

**A PENALTY FUNCTION METHOD FOR
CONSTRAINED MOLECULAR DYNAMICS SIMULATION**

By

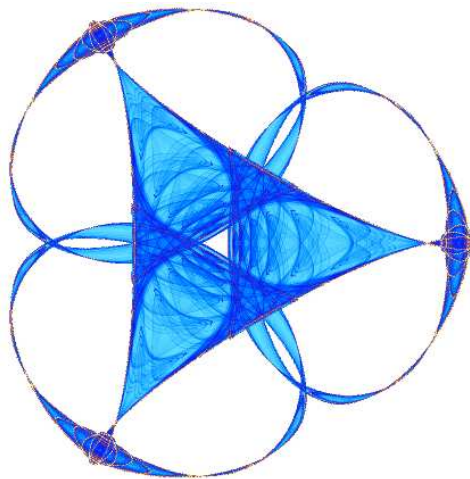
Ajith Gunaratne

and

Zhijun Wu

IMA Preprint Series # 2187

(January 2008)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA
400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455-0436

Phone: 612-624-6066 Fax: 612-626-7370

URL: <http://www.ima.umn.edu>

A PENALTY FUNCTION METHOD FOR CONSTRAINED MOLECULAR DYNAMICS SIMULATION

AJITH GUNARATNE AND ZHIJUN WU

Abstract. We propose a penalty-function method for constrained molecular dynamics simulation by defining a quadratic penalty function for the constraints. The simulation with such a method can be done by using a conventional, unconstrained solver only with the penalty parameter increased in an appropriate manner as the simulation proceeds. More specifically, we scale the constraints with their force constants when forming the penalty terms. The resulting force function can then be viewed as a smooth continuation of the original force field as the penalty parameter increases. The penalty function method is easy to implement and costs less than a Lagrange multiplier method, which requires the solution of a nonlinear system of equations in every time step. We have first implemented a penalty function method in CHARMM and applied it to protein Bovine Pancreatic Trypsin Inhibitor (BPTI). We compared the simulation results with Verlet and Shake, and found that the penalty function method had high correlations with Shake and outperformed Verlet. In particular, the RMSD fluctuations of backbone and non-backbone atoms and the velocity auto correlations of C_α atoms of the protein calculated by the penalty function method agreed well with those by Shake. We have also tested the method on a group of argon clusters constrained with a set of interatomic distances in their global energy minimum states. The results showed that the method was able to impose the constraints effectively and the clusters tended to converge to their energy minima more rapidly than not confined by the constraints.

Key Words. Constrained molecular dynamics, Verlet algorithm, Shake algorithm, Lagrange multipliers method, penalty function method.

1. Introduction

Molecular dynamics simulation can be used to study many different dynamic properties of proteins, but a long sequence of iterations has to be carried out even for small protein motions due to the small time step ($1.0\text{e-}15\text{sec}$) required [23]. The bonding forces are among those causing fast protein vibrations that require small time steps to integrate, but they may be replaced by a set of bond length constraints, to increase the step size and hence the simulation speed [12]. Several Lagrange multiplier types of methods have been developed for constrained molecular dynamics simulation. However, in all these methods, the multipliers have to be determined in every time step by solving a nonlinear system of equations so

Received by the editors in 2007 and, in revised form, in 2008.

2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.

This research was supported partially by the NIH/NIGMS grant R01GM081680 and the Institute of Mathematics and Its Applications with funds provided by the NSF of America.

that the new iterate can satisfy the constraints [3]. Depending on the number of constraints, the additional computational cost can be large, given the fact that the force field calculation in every time step is at most $O(n^2)$, while the solution of the nonlinear system of equations may require $O(m^3)$, where n is the number of particles in the system and m the number of constraints.

In this paper, we propose a so-called penalty function method [18] for constrained molecular dynamics. In this method, a special function is defined so that the function is minimized if the constraints are satisfied. By adding such a function in the potential energy function, the constraints can then be removed from the system, and the simulation can be carried out in a conventional, unconstrained manner. The advantage of using a penalty function method is that it is easy to implement, and does not require solving a nonlinear system of equations in every time step. The disadvantage of the method is that the penalty parameter, i.e., the parameter used to scale the penalty function, is hard to control and in principle, needs to be large enough for the penalty function to be truly effective, which on the other hand, may cause numerical instabilities when used in simulation [10]. It may also arguably be a disadvantage that the penalty function method only forces the constraints to be satisfied approximately but not completely. In any case, the method may possibly be used as an alternatively and computationally more efficient approach for constrained molecular dynamics simulation than the Lagrange multiplier types of methods.

We have first implemented a penalty function method in CHARMM [7] and tested it on protein Bovine Pancreatic Trypsin Inhibitor (BPTI) by following a similar experiment done by Gunsteren and Karplus in [12] for the Shake algorithm [22]. In this implementation, we removed the bond length potentials from the potential energy function and introduced the corresponding bond length constraints. For each of the bond length constraints, we constructed a quadratic penalty function and inserted it into the potential energy function. For each different type of bond, we also scaled the corresponding penalty function with the force constant of the bond so that the resulting function had the same form as the original bond length potential if without multiplied by the penalty parameter. In this way, the resulting force field becomes simply a continuation of the original force field as the penalty parameter changes continuously from 1 to a value > 1 . We conducted a simulation on BPTI with the penalty function method, and compared the results with Verlet and Shake, and found that the penalty function method had a high correlation with the Shake and outperformed the Verlet. In particular, the root-mean-square-deviations (RMSD) of the backbone and non-backbone atoms and the velocity auto correlations of the C_α atoms of the protein calculated by the penalty function method agreed well with those by Shake. Note again that the penalty function method requires no more than just applying a conventional, unconstrained simulation algorithm such as the Verlet algorithm to the potential energy function expanded with additional penalty terms for the bond length constraints.

We have also tested the penalty function method on a group of argon clusters with the equilibrium distances for a selected set of molecular pairs as the constraints. Here by the equilibrium distances we mean the distances for the pairs of argon molecules when the clusters are in their global energy minimal states. We generated these distances by using the global energy minimal configuration of the clusters published in previous studies [19]. A penalty function was constructed for each of the constraints and incorporated into the potential energy function of the cluster. The simulation was then conducted by using a conventional, unconstrained

simulation method, i.e., the Verlet algorithm [26], with the extended potential energy function. Here, there were no substantial algorithmic changes or computational overheads required due to the addition of the constraints. The simulation results showed that the penalty function method was able to impose the constraints effectively and the clusters tended to converge to their lowest energy equilibrium states more rapidly than not confined by the constraints.

We introduce the theory and the methods for constrained molecular dynamics simulation in Section 2 and describe the penalty function method in Section 3. In Section 4, we present the results on BPTI and their comparisons with the Verlet and the Shake. In Section 5, we present the results of using the penalty function method on argon cluster simulation. We conclude the paper in Section 6.

2. Constrained Molecular Dynamics Simulation

Based on the theory of classical mechanics, the trajectory of molecular motion between two molecular states minimizes the total action of the motion [15]. Let $x(t)$ be the configuration of the molecule at time t , $x = \{x_i : x_i = (x_{i,1}, x_{i,2}, x_{i,3})^T, i = 1, \dots, n\}$, where x_i is the position vector of atom i and n the total number of atoms in the molecule. Given beginning and ending time t_0 and t_e , $x(t)$ in $[t_0, t_e]$ defines a trajectory connecting two molecular states $x_0 = x(t_0)$ and $x_e = x(t_e)$. Let $L(x, x', t)$ be the difference of the kinetic and potential energy of the molecule at time t . The functional L is called the Lagrangian of the molecule. Let S be the action of the molecule in $[t_0, t_e]$. Then, S is defined as the integral of the Lagrangian in $[t_0, t_e]$, and according to the least action principle [15], the trajectory x minimizes the action S of the molecular motion in $[t_0, t_e]$,

$$(2.1) \quad \min \left[S(x) = \int_{t_0}^{t_e} L(x, x', t) dt \right]$$

Theorem 2.1. *Let L be a continuously differentiable functional. Let x be a solution of problem (2.1). Then, x satisfies the following Euler-Lagrange Equation,*

$$(2.2) \quad \frac{\partial L(x, x', t)}{\partial x'} - \frac{d}{dt} \left[\frac{\partial L(x, x', t)}{\partial x} \right] = 0$$

Proof: Let δx be a small variation of x and $\delta x(t_0) = \delta x(t_e) = 0$. By the principle of variation, the necessary condition for x to be a solution of problem (2.1) is that,

$$(2.3) \quad \delta S = \int_{t_0}^{t_e} \left(\frac{\partial L(x, x', t)}{\partial x} \delta x + \frac{\partial L(x, x', t)}{\partial x'} \delta x' \right) dt = 0$$

Since $\delta x' = \delta \left(\frac{dx}{dt} \right) = d \left(\frac{\delta x}{dt} \right)$, we obtain, after integrating the second term of (2.3) by parts,

$$(2.4) \quad \delta S = \int_{t_0}^{t_e} \left(\frac{\partial L(x, x', t)}{\partial x} - \frac{d}{dt} \left[\frac{\partial L(x, x', t)}{\partial x'} \right] \right) \delta x dt = 0$$

Since δS should be zero for all δx , the integrand of (2.4) must be zero and (2.2) follows. \square

Corollary 2.2. *Let $L = \frac{x'^T M x'}{2} - E(x)$, where M is the mass matrix of a molecule and E the potential energy. Then, a necessary condition for x to minimize an action S is that,*

$$(2.5) \quad Mx'' = -\nabla E(x)$$

Proof: It follows from Theorem 2.1 and the facts that $\frac{d}{dt} \frac{\partial L}{\partial x'} = Mx''$ and $\frac{\partial L}{\partial x} = -\nabla E$. \square

Equation (2.5) is well known as the equation of motion for a molecule of n atoms. It can be equivalently stated as,

$$(2.6) \quad m_i x_i'' = f_i(x_1, \dots, x_n), \quad f_i = -\frac{\partial E}{\partial x_i}, \quad i = 1, \dots, n$$

where m_i and f_i are the mass and force for atom i , respectively and $M = \text{diag}[m_1, \dots, m_n]$.

Note that Theorem 2.1 and Corollary 2.2 imply that a trajectory that minimizes the molecular action between two system states necessarily satisfies the classical mechanical equation of motion. In other words, the solution of the equation of motion can be considered as an attempt for the minimization of the molecular action of motion.

Verlet [26] developed an algorithm, now called the Verlet algorithm, for numerically integrating the equations in (2.6), started with a given set of initial positions and velocities for the atoms. There are two versions of the Verlet algorithm, the position Verlet and the velocity Verlet [25], as given in formulas (2.7) and (2.8), respectively:

Position Verlet:

$$(2.7) \quad \begin{aligned} x_i^{k+1} &= 2x_i^k - x_i^{k-1} + h^2 f_i^k / m_i, \\ i &= 1, \dots, n, \quad k = 1, 2, \dots \end{aligned}$$

Velocity Verlet:

$$(2.8) \quad \begin{aligned} x_i^{k+1} &= x_i^k + h v_i^k + h^2 f_i^k / (2m_i), \\ v_i^{k+1} &= v_i^k + h(f_i^k + f_i^{k+1}) / (2m_i), \\ i &= 1, \dots, n, \quad k = 1, 2, \dots \end{aligned}$$

Being symplectic, the velocity Verlet preserves the energy and volume of the molecular system and exhibits superior numerical stability for long time simulation [2]. However, the simulation has to be carried out with a small time step (in order of 1.0e-15sec) to keep up with all rapid atomic level movements. The potential of simulating molecular motions in longer time scales beyond pico- or nano-seconds has therefore been limited. For proteins, the bonding forces are believed among those responsible for fast protein vibrations that require small time steps to integrate. Therefore, one of the approaches to increase the step size and hence the simulation speed is to remove the bonding forces from the force field while enforcing them through a set of bond length constraints. The simulation can then be done by integrating the constrained equation of motion with larger time steps [12].

Let $g = \{g_j : j = 1, \dots, m\}$ be a vector of functions that can be used to define the constraints on the molecule. The constrained simulation problem can then be

considered as a constrained least action problem.

$$(2.9) \quad \begin{aligned} \min [S(x) &= \int_{t_0}^{t_e} L(x, x', t) dt] \\ \text{subject to } g(x) &= 0 \end{aligned}$$

Then, by the theory of constrained optimization, a necessary condition for a molecular trajectory x between x_0 and x_e to be a solution of problem (2.9) is that,

$$(2.10) \quad \begin{aligned} \delta S(x) + \sum_{j=1}^m \lambda_j \delta g_j(x) &= 0 \\ g(x) &= 0 \end{aligned}$$

where λ_j is a vector of Lagrange multipliers [5].

Theorem 2.3. *Let $L = \frac{x'^T M x'}{2} - E(x)$, where M is the mass matrix of a molecule and E the potential energy. Then, a necessary condition for x to minimize an action S subject to $g(x) = 0$ is that,*

$$(2.11) \quad \begin{aligned} Mx'' &= -\nabla E(x) - G(x)^T \lambda, \\ g(x) &= 0 \end{aligned}$$

where λ is a vector of Lagrange multipliers and $G(x)$ the Jacobian of $g(x)$.

Proof: For $L = \frac{x'^T M x'}{2} - E(x)$, condition (2.10) translates to,

$$(2.12) \quad \begin{aligned} Mx'' &= -\nabla E(x) - \sum_{j=1}^m \lambda_j \nabla g_j(x), \\ g(x) &= 0 \end{aligned}$$

and hence to (2.11) with $G = [\nabla g_1, \nabla g_2, \dots, \nabla g_m]^T$. \square

For each atom, equation (2.11) can be written as,

$$(2.13) \quad \begin{aligned} m_i x_i'' &= f_i(x_1, \dots, x_n) + \sum_{j=1}^m \lambda_j g_{ji}(x_1, \dots, x_n), \\ 0 &= g_j(x_1, \dots, x_n) \\ j &= 1, \dots, m, \quad i = 1, \dots, n. \end{aligned}$$

where

$$(2.14) \quad f_i = -\frac{\partial E}{\partial x_i}, \quad g_{ji} = -\frac{\partial g_j}{\partial x_i}, \quad j = 1, \dots, m, \quad i = 1, \dots, n.$$

Note that in (2.13), the right-hand side of the first equation can be treated as a single force function (with the original force function plus a combination of the derivatives of the constraint functions), and therefore, the equation can be integrated in the same way as equation (2.6) by the Verlet algorithm, except that in every step, the Lagrange multipliers λ_j , $j = 1, \dots, m$, have to be determined so that the new positions x_i , $i = 1, \dots, n$, for the atoms satisfy the constraints $g_j(x_1, \dots, x_n) = 0$, $j = 1, \dots, m$. Indeed, several algorithms have been developed along this line including the Shake [22] and Rattle [1] algorithms, corresponding to

the position and velocity Verlet algorithms for unconstrained simulation, respectively.

Shake:

$$\begin{aligned}
 (2.15) \quad x_i^{k+1} &= 2x_i^k - x_i^{k-1} + h^2(f_i^k + \sum_{j=1}^m \lambda_j^k g_{ji}^k)/m_i, \\
 0 &= g_j(x_1^{k+1}, \dots, x_n^{k+1}) \\
 j &= 1, \dots, m, \quad i = 1, \dots, n \quad k = 1, 2, \dots
 \end{aligned}$$

Rattle:

$$\begin{aligned}
 (2.16) \quad x_i^{k+1} &= x_i^k + hv_i^k + h^2(f_i^k + \sum_{j=1}^m \lambda_j^k g_{ji}^k)/(2m_i), \\
 v_i^{k+1} &= v_i^k + h(f_i^k + \sum_{j=1}^m \lambda_j^k g_{ji}^k + f_i^{k+1} + \sum_{j=1}^m \lambda_j^{k+1} g_{ji}^{k+1})/(2m_i), \\
 0 &= g_j(x_1^{k+1}, \dots, x_n^{k+1}) \\
 j &= 1, \dots, m, \quad i = 1, \dots, n \quad k = 1, 2, \dots
 \end{aligned}$$

In both Shake and Rattle, a nonlinear system of equations needs to be solved in every step so that the new iterates can satisfy the constraints. A Gauss-Seidel method has been used in the algorithms for the solution of the nonlinear system of equations [20]. Barth et al [3] developed an SOR (Successive Over-Relaxation) method to improve the performance of the Gauss-Seidel approach, and also tested a Newton-type method. The Gauss-Seidel method runs fast for each iteration but converges slow, while the Newton-type method converges quickly but requires expensive computation in each iteration step. As a result, all the methods are somehow equivalently costly, and in the worst case, take $O(m^3)$ floating point operations in every time step of the simulation, where m is the number of constraints [3]. Note that simulation without constraints requires at most $O(n^2)$ floating point operations per time step, where n is the number of atoms. When the number of constraints is comparable to the number of atoms, the computational overhead for constrained dynamics will certainly be more than significant.

3. The Penalty Function Method

The penalty function method has a long history of being used as an alternative approach to constrained optimization. The idea is simply to combine the objective function and the constraints so that the objective function and the constraint violation are both minimized when the combination is minimized. The solution of the original constrained optimization problem can then be achieved by solving an unconstrained one [18]. As described in previous section, constrained molecular dynamics simulation is essentially seeking the solution to a constrained optimization problem - a constrained least action problem. Therefore, the problem may be approached by a penalty function method as well.

Courant [8] first proposed a penalty function method using the squared Euclidean norm of the constraint violations as the penalty term. Fletcher [11] studied the theoretical basis for a class of exact penalty function methods for the solution of equality constrained optimization problems. The exact penalty function methods were further investigated by Di Pillo [9]. The idea of Fletcher of defining a class of smooth penalty functions and that of Courant of using a quadratic penalty term were also combined [21]. Variants of the penalty function methods also include using so-called barrier functions to prevent optimization from becoming infeasible by setting an infinitely large barrier around the border of the feasible region [24].

Let f be the objective function and $g = \{g_j : j = 1, \dots, m\}$ be a set of constraint functions. Consider a general equality constrained optimization problem,

$$(3.1) \quad \begin{aligned} \min \quad & f(x_1, \dots, x_n) \\ \text{subject to} \quad & g_j(x_1, \dots, x_n) = 0 \quad j = 1, \dots, m. \end{aligned}$$

The unconstrained optimization problem with a quadratic penalty function for (3.1) can be defined as follows,

$$(3.2) \quad \min \quad f(x_1, \dots, x_n) + \mu \sum_{j=1}^m |g_j(x_1, \dots, x_n)|^2$$

where μ is a parameter called the penalty parameter. In principle, the solution for problem (3.1) can be recovered by solving problem (3.2) with the parameter μ gradually increased to infinity.

A so-called exact penalty function can also be defined, such as using the l_1 -norm. Then, problem (3.2) becomes,

$$(3.3) \quad \min \quad f(x_1, \dots, x_n) + \mu \sum_{j=1}^m |g_j(x_1, \dots, x_n)|$$

and the solution for problem (3.1) can be recovered by solving problem (3.3) with the parameter μ only raised to a sufficiently large value.

If the constraints are inequalities, i.e., $g_j(x_1, \dots, x_n) \geq 0$, $j = 1, \dots, m$, the penalty functions in (3.2) and (3.3) can still be used in the same way as for equality constraints, only with g_j replaced by g_j^- for all j , where $g_j^- = \min\{g_j, 0\}$ gives the amount of violation for constraint j . Another approach is to introduce a barrier function for each constraint. Then, the problem becomes minimizing the combination of the objective function and the barrier functions such as the following,

$$(3.4) \quad \min \quad f(x_1, \dots, x_n) - r \sum_{j=1}^m \log [g_j(x_1, \dots, x_n)]$$

where $\log[g_j(x_1, \dots, x_n)]$ is called the log barrier function for $g_j(x_1, \dots, x_n)$ as the function is not defined when $g_j(x_1, \dots, x_n) < 0$ and is infinity when $g_j(x_1, \dots, x_n) = 0$. The parameter r is used to control the barrier term. In principle, the solution of the original constrained optimization problem can be asymptotically approached by solving problem (3.4) as r is gradually decreased to zero.

In this work, we will only use the formulation in (3.2) for the development of the penalty function method for constrained molecular dynamics simulation. The primary reasons are that in this work, we only consider the equality constraints, and the squared Euclidean norm used in (3.2) also provides smoother properties

than the l_1 -norm in (3.3) for optimization. By using the formulation in (3.2), the constrained least action problem as given in (2.9) can be transformed to,

$$(3.5) \quad \min S(x) + \mu \frac{\|g(x)\|^2}{2}$$

where $\|\cdot\|$ is the Euclidean norm and $g = (g_1, \dots, g_m)^T$. In principle, a solution for the constrained least action problem (2.9) can be obtained by solving a sequence of problems in (3.5) with μ selected from an increasing sequence of parameters $\{\mu^k\}$.

Theorem 3.1. *Let $\mu = \mu^k$ and $\mu^k \uparrow \infty$ as $k \rightarrow \infty$. Let x^k be a global solution to (3.5) with $\mu = \mu^k$, and $x^k \rightarrow x^*$ as $k \rightarrow \infty$. Then, $g(x^k) \rightarrow 0$ as $x^k \rightarrow x^*$, and x^* is a global solution to the constrained least action problem (2.9).*

Proof: Let $\phi(x, \mu) = S(x) + \frac{\mu\|g(x)\|^2}{2}$. Then,

$$(3.6) \quad \phi(x^k, \mu^k) \leq \phi(x^{k+1}, \mu^k) \leq \phi(x^{k+1}, \mu^{k+1})$$

showing that the sequence of global minima $\{\phi(x^k, \mu^k)\}$ of (3.5) is non-decreasing. By using the facts that $\phi(x^k, \mu^k) \leq \phi(x^{k+1}, \mu^k)$ and $\phi(x^{k+1}, \mu^{k+1}) \leq \phi(x^k, \mu^{k+1})$, we have,

$$(3.7) \quad \phi(x^{k+1}, \mu^{k+1}) - \phi(x^{k+1}, \mu^k) \leq \phi(x^k, \mu^{k+1}) - \phi(x^k, \mu^k)$$

and

$$(3.8) \quad (\mu^{k+1} - \mu^k) (\|g(x^k)\|^2 - \|g(x^{k+1})\|^2) \geq 0$$

It follows that $\{\|g(x^k)\|^2\}$ is non-increasing. Since $\phi(x^k, \mu^k) \leq \phi(x^{k+1}, \mu^k)$, $\{S(x^k)\}$ is also non-decreasing.

Let S^* be the global minimum of (2.9). Then, $\phi(x^k, \mu^k) \leq \phi(x, \mu^k) = S^*$, where $x = \text{global argmin}\{S(x) : g(x) = 0\}$. Then,

$$(3.9) \quad S(x^k) + \mu^k \|g(x^k)\|^2 \leq S^*$$

Since $\{S(x^k)\}$ is non-decreasing and $\{\mu^k\}$ is increasing, $g(x^k) \rightarrow 0$, and it follows that if $x^k \rightarrow x^*$, $g(x^*) = 0$ and $S(x^*) \leq S^*$. By the definition of S^* , $S(x^*) \geq S^*$, and therefore, $S(x^*) = S^*$. \square

We now define an extended Lagrangian

$$(3.10) \quad \tilde{L}(x, x', t) = L(x, x', t) + \mu \|g(x)\|^2 / (t_e - t_0) / 2$$

Then, problem (3.5) can be written in the following form,

$$(3.11) \quad \min \tilde{S}(x) = \int_{t_0}^{t_e} \tilde{L}(x, x', t) dt$$

By applying Theorem 2.1 and Corollary 2.2 to (3.11), we obtain the extended equation of motion as the necessary condition for any x to be a solution to problem (3.5),

$$(3.12) \quad Mx'' = -\nabla E(x) - \mu G(x)^T g(x)$$

where G is the Jacobian of g . The following theorem shows that a solution to problem (2.9) that satisfies the necessary condition (2.11) for the problem can be obtained by solving the extended equation of motion (3.12) with μ increasing to infinity. The solution is equivalent to the one that can be obtained by using a Lagrange multiplier type method.

Theorem 3.2. *Let $\mu = \mu^k$ and $\mu^k \uparrow \infty$ as $k \rightarrow \infty$. Let x^k be a solution to problem (3.5) with $\mu = \mu^k$, and $x^k \rightarrow x^*$ as $k \rightarrow \infty$. Let G be the Jacobian of g and $G(x^*)$ be of full rank. Then, x^* satisfies the necessary condition (2.11) for x^* to be a solution to the constrained least action problem (2.9).*

Proof: Based on (3.12), for each pair of (x^k, μ^k) , necessarily,

$$(3.13) \quad M[x^k]'' = -\nabla E(x^k) - \mu^k G(x^k)^T g(x^k)$$

Let $\lambda^k = \mu^k g(x^k)$. Then,

$$(3.14) \quad M[x^k]'' = -\nabla E(x^k) - G(x^k)^T \lambda^k$$

and

$$(3.15) \quad \begin{aligned} \lambda^k &= -[G(x^k)^T]^+ [M[x^k]'' + \nabla E(x^k)] \\ &\rightarrow -[G(x^*)^T]^+ [M[x^*]'' + \nabla E(x^*)] = \lambda^* \end{aligned}$$

where $[G(x^k)^T]^+$ is the pseudo-inverse of $G(x^k)^T$. Then, $g(x^k) = \frac{\lambda^k}{\mu^k} \rightarrow \frac{\lambda^*}{\mu^k} \rightarrow 0$.

It follows that,

$$(3.16) \quad \begin{aligned} M[x^*]'' &= -\nabla E(x^*) - G(x^*)^T \lambda^* \\ g(x^*) &= 0 \end{aligned}$$

□

In the atomic form, equation (3.12) can be written as,

$$(3.17) \quad \begin{aligned} m_i x_i'' &= f_i(x_1, \dots, x_n) + \mu \sum_{j=1}^m g_{ji}(x_1, \dots, x_n) g_j(x_1, \dots, x_n) \\ f_i &= -\frac{\partial E}{\partial x_i}, \quad g_{ji} = -\frac{\partial g_j}{\partial x_i}, \quad i = 1, \dots, n \end{aligned}$$

By treating the entire right-hand side of each equation in (3.17) as a force function, we can then apply standard Verlet algorithms to obtain our numerical formulas for the solution of the equations in (3.17):

Penalty Position Verlet:

$$(3.18) \quad \begin{aligned} x_i^{k+1} &= 2x_i^k - x_i^{k-1} + h^2(f_i^k + \mu \sum_{j=1}^m g_{ji}^k g_j^k)/(m_i), \\ i &= 1, \dots, n, \quad k = 1, 2, \dots \end{aligned}$$

Penalty Velocity Verlet:

$$(3.19) \quad \begin{aligned} x_i^{k+1} &= x_i^k + h v_i^k + h^2(f_i^k + \mu \sum_{j=1}^m g_{ji}^k g_j^k)/(2m_i), \\ v_i^{k+1} &= v_i^k + h(f_i^k + f_i^{k+1} + \mu \sum_{j=1}^m (g_{ji}^{k+1} g_j^k + g_{ji}^{k+1} g_j^{k+1}))/m_i, \\ i &= 1, \dots, n, \quad k = 1, 2, \dots \end{aligned}$$

Note that formulas (3.18) and (3.19) do not involve solving nonlinear systems and can therefore be updated much more efficiently than Shake and Rattle. However,

the parameter μ needs to be selected appropriately. It is required to be sufficiently large. There is also an issue that for different penalty terms, different scales may need to be used for the parameters. We discuss these issues in greater details in the specific implementations of the algorithms in the following sections.

4. Results on Bovine Pancreatic Trypsin Inhibitor

We have implemented a penalty function algorithm in molecular dynamics simulation software CHARMM developed by Brooks, et al [7] and tested it for simulation of protein BPTI (bovine pancreatic trypsin inhibitor). The simulation was performed with three different schemes, one with Verlet (VL) algorithm, one with Shake (SH), and one with Penalty (PL). The bond lengths were used to generate the constraints for the protein in SH and PL runs. There were no external solvent molecules included except for four water molecules in specified spots in the protein. The protein is chosen for this study because it has been well studied and widely used as a test case for various simulations [12], [13], [16], [17]. To compare the methods and determine if they sample approximately the same part of phase space, various statistical properties were analyzed, including the average fluctuations and correlation functions for various physical quantities.

The CHARMM potential energy function is defined as follows.

$$\begin{aligned}
 (4.1) \quad E = & \sum_{Bonds} k_b(b - b_0)^2 + \sum_{Angles} k_\theta(\theta - \theta_0)^2 \\
 & + \sum_{Dihedrals} k_\phi(1 + \cos(n\phi - \delta)) + \sum_{Improper} k_\omega(\omega - \omega_0)^2 \\
 & + \sum_{Urey-Bradley} k_u(u - u_0)^2 \\
 & + \sum_{Non-bonded} \varepsilon_{i,j} \left[\left(\frac{R_{i,j}^{min}}{r_{i,j}} \right)^{12} - \left(\frac{R_{i,j}^{min}}{r_{i,j}} \right)^6 \right] + \frac{q_i q_j}{\varepsilon r_{i,j}}
 \end{aligned}$$

There are several versions of the CHARMM force field. We used CHARMM22 (released in 1991). The first term in the energy function accounts for the bond stretches where k_b is the bond force constant and $(b - b_0)$ is the distance from equilibrium that the atoms have moved. The second term in the equation accounts for the bond angles where k_θ is the angle force constant and $(\theta - \theta_0)$ is the angle from equilibrium between 3 bonded atoms. The third term is for the dihedral angles where k_ϕ is the dihedral force constant and n is the multiplicity of the function, ϕ is the dihedral angle and δ is the phase shift. The fourth term accounts for the improper angles, that are out of plane bending, where k_ω is the force constant and $(\omega - \omega_0)$ is the out of plane angle. The Urey-Bradley component comprises the fifth term, where k_u is the respective force constant and u is the distance between the 1,3 atoms in the harmonic potential. Non-bonded interactions between (i, j) pairs of atoms are represented by the last two terms. By definition, the non-bonded forces are only applied to atom pairs separated by at least three bonds. The van Der Waals energy is calculated with a standard 12-6 Lennard-Jones potential and the electrostatic energy with a Coulomb potential. In the Lennard-Jones potential above, the R^{min} term is not the minimum of the potential, but rather where the Lennard-Jones potential crosses the x-axis.

The protein BPTI has 58 amino acid residues [4]. It consists of 454 atoms. With the four water molecule, the total number of atoms included in the simulation was

458 (without counting the hydrogen atoms). When the bond-length constraints were applied, the bond stretching potential terms were omitted from the potential energy function, and the bond lengths except for the hydrogen bonds were kept fixed by the constraints. Note that the constraint for the bond-length between a bonded pair of atoms i and j can be formed by using the equality constraint

$$(4.2) \quad r_{i,j} - d_{i,j} = 0, \quad \text{with} \quad r_{i,j} = \|x_i - x_j\| \quad \text{or}$$

$$(4.3) \quad r_{i,j}^2 - d_{i,j}^2 = 0, \quad \text{with} \quad r_{i,j} = \|x_i - x_j\|,$$

where $d_{i,j}$ are the equilibrium distance between atoms i and j . The equation (4.3) is smoother than (4.2) and has been used in Shake because the Shake algorithm requires solving such equations in every time step and it is important to keep the equations differentiable. However, in the penalty function method, both (4.2) and (4.3) can be used since the derivatives of the equations are only used for force field calculations in reasonably defined domains. The equation (4.2), when used to form a penalty term, is actually more stable than (4.3), because the penalty is amplified (squared) in (4.3). For this reason, we used (4.2) in the implementation of the penalty function method in CHARMM. The penalized energy function becomes the following,

$$(4.4) \quad E = \mu \sum_{\text{Bonds}} k_{i,j} (r_{i,j} - d_{i,j})^2 + \sum_{\text{Angles}} k_{\theta} (\theta - \theta_0)^2 \\ + \sum_{\text{Dihedrals}} k_{\phi} (1 + \cos(n\phi - \delta)) + \sum_{\text{Impropers}} k_{\omega} (\omega - \omega_0)^2 \\ + \sum_{\text{Urey-Bradley}} k_u (u - u_0)^2 \\ + \sum_{\text{Non-bonded}} \varepsilon_{i,j} \left[\left(\frac{R_{i,j}^{\text{min}}}{r_{i,j}} \right)^{12} - \left(\frac{R_{i,j}^{\text{min}}}{r_{i,j}} \right)^6 \right] + \frac{q_i q_j}{\varepsilon r_{i,j}}$$

where the original bond-length energy (the first term) is replaced by a penalty function for the bond-length constraints. Note that the penalty term for each bond-length is multiplied by a constant $k_{i,j}$. The term can then be scaled by using an appropriate value for $k_{i,j}$. In our implementation, we simply used the corresponding force constant for each $k_{i,j}$. Coincidentally, the penalized energy function then becomes exactly the original energy function when $\mu = 1$ and is a continuation from the original energy function for any $\mu > 1$. Figure 1 shows the flow chart of the simulation program using the penalty function method. It is the same as an unconstrained simulation program except that a penalized energy function (4.4) is used and the penalty parameter μ in the function can be adjusted to control the effects of the bonds on the simulation. Note that in our implementation, the penalty parameter was changed gradually from value (0.7) less than 1 to a value (1.7) beyond 1 during the simulation.

We implemented the penalty function method (PL for short) using the velocity Verlet in CHARMM and compared the simulation results on BPTI with the original velocity Verlet (VL for short) and Shake (SH for short). A time step $h = 1.0e - 3ps$ was used. Results with $h = 2.0e - 3ps$ and other larger time steps were also collected for VL runs. For SH and PL, the relative tolerance for constraint satisfaction was set to $1.0e - 5$. Of course, the accuracy of the simulation depended not only on the constraint tolerance but also the step size.

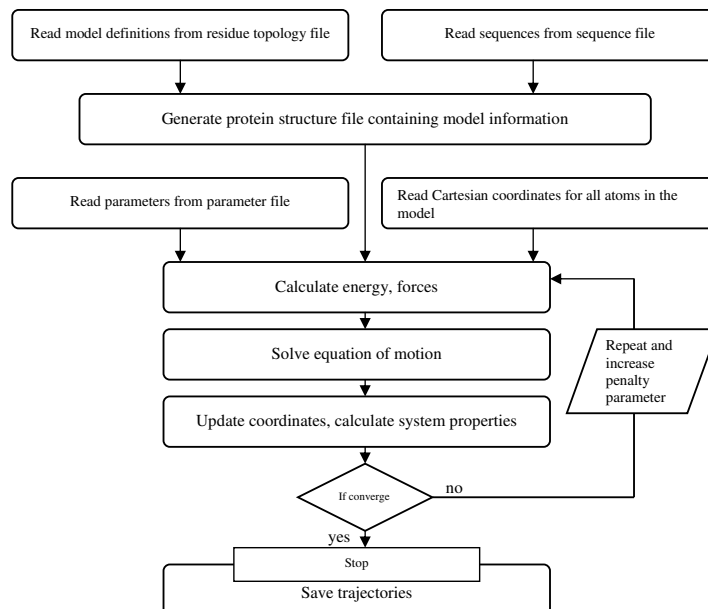


FIGURE 1. The flow chart of the simulation program with the penalty function method. The procedure is the same as in unconstrained simulation, but a penalized energy function is used, with an adjustable penalty parameter to control the effects of the bonds.

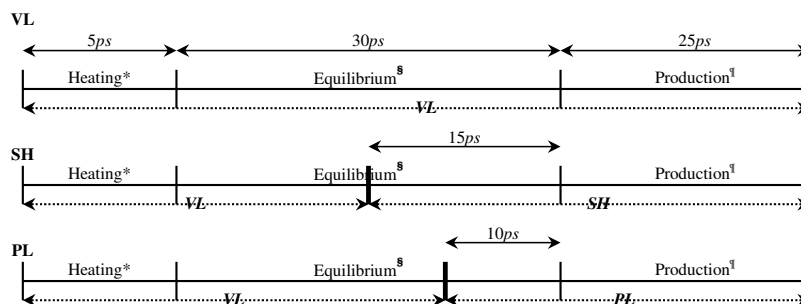


FIGURE 2. Simulation time for VL, SH and PL. *Heating - bring the system to normal temperature; Equilibrium - the time for the system to reach the equilibrium; Production - stable dynamic results for analysis.

The initial BPTI coordinate file was downloaded from PDB Data Bank [4], (<http://www.rcsb.org/pdb/>) which contained 454 atoms and 60 water molecules (without hydrogen atoms). Out of 60 water molecules, 4 inside the protein were carefully selected and put into the system. The hydrogen atoms were then added to form the starting structure for simulation.

The steepest descent method was first used to minimize the energy. This was performed to eliminate the strain present in the starting structure. At the beginning the energy for the starting structure was $44906.75 \text{ kcal/mol}$. Total 2999 energy minimization steps were conducted and the energy was minimized to $-1137.49 \text{ kcal/mol}$.

TABLE 1. Computing time for VL, SH and PL

| Scheme | Computing time* |
|--------|-----------------|
| VL | 1.14 hours |
| SH | 1.25 hours |
| PL | 1.14 hours |

*Computing time for the 25ps simulation after equilibrium.

The energy minimization process took 11.97 minutes elapsed time and 3.28 minute CPU time on an Alpha 500 Mhz workstation.

A minimized structure represents the molecule at a temperature close to absolute zero 3.42K. Therefore, the system requires to be brought back to a normal temperature. Heating was accomplished by initially assigning random velocities to the atoms according to a Gaussian distribution appropriate for that low temperature and then running dynamics simulation with VL. The temperature was then increased gradually by assigning greater random velocities to the atoms at every 0.05ps from 3.42K to 300K. The entire heating process took 5000 simulation steps with 0.001ps time step, which corresponded to total 5ps simulation time (Figure 2). When simulation started, the temperature rose rapidly. The conversion of kinetic energy to potential energy was fast. However, the increase in temperature slowed down when the system aged.

The total simulation time for VL was 55ps, but only final 25ps were used for analysis. We observed that BPTI reached the equilibrium in first 30ps (Figure 2). Usually, this period is long for protein, because of the high connectivity of the covalently bonded system and the long range of electrostatic potentials.

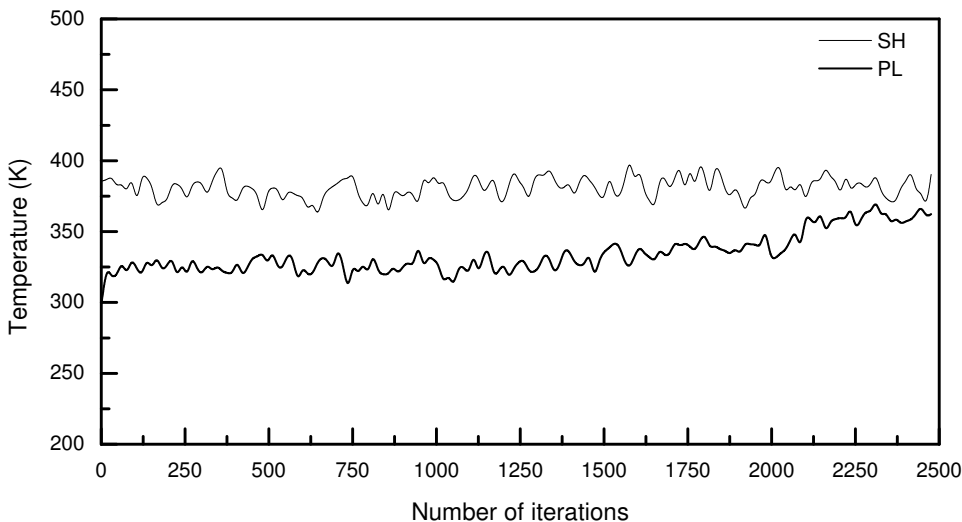


FIGURE 3. The variation of temperature (K) of BPTI in the 25ps production simulations by SH and PL. PL started the same temperature as VL but gradually approached to that for SH as the penalty parameter was increased.

To achieve the equilibrium state for SH and PL, we first performed 15ps and 20ps simulations with VL and then started SH and PL with initial positions and

velocities taken from the final step of VL respectively (Figure 2). We then also ran SH and PL for 25ps for analysis (Figure 2). The computing time for each simulation is presented in Table 1. It showed that VL, SH and PL required 2.44, 3.00 and 2.44 minutes of computing time per picosecond simulation on an Alpha workstation. We recorded the coordinates of the trajectories every 0.01ps. The results in the final 25ps of the simulations were used to obtain the dynamical and statistical properties of the system.

Figure 3 shows the temperature distribution in the 25ps simulation by SH and PL. The variation of the temperature showed that there was a difference between SH and PL at the beginning of the simulation. The temperature for PL started at 300K, the same as VL, but gradually increased and eventually approached to that for SH. This indicated that the simulation by PL started with a condition similar to that by VL but then changed to SH later when the penalty parameter is fully adjusted to an appropriate value.

The average backbone root mean square (RMS) fluctuations are plotted as a function of residue number in Figure 4. The graphs show a great correlation between the fluctuations by SH and PL. On the other hand, VL simulation produced large fluctuations for 10 TYR, 13 PRO, 15 LYS, 27 ALA, 45 PHE and 47 SER residues, which were disagreed with those by SH and PL.

The root mean square fluctuations of C_α atoms in the simulations are plotted in Figure 5. Similar to the average backbone root mean square fluctuations, the C_α fluctuations by PL and SH again had strong correlations. The average root mean square fluctuations of HN and the non-backbone atoms by SH and PL correlated as well as shown in Figure 6 and 7 except for some discrepancies around residues 54 to 58.

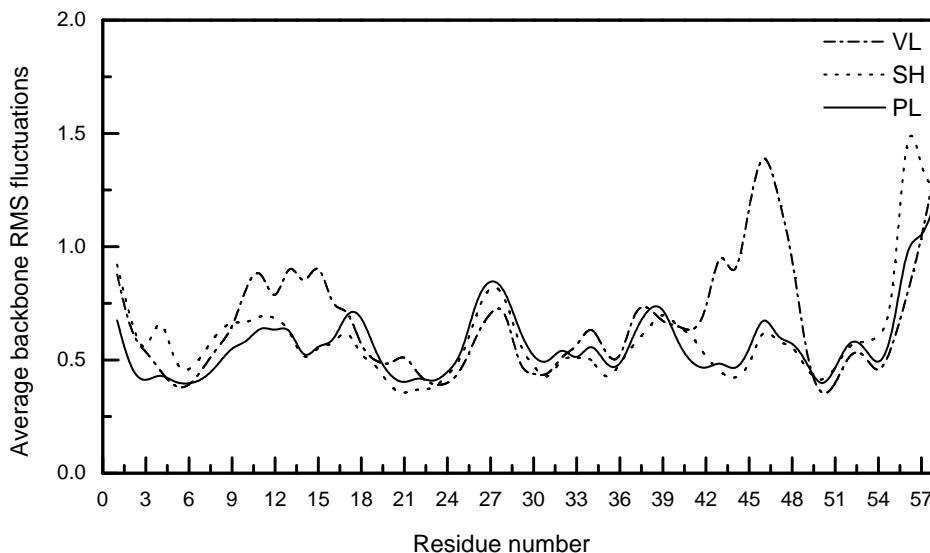


FIGURE 4. The average backbone RMS fluctuations of the residues in the 25ps production simulations.

Figure 8 shows the normalized velocity auto correlations calculated for 51 CYS C_α using the trajectories produced by VL, SH, and PL. For the demonstration purpose, the correlations over a 10ps time period are shown. The first curve is for VL run with an auto correlation time equal to 0.01ps. The auto correlation time for

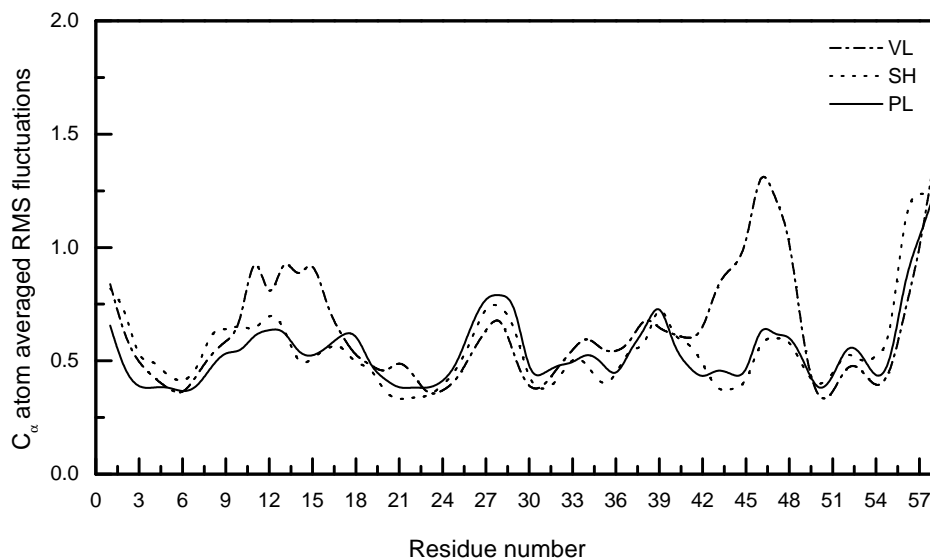


FIGURE 5. The average C_α RMS fluctuations in the $25ps$ production simulations.

the second curve is $0.02ps$ and is half the resolution of the first one. The third and fourth curves are for SH and PL runs, respectively, both with the auto correlation time equal to $0.01ps$. The curves for SH and PL showed similar correlations with that for VL in $0.02ps$ resolution, suggesting that both SH and PL are roughly faster than VL by two folds.

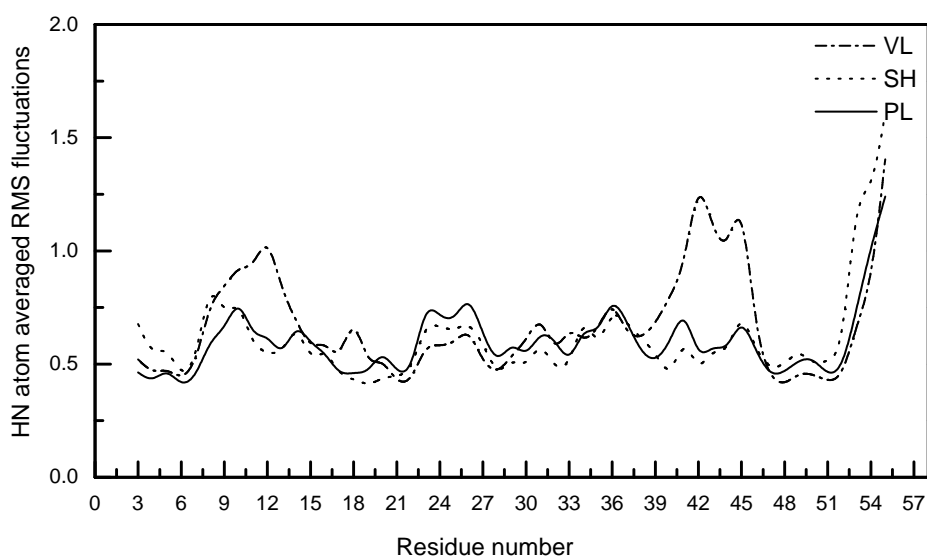


FIGURE 6. The average RMS fluctuations of the HN atoms in the $25ps$ production simulations.

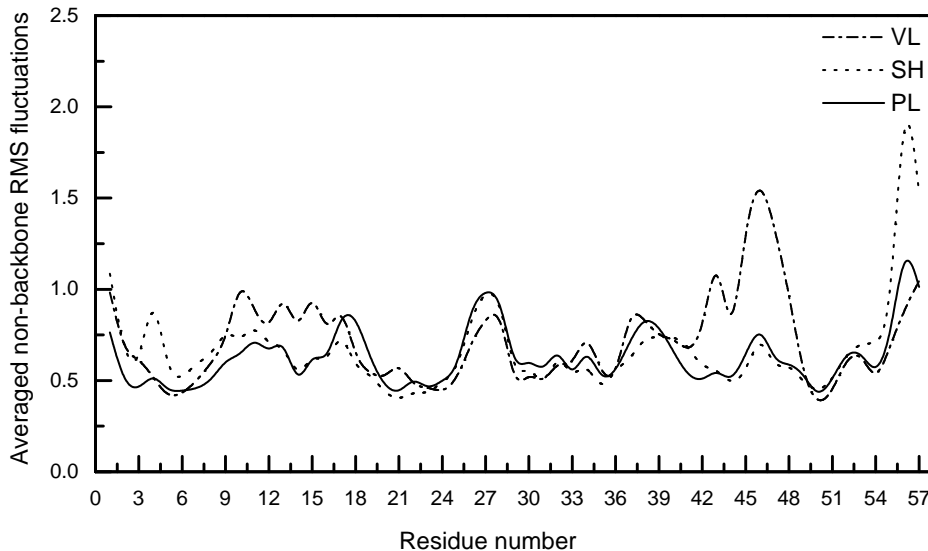


FIGURE 7. The average RMS fluctuations of the non-backbone atoms in the 25ps production simulations.

5. Results on Argon Clusters

The constrained molecular dynamics simulation can be used not only for reducing the fast vibrations due to the bonding forces, but also as a general scheme to incorporate any types of conformational constraints into the simulation so that the simulation can be guided towards preferred directions. For example, in simulation of protein folding, the prior knowledge on the folded structures such as bond lengths and bond angles, contact distances, or NMR distances can often be included as constraints in the simulation.

We have also tested the penalty function method on handling more general distance constraints. As a test case, we have selected a group of argon clusters and applied the penalty function method to the simulation of the equilibrium of the clusters. Argon clusters have been well studied in chemistry and material sciences, and been used as model systems for molecular dynamics simulation and energy minimization [28]. An argon cluster is formed by a set of argon molecules with van der Waals interactions among them. The van der Waals forces can be approximated by using so-called Lennard-Jones functions. More specifically, let x_i and x_j be the position vectors for two argon molecules, i and j , then their interacting potential can be defined by the following Lennard-Jones function.

$$(5.1) \quad h(r_{i,j}) = 4\epsilon \left[\left(\frac{\sigma}{r_{i,j}} \right)^{12} - \left(\frac{\sigma}{r_{i,j}} \right)^6 \right]$$

where $r_{i,j} = \|x_i - x_j\|$ is the distance between molecule i and j , $\epsilon \approx 165.4e-23$ J the minimum value of the potential, and $\sigma \approx 3.405$ the distance for the potential to achieve the minimum. By using such a potential function, the total potential energy $E(x)$ for a cluster of n argon molecules with a configuration $x = (x_1, \dots, x_n)^T$ can be calculated by the formula,

$$(5.2) \quad E(x) = \sum_{i=1}^n \sum_{j=i+1}^n h(r_{i,j}) = \sum_{i=1}^m \sum_{j=i+1}^n 4\epsilon \left[\left(\frac{\sigma}{r_{i,j}} \right)^{12} - \left(\frac{\sigma}{r_{i,j}} \right)^6 \right]$$

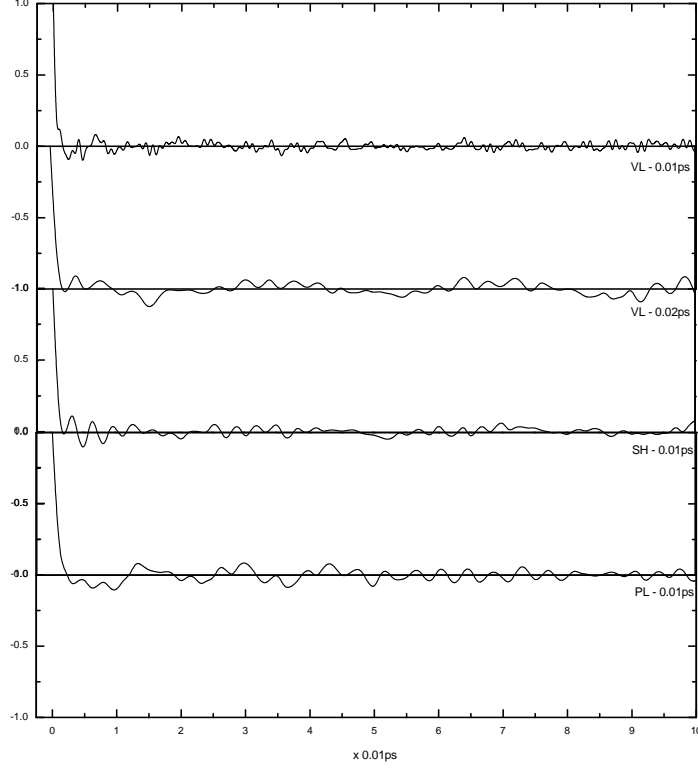


FIGURE 8. The velocity auto correlations of the C_α atom of 51 CYS based on the trajectories produced by VL, SH, and PL in a time period of $0.1ps$.

and the force $f_i(x)$ on molecule i by

$$\begin{aligned}
 (5.3) \quad f_i(x) &= \sum_{j=1, j \neq i}^n \frac{\partial h(r_{i,j})}{\partial x_i} \\
 &= \sum_{j=1, j \neq i}^n 48\epsilon\sigma^{-2} \left[\left(\frac{\sigma}{r_{i,j}} \right)^{14} - 0.5 \left(\frac{\sigma}{r_{i,j}} \right)^8 \right] (x_i - x_j)
 \end{aligned}$$

By choosing appropriate units for the physical quantities (σ for length, 4ϵ for energy, $48\epsilon\sigma^{-2}m$ for mass, where m is the mass for argon molecule), the above formulas can be simplified to,

$$(5.4) \quad E(x) = \sum_{i=1}^n \sum_{j=i+1}^n h(r_{i,j}) = \sum_{i=1}^m \sum_{j=i+1}^n \left[\left(\frac{1}{r_{i,j}} \right)^{12} - \left(\frac{1}{r_{i,j}} \right)^6 \right]$$

and the force $f_i(x)$ on molecule i by

$$\begin{aligned}
 (5.5) \quad f_i(x) &= \sum_{j=1, j \neq i}^n \frac{\partial h(r_{i,j})}{\partial x_i} \\
 &= \sum_{j=1, j \neq i}^n m \left[\left(\frac{1}{r_{i,j}} \right)^{14} - 0.5 \left(\frac{1}{r_{i,j}} \right)^8 \right] (x_i - x_j)
 \end{aligned}$$

and the equation of motion for the cluster can be written in the following form.

$$(5.6) \quad \frac{d^2 x_i}{dt^2} = \sum_{j=1, j \neq i}^n \left[\left(\frac{1}{r_{i,j}} \right)^{14} - 0.5 \left(\frac{1}{r_{i,j}} \right)^8 \right] (x_i - x_j), \quad i = 1, \dots, n$$

where the time unit is $0.299ps$. The equations in (5.6) can be integrated using the Verlet algorithm with a step size equal to 0.032 as shown in [26].

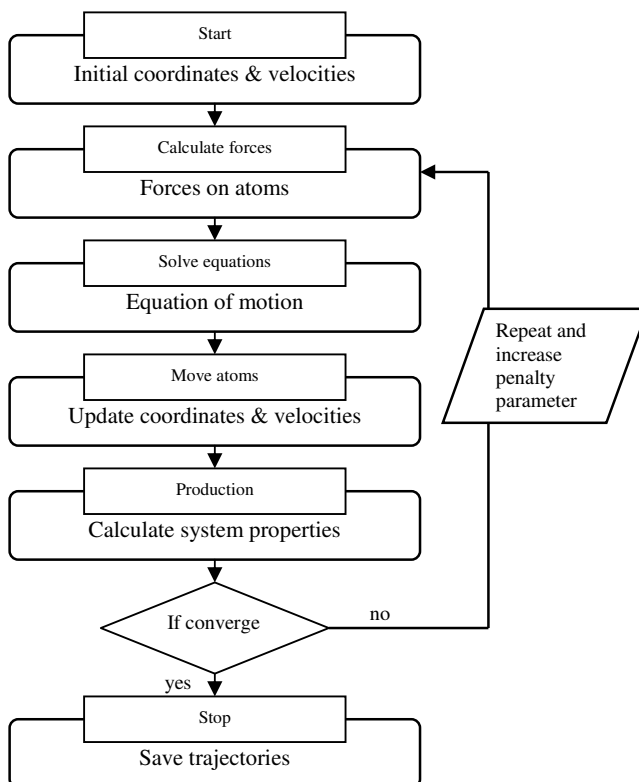


FIGURE 9. The flow chart of the penalty function algorithm for cluster simulation.

We considered a group of argon clusters with up to 147 molecules and performed the dynamics simulation for the systems. For each cluster, we started with a structure nearby the global energy minimum of the cluster. We first applied the position Verlet (VL) to the cluster. We then generated a set of distance constraints from the global energy minimum configuration of the cluster and applied the penalty position Verlet (PL) to the cluster with the generated constraints. Both trajectories produced by VL and PL were recorded and compared, and their equilibrium states were analyzed. Note that it is well-known that the global energy minimum for an argon cluster is very difficult to find. So far, the global energy minimum was found only for a small group of clusters (of less than 147 molecules [19], [29]). The starting structures and the distance constraints were then generated from these configurations.

In general, any number of distance constraints can be applied to the system, but in our implementation, we used a randomly selected fraction of total number of distances in the global energy minimum configuration as the constraints. Different

from the implementation of the penalty function method in CHARMM, here we used the following constraint function for each of the distances.

$$(5.7) \quad r_{i,j}^2 - d_{i,j}^2 = 0, \quad \text{with } r_{i,j} = \|x_i - x_j\|,$$

where $d_{i,j}$ is the distance between atoms i and j in the global energy minimum configuration of the cluster. The penalized energy and force functions can then be formulated as,

$$(5.8) \quad E(x) = \mu \sum_{(i,j) \in S} (r_{i,j}^2 - d_{i,j}^2)^2 + \sum_{i=1}^m \sum_{j=i+1}^n \left[\left(\frac{1}{r_{i,j}} \right)^{12} - \left(\frac{1}{r_{i,j}} \right)^6 \right]$$

and

$$(5.9) \quad f_i(x) = \mu \sum_{(i,j)/(j,i) \in S} 4m (r_{i,j}^2 - d_{i,j}^2) (x_i - x_j) \\ + \sum_{j=1, j \neq i}^n m \left[\left(\frac{1}{r_{i,j}} \right)^{14} - 0.5 \left(\frac{1}{r_{i,j}} \right)^8 \right] (x_i - x_j), \quad i = 1, \dots, n$$

and the equation of motion becomes,

$$(5.10) \quad \frac{d^2 x_i}{dt^2} = \mu \sum_{(i,j)/(j,i) \in S} 4 (r_{i,j}^2 - d_{i,j}^2) (x_i - x_j) \\ + \sum_{j=1, j \neq i}^n \left[\left(\frac{1}{r_{i,j}} \right)^{14} - 0.5 \left(\frac{1}{r_{i,j}} \right)^8 \right] (x_i - x_j), \quad i = 1, \dots, n$$

where S is the set of selected pairs of molecules with distance constraints. We integrated the equations in (5.10) by using the position Verlet. A flow chart for the penalty function algorithm as applied to argon cluster simulation is shown in Figure 9. Note that in this implementation of the penalty function algorithm, a step size of 0.032 was used, and the penalty parameter was increased by 1 in every 500 time steps during the whole simulation.

Figure 10 shows the changes in potential energy for cluster 24 in 9000 time steps simulated by Verlet (VL) and the penalty function method (PL). Within this period of time, the potential energy of the trajectory produced by the penalty function method decreased gradually towards the global energy minimum of the cluster while the trajectory produced by Verlet remained oscillating at a high energy level. Similar results were observed on other clusters. Some of them showed even faster convergence of the trajectory to the global energy minimum of the clusters, as shown in Figure 11 for cluster 13, where the trajectory approached the global energy minimum in 3000 time steps.

6. Concluding Remarks

In this paper, we have proposed a so-called penalty function method for constrained molecular dynamics. In this method, a special function is defined so that the function is minimized if the constraints are satisfied. By adding such a function in the potential energy function, the constraints can then be removed from the system, and the simulation can be carried out in a conventional, unconstrained manner. The advantage of using a penalty function method is that it is easy to implement, and does not require solving a nonlinear system of equations in every time step. The disadvantage of the method is that the penalty parameter, i.e., the parameter used to scale the penalty function, is hard to control and in principle, needs to be large enough for the penalty function to be truly effective, which

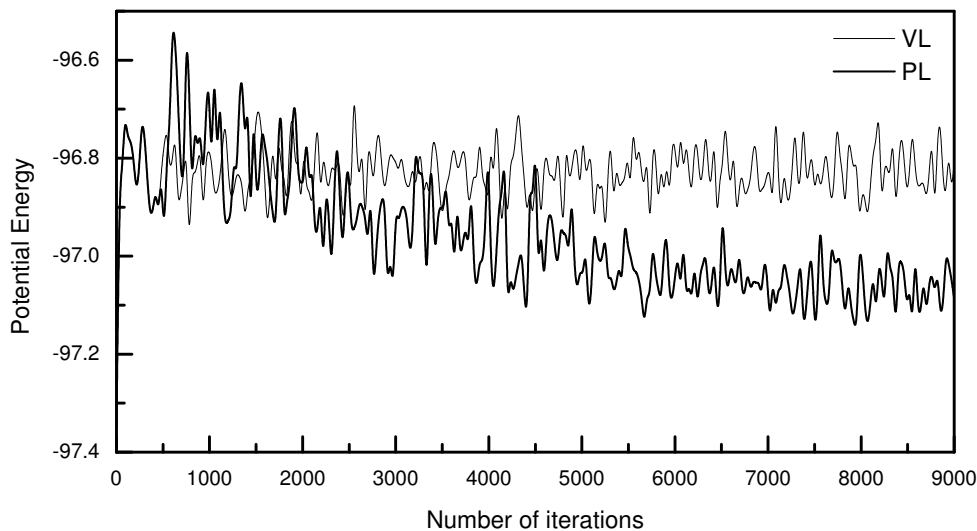


FIGURE 10. Changes in potential energy of argon cluster 24. Solid and dotted lines show the potential energy of the trajectory produced by the Verlet (VL) and penalty function (PL) methods, respectively. Here, randomly selected 50% of all distances were constrained to their distances in the global energy minimum configuration (-97.349).

on the other hand, may cause numerical instabilities when used in simulation. It may also arguably be a disadvantage that the penalty function method only forces the constraints to be satisfied approximately but not completely. In any case, the method may possibly be used as an alternative and computationally more efficient approach for constrained molecular dynamics simulation than the Lagrange multiplier types of methods.

We have first implemented a penalty function method in CHARMM and tested it on protein Bovine Pancreatic Trypsin Inhibitor (BPTI) by following a similar experiment done by Gunsteren and Karplus for the Shake algorithm. In this implementation, we removed the bond length potentials from the potential energy function and introduced the corresponding bond length constraints. For each of the bond length constraints, we constructed a quadratic penalty function and inserted it into the potential energy function. For each different type of bond, we also scaled the corresponding penalty function with the force constant of the bond so that the resulting function had the same form as the original bond length potential if without multiplied by the penalty parameter. In this way, the resulting force field becomes simply a continuation of the original force field as the penalty parameter changes continuously from 1 to a value > 1 . We conducted a simulation on BPTI with the penalty function method, and compared the results with Verlet and Shake, and found that the penalty function method had a high correlation with the Shake and outperformed the Verlet. In particular, the root-mean-square-deviations (RMSD) of the backbone and non-backbone atoms and the velocity auto correlations of the *C_{alpha}* atoms of the protein calculated by the penalty function method agreed well with those by Shake. Note again that the penalty function method

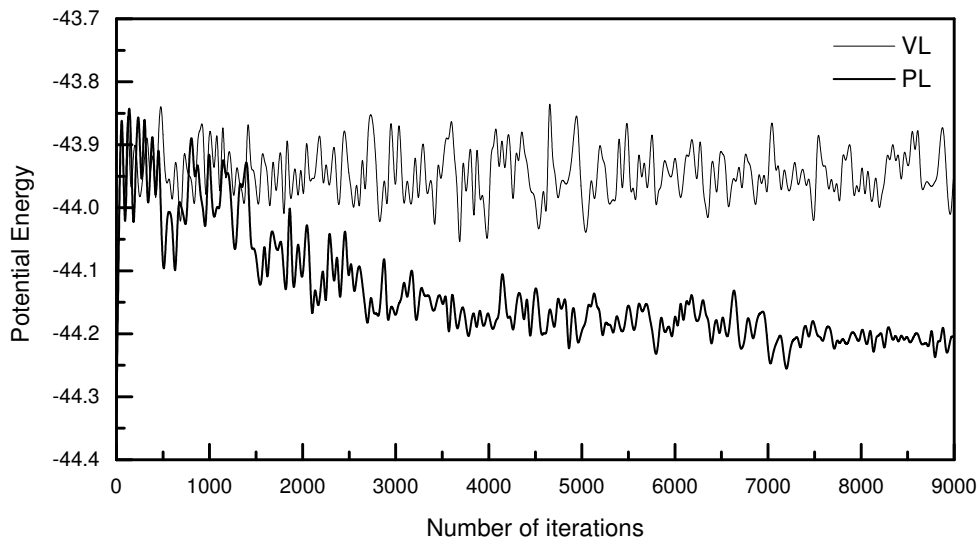


FIGURE 11. Changes in potential energy of the trajectory for argon cluster 13 produced by the penalty function method. Here, randomly selected 60% of all distances were constrained to their distances in the global energy minimum configuration. The trajectory already approached to the global energy minimum (-44.3) of the cluster in 3000 time steps while the trajectory generated by the Verlet remained in high energy.

requires no more than just applying a conventional, unconstrained simulation algorithm such as the Verlet algorithm to the potential energy function expanded with additional penalty terms for the bond length constraints.

We have also tested the penalty function method on a group of argon clusters with the equilibrium distances for a selected set of molecular pairs as the constraints. Here by the equilibrium distances we mean the distances for the pairs of argon molecules when the clusters are in their global energy minimal states. We generated these distances by using the global energy minimal configuration of the clusters published in previous studies. A penalty function was constructed for each of the constraints and incorporated into the potential energy function of the cluster. The simulation was then conducted by using a conventional, unconstrained simulation method, i.e., the Verlet algorithm [26], with the extended potential energy function. Here, there were no substantial algorithmic changes or computational overheads required due to the addition of the constraints. The simulation results showed that the penalty function method was able to impose the constraints effectively and the clusters tended to converge to their lowest energy equilibrium states more rapidly than not confined by the constraints.

Acknowledgments

We would like to thank Di Wu and Peter Vedell for helpful comments and suggestions on the paper, and Department of Mathematics and College of Liberal Arts and Sciences of Iowa State University for their support on this work.

References

- [1] H. C. Andersen, Rattle: A Velocity Version of Shake Algorithm for Molecular Dynamic Calculations, *Journal Computational Physics*, 52, 1983, pp. 24 - 34.
- [2] V. I. Arnold, *Mathematical Methods of Classical Mechanics*, volume 60 of Springer Graduate Texts in Mathematics, Springer-Verlag, 1975.
- [3] E. Barth, K. Kuczera, B. Leimkuhler, R. D. Skeel, Algorithms for Constrained Molecular Dynamics, *Journal of Computational Chemistry*, 16, 1995, pp. 1192 - 1209.
- [4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, The Protein Data Bank, *Nucleic Acids Research*, 28, 2000, pp. 235 - 242.
- [5] D. P. Bertsekas, *Constrained Optimization and Lagrange Multipliers Methods*, Academic Press, New York, 1982.
- [6] B. R. Brooks, M. Karplus, B. M. Pettitt, *Proteins: A Theoretical Perspective of Dynamics, Structure and Thermodynamics*, John Wiley & Sons, 1988.
- [7] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, M. Karplus, CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, *Journal of Computational Chemistry*, 4, 1983, pp. 187 - 217.
- [8] R. Courant, *Variational Methods for the Solution of Problems of Equilibrium and Vibrations*, *Bulletin of the American Mathematical Society*, 49, 1943, pp. 1 - 23.
- [9] G. Di Pillo, Exact Penalty Methods, in: *Algorithms for Continuous Optimization: the state-of-the-art*, E. Spedicato (ed.), Kluwer Academic Publishers, Boston, 1994, pp. 1 - 45.
- [10] J. P. Dussault, Numerical Stability and Efficiency of Penalty Algorithms, *SIAM Journal on Numerical Analysis*, 32(1), 1995, pp. 296 - 317.
- [11] R. Fletcher, A Class of Methods for Nonlinear Programming with Termination and Convergence Properties, in *Integer and Nonlinear Programming*, J. Abadie (ed.), North-Holland Publishing Company, Amsterdam, London, 1970.
- [12] W. F. V. Gunsteren, M. Karplus, Effect of Constraints on Dynamics of Macromolecules, *Macromolecules*, 15, 1982, pp. 1528 - 1544.
- [13] M. Karplus, J. A. McCammon, Protein Structural Fluctuations During a Period of 100ps, *Nature London*, 277, 1979, pp. 578 - 585.
- [14] J. E. Lennard-Jones, A. E. Ingham, *Proceedings of the Royal Society of London*, A 107, 1925, pp. 636 - 653.
- [15] J. B. Marion, S. T. Thornton, *Classical Dynamics of Particles and Systems*, Harcourt Brace & Company, 1995.
- [16] J. A. McCammon, P. G. Wolynes, M. Karplus, Picosecond Dynamics of Tyrosine Side Chains in Proteins, *Biochemistry*, 18, 1979, pp. 927 - 942.
- [17] J. A. McCammon, B. R. Gelin, M. Karplus, Dynamics of Folded Proteins, *Nature*, 267, 1977, pp. 585 - 590.
- [18] J. Nocedal, S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
- [19] J. A. Northby, Structure and Binding of Lennard-Jones Clusters: 13 N 147, *Journal of Chemical Physics*, 87(10), 1987, pp. 6166 - 6177
- [20] J. M. Ortega, W. C. Rheinboldt, *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press, 1970.
- [21] T. Rapcsak, Global Lagrange Multiplier Rule and Smooth Exact Penalty Functions for Equality Constraints, *Nonlinear optimization and related topics*, eds., 2000, pp. 351 -368.
- [22] J. P. Ryckaert, F. H. Ciccoliti, H. J. C. Berendsen, Numerical-Integration of Cartesian Equations of Motion of a System with Constraints - Molecular-Dynamics of N-Alkanes, *Journal of Computational Physics*, 23, 1977, pp. 327 - 341.
- [23] T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide*, Springer, 2003.
- [24] H. P. Schwefel, *Evolution and Optimum Seeking*, John Wiley and Sons, 1995, pp. 351 - 368.
- [25] W. C. Swope, H. C. Andersen, P. H. Berens, K. R. Wilson, A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to small water clusters, *Journal of Chemical Physics*, 76, 1982, pp. 637 - 649.
- [26] L. Verlet, Computer Experiments on Classical Fluids I, Thermo Dynamical Properties of Lennard Jones Molecules, *Physical Review*, 159, 1967, pp. 98 - 103.
- [27] L. T. Wille, J. Vennik, Computational Complexity of the Ground-state Determination of Atomic Clusters, *Journal of Physics*, 18, 1985, pp. L419 - L422.
- [28] L. T. Wille, Minimum-Energy Configurations of Atomic Clusters: New Results Obtained by Simulated Annealing, *Chemical Physics Letter*, 133, 1987, pp. 405 - 410.

- [29] G. L. Xue, Improvement of the Northby Algorithm for Molecular Confirmation: Better Results, accepted for publication in journal of global optimization.

Department of Mathematics, Florida A & M University, Tallahassee, FL 32307, USA

E-mail: ajith.gunaratne@famu.edu

URL: <http://www.famu.edu/math/UserFiles/File/faculty/ajith/index.htm>

Department of Mathematics, Iowa State University, Ames, IA 50010, USA

E-mail: zhijun@iastate.edu

URL: <http://orion.math.iastate.edu/wu>