Finding Solutions of the Inverse Kinematics Problems in Computer-aided Drug Design

Ming Zhang  Lydia E. Kavraki
{mzhang,kavraki}@cs.rice.edu
Department of Computer Science, Rice University, Houston, TX 77005

Abstract

The efficient computation of low-energy molecular conformations is of critical importance to problems ranging from protein folding to computer-assisted drug design. Despite the growing number of papers on three-dimensional conformational search, several questions remain open. In this paper we investigate one such question relating to molecular inverse kinematics problems. In these problems we are given an initial conformation of a molecule and the target positions of some feature atoms of the molecule. We wish to automatically compute a new conformation of the molecule that brings the feature atoms to their target positions. We first show how to derive a system of polynomial equations from the geometric constraints of the feature atoms. In contrast with previous work, we do not attempt to solve the system of equations directly, which is computationally expensive. Instead, we adopt a technique based on the Groebner basis from algebraic geometry and develop a novel subdivision algorithm to approximate the real solutions. The approximated solutions can then be used as the starting conformations for existing (heuristic) energy minimization procedures that try to satisfy the target positions of feature atoms and reduce the overall energy of the conformation. To our knowledge, this is the first time that a rigorous algebraic methodology has been used to approximate molecular inverse kinematics solutions and the first time that a subdivision algorithm has been developed to efficiently locate the solutions.

Keywords: molecular conformation, inverse kinematics, polynomial equations, real root location, subdivision

1 Introduction

The efficient computation of low-energy molecular conformations is of critical importance to problems ranging from protein folding to drug design. A number of recent papers investigate
how to produce reasonable conformations for small molecules [6, 21], large molecules such as proteins [5], and molecular complexes [13, 16]. The number of varying torsional angles, bond angles, and bond lengths in a molecule (that is, the degrees of freedom - DOF - of a molecule) are in the order of a few tens even for small molecules [9, 23]. As a result, any systematic enumeration of conformations is prohibitively expensive and recent work has investigated a variety of randomized and heuristic approaches [10, 12, 21]. Today it is widely recognized that molecular motion is tied to function and as a result the generation of relevant molecular conformation snapshots has gained renewed attention [1].

In this paper we deal with a specific class of conformational search problems which we refer to as inverse kinematics problems. Our terminology is borrowed from the robotics literature as explained below. In the problems we consider, the target positions of some feature atoms of the molecule are user-specified. Then the values of the DOF of the molecule (torsional angles, bond angles and lengths) are automatically computed so that the feature atoms can achieve their target positions. A pictorial illustration is offered in Figure 1. As we will discuss below, computing the desired values of the DOF is extremely expensive, that is why in this paper we develop an algorithm to approximate the values of the DOF using a subdivision scheme.

Figure 1: (a) Ligand A and the feature atoms (black dots) in the initial conformation. (b) Now black dots indicate target positions of feature atoms of ligand A. The ligand changes its shape to fit the constraints. The identification of the target positions depends mainly on the chemical characteristics of the receptor.

The solution of molecular inverse kinematics problems can directly benefit a number of applications. One important class of applications relate to computer-assisted drug design. A fundamental assumption for rational drug design is that drug activity is obtained through the molecular binding of one molecule (the ligand) to the pocket of another, usually larger, molecule (the receptor, which is commonly a protein). In their active, or binding, conformations, the molecules exhibit geometric and chemical complementarity, both of which are essential for successful drug activity [13]. Very often the binding receptor pocket is unknown but a number of ligands have been experimentally found to interact with that specific pocket. Then chemists are interested in identifying the pharmacophore present in these ligands. The pharmacophore is a set of features in a specific 3D arrangement contained in all the active
conformations of the considered molecules (in Figure 1(b), the black dots could constitute a pharmacophore). A prevailing hypothesis is that the pharmacophore is the part, or parts, of the molecule that is responsible for drug activity, while the rest of the molecule is a scaffold for the pharmacophore’s features. The identification of pharmacophore can involve the solution of inverse kinematics problems [7]. Also, once the pharmacophore is found other molecules can be tested to see if they fit the pharmacophore using an inverse kinematics procedure [7]. Even when the docking site is known, it is frequently relevant to match some features of the cavity by placing features of the ligand in predefined positions or by displacing receptor side chains [10, 13] – in both cases inverse kinematics problems are defined. Other applications where inverse kinematics solutions are important include the placement of side chains and loops in protein folding [20] and the computation of protein folding pathways [22].

**Inverse Kinematics Problems and Our Contribution** We use the term inverse kinematics to relate the resemblance of these problems to robot kinematics problems. In robotics, a forward kinematics procedure takes as input the values of the DOF of the robot (e.g., joint angles) and produces its shape in space. An inverse kinematics procedure may take as input the Cartesian coordinates \((x, y, z)\) of the tip of a manipulator and produce as output the values of the DOF (e.g., the joint angles) that will enable the manipulator to reach position \((x, y, z)\). It has been shown that molecules can be modeled as tiny robots [4, 7] and hence we can talk about forward and inverse kinematics for molecules in the same way that we talk about forward and inverse kinematics for robots. In general, the position of any point on the robot (atom in the molecule) is represented as a polynomial or a trigonometric function. When feature points of the robot (atoms of the molecule) are set to the user-specified target values, a system of trigonometric or polynomial equations is obtained [14]. Consequently, the solution of a system of trigonometric or polynomial equations becomes the solution of inverse kinematics problems.

Inverse kinematics problems have been studied using randomized methods and energy minimization techniques while working with trigonometric equations directly [12]. These approaches suffer from the lack of good initial guesses for the minimizations. As a result, the methods can be trapped in local minimums before finding the solutions [12, 2].

Algebraic geometry methods have also been used to solve systems of polynomial equations resulting from inverse kinematics problems. However, finding the solutions of systems of multi-variate polynomials is by no means an easy task [6, 15]. A system of several (~5) general polynomials with several (~5) variables of moderate degrees (~10) can take several days to be solved in the modern computers, if the computation luckily terminates before running out of the all the available resources. Currently, there are no good, general solvers to solve multi-variate (non-linear) polynomial equations [15, 19]. Other earlier work on molecular inverse kinematics problems has used distance geometry, multi-variate resultant theory, and matrix methods [6, 14, 15, 17].

Realizing how expensive and computationally impractical it may be to compute the solutions of the polynomial equations, we propose a novel approach. First, since only real
solutions of the polynomial equations are meaningful in practice, we focus on real solutions and ignore the (non-real) complex solutions. Second, instead of computing the real solutions, we will approximate them using a subdivision scheme. As discussed above, the approximated real solutions of the polynomial equations are good initial conformations for a subsequent optimization (minimization) process [12]. Approximating the solutions usually takes much less time than computing the solutions. Therefore our approach exhibits the potential for dealing with high dimensional inverse kinematics problems.

**Organization** This paper is organized as follows. Section 2 reviews the derivation of molecular equations from the geometric constraints on the feature atoms and defines the molecular inverse kinematics problems. In Section 3 we review a technique based on the Groebner basis from algebraic geometry [3] for isolating and locating real roots of molecular equations. In Section 4, we introduce a subdivision algorithm to approximate the real solutions of the molecular inverse kinematics problems defined in Section 2. We conclude with a discussion of the subdivision algorithm and future work in Section 5.

## 2 Inverse Kinematics Problems

In this section, we define the inverse kinematics problems. In most molecular kinematics studies, the van der Waals radii, electric charges, bond lengths, and bond angles are considered constant, while the torsional angles are allowed to change [7, 9, 16]. We follow this assumption here.

We further group atoms into atomgroups. An atomgroup is a (maximum) set of connected atoms such that none of the bonds among the atoms rotate. Using atomgroups instead of individual atoms can considerably speed the calculation of molecular conformations [23]. For simplicity, we also assume that there are no cycles of atomgroups in the molecule. When one atomgroup is chosen as the root (anchor), the molecule becomes a tree with the atomgroups at the nodes.

### 2.1 Molecular Equations

**Attaching local frames** We attach local frames to atomgroups to facilitate the calculation of atom positions. A local frame $F_i = \{Q_i; u_i, v_i, w_i\}$ is attached to atomgroup $g_i$ as follows (Figure 2(a)): $Q_i$ is the child atom (in $g_i$) of bond $b_i$; $w_i$ is the unit vector along bond $b_i$ pointing toward $g_{i-1}$; $u_i$ is an arbitrary unit vector perpendicular to $w_i$; $v_i$ is perpendicular to $w_i$ and $u_i$.

**Relational matrices** We now derive the relations between neighbor local frames which will be used to calculate atom positions. Suppose the frame at atomgroup $g_i$ is $F_i =$
Figure 2: (a) Local frame $F_i = \{Q_i; u_i, v_i, w_i\}$ at atom group $g_i$. (b) Local frames $F_i = \{Q_i; u_i, v_i, w_i\}$ at atom group $g_i$ and $F_{i-1} = \{Q_{i-1}; u_{i-1}, v_{i-1}, w_{i-1}\}$ at atom group $g_{i-1}$.

$\{Q_i; u_i, v_i, w_i\}$ and the frame at its parent atom group $g_{i-1}$ is $F_{i-1} = \{Q_{i-1}; u_{i-1}, v_{i-1}, w_{i-1}\}$. Let the torsional angle of bond $b_i$ be $\theta_i$ (Figure 2(b)). For each atom $A$ in atom group $g_i$, the coordinates $(x_i, y_i, z_i)$ in $F_i$ and the coordinates $(x_{i-1}, y_{i-1}, z_{i-1})$ in $F_{i-1}$ are related by

$$(x_{i-1} \ y_{i-1} \ z_{i-1} \ 1)^t = R_i \cdot (x_i \ y_i \ z_i \ 1)^t,$$

where

$$R_i = \text{Constant matrix} \cdot \begin{pmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$  

Updating atom positions We now calculate the position of any atom $A$ in atom group $g_i$. Suppose $g_i, g_{i-1}, \ldots, g_0$ is a sequence of atom groups, where $g_j$ is the parent atom group of $g_{j+1}$, $0 \leq j \leq i - 1$, and $g_0$ is the root atom group. Then the position of $A$ is

$$(x \ y \ z \ 1)^t = R_1 \ldots R_i \cdot (x_i \ y_i \ z_i \ 1)^t,$$

where $(x_i, y_i, z_i)$ are the coordinates of $A$ in the local frame at atom group $g_i$. (Here we assume that the Euler angles are known and the Euler matrix has been multiplied into the constant matrix in $R_1$ (Equation 2).)

2.2 Inverse Kinematics Problems

We can now formulate the molecular inverse kinematics problems. Given (i) a molecule in an initial conformation, and (ii) the target positions $(a_i, b_i, c_i)$ of some feature atoms, solve for the values of torsional angles so that the feature atoms reach their target positions.
For each feature atom $A_i$, if the target position is $(a_i, b_i, c_i)$, then

$$(a_i \ b_i \ c_i \ 1)\cdot R_1 \ldots R_i (x_i \ y_i \ z_i \ 1),$$

where $(x_i, y_i, z_i)$ are the (known, fixed) local coordinates of the feature atom $A_i$.

There are three equations in (3) --- the last coordinate gives the identity $1 = 1$. Each of these three equations in (3) is linear in $\cos(\theta_j), \sin(\theta_j), j = 1, \ldots, i$, respectively. Instead of dealing with the trigonometric equations directly, we convert the cosine and sine functions into rational functions using the standard transformation

$$\cos(\theta_j) = \frac{1 - t_j^2}{1 + t_j^2}, \quad \sin(\theta_j) = \frac{2t_j}{1 + t_j^2},$$

where $t_j = \tan(\theta_j/2)$. Multiplying the common divisors to both sides of these equations, we obtain three polynomial equations in $t_1, \ldots, t_i$. Each of these three equations is quadratic in each of the variables $t_1, \ldots, t_i$. We call these polynomial equations \textit{molecular equations}.

Suppose the number of rotatable bonds of the molecule is $n$, and the number of feature atoms is $l$. Then the total number of molecular equations is $3(l - 1)$; each of these equations has total degree $2n$ in $t_1, \ldots, t_n$. By Bézout’s theorem [3], there are up to $(2n)^{3(l - 1)}$ complex solutions of these equations (counting multiplicities) if there are finitely many solutions. These solutions are the solutions of the molecular inverse kinematics problems specified by (3). Since only real solutions are meaningful for our problem, we will target the real solutions only. Assuming that the number of solutions of the molecular equations is finite, we will adopt a technique based on the Groebner basis from algebraic geometry [3] and develop a subdivision algorithm to approximate the real solutions.

3 Locating Real Roots

In this section, we review some background algebraic geometry on locating and isolating real solutions of polynomial equations. Usually, computing the solutions of polynomial equations can be a very difficult task [15]. Instead of computing the solutions, we will approximate the solutions by identifying the regions where the solutions reside. The technique described in this section will help to isolate and locate real solutions of polynomial equations. For more information, see [3, 18].

Throughout this section, we let $k$ be a subfield of $\mathbb{R}$ (real numbers), usually $k = \mathbb{Q}$ (rational numbers), or a finite extension field of $\mathbb{Q}$. Let $I$ be an ideal in the polynomial ring $k[x_1, \ldots, x_n]$ and we denote the set of real solutions of $I$ by $V(I) = \{ a \in \mathbb{R}^n | f(a) = 0, \forall f \in I \}$. Assume that $V(I)$ is finite, i.e., the polynomial equations of $I$ have finitely many real solutions.
Mapping $m_h$ Let $A = k[x_1, \ldots, x_n]/I$ be the quotient ring. Then $A$ is a vector space over $k$. For any $h \in k[x_1, \ldots, x_n]$, define

$$m_h : A \longrightarrow A, \quad m_h([g]) = [h] * [g] = [h \cdot g] \in A,$$

where the brackets represent the equivalent classes. Then $m_h$ is a linear mapping from $A$ to $A$. Fix a monomial order, and let $B$ be a monomial basis of $A$ as a vector space over $k$. Then we can represent $m_h$ by its matrix with respect to basis $B$.

Bilinear form $S_h$ For any $h \in k[x_1, \ldots, x_n]$, define bilinear form

$$S_h : A \times A \longrightarrow k, \quad S_h([f], [g]) = \text{Trace}(m_{hf \cdot fg}),$$

(5)

where the trace of a matrix is the sum of the diagonal entries, and $m_{hf \cdot fg}$ is a mapping defined by the polynomial $h \cdot f \cdot g$. It is easy to see that $S_h$ is symmetric, i.e., $S_h([f], [g]) = S_h([g], [f])$. Let $M_h$ be the matrix of $S_h$ with respect to basis $B$. Then $M_h$ is symmetric. Specifically, suppose $B = \{v_1, \ldots, v_d\}$, then $M_h$ has dimension $d \times d$, and the $(i, j)$ entry of $M_h$ is $\text{Trace}(m_{h \cdot v_i \cdot v_j})$.

Decomposition of $\mathbb{R}^n$ Since we seek only the real solutions of polynomial equations, the search space for the solutions of the molecular inverse kinematics problems is $\mathbb{R}^n$. For any $h \in k[x_1, \ldots, x_n]$, define

$$H^+ = \{a \in \mathbb{R}^n \mid h(a) > 0\}, \quad H^- = \{a \in \mathbb{R}^n \mid h(a) < 0\}, \quad H^0 = \{a \in \mathbb{R}^n \mid h(a) = 0\}.$$

Note that $H^+, H^-, H^0$ do not intersect with each other, and $\mathbb{R}^n = H^+ \cup H^- \cup H^0$. The polynomial $h$ is called a decomposition function. Our goal is to locate the solutions of ideal $I$ in $H^+, H^-, H^0$ by properly choosing the decomposition functions $h$.

Main theorems The signature of a matrix is the difference between the number of positive eigenvalues and the number of negative eigenvalues. The signature of a matrix $M$ is denoted by $\sigma(M)$. We state two main theorems below. The second theorem holds the key to the algorithm we develop in Section 4.

Theorem 1 [3] Assume that $V(I)$ is finite. Let $S_h$ be the bilinear form defined by Equation (5) and $M_h$ be its matrix. Then the signature of $M_h$

$$\sigma(M_h) = \#\{a \in V(I) \mid h(a) > 0\} - \#\{a \in V(I) \mid h(a) < 0\}. \quad \square$$

Theorem 2 Assume that $V(I)$ is finite. The numbers of solutions of $I$ in each of $H^+$, $H^-$, $H^0$ are

$$\#\{a \in V(I) \mid h(a) > 0\} = (\sigma(M_1) + \sigma(M_{12}))/2,$$

$$\#\{a \in V(I) \mid h(a) < 0\} = (\sigma(M_{12}) - \sigma(M_h))/2,$$

$$\#\{a \in V(I) \mid h(a) = 0\} = \sigma(M_1) - \sigma(M_{12}),$$

where $M_1$ is the matrix of the bilinear form defined by the constant function 1.
Proof: The theorem follows from these equalities:

\[
\begin{align*}
\sigma(M_h) &= \# \{ a \in V(I) \mid h(a) > 0 \} - \# \{ a \in V(I) \mid h(a) < 0 \}, \\
\sigma(M_{I_2}) &= \# \{ a \in V(I) \mid h^2(a) > 0 \} \\
&= \# \{ a \in V(I) \mid h(a) > 0 \} + \# \{ a \in V(I) \mid h(a) < 0 \}, \\
\sigma(M_I) &= \# \{ a \in V(I) \}. \quad \Box
\end{align*}
\]

To obtain the solution counts in each of \(V(I) \cap H^+, V(I) \cap H^-, V(I) \cap H^0\), we need to compute the signatures \(\sigma(M_h), \sigma(M_{I_2}), \sigma(M_I)\). There are several ways to determine the signature of a real symmetric matrix. We can solve for the eigenvalues of the matrix and obtain the signature. However this is expensive and unnecessary since we do not really need the eigenvalues. A better way is to use the characteristic polynomials of symmetric matrices. The number of sign changes of the coefficient sequence of the characteristic polynomial of a matrix gives the number of the positive and negative eigenvalues and hence the signature [3]. The most reliable way we know to compute the signature of a symmetric matrix is to use the Bunch-Kaufman factorization [11]. A symmetric matrix \(M\) is factored into \(M = QDQ^T\) where \(D\) is symmetric and block diagonal with 1-by-1 or 2-by-2 diagonal blocks, and \(Q\) is a non-singular matrix. The signature of \(M\) is the difference between the number of positive 1-by-1 blocks and number of negative 1-by-1 blocks of \(D\).

In summary, instead of computing the solutions of a system of polynomial equations, we try to isolate and locate the real solutions. We choose a decomposition function \(h\), and partition the search space \(\mathbb{R}^n\) into three non-intersecting regions \(H^+, H^-, H^0\). By computing the number of positive and negative eigenvalues of some matrices induced by \(h\), we can calculate the number of real solutions of the polynomial equations in each of \(H^+, H^-, H^0\).

### 4 Approximating Inverse Kinematics Problems

From Section 2, we have derived a system of molecular equations from the geometric constraints of the feature atoms. Using Theorem 2 from Section 3, we can count the numbers of real solutions in regions \((H^+, H^-, H^0)\) of the search space \(\mathbb{R}^n\). In this section, we develop a subdivision algorithm to approximate the real solutions of the molecular equations, hence the solutions of the molecular inverse kinematics problems. Below we first identify the intervals in which the coordinates of the real solutions lie. Then we identify the boxes (regions in the search space \(\mathbb{R}^n\)) where the real solutions reside. Both steps are accomplished by properly choosing the decomposition functions \(h\).

**Identification of intervals** Suppose the molecular equations involve \(n\) variables \(t_1, \ldots, t_n\). We give the following algorithm to identify the intervals in which the coordinates of the real solutions lie:
• First we choose \( h(t_1, \ldots, t_n) = t_1 - a_1 \) and the search space \( \mathbb{R}^n \) is decomposed into two half spaces and a plane. We compute the number of real solutions in each part: \( n_1 \) is the number of solutions whose \( t_1 \) coordinates are less than \( a_1 \); \( p_1 \) is the number of solutions whose \( t_1 \) coordinates are bigger than \( a_1 \) (Figure 3).

\[
\begin{array}{c}
| n_1 \\
\ h(t_1, \ldots, t_n) = t_1 - a_1 \\
\ a_1 \\
| \ \\
\ \ \\
\ v \\
\ p_1 \\
\ v \ \\
\ a_1 \\
\ v \\
\ v \ \\
\ v \ \\
\ t_1
\end{array}
\]

Figure 3: Solution counts \( n_1 \) and \( p_1 \) for decomposition function \( h = t_1 - a_1 \).

• If \( p_1 > 0 \), then we choose \( h(t_1, \ldots, t_n) = t_1 - a_2 \), where \( a_2 > a_1 \) and compute \( p_2 \) (Figure 4).

\[
\begin{array}{c}
| n_1 \\
\ h(t_1, \ldots, t_n) = t_1 - a_2 \\
\ a_1 \\
\ a_2 \\
| \ \\
\ p_1 \\
\ p_2 \\
\ v \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ a_1 \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \\
\ v \ \

intervals that contain solutions, we can approximate the coordinates of the real solutions to an arbitrary precision — location.

Therefore, we can obtain a sequence of intervals \((l_1, u_1), \ldots, (l_q, u_q)\) for each variable \(t_i, i = 1, \ldots, n\). Each interval \((l_j, u_j)\) guarantees that there exists a real solution whose \(t_i\) coordinate lies between \(l_j\) and \(u_j\), \(1 \leq j \leq q\).

**Identification of boxes** Now we have sequences of intervals for the coordinates of the solutions. We still need to integrate the coordinates into the solutions — not all the combinations of the coordinates form solutions of the molecular equations. Below we first give an algorithm to combine two sequences of intervals (for two variables) to form partial solutions (identification of rectangles). Then we use the information of the rectangles to form the complete solutions of the molecular equations (identification of boxes).

Let us check the interval sequences \(\{(l_{a_1}, u_{a_1}), \ldots, (l_{a_q}, u_{a_q})\}\) for \(t_1\) and \(\{(l_{b_1}, u_{b_1}), \ldots, (l_{b_{q'}}, u_{b_{q'}})\}\) for \(t_2\). The number of solutions of the molecular equations is fixed, say \(q\), no matter whether we count them with respect to \(t_1\) or \(t_2\), \(q' = q'' = q\). As illustrated in Figure 5, there is a solution in each interval \((l_{a_i}, u_{a_i})\) and interval \((l_{b_j}, u_{b_j})\). Therefore, each shaded rectangle contains exactly one solution inside.

![Figure 5: The interval sequence \(\{(l_{a_1}, u_{a_1}), \ldots, (l_{a_q}, u_{a_q})\}\) for \(t_1\) and interval sequence \(\{(l_{b_1}, u_{b_1}), \ldots, (l_{b_{q'}}, u_{b_{q'}})\}\) for \(t_2\). Each shaded rectangle contains exactly one solution.](image)

We abstract Figure 5 into a matrix \(E = (e_{ij})\) of dimension \(q \times q\): the number of solutions in the shaded rectangle \((l_{a_i}, u_{a_i}) \times (l_{b_j}, u_{b_j})\) is \(e_{ij}\). Then the solution count problem can be rephrased using the matrix model. The entries \(e_{ij}\) of matrix \(E_{q \times q}\) are either 1 or 0. The sum of each row is 1; the sum of each column is 1. Moreover, at any \((i, j), 1 \leq i, j \leq q\), we break the matrix \(E\) into four submatrices

\[
\begin{pmatrix}
E_{11} & E_{12} \\
E_{21} & E_{22}
\end{pmatrix},
\quad
E_{11} = \{ e_{kl} \mid k < i, l < j \},
\quad
E_{12} = \{ e_{kl} \mid k < i, l \geq j \},
\quad
E_{21} = \{ e_{kl} \mid k \geq i, l < j \},
\quad
E_{22} = \{ e_{kl} \mid k \geq i, l \geq j \}.
\]

It is easy to see that the sum of the entries of \(E_{11}, E_{22}\) and the sum of the entries of \(E_{12}, E_{21}\) can be computed. In fact, if we choose a decomposition function \(h = (t_1 - l_{a_{n-i}}) \cdot (t_2 - u_{b_j})\), then all the real solutions which make \(h\) positive are counted in either \(E_{12}\) or \(E_{21}\); all the real solutions which make \(h\) negative are counted in either \(E_{11}\) or \(E_{22}\).
Next we show how to determine the non-zero entries of $E$ and hence identify the rectangles where the molecular equations have a real solution.

- Choose $h_1 = (t_1 - la_q) * (t_2 - ub_1)$. If the solution count $r_1$ in $H_1$ is $q$, then $c_{11} = 1$ and $c_{1j} = 0$, $j = 2, \ldots, q$.

- Otherwise, $c_{11} = 0$. Let $h_2 = (t_1 - la_q) * (t_2 - ub_2)$ and compute the solution count $r_2$ in $H_2$. If $r_2 > r_1$, then $c_{12} = 1$ and all other entries in the first row are zero. Otherwise, $r_1 > r_2$ (since $r_1 \neq r_2$) and $c_{12} = 0$. Let $h_3 = (t_1 - la_q) * (t_2 - ub_3)$ and continue the above process until the non-zero entry in the first row of $E$ is identified.

- Repeat the process to identify the non-zero entries of other rows of matrix $E$.

Now we know the $(t_1, t_2)$ coordinates (partial solutions) of the real solutions of the molecular equations. By examining the intervals $\{(la_1, wa_1), \ldots, (la_q, wa_q)\}$ for $t_1$ and $\{(l_1, u_1), \ldots, (l_q, u_q)\}$ for $t_2$, we get the $(t_i, t_i)$, $i = 2, \ldots, q$, coordinates of the real solutions of the molecular equations. Combining all the $(t_1, t_i)$ information together, we are able to locate the real solutions of the molecular equations in the boxes in $\mathbb{R}^q$. The 3-dimensional case is illustrated in Figure 6.

![Figure 6](image)

Figure 6: Identifying solutions in boxes. A "x" indicates a solution in the corresponding rectangle. Each level has exactly one solution; so the box to which the "x" on the two sides project contains the solution.

## 5 Discussion

In this paper, we developed an algorithm to approximate the real solutions of molecular inverse kinematics problems using a subdivision scheme. Intervals which contain the real solution coordinates are first identified and then combined to locate the solutions of the molecular equations in the entire space.

In this paper, we have assumed that only the torsional angles can change while the bond angles and bond lengths are fixed. However, our subdivision algorithm still works when the
bond angles, bond lengths together with the torsional angles vary. The molecular equations
have to be amended though to account for changes of bond lengths and bond angles.

In Section 4, we assumed that the solutions lie inside the intervals \( (l_i, u_i) \), not the bound-
aries \( l_i \) or \( u_i \). If some of the interval boundaries \( l_i \) or \( u_i \) happen to be the solutions of the
molecular equations, we can slightly adjust the intervals so that the solutions still lie inside
the intervals. We also assumed that the refined intervals either contain one solution or have
no solutions inside. If more than one solutions of the molecular equations have the same
\( t_i \) coordinate, then no matter how we refine the intervals, some intervals always have more
than one solution. The subsequent analysis on identification of rectangles containing the
partial solutions still works with a slight modification.

The biggest challenge of the algorithm in this paper arises from the computation of the
Groebner basis. The Groebner basis computation has double exponential complexity if all
the arithmetics are carried out using exact numbers. We are working on the implementation
of floating point arithmetics, and a rigorous mathematical proof of the effectiveness and
efficiency of the floating point arithmetics. We are also working on determining the sensitivity
and tolerance of our algorithm with respect to the input data errors.

In closing, we would like to emphasize that there are no satisfactory algorithms for the
molecular inverse kinematics problems that we examine in this paper. As discussed in Section
1, randomized approaches suffer from local minima, while algebraic geometry methods that
try to compute exact solutions are prohibitively expensive. Given that approximating real
solutions is much faster than actually computing solutions, our approach presents a novel
direction of research that has the potential to deal with high dimensional inverse kinematics
problems.

Acknowledgment Work on this paper by Ming Zhang and Lydia Kavraki is supported
in part by ATP 003604-0120-1999, NSF IRI-970228, NSF CISE SA1728-21122N, a Whitaker
Biomedical Engineering Grant and a Sloan Fellowship awarded to Lydia Kavraki. We like to
thank Professor Dan Sorrensen at Rice University for referring us to LAPACK on computing
the signatures of symmetric matrices. We would also like to thank Miguel Teodoro, Brian
Chen and Andrew Ladd for useful discussions.

References


tern Matching in Files of Three-dimensional Chemical Structures: Comparison of
Conformational-searching Algorithms for Flexible Searching, Journal of Chemical In-
formation and Computer Sciences, 34, 197-206.


6 Appendix

In this section, we briefly review the computation of Groebner basis using Buchberger’s S-polynomial algorithm. For more information on the Groebner basis computations and its applications, see [3].

**Definition** Let \( f, g \in k[x_1, \ldots, x_n] \) be nonzero polynomials. Fix a monomial order and let the leading terms of \( f, g \) be

\[
LT(f) = cx^\alpha, \quad LT(g) = dx^\beta,
\]

where \( c, d \in k \) are constants, and \( x^\alpha \) is an abbreviation of \( x_1^{\alpha_1} \ldots x_n^{\alpha_n} \). Let \( x^\gamma \) be the least common multiple of \( x^\alpha \) and \( x^\beta \). The S-polynomial of \( f, g \) is

\[
S(f, g) = \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g.
\]

**Buchberger’s Algorithm**

**Input:** \( F = \{f_1, \ldots, f_s\} \)

**Output:** a Groebner basis \( G = \{g_1, \ldots, g_t\} \) for \( I = \langle F \rangle \) with \( F \subset G \)

\[
G \leftarrow F
\]

**Do**

\[
G' \leftarrow G
\]

**For each pair** \( (p, q), p \neq q, p, q \in G' \), do

\[
S \leftarrow S(p, q)
\]

Compute the remainder \( r \) of \( S \) w.r.t \( G' \) (simple division by \( G' \))

If \( r \neq 0 \), then \( G \leftarrow G \cup \{r\} \)

**Until** \( G = G' \)

The Groebner basis computation has been implemented in several institutions. Professor Buchberger and Windsteiger have their version available through ftp from melmac.risc.uni-linz.ac.at.

15