(X_1(t+1),X_2(t+1),\cdots,X_N(t+1)) = (X_1(t),X_2(t),\cdots,X_N(t)) + \left(1-\frac{1}{t+2}\right)*nowstep*(1,1,\cdots,F_N(t)) \tag{7}$$

计算此时势能函数值为U_1，并令$t=t+1$.

Step3： 若$U_1<U_0$，则记$U_0=U_1$，转$Step2$，否则，如果$U_1\geqslant U_0$且$nowstep>step_{\min}$,则令$nowstep*=0.99$,转向$Step2$.

Step4：算法停止，输出此时的势能值U_0以及团簇构型.

注记：

1. 参数ε对SC团簇基态结构的预测没有影响，可以根据实验结果动态调整；

2. 参数n,m对于本研究对象Au_{20}是固定的。参数a,c可以变化，就文献来看，一般如是选取。而且参数a,c的选取对势能函数最优值的求解没有影响，所以在计算时可以把它们都取为 1 以简化计算；

3. 关于公式(5)(6)(7)中的系数$1-\frac{1}{t+2}$，是为了限制合力的大小，使拟物算法在初始阶段摆动的幅度不会太大。例如在初始阶段$t\to 0$时，加上系数，可以使得力F仅为原始的$\frac{1}{2}$；而当t趋于稳定，如$t\to\infty$时，力F接近全力。这是一种改进策略。

5.2.1.2 求解团簇优化问题的拟物拟人算法

输入：原子个数N，最大不改进次数$MaxNolmproveNum$.

输出：求得最好的势能函数值U_0以及团簇构型

算法具体过程：

Step1：初始化参数——跳坑次数$nHope=0$，精度$\varepsilon=1.0e-6$，N为原子数，不改进次数$MaxNoImproveNum=0$，$U_{best}(U)$表示团簇原子数为N时的当前最优值.

Step2：在半径为$\left(1.1053N^{\frac{1}{3}}-1\right)5^{\frac{1}{6}}2^{-\frac{2}{3}}$的球内随机产生原子团簇的初始格局(不能出现重合的位置坐标)$(X_1(t),X_2(t),\cdots,X_N(t))$，($X_i(t)$为第 i 个原子 t 时刻的坐标)，调用**变方向变步长拟物算法**对该格局极小化后的格局为X_{odd}，对应的势能为U_{odd}.

Step3：　如果$U_{old}<U_{best}(N)+\frac{\varepsilon}{2}$，转$Step7$；如果$NoImproveNum<MaxNoImproveNum$，转$Step4$；否则，转$Step6$.

Step4：计算每个原子的能量，选取能量最大的前 3 个(此处可以选择 1 至 5 个)随机放置在几何中心附近(不能够与原来的原子重合)。用此法改变原格局X_{odd}，再利用**变方向变步长拟物算法**极小化，此时得到的新格局X_{new}，对应势能为U_{new}，转$Step5$.

Step5：　若$U_{new}<U_{best}(N)-\frac{\varepsilon}{2}$，则记$U_{old}=U_{new}$，$X_{old}=X_{new}$，$NoImproveNum=0$，转$Step3$；否则，$NoImproveNum++$，转$Step3$.

Step6：对格局X_{old}调用全局扰动策略(使所有的原子按随机的方向运动一小段距离，为一种邻域搜索策略。随机生成一个作用力，作用在每一个原子上，暂定这个作用力的数量级为**变方向变步长拟物算法**中最小作用力的十分之一，此处应该需要根据实验结果进行调整)，若扰动后新的格局能量变小，则更新X_{old}和U_{old}，直到连续多次扰动能量都不能下降为止。此时若$U_{old}<U_{best}(N)+\frac{\varepsilon}{2}$，则转$Step7$；否则转$Step8$.

Step7：成功停机，输出当前的势能值U_{old}以及格局.

Step8：失败停机.

5.2.1.3　改进的$Basin-Hopping(BH)$算法

然$Basin-Hopping(BH)$算法简单、执行速度较快，但是算法的随机性较大，对团簇的势能值的改进能力有限。故本文采用改进的$Basin-Hopping(BH)$算法。算法的执行流程如下：

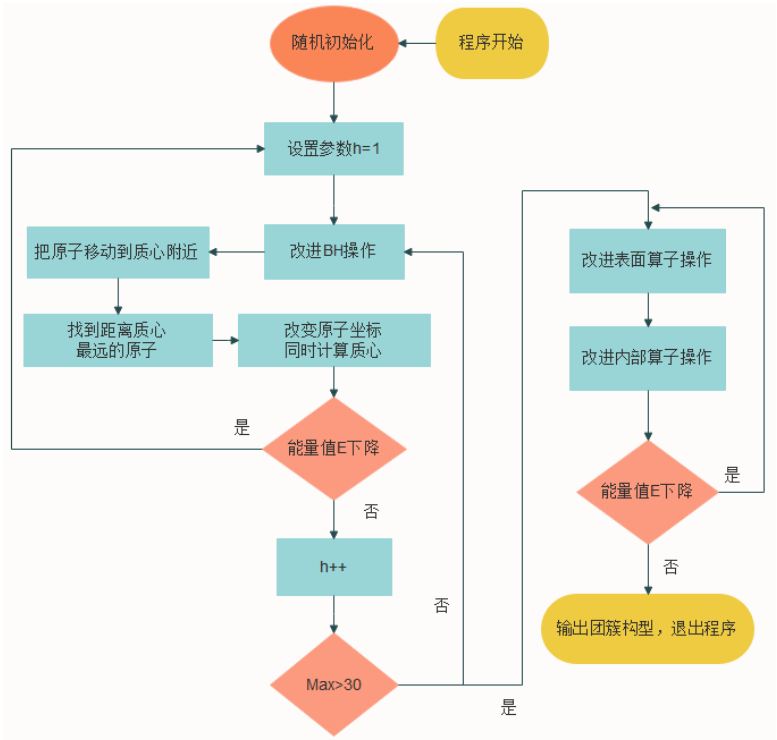


图 14 BH 算法整体执行的流程

Step1：设置参数$h=1$，连续最大优化次数$Max=30$，对当前团簇格局$X(x_1,y_1,z_1,\ldots,x_n,y_n,z_n)$（$n$为原子数目）进行局部优化，得到新的团簇格局$S_0$；

Step2：将S_0中每个原子的坐标随机地加上或减去一个正实数r，即$x_i=x_i+rand(-r,r)$，$y_i=y_i+rand(-r,r)$，$zi=zi+rand(-r,r)\,(i=1,2,3,\ldots,n)$，得到新的团簇格局$S_1$；

Step3：计算S_1的质心坐标(x,y,z)，其中$x=\dfrac{(x_1+x_2+\cdots+x_n)}{n}$，$y=\dfrac{(y_1+y_2+\cdots+y_n)}{n}$，$z=\dfrac{(z_1+z_2+\cdots+z_n)}{n}$；

Step4：找到S_1中离质心坐标欧式距离最远的原子k，坐标为(x_k,y_k,z_k)；

Step5：将k移动到团簇的质心附近，即$x_k=x+rand(-r,r)$，$y_k=y+rand(-r,r)$，$zk=z+rand(-r,r)$，此时得到新的团簇格局S_2；

Step6：对S_2进行局部优化得到新的团簇格局S_3；

Step7：若S_3的势能值$<S_0$的势能值，则$S_0=S_3$，$h=1$跳转到继续执行$Step2$；若S_3的势能值$\geq S_0$的势能值，则$h=h+1$；

Step8：若$h\leq Max$，跳转到$Step2$继续执行，否则结束操作；

注记：将BH操作放在优化的第一步可以在一个较短的时间内将团簇结构优化为一个能量值相对较低的构型。然后将改进的表面算子操作和改进的内部算子操作作为一个整体循环放在改进BH操作的下一步继续执行。因为当团簇结构优化的余地不大时，表面算子操作和内部算子操作可以对团簇结构起到很好的优化作用。将其放在BH操作后，可以在BH操作无法继续优化团簇构型时，继续优化团簇结构，降低团簇的能量值。

另外，这两个操作可以互补，作为一个整体循环可以更好地优化团簇结构，通常高能量的原子位于团簇结构的表面，经过内部算子操作，表面的原子会被移动到团簇的内部，进而经过局部优化后团簇表面的原子排列会发生改变。利用表面算子操作可以对团簇的表面进行优化，使得团簇的表面更光滑平整。

5.2.1.4 全局优化模型

1. 模型准备

启发式算法的本质是局部搜索方法，这样无法避免地会陷入局部最优的困境中，接着通过启发式策略跳出当前局部最优困境，继续进行局部优化，再跳出局部最优，如此反复直到找到一个可以接受的结果。在团簇结构优化中也是如此，Au_{32}团簇结构优化时（如BH操作、表面算子操作和内部算子操作）会经历多次“跳坑”操作和多次的局部优化操作。局部优化操作可以优化当前的团簇格局，找到一个在其附近的局部最优格局，对应于势能函数就是找到一个局部极小值点。

2. 建立模型

本文采用的局部优化算法为**拟牛顿法**，是一种基于牛顿法并加以改进的方法。如下先介绍牛顿法。牛顿法的基本思想是：在极小值点附近对目标函数$f(X)$做二阶（甚至更高阶，但对于本模型，经验证二阶足以达到效果，阶数过高会增加计算负担）$Taylor$展开，进而找到下一个极小值点的估计值，反复迭代直至误差在容许的范围内。对多维目标函数做在X_k处做二阶泰勒展开式

$$\phi(X)=f(X_k)+\nabla f(X_k)\cdot(X\text{-}X_k)+\frac{1}{2}\cdot^{(X\text{-}X_k)^T}\nabla^2 f(X_k)\cdot(X\text{-}X_k) \tag{8}$$

其中，∇f为目标函数f的梯度向量，记为g；$\nabla^2 f$为目标函数f的海森矩阵，记为H。对上式中$\phi(X)$求驻点，即两边同时作用一个梯度算子，可以得到$X=X_k-H_k^{-1}\cdot g_k$。如果给定初始点x_0，则可以构造出迭代式$x_{k+1}=x_k-H_k^{-1}\cdot g_k, k=0,1,...$来逼近局部的极小值点，其中$d_k=-H_k^{-1}\cdot g_k$，称为牛顿方向。

考虑到在上述方法中需要求目标函数的二阶偏导数，计算复杂度较高，而且无法保证目标函数f的海森矩阵$\nabla^2 f$正定，进而导致上述方法失败。于是本文采用拟牛顿法，其主要思想为：近似构造海森矩阵，模拟牛顿法的条件，进而优化目标函数。不同的构造海森矩阵的方法对应不同的拟牛顿法，这里使用$L-BFGS$算法。具体流程如下：

Step1：给定搜索起点X_0，误差范围ε，令初始迭代矩阵$B_0=I$，迭代步数$k=0$；

Step2：确定搜索方向$d_k=-B_k^{-1}\cdot g_k$，计算步长$\lambda_k=\min f(x_k+\lambda d_k)$；

Step3：带入步长λ_k计算$s_k=\lambda_k d_k$，计算$X_{k+1}=X_k+s_k$；

Step4：用X_{k+1}计算g_{k+1}，若$\|g_{k+1}\|<\varepsilon$，则结束算法；

Step5：计算$y_k=g_{k+1}-g_k$，计算$B_{k+1}=B_k+\left(\frac{y_k y_k^T}{y_k^T s_k}-\frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}\right)$；

Step6：令$k=k+1$，跳转至$Step3$；

注记：在Au_{32}团簇结构优化中会多次用到局部优化操作，因此一个好的局部优化算法能够节省大量的运行时间，大幅度提高算法的搜索速度。实验表明$L-BFGS$对SC势能Au_{32}团簇能够快速有效地搜索到局部最优结构。本文采取将数据分组，对每组数据求得局部最优结构后进行比较，进而得到全局最优结构。

5.2.2 模型求解

通过以上模型和算法，得到Au_{32}的最优结构图：

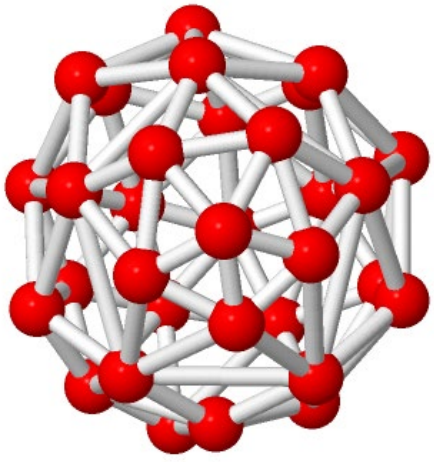


图 15 Au_{32}的最优结构图

Au_{32}的最优结构图类似“足球”形状，由 12 块五边形，20 块六边形组成。

该结构具有高度对称性和稳定性，内部各键键长接近定值，对于Au_{32}能够保持此种结构，除整体稳定性之外，还与平衡距离有关，$Au-Au$之间平衡距离非常接近。这样在该结构中没有出现过短或过长的键使能量升高。使用序列参数来分析Au_{32}团簇中各种原子的分布规律。R表示原子到团簇结构中心的距离，形式为：

$$R=\frac{1}{n}\sum_{i=1}^{n}\sqrt{x_i^2+y_i^2+z_i^2} \tag{9}$$

式中的(x_i,y_i,z_i)为团簇原子的坐标。小R值或大R值分别表示原子处于团簇的中心或表面。通过计算可知，该结构下所有的金原子几乎均位于同一球面上，几何性质十分稳定。同时还可以根据表面自由能计算看出，该结构的表面自由能最小，也说明了它的稳定性。

5.2.3 模型总结

本模型详细地说明了对Au_{32}团簇结构优化使用的研究方法。主要包括使用启发式扰动策略进行“跳坑”操作，并运用拟牛顿法($L-BFGS$)进行构型的局部优化。

第一部分介绍了结合改进策略的变方向变步长拟物算法以及拟人算法，最后构造了求解团簇优化问题的拟人算法，并将其应用到预测Au_{32}团簇基态结构问题中。

第二部分针对BH操作随机性较大的问题，给出改进的BH算法，增加了距离质心最远的原子移动到质心附近的操作步骤，大幅降低团簇的势能。

第三部分介绍局部优化模型中使用的局部优化算法——拟牛顿法($L-BFGS$)，并给出了具体的算法流程。

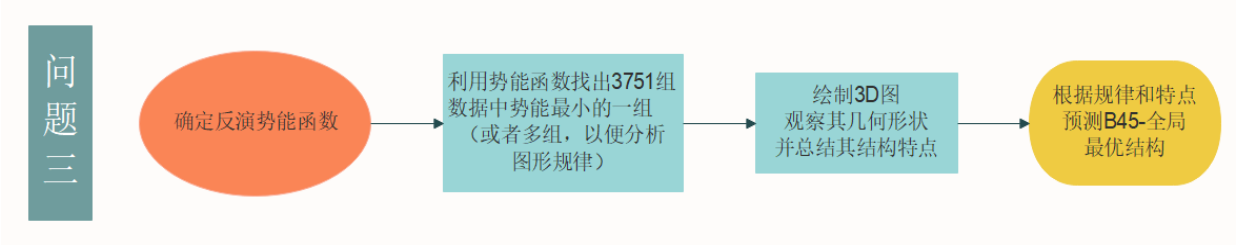
5.3 问题三模型的建立与求解

5.3.1 模型假设

假设构型的总势能是仅仅关于原子间距离的函数；

假设团簇中的各原子处于平衡状态。

5.3.2 模型建立



5.3.2.1 反演势能函数法

已知团簇内原子受力平衡，则对B_{45}^-中 45 个原子进行受力分析，对于每一个原子均有$\sum_{i=1}^{44}\vec{F_i}=\vec{0}$，由此推出原子的势能函数为：

$$V=\sum_{j=1}^{45}\sum_{i=1}^{44}P(r_{ij}) \tag{10}$$

其中，$P(r_{ij})$为多项式函数，i,j遍历构型中的每一个原子。得到的势能图如下：

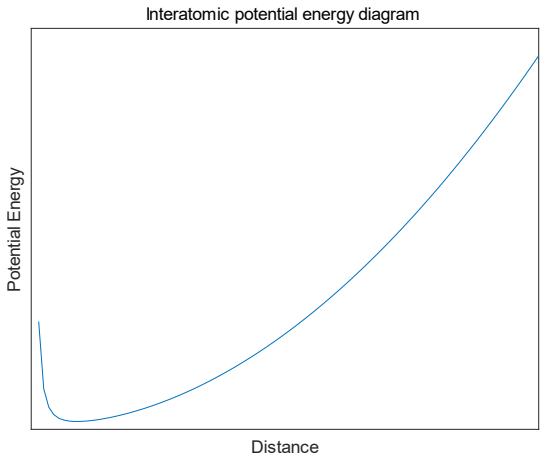


图 16 无修偏条件下的势能图

考虑到研究对象B_{45}^-为带电子体，仅仅考虑两两原子间的作用力会造成很大偏差（而上式势能函数仅考虑“对势”），在此引入多体势函数。多体势理论认为：在多原子的系统中，改变一个原子的位置，必将影响整体系统中一定范围的电子云分布，从而影响到其他原子间的相互作用。故为了确保结果的准确性，多原子体系的势函数应使用多体势表述。

总势能：$V(r)=\sum_{i<j}V_2(r_{ij})+\sum_{i\neq j,k;j<k}V_3(r_{ij},r_{ik},r_{jk})$；

二体势：$V_2(r_{ij})=\left(A_{ij}{r_{ij}}^{-p_{ij}}-B_{ij}r_{ij}^{-q_{ij}}\right)exp\left(\frac{\delta_{ij}}{r_{ij}-{a^1}_{ij}}\right)$；

三体势：$V_3(r_{ij},r_{ik},r_{jk}) = \lambda_{ijk} exp\left(\frac{\gamma_{ij}}{r_{ij} - a^2_{ij}} + \frac{\gamma_{ik}}{r_{ik} - a^2_{ik}}\right) \times \left(cos(\theta_{jik}) + \frac{1}{3}\right)^2$；

其中，θ_{jik} 是 ji 键与 ik 键之间的键角。

体势数越大，表现为图像越稳定，模拟出的结构稳定性更良好。结合实际情况，在此采用三体势函数。

在此基础上，得到改进后的总势能函数为：

$$V = \sum_{j=1}^{45} \sum_{i=1}^{44} P(r_{ij}) \left(1 - \tanh(\gamma \cdot r_{ij})\right) \tag{11}$$

其中，γ 为修偏参数。且要保证势函数具有以下性质：

性质 1：至少有一个极小点，并且在核间距 R 等于其平衡值 R_e 时，系统的势能等于离解能 D_e 的负值，即 $V(R_e) = D_e$；

性质 2：当核间距趋于无穷大时，势能曲线有正确的渐近行为，即 $\lim\limits_{R \to \infty} V(R) =$ 常量；

性质 3：当核间距趋于零时，势能趋于无穷大，即 $\lim\limits_{R \to \infty} V(R) \to \infty$；

假设有修偏系数 γ，多项式函数 $P(r_{ij})$ 的系数未知，将题给数据代入，列有相应方程，求解线性方程组得到多项式函数 $P(r_{ij})$ 的系数，从而得到势能函数。经修偏后的图像反映地性质较为清晰，如图：

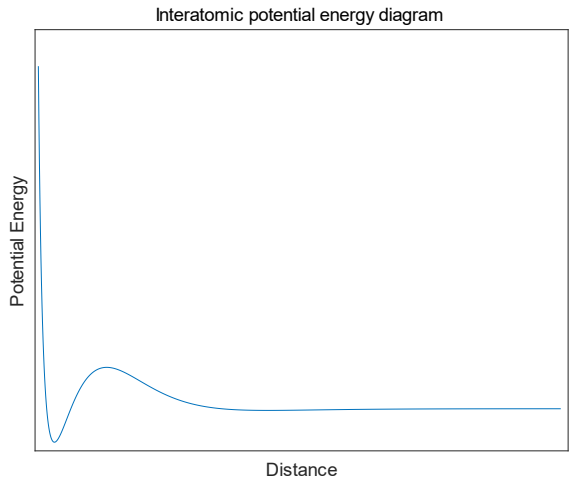


图 17 含有修偏系数的势能函数

对比两幅图，可以清晰地看到，经过修偏，势能函数趋于稳定状态，准确性大大提高，由此可见修偏对于模型的重要性。进而也验证了本模型的良好性。

5.3.3 模型求解

根据 3751 组数据，首先利用 *Python* 编程，根据上述**反演势能函数**找到能量最低的 6 组数据，进而利用 *MATLAB — Molecule Viewer* 进行可视化，得到如下结构图（依次按能量升序排列，即图 18 为能量最小的结构图）：

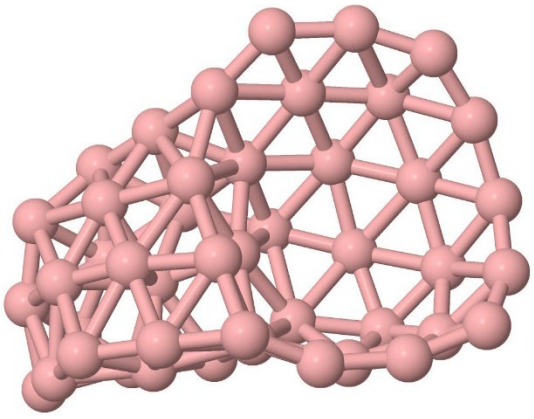


图 18 第 273 团簇结构

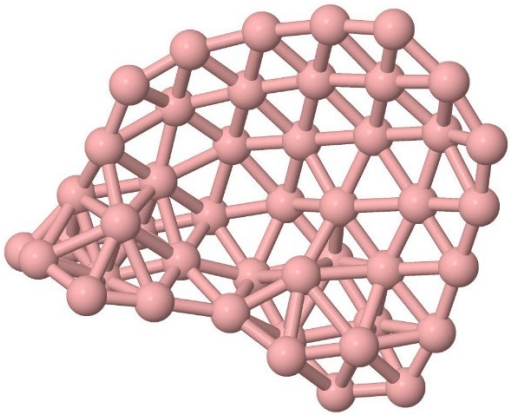


图 19 第 3488 团簇结构

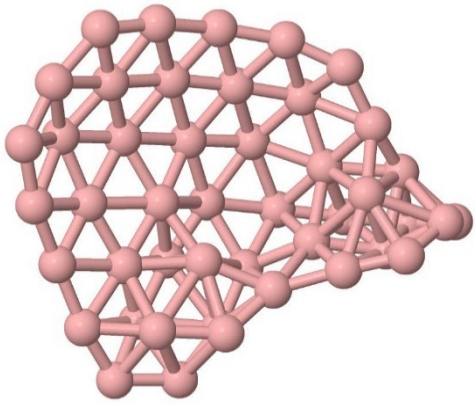


图 20 第 1644 团簇结构

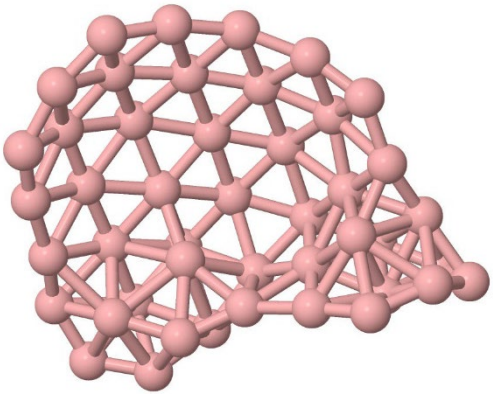


图 21 第 3470 团簇结构

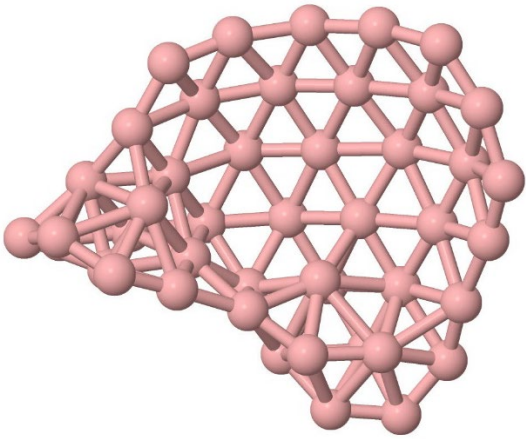


图 22 第 955 团簇结构

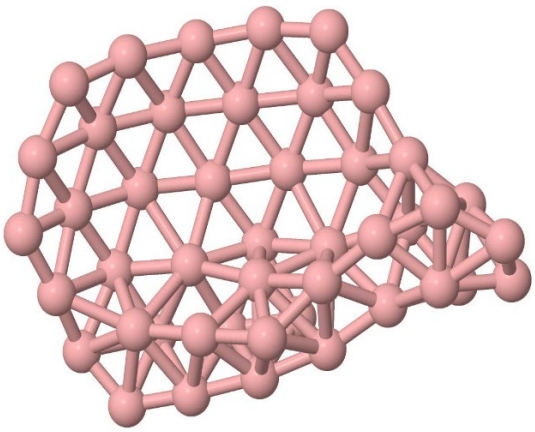


图 23 第 3368 团簇结构

可以观察到，由能量最低升序排列，能量最小的 5 个团簇（图 19-图 23）结构较为相似，且均具有一个较为集中的“中心点”，比对能量最小的团簇（图 18），发现能量越小的团簇结构，其越像一个类似中心点的“中心”靠拢。

综上所述，本文猜测对于B_{40}^-，其能量最低的结构（全局最优结构）为船形封闭结构，其具体结构如图所示：

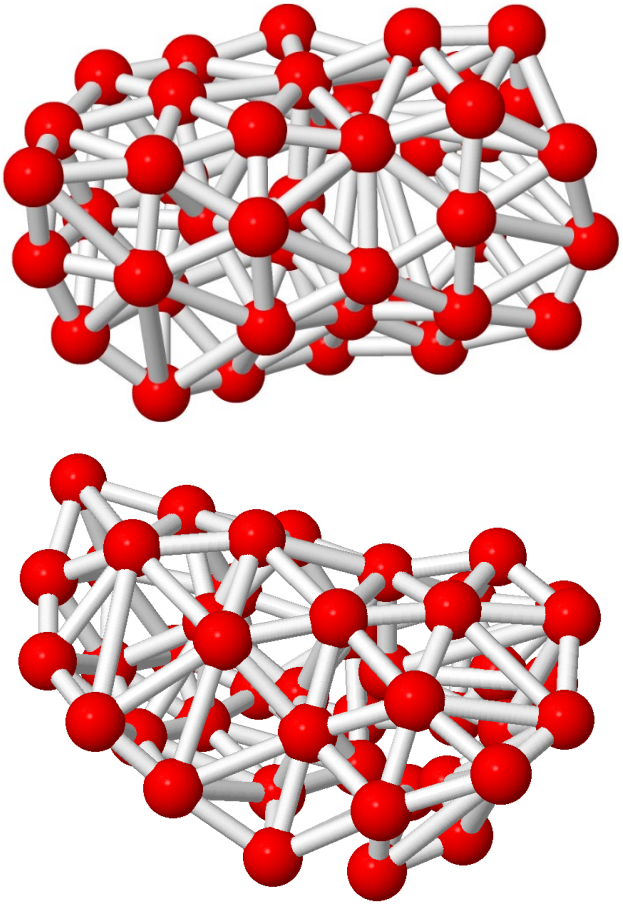


图 24 B_{40}^-的最优结构

B_{40}^-的最优结构类似于两张带有弧度的网状结构对称连结在一起构成的封闭结构。

5.3.4 模型总结

本模型基于三体势函数得到带有修偏系数γ的总势能函数。根据函数图像找到势能最低的几组构型，分析构型特点，进而预测B_{45}^-全局最优结构。

5.4 问题四模型的建立与求解

问题四基于问题三中的反演势能函数和算法，得到异构体。进而把得到的异构体当作基本数据，分析B_{40}^-异构体构型的特点，找出能量较小时构型的规律，再利用差分进化算法对B_{40}^-的团簇结构进行预测和检验。

针对以上思路，建立如下模型：

5.4.1 模型建立

差分进化法是从一个随机的解出发，按照某种规律，以一定的概率在整个求解范围内搜索最优解。其基本思想是：在D维空间中，假设每一代种群中有n个粒子，在种群中搜索方向及步长的信息，经过交叉、变异操作，产生一个临时种群。对上面存在的两个种群进行一对一的选择操作，得到下一代种群。按上述方法对种群进行不断进化，达到满意的结果未止。算法流程图如下所示：

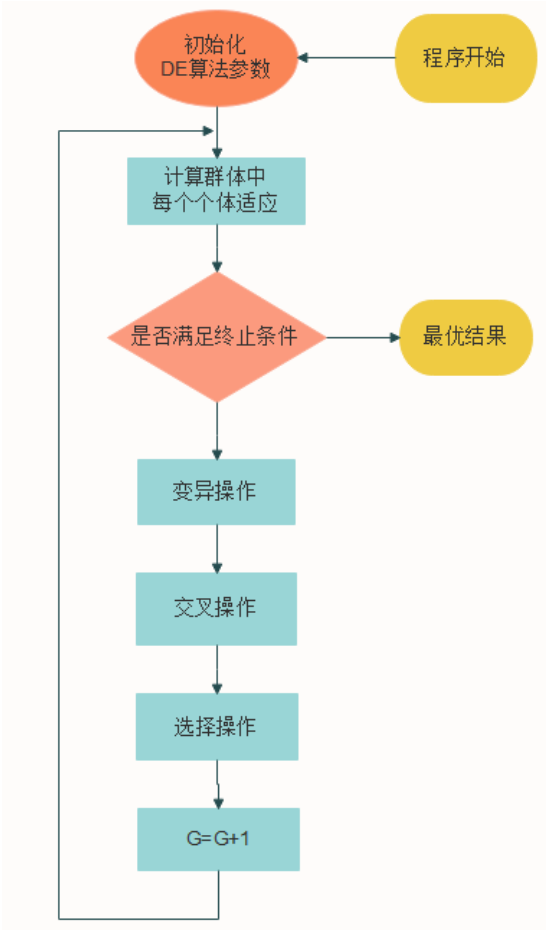


图 25 差分进化算法的流程示意图

Step1：在D维空间初始化种群，将第i个粒子的空间位置记作：$X_i^0=[x_{i,1}^0,x_{i,2}^0,...,x_{i,D}^0]$；

Step2：对每个粒子按照公式进行变异操作：$V_i^{t+1}=X_{r_1}^t+F\left(X_{r_2}^t-X_{r_3}^t\right)$，$F$为缩放因子，可以用来控制差分向量放大的程度，$F\in[0,2]$。$r_1,r_2,r_3\in\{1,2,...,M\}$，它们互不相同且与$i$也不同。

Step3：对X_i^t和变异体V_i^{t+1}按照如下公式进行交叉操作，得到个体$u_{i,j}^{t+1}$，即

$$u_{i,j}(t+1)=\begin{cases}v_{i,j}(t+1), & If(rand(j)\leqslant(CR)\ or\ j=rand(i))\\ x_{i,j}(t), & Otherwise\end{cases} \tag{12}$$

其中，CR为交叉概率常数，且$CR\in[0,1]$；$rand(j)$是均匀分布的随机数，且$rand(j)\in[0,1]$.

Step4：根据如下公式对试验个体$u_i(t+1)$和$x_i(t)$进行比较，如果优化的是最小化问题，则选择子代时就要选目标函数值较低的个体，即

$$x_i(t+1)=\begin{cases}u_i(t+1) & f(u_i(t+1))\le f(x_i(t))\\ x_i(t) & otherwise\end{cases},i=1,2,...,M \qquad (13)$$

Step5：如果输出结果满足算法终止条件，那么停止运算，否则，返回步骤中继续运算。

在差分进化算法中主要涉及三个参数：缩放因子F、群体规模M、以及交叉概率CR的设定。对于缩放因子F一般取值范围在$[0,2]$之间，通常情况下选取0.5；群体规模M一般是取介于$5\times n$与$10\times n$之间得数，但是不能少于4，否则就将无法进行变异操作；交叉概率CR一般在$[0,1]$之间选取。当CR很大时，尽管其收敛速度会加快，但是容易出现早熟现象，通过大量研究实验得出比较好的选择应在0.3左右。

5.4.2 模型求解

图 26 B_{40}^-全局优化结构

该结构具有高度对称性和较强的稳定性。

1. 几何分析：

B_{40}的该种构型具有类富勒烯结构，但是本身又结合相对紧密，相比于B_{45}有着更紧凑的结构。通过计算不同键键长，发现其键长接近定值，整体稳定性较好。

2. 成键分析：

我们利用$QTAIM$方法对B_{40}团簇进行了电子密度的拓扑分析，来揭示其成键本质。其电子密度相对均匀，体现了良好的电子结构上的稳定性。

3.动力学分析：

为了验证内嵌复合物的动力学稳定性，我们利用$DMol3$代码在$300K$时对B_{40}进行了$BO-MD$模拟。在动力学模拟过程中，硼笼表现出微小的变形，但整个结构始终保持完整。此外，我们还考查了在$900K$高温下团簇的动力学稳定性，尽管模拟过程中硼团簇整体有轻微变形，个别边缘位置上的原子有些许移动，但在$3ps$后，依然保持了原先的完整结构。显然，这种团簇构型不仅在室温下稳定，在高温下也表现出较高的稳定性。

5.4.3 模型总结

本模型采用差分进化算法优化B_{40}^-的结构，它不依赖于问题信息，通用性较强，容易与其它算法相互结合，构造出性能更优的算法，最终预测得到B_{40}^-的结构。

六、 分析与总结

6.1 模型的分析

6.1.1 模型的优点

1. 问题二模型中的改进的BH算法克服了传统算法随机性较大的缺点，使得模型结果更为稳定；

2. 问题二模型中的变方向变步长拟物算法限制了合力的大小，使得在初始阶段势能的摆动幅度不会过大，保证了结构的稳定性；另外，该模型的特点使得N个原子都进行了运动，提高了算法总体上的下降速度[4]；

3. 问题三模型中的反演势能函数法将多体势考虑在内，相比于普通的对势更具体化，对于B_{45}^-全局最优结构的预测有重要启示作用；

4. 问题四模型中的差分进化算法具有利用群体全局信息和个体局部信息指导算法进一步搜索的能力，可以很高效地应用到构型预测中，且与遗传算法、粒子群优化算法相比，差分进化算法对B团簇结构进行优化，具有更显著的收敛特性，所需的计算时间更短。

6.1.2 模型的缺点

模型计算时，时间复杂度高、空间复杂度高，代码实现困难。

问题三与四模型的研究对象B_{45}^-带电子，在建立模型时未考虑原子的电子轨道变化对结构的影响。

差分进化算法的研究成果缺乏系统性，相当分散。

6.2 启示与展望

随着物理、化学等学科的发展，寻找团簇结构的最稳态问题也逐渐成为一个重要的研究方向。研究原子团最稳态的团簇结构对于人们认识新材料、合成新型催化剂有着重要意义。但团簇结构优化问题已经证实为NP难问题，目前还没有多项式时间的精确算法可以解决此类问题。目前求解此问题主要是借助启发式算法。

本文的主要工作是运用启发式算法对SC**势能函数**描述的Au和**反演势能函数**描述的B团簇进行基态结构的预测。问题一的模型介绍了SC势能函数，并将最优结构与能量联系起来，预测出Au_{20}的全局最优结构为**正四面体状结构**；问题二的模型通过**改进的**BH**算法**，动态预测出Au_{32}的全局最优结构为类似“足球”形状，由 12 块五边形，20 块六边形组成；问题三的模型选取适当的势能模型描述团簇原子间的作用力，于是引入多体势函数，并对偏差进行了修正；问题四的模型利用**智能优化算法**中的**差分进化算法**对B原子团簇进行结构优化，预测出B_{40}^-的结构为近球形，具有高度的对称性和较强的稳定性。

结合本文所做工作，提出以下几点展望：

1. 在算法的执行过程中参数可以进行动态调整。在当前的算法中，参数往往是一个固定的值，实际上在算法不同执行环节，团簇结构也在随之动态变化，因此参数也需要随着团簇结构的不断变化而不断进行适应性的调整，甚至对于出现一定偏差的算法，可以设置修偏参数进行修正。

2. 团簇基态结构预测是一项系统而复杂的工作，本文只是研究了团簇结构预测方法中较为简单的几个问题，通过观察最优团簇结构信息进行启发操作，创造出更适用的启发式算法，并将其应用在更复杂团簇结构优化问题上，这是我们以后所要面临和解决的问题。

七、 参考文献

[1] 许如初,倪海文,黄文奇.预测 Au_(13-75)团簇基态结构的启发式算法[J].中国科学:物理学 力学 天文学,2012,42(02):134-140.

[2] 倪海文. 团簇基态结构预测的高效启发式算法[D].华中科技大学,2011.

[3] 熊荆武. 预测 Au 团簇基态结构的启发式优化算法[D].华中科技大学,2017.

[4] 姜慧. 基于智能优化算法的小型团簇结构研究[D].浙江师范大学,2012.

[5] [J]ScienceVolume 299, Issue 5608. 2003. PP 864-867.

[6] [J]Chemicals & Chemistry2018. PP 3117.

[7] 周营成. 基于群智能和机器学习的新型纳米团簇结构预测研究[D].北京化工大学,2020.

[8] 刘暾东,李泽鹏,季清爽,邵桂芳,范天娥,文玉华.基于改进 Basin-Hopping Monte Carlo 算法的 Fe_n-Pt_m(5≤n+m≤24)合金团簇结构优化[J].物理学报,2017,66(05):57-67.

[9] 田凯. 笼状硼基团簇结构及电子性质的第一性原理计算研究[D].河南大学,2007.

[10] Nanotechnology - Nanotubes; Researchers from Sichuan University Report Details of New Studies and Findings in the Area of Nanotubes (Tuning of Structure Evolution and Electronic Properties through Palladium-Doped Boron Clusters: PdB16 as a Motif for Boron-Based Nanotubes) [J]. Chemicals & Chemistry, 2020: 5788-.

[11] Kalita Amlan J et al. Double aromaticity in a BBe6H6+ cluster with a planar hexacoordinate boron structure.[J]. Chemical communications (Cambridge, England), 2020,

八、 附录

附录列表：

原子之间作用力F
变方向变步长拟物算法
BH算法
反演力函数算法
反演势能函数

1. 原子之间作用力F

```
import pandas as pd
import numpy as np
import scipy
from scipy.spatial.distance import pdist, squareform
from scipy.optimize import fmin_l_bfgs_b
import os
path = r'C:\Users\86166\Desktop\university\2021 春\mathorcup\题目\B\附件\Au20_OPT_1000'
path_list = os.listdir(path)
path_list.sort(key=lambda x:int(x.split('.')[0]))
poten_energy = []
area = []

for file in path_list:
    with open(path + "/" + file, 'r', encoding='utf8') as fp:
        F_t = 0
        n = 10
        m = 8
        c = 34.408
        a = 1
        inf, eng, x, y, z = [], [], [], [], []
        inf.append(fp.readline().strip())
        eng.append(fp.readline().strip())
        for n, j in enumerate(fp.readlines()):
            j = j.split()
            x.append(eval(j[1]))
            y.append(eval(j[2]))
            z.append(eval(j[3]))
        data = pd.DataFrame(zip(x, y, z), columns=('X', 'Y', 'Z'))
```

```
        distance = pdist(data, 'euclidean')  # 计算数组 X 样本之间的欧式距离 返回值为 Y 为压缩距离元组或矩阵
        distance = scipy.spatial.distance.squareform(distance, force='no', checks=True)
        distance = pd.DataFrame(distance)
        rij = distance.where(distance != 0)
        a_rij = a / rij
        v_r = (a_rij.where(a_rij != 'NaN')) ** n

        v_d = (a_rij.where(a_rij != 'NaN')) ** m
        v_ri = 1 / 2 * v_r.apply(lambda x: x.sum(), axis=1)
        v_di = -c * (v_d.apply(lambda x: x.sum(), axis=1)) ** 1 / 2
        # 势能函数
        Ex = (v_ri + v_di).sum()
        poten_energy.append(Ex)
        area.append(file)
        # print("{}:".format(file), Ex)
        # # 受力大小
        x_tot = 0
        y_tot = 0
        z_tot = 0
        Nlen = len(x)
        # print(Nlen)
        for i in range(Nlen):
            x_tot += x[i]
            y_tot += y[i]
            z_tot += z[i]
        tmp = [x_tot, y_tot, z_tot]
        XX_tot = np.mat(tmp)
        # print(kk)
        for i in range(Nlen):
            mxt_i = [x[i], y[i], z[i]]
            for j in range(Nlen):
                if i == j:
                    continue
                xi_xj = mxt_i - np.mat([x[j], y[j], z[j]])
                kk = 5 * a ** 10 * rij[i][j] ** (-12) - 4 * a ** 4 * c * rij[i][j] ** (-6)
            F_t += (kk * np.linalg.norm(xi_xj)) ** 2
        F_t = kk * ((F_t / Nlen) ** (1/2))
        print("{}".format(file), F_t)


# print(min(poten_energy))
# print(poten_energy.index(min(poten_energy)))
# print("文件是:", area.index(poten_energy.index(min(poten_energy))))
```

```
72.
73. #       d=0.6
74. #       w1,w2,w3=0.001,0.004,0.008
75. # miou=1/2*np.linalg.norm(x,ord=2,keepdims=False)
76. # dm1 = pdist(data, lambda u, v: ((w1*(u-v)**2).sum()))
77. # dm1=scipy.spatial.distance.squareform(dm1, force='no', checks=True)
78. # dm2 = pdist(data, lambda u, v: ((w2*(u**2-v**2)**2).sum()))
79. # dm2=scipy.spatial.distance.squareform(dm2, force='no', checks=True)
80. # dm3 = pdist(data, lambda u, v: ((w3*(u**3-v**3)**2).sum()))
81. # dm3=scipy.spatial.distance.squareform(dm3, force='no', checks=True)
82. # beta=(1/2*(dm1+dm2+dm3)-190*d**2)
83. # beta=pd.DataFrame(beta)
84. # beta=(beta.where(beta>0,0))**2
85. # beta=np.sum(np.sum(beta))
86. # E_new=Ex+beta+miou
```

2. 变方向变步长拟物算法

```
1. import pandas as pd
2. import numpy as np
3. import scipy
4. from scipy.spatial.distance import pdist, squareform
5. from scipy.optimize import fmin_l_bfgs_b
6. import os
7. poten_energy = []
8. area = []
9. step_max = 0.24
10. step_min = 0.01
11. now_step = step_max
12. eds = 1e-6
13. N = 20
14. R = (1.5053 * N ** (1.0 / 3) - 1) * 5 ** (1.0 / 6) * 2 ** (-2.0 / 3)
15. Fn = []
16. F_t = 0
17. x_n = []
18. times = 0
19. t_max = 100
20. x_ans = []
21.
22. def calEX(x_n):
23.     F_t = 0
24.     n = 10
25.     m = 8
26.     c = 34.408
```

```
    a = 1
    inf, eng, x, y, z = [], [], [], [], []
    for i in range(len(x_n)):
        x.append(x_n[i][0])
        y.append(x_n[i][1])
        z.append(x_n[i][2])
    data = pd.DataFrame(zip(x, y, z), columns=('X', 'Y', 'Z'))
    distance = pdist(data, 'euclidean')  # 计算数组 X 样本之间的欧式距离 返回值为 Y 为压缩距离元组或矩阵
    distance = scipy.spatial.distance.squareform(distance, force='no', checks=True)
    distance = pd.DataFrame(distance)
    rij = distance.where(distance != 0)
    a_rij = a / rij
    v_r = (a_rij.where(a_rij != 'NaN')) ** n

    v_d = (a_rij.where(a_rij != 'NaN')) ** m
    v_ri = 1 / 2 * v_r.apply(lambda x: x.sum(), axis=1)
    v_di = -c * (v_d.apply(lambda x: x.sum(), axis=1)) ** 1 / 2
    # 势能函数
    Ex = (v_ri + v_di).sum()
    return Ex

def calFN(x_n):
    xi_xj = 0
    F_t = 0
    n = 10
    m = 8
    c = 34.408
    a = 1
    inf, eng, x, y, z = [], [], [], [], []
    for i in range(len(x_n)):
        x.append(float(x_n[i][0]))
        y.append(float(x_n[i][1]))
        z.append(float(x_n[i][2]))
    data = pd.DataFrame(zip(x, y, z), columns=('X', 'Y', 'Z'))
    distance = pdist(data, 'euclidean')  # 计算数组 X 样本之间的欧式距离 返回值为 Y 为压缩距离元组或矩阵
    distance = scipy.spatial.distance.squareform(distance, force='no', checks=True)
    distance = pd.DataFrame(distance)
    rij = distance.where(distance != 0)
    a_rij = a / rij
    v_r = (a_rij.where(a_rij != 'NaN')) ** n

    v_d = (a_rij.where(a_rij != 'NaN')) ** m
```

```
    v_ri = 1 / 2 * v_r.apply(lambda x: x.sum(), axis=1)
    v_di = -c * (v_d.apply(lambda x: x.sum(), axis=1)) ** 1 / 2
    # 势能函数
    Ex = (v_ri + v_di).sum()
    poten_energy.append(Ex)
    # # 受力大小
    x_tot = 0
    y_tot = 0
    z_tot = 0
    Nlen = len(x)
    for i in range(Nlen):
        x_tot += x[i]
        y_tot += y[i]
        z_tot += z[i]
    tmp = [x_tot, y_tot, z_tot]
    XX_tot = np.mat(tmp)
    for i in range(Nlen):
        kk = 0
        mxt_i = [x[i], y[i], z[i]]
        for j in range(Nlen):
            if i == j:
                continue
            xi_xj = mxt_i - np.mat([x[j], y[j], z[j]])
            kk = 5 * a ** 10 * rij[i][j] ** (-12) - 4 * a ** 4 * c * rij[i][j] ** (-6)
        Fn.append (kk * np.linalg.norm(xi_xj))
        F_t += (kk * np.linalg.norm(xi_xj)) ** 2
    F_t = (F_t / N) ** (1 / 2)
    return Fn, F_t

def FN(Fi, pos, f_n):
    for i in range(len(Fn)):
        if i == pos:
            f_n[i] = Fi
        else:
            f_n[i] = 1

# def step2(F_tot, t):
#     if F_tot < eds | t > t_max:
#         return -1
#     pre = x_n
#     U_0 = calEX(x_n)
#     for i in range(len(x_n)):
#         x_n[i] += (1 - 1.0 / (t + 2)) * now_step * f_n[i]
#     FN(Fn[t], t + 1)
```

```
#     Fn = calFN(x_n)
#     U_1 = calEX(x_n)
#     if U_1 < U_0:
#         U_0 = U_1
#         x_ans = x_n
#     if (now_step > step_min & U_1 >= U_0):
#         now_step *= 0.99
#         return step2(F_t, t + 1)

fp = open(r'C:\Users\86166\Desktop\mathor_B\坐标\randP.txt')
inf, eng = [], []
inf.append(fp.readline().strip())
eng.append(fp.readline().strip())
for n, j in enumerate(fp.readlines()):
    j = j.split()
    x_n.append(j)
Fn, F_tot = calFN(x_n)
f_n = []
x_ans = x_n
t = 1
FN(Fn[1], 1, f_n)
while F_tot < eds | t > t_max:
    U_0 = calEX(x_n)
    for i in range(len(x_n)):
        x_n[i] += (1 - 1.0 / (t + 2)) * now_step * f_n[i]
    f_n = FN(Fn[t], t, f_n)
    Fn, F_tot= calFN(x_n)
    U_1 = calEX(x_n)
    if U_1 < U_0:
        U_0 = U_1
        x_ans = x_n
    if (now_step > step_min & U_1 >= U_0):
        now_step *= 0.99
    t = t + 1

for i in range(len(x_n)):
    for j in range(len(x_n[0])):
        print(x_n[i][j] + " ")
    print("\n")
# step2(f_n, 1)
```

2. 反演力函数算法

```
1.  load B0
```

```
2.
3.  CellB=cell([45,44]); % 原始 cell
4.  DistB=zeros([45,44]); % 距离
5.  ArrayB=zeros([135,44]); % 将 CellB 展开
6.
7.  % 计算距离和生成 CellB
8.  for i = 1:45
9.      for j = 1:45
10.         if j < i
11.             CellB{i,j}=B0(j,:)-B0(i,:);
12.             DistB(i,j)=sqrt((B0(j,1)-B0(i,1))^2+(B0(j,2)-B0(i,2))^2+(B0(j,3)-B0(i,3))^2);
13.         end
14.         if j > i
15.             CellB{i,j-1}=B0(j,:)-B0(i,:);
16.             DistB(i,j-1)=sqrt((B0(j,1)-B0(i,1))^2+(B0(j,2)-B0(i,2))^2+(B0(j,3)-B0(i,3))^2);
17.         end
18.     end
19. end
20.
21. % 将 CellB 展开
22. for i = 0:44
23.     for k = 1:44
24.         ArrayB(3*i+1:3*i+3,k)=CellB{i+1,k}';
25.     end
26. end
27.
28. F=linspace(1,-134,136); % 多项式的系数，按次幂排列
29. Sum=zeros([136,1]);
30. AllB=zeros([135,136]); % 最终系数矩阵
31.
32. % 计算系数矩阵
33. for i = 0:44
34.     for k = 1:3
35.         for j = 1:44
36.             Sum=Sum+ArrayB(3*i+k,j)*(DistB(i+1,j).^F)';
37.         end
38.         AllB(3*i+k,:)=Sum';
39.         Sum=zeros([136,1]);
40.     end
41. end
42.
43. Zero=zeros([135,1]);
44. One=ones([135,1]);
45.
```

```
% 求解
Solve=AllB(:,2)\[AllB(:,1),AllB(:,3:136)];

% % 绘制图像
x=linspace(0.022,1.5,100);
y=zeros([1,100]);
S=0;
for i = 1:100
    for j = 1:5 % 取前 j 项
        S=S+Solve(1,j)*x(1,i)^(F(1,j)+1);
    end
    y(1,i)=S;
    S=0;
end
figure(2)
plot(x,y/100)
set(gca,'xtick',[],'xticklabel',[])
set(gca,'ytick',[],'yticklabel',[])
title('Interatomic potential energy diagram')
xlabel('Distance')
ylabel('Potential Energy')

%
% syms x f;
%
% str=num2str(Solve(1,2));
% for i = 1:135
%     if i ~= 2
%         str = [str,num2str(Solve(1,i)),'*x^',num2str(F(1,i))];
%     end
% end
% f = str2sym(str);
% fi = int(f,x);
```

3. 反演势能函数

```
load Energy
load BCoord

gamma=0.5; % 修偏系数
Long=50; % 数据长度

F=linspace(2,-Long+2,Long); % 多项式的系数，按次幂排列
Sum=zeros([Long,1]);
```

```
AllB=zeros([Long,Long]); % 最终系数矩阵

for k = 1:Long
    Sum=zeros([Long,1]);
    CellB=cell([45,44]); % 原始 cell
    DistB=zeros([45,44]); % 距离

    % 计算距离和生成 CellB
    for i = 1:45
        for j = 1:45
            if j < i
                CellB{i,j}=BCoord{1,k}(j,:)-BCoord{1,k}(i,:);
                DistB(i,j)=sqrt((BCoord{1,k}(j,1)-BCoord{1,k}(i,1))^2+(BCoord{1,k}(j,2)-BCoord{1,k}(i,2))^2+(BCoord{1,k}(j,3)-BCoord{1,k}(i,3))^2);
            end
            if j > i
                CellB{i,j-1}=BCoord{1,k}(j,:)-BCoord{1,k}(i,:);
                DistB(i,j-1)=sqrt((BCoord{1,k}(j,1)-BCoord{1,k}(i,1))^2+(BCoord{1,k}(j,2)-BCoord{1,k}(i,2))^2+(BCoord{1,k}(j,3)-BCoord{1,k}(i,3))^2);
            end
        end
    end

    % 计算系数矩阵
    for i = 1:45
        for j = 1:44
            Sum=Sum+(1-tanh(gamma*DistB(i,j)))*(DistB(i,j).^F)';
        end
    end

    CellB=[];
    DistB=[];
    AllB(k,:)=Sum';
    Sum=zeros([Long,1]);
end

% % 绘制图像
% Solve=Energy(1:Long,1)\AllB;
% % 绘制图像
% x=linspace(0.2,10,1000);
% y=zeros([1,1000]);
% S=0;
% for i = 1:1000
%     for j = 1:5 % 取前 j 项
```

```
%         S=S+Solve(1,j)*(1-tanh(gamma*x(1,i)))*(x(1,i)^F(1,j));
%     end
%     y(1,i)=S;
%     S=0;
% end
% figure(1)
% plot(x,-y)


Solve=Energy(1:Long,1)\AllB;
% 绘制图像
x=linspace(0.2,10,1000);
y=zeros([1,1000]);
S=0;
for i = 1:1000
    for j = 1:5 % 取前 j 项
        S=S+Solve(1,j)*(1-tanh(gamma*x(1,i)))*(x(1,i)^F(1,j));
    end
    y(1,i)=(-S-0.005)*exp(-x(1,i));
    S=0;
end
figure(1)
plot(x,y)
set(gca,'xtick',[],'xticklabel',[])
set(gca,'ytick',[],'yticklabel',[])
title('Interatomic potential energy diagram')
xlabel('Distance')
ylabel('Potential Energy')
%
% y=zeros([1,100]);
% x=linspace(1,100,100);
% for k = 1:100
%     P=0;
%     SumDist=0;
%     CellB=cell([45,44]); % 原始 cell
%     DistB=zeros([45,44]); % 距离
%
%     % 计算距离和生成 CellB
%     for i = 1:45
%         for j = 1:45
%             if j < i
%                 CellB{i,j}=BCoord{1,k}(j,:)-BCoord{1,k}(i,:);
%                 DistB(i,j)=sqrt((BCoord{1,k}(j,1)-BCoord{1,k}(i,1))^2+(BCoord{1,k}(j,2)-
BCoord{1,k}(i,2))^2+(BCoord{1,k}(j,3)-BCoord{1,k}(i,3))^2);
```

```
94. %            end
95. %            if j > i
96. %                CellB{i,j-1}=BCoord{1,k}(j,:)-BCoord{1,k}(i,:);
97. %                DistB(i,j-1)=sqrt((BCoord{1,k}(j,1)-BCoord{1,k}(i,1))^2+(BCoord{1,k}(j,2)-
    BCoord{1,k}(i,2))^2+(BCoord{1,k}(j,3)-BCoord{1,k}(i,3))^2);
98. %            end
99. %        end
100.    %    end
101.    %
102.    %    for i = 1:45
103.    %        for j = 1:44
104.    %            for n = 1:8
105.    %                P=P+Solve(1,n)*DistB(i,j)^F(1,n);
106.    %            end
107.    %            SumDist=SumDist+(1-tanh(gamma*DistB(i,j)))*P;
108.    %            P=0;
109.    %        end
110.    %    end
111.    %    y(1,k)=SumDist;
112.    %    SumDist=0;
113.    % end
114.    % figure(2)
115.    % scatter(x,y,1)
```