Algebraic Algorithms for Structure Determination in Biological Chemistry

IOANNIS Z. EMIRIS,[1] EPAMINONDAS D. FRITZILAS,[1] DINESH MANOCHA[2]

[1]*Department of Informatics and Telecommunications, National University of Athens, Panepistimiopolis 15784, Greece*

[2]*Department of Biology, National University of Athens, Panepistimiopolis 15784, Greece*

[3]*Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, USA*

Received 30 April 2005; accepted 5 May 2005
Published online 12 July 2005 in Wiley InterScience (www.interscience.wiley.com).
DOI 10.1002/qua.20703

ABSTRACT: Several problems in computational chemistry, structural molecular biology, and biological chemistry can be solved by symbolic-numerical algorithms. We introduce suitable algebraic tools and then survey their usage in concrete applications. In particular, questions on molecular structure can be modeled by systems of polynomial equations, mainly by drawing on techniques from robot kinematics. Resultant-based algorithms, including sparse resultants and their matrix formulae, are described in order to reduce the solving of polynomial systems to numerical linear algebra. As an illustration, we focus on computing all conformations of cyclic molecules and on matching pharmacophores under distance constraints; in both cases, the number of independent degrees of freedom is relatively small. We summarize some existing results as well as sketch some original work. Both lead to complete and accurate solutions for those problems in the sense that our algorithms output all solutions with sufficiently high precision for the needs of biochemical applications. Int J Quantum Chem 106: 190–210, 2006

Key words: molecular conformation; geometric constraint; pharmacophore; algebraic algorithm; symbolic-numerical method

Correspondence to: I. Emiris; e-mail: emiris@di.uoa.gr

Contract grant sponsor: Greek Ministry of Educational Affairs and the European Union.
Contract grant number: Pythagoras - 70/3/7392.
Contract grant sponsor: Research Council of National University of Athens, Greece.
Contract grant number: Kapodistrias - 70/4/6452.
Contract grant sponsor: Army Research Office.
Contract grant numbers: DAAD19-02-1-0390; W911NF-04-1-0088.
Contract grant sponsor: National Science Foundation.
Contract grant numbers: 0400134; 0118743.
Contract grant sponsor: Office of Naval Research.
Contract grant number: N00014-01-1-0496.
Contract grant sponsor: Defense Advanced Research Projects Agency/Research, Development & Engineering Command.
Contract grant number: N61339-04-C-0043.
Contract grant sponsor: Intel.

1. Introduction

Identifying molecular structure is a critical question in molecular biology and biological chemistry, because the activity of a molecule strongly depends on its three-dimensional structure and geometric complementarity to other molecules. Over the past few decades, the use of computer tools has gained considerable importance in areas such as pharmaceutical drug design, molecular modeling, and docking. This article analyzes symbolic-numerical computational methods for modeling geometric problems concerning molecules as algebraic systems, as well as algorithms for solving these systems.

We sketch a general methodology for modeling such problems in algebraic terms so that the resulting polynomial system has small dimension. We provide a brief introduction to powerful algorithms for studying basic properties of systems of simultaneous polynomial equations, in particular for counting and computing their common roots. From the solutions one can compute the conformers that satisfy the requirements of the initial geometric problem. We illustrate the application of these tools to computing the conformations of ring molecules and solving pharmacophore-like distance constraints. In both cases, we enumerate all possible solutions to the molecular problem, which provides a complete and rigorous way to study the three-dimensional structure. The illustrative applications are based on our earlier studies [1–4].

We emphasize problems with relatively few degrees of freedom, usually up to 10 or 20. In these cases, we strive for a procedure for equation solving with the following characteristics:

Efficiency: Certain applications demand real time performance, for instance, when the problem at hand is a subproblem of a larger question to be solved repeatedly.

Robustness: The algorithm should be able to compute all real solutions.

Accuracy: The input data obtained from experiments or sensors may not be exact. As a result, the algorithm for equation-solving should be well conditioned.

Our tools, based on advanced system-solving methods, will satisfy all these goals.

Solving algebraic systems is a well-studied task of high current research interest. Methods for this problem can be distinguished in two large families, namely algebraic and numerical algorithms whose boundaries, nonetheless, are unclear. The first family includes Gröbner bases and resultants; in both cases, a numerical subtask is typically executed at the end. Purely numerical approaches include Newton-based iterative methods, exclusion algorithms, and homotopy continuation. A full account of different alternatives is beyond the scope of this work.

We present resultants of polynomial systems and how they are expressed by matrices, which are called resultant matrices. We concentrate on matrices of the Sylvester (and Macaulay) type, including techniques for exploiting the sparsity of the input equations. The principal merits of algorithms based on resultants and their matrices are their completeness and accuracy, compared with purely numerical techniques, and their speed, compared with purely symbolic algorithms. Furthermore, resultant-based methods can handle approximate inputs or inputs of limited accuracy and produce the best possible output under some measure.

The size of Sylvester-type resultant matrices is exponential in the number of unknowns, i.e., the number of equations. But in system solving, the operations carried out online include only numerical operations on the resultant matrices. The matrix construction phase can be executed off-line, with symbolic polynomial coefficients. Hence, for systems with dimension up to 10, resultant matrices seem to be the method of choice, since they combine the veracity of symbolic algorithms with the speed of numerical linear algebra. We have in mind systems with equations of moderate total degree, typically up to 4 or 6, characterized by some sparsity of the polynomials.

Let us now go back to our original problems, dealing with the molecular structure. Most algebraic approaches deal with the problem of computing the molecular structure under a set of constraints imposed on it. A feature that is common to all of them is the reduction of the geometric constraints to a set of algebraic equations, which are then attacked with various methods. This means that, in general, there are two different stages, which vary according to the problem in question and according to the modeler's choices: the modeling part and the equation-solving part. The overall procedure can be outlined as follows:

Input: The physical description of a molecule in terms of point coordinates, distances, and/or angles and a set of constraints imposed on the molecular structure.

Output: A complete description of every possible conformation (alternatively, we may simply output a bound on the number of possible conformations)

There are several major steps:

1. Formulate the problem in terms of polynomial equations. This can be done either with an algorithmic methodology like the one described in the sequel, or with an ad hoc approach, suitable for the particular problem at hand.
2. Optionally compute bounds on the number of possible common solutions. In most cases, we focus on the real solutions, which have a direct interpretation for the physical problem.
3. Use some matrix-based elimination methods, such as those sketched below, to reduce the nonlinear polynomial system to an eigenproblem of an appropriate matrix. An alternative is to use symbolic computation, such as Gröbner bases or resultants, to reduce the polynomial system to the solution of a single univariate polynomial.
4. Numerically solve the eigenproblem or compute the roots of the univariate polynomial. In both cases, with some post-processing, one can retrieve all roots of the initial polynomial system.
5. From the (real) roots of the system, construct the corresponding molecular conformations.

This procedure will be illustrated for two concrete problems related to molecular conformations, solved by tools relying on resultants and linear algebra operations.

An overview of this article follows. Section 2 points to related work. Section 3 discusses modeling of molecular structure in algebraic terms. Section 4 examines our symbolic and numerical algorithms for studying and solving systems of polynomial equations. In Section 5 we focus on determining the structure of ring molecules as an illustration of resultant methods. Section 6 discusses original contributions to the problem of satisfying pharmacophore-like distance constraints. A summary of our contributions concludes this study in Section 7.

2. Related Work

Many methods have been proposed for solving problems of geometric nature concerning molecular structure. In most cases, the final goal is to compute all or some of the conformations satisfying a set of prescribed geometric constraints, (e.g., Refs. [5–9]). To achieve this goal, one has to conduct an efficient search the conformational space of the molecule at hand, a task for which various heuristic, analytical, and hybrid techniques have been developed.

In particular, there are several approaches in molecular modeling, whose roots can be traced back in robot kinematics, a research area that studies the motion of articulated mechanisms. The main premise for this interaction is the observation that various structural requirements on molecules can be modeled as macroscopic kinematic constraints. The relationship of molecules to robots is obvious, once bonds are thought of as rigid joints and atoms as the links or articulations of the mechanism. In kinematic terms, the molecule is equivalent to a serial mechanism in which each pair of consecutive axes intersects at a link. As in robotics, a suitable modeling methodology can help formulate the constraints imposed on the molecular structure as polynomial equations. Thereafter, their solution can be achieved with several tools originating from symbolic algebra and numerical linear algebra. This approach will be further discussed below.

Formalizing the rigidity constraints into algebraic terms may be harder than one imagines, especially because we try to minimize the dimension of the resulting polynomial system. Direct approaches exist for specific problems. For instance, using Euclidean geometry and the "flap" angles, small ring molecules have been modeled optimally in Refs. [2, 9] (cf. Section 5.1). The latter paper proposes an object-oriented implementation, in MAPLE, of geometric objects so as to automate the derivation of algebraic equations from geometric constraints. Distance geometry is a very general and powerful approach for studying rigidity. Among other things, it provides a rather general modeling approach, which again leads to algebraic systems [2, 10–12] (cf. Section 5.1).

Constraints entailed by rings have been extensively studied by Gō and Scheraga during the 1970s [13, 14]. These investigators established the main viewpoint, namely of considering the values of bond lengths and bond angles as fixed, while letting only the torsional dihedral angles vary. This simplification significantly reduces the problem complexity, while, at a first level of approximation, yields conformations close to energy minima. In solving the resulting algebraic systems, several ad

hoc methods have been proposed (see Refs. [11, 12]).

The main contribution of Ref. [4] is the use of matrix methods for efficiently solving the closure problem for cyclic molecules with 6–8 rotational degrees of freedom. The authors use a result concerning the inverse kinematics of a serial mechanism with six revolute joints [15] and reduce the ring closure problem to the eigenproblem of an appropriate matrix. Extending to molecules with more than six varying dihedrals has been approached by a grid search of the space of the two last dihedrals, each grid point giving rise to a six-dimensional subproblem [4, 13].

In Ref. [2], the authors approach the same problem, namely ring closure for molecules with 5–7 atoms. Thereby, they apply modern algebraic algorithms based on three different types of resultant matrices: Bézout resultant, sparse (or toric) resultant and simple Sylvester resultants in cascade. They provide a case analysis for their algebraic tools and compare the accuracy of output, as well as the speed of solution. They also study the stability of the resulting molecular conformations.

The computation of all conformations of a tripeptide loop, constrained to dock exactly on an existing protein scaffold, is another problem that has been attacked with algebraic methods. Wedemeyer and Scheraga [16] use geometrical arguments to formulate the problem as a polynomial system. By applying Sylvester resultants in cascade, they reduce the system to a univariate polynomial equation. Other approaches to the same problem, demonstrating alternative modeling and solution methodologies, as well as search techniques to deal with loops longer than three residues, are described in Refs. [17, 18].

Zhang et al. [19] propose an algebraic approach for a problem closely related to molecular docking: the computation of conformations that place certain atoms of a molecule in some prescribed positions in space. Using a modeling algorithm described in Ref. [20] they formulate the problem as an algebraic system. They solve the latter with an efficient subdivision algorithm, aiming only at the real roots.

Incorporating energy minimization into the conformational analysis is the final objective. In most cases, it is suggested to perform it independently, after the geometric constraints have been satisfied (e.g., Refs. [7, 21]). However, LaValle et al. [22] propose an algorithm for computing conformations of both acceptable geometry, according to some constraints, and low energy. Although they do not actually use algebraic solution tools, but adopt a heuristic minimization strategy instead, their modeling is once again based on robot kinematics.

Finally, we should mention that apart from the algebraic methods discussed above, there are plenty of different, mainly optimization-oriented approaches for the solution of geometric constraints imposed on small molecules. Many of these approaches have been developed in the context of chemical database screening; for a review, we refer to Ref. [6]. In general, they explore the conformational space with different strategies, seeking to minimize an appropriate cost function.

3. Molecular Modeling

This section draws upon the theory and algorithms of inverse kinematics in robotics to derive robust procedures for modeling problems related to molecular structure in algebraic terms. The inverse kinematics problem for general serial mechanisms has been a fundamental problem in robotics literature for more than three decades [23]. Robot manipulators are modeled as a rigid serial chain consisting of revolute and prismatic joints. The position and orientation of the end effector are a direct function of the joint variables. In most robotics applications, we are given the pose of the end effector, and the problem of inverse kinematics corresponds to computing the joint displacements for that pose.

We model the molecular chain as a rigid structure. In particular, we adopt two widely used simplifications to describe the conformational space of the molecules [24]. First, we consider the atoms to be points, whose positions can be uniquely identified in a three-dimensional (3D) coordinate system. These points are interconnected with a set of bonds, forming the so-called molecular graph. Second, we assume that the conformational flexibility of a molecule is only due to the ability of its parts to rotate about the axes defined by the single covalent bonds. This means that the bond lengths and angles are constrained to have some fixed values, while certain torsional angles are allowed to vary, accounting for the conformational flexibility of the molecule. From this point of view, the molecule looks like a macroscopic robotic mechanism with revolute joints. This has been a key observation for several approaches that successfully applied ideas from robotics on the study of molecules.

Let us now present a general methodology that has been used in several approaches, e.g. [4, 17, 19,

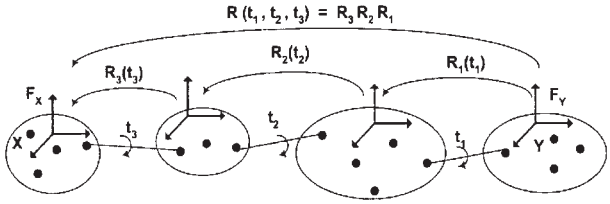

FIGURE 1. Coordinate transformations along the molecular chain.

22]. It is based on the idea of attaching several local coordinate systems on appropriate points of the molecule. As a first step, the molecule is split into rigid parts, i.e., groups of atoms that do not contain any rotating bonds, and therefore behave as rigid bodies. For example, the peptide bond defines a rigid part of a protein. Furthermore, we may consider rings to be rigid, as they have limited conformational flexibility or, in general, we can consider any fragment of the molecule rigid whose flexibility is not of interest to us. The idea of splitting the molecule into rigid atom groups was formalized in Ref. [20].

Having defined the rigid parts, we proceed by attaching on each a local coordinate system. Each atom belonging to the group has fixed and known coordinates with respect to the local system. The idea of the construction is illustrated in Figure 1. Let us consider two atoms, X and Y, and the corresponding local frames F_X and F_Y. The general goal is to compute matrix R that transforms the coordinates of an arbitrary point from frame F_Y to frame F_X. Matrix R is the product of matrices $R_1, \ldots, R_n$, each corresponding to a stepwise coordinate transformation along the bond path connecting the two atom groups. Each matrix R_i depends on the corresponding torsional angle t_i, which makes matrix R a function of all angles $t_1, \ldots, t_n$. Obviously, each R_i also depends on the relative position and orientation of the local frames, but since we fix a standard way to place them on the molecule, the only variables left undetermined are the torsional angles. In summary, to compute R as a function of the torsional angles, one needs a convention to place the local frames and a corresponding formula to compute the R_i values. This allows us to cast a geometric constraint involving the atoms X and Y as a trigonometric equation, containing the torsional angles as unknowns (cf. Section 6.1).

A standard method to attach the local coordinate systems on the molecule is the Denavit–Hartenberg convention (cf. Ref. [23]). We briefly sketch it below,

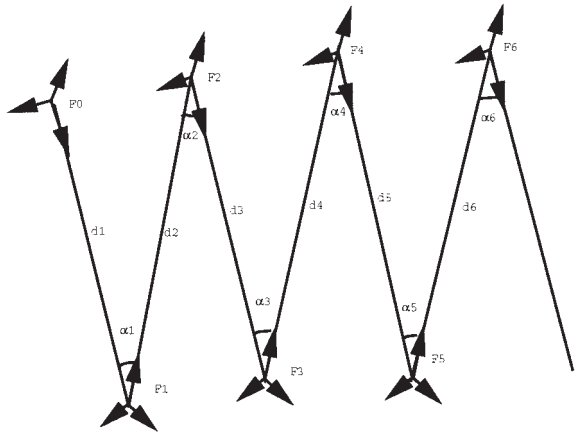

FIGURE 2. Denavit–Hartenberg formalism for a molecular chain with all bonds rotating.

for the case illustrated in Figure 2, where all bonds of the molecular chain are rotating. The case depicted in Figure 3 is somewhat different, since the peptide bond is regarded as rigid. This affects the placement of the local frames, but the basic idea and the transformation matrices remain essentially the same. Although we use terminology from the robotics literature, the molecular problem is equivalent, except that it is described using chemical bonds, bond lengths, and bond angles, as opposed to links, length of the links, and joint angles. Each link is represented by the line along its joint axis and the common normal to the next joint axis. In the

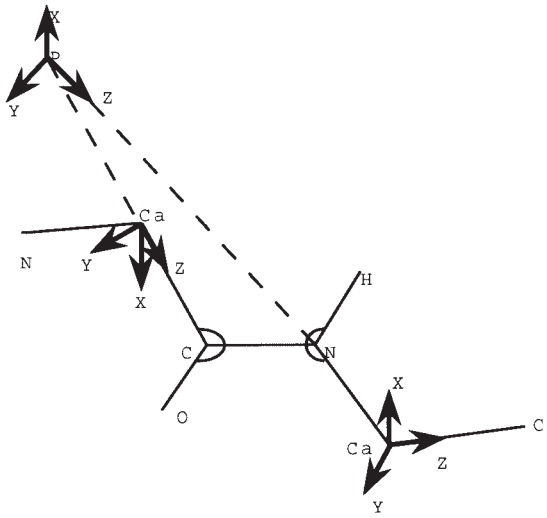

FIGURE 3. Denavit–Hartenberg formalism for a peptide unit. The peptide bond is considered rigid.

case of parallel joints, any of the common normals can be chosen. The links of the manipulator are numbered from 1 to n. The base link is 1, and the outermost link or hand is n. A coordinate system is attached to each link with which to describe the relative arrangements among the various links. The coordinate system attached to the ith link is numbered i.

The 4×4 transformation matrix relating the $(i-1)$st coordinate system to the ith coordinate system is [23]:

$$R_i = \begin{pmatrix} \cos t_i & -\sin t_i \cos \alpha_i & \sin t_i \sin \alpha_i & k_i \cos t_i \\ \sin t_i & \cos t_i \cos \alpha_i & -\cos t_i \sin \alpha_i & k_i \sin t_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where t_i is the ith joint rotation angle, α_i is the twist angle between the axes of joints i and $i+1$, k_i is the length of link $i+1$, and d_i is the offset distance at joint i.

For a given robot with revolute joints, we are given the α_i, d_i, and k_i values. For the inverse kinematics problem, we are also given the pose of the end-effector, attached to link n. This pose is described with respect to the base link or link 1. We represent this pose by matrix R_{hand}. Hence, the problem of inverse kinematics corresponds to computing the joint angles, $t_1, t_2, \ldots, t_n$ such that

$$R_1 R_2 \cdots R_n = R_{\text{hand}} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & s_1 \\ r_{21} & r_{22} & r_{23} & s_2 \\ r_{31} & r_{32} & r_{33} & s_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The left-hand side entries of the matrix equation given above are functions of the sines and cosines of the joint angles. Furthermore, this matrix equation corresponds to 12 scalar equations. Since the matrix formed by the first 3 rows and 3 columns of R_{hand} is orthonormal, only 6 of the 12 equations are independent. Thus, the problem of inverse kinematics of general manipulators with n joints corresponds to solving 6 equations for n unknowns.

The trigonometric equations can be transformed to algebraic by applying the half-tangent substitution on the sines and cosines and eliminating the denominators:

$$\sin t_i = \frac{2x_i}{1+x_i^2} \qquad \cos t_i = \frac{1-x_i^2}{1+x_i^2},$$

$$\text{where} \quad x_i = \tan\left(\frac{t_i}{2}\right). \quad (1)$$

For $n = 6$, the given system of equations has a finite number of solutions. In particular, it has been shown that the total number of solutions is 16 for $6R$ manipulators [25]. An ad hoc resultant formulation, exploiting sparsity, has been presented in Ref. [26], and it has been used in designing a real-time algorithm taking tens of milliseconds on the workstations used in Ref. [15]. However, in many instances, there are no real solutions to the problem. In such cases, we are interested in a pseudo-inverse kinematic version of the problem, which corresponds to the "closest real solution." It is defined in the following manner: Find the joint angles that minimize the Euclidean distance between the actual position of the end-effector and the desired position, given the orientation. This reduces to a constrained optimization problem. However, the minimizing function as well as the constraints are algebraic. For a chain with n links, computing the global minima reduces to six equations in n unknowns as well. No good algorithms are known for solving the pseudo-inverse kinematics problem.

Zhang and Kavraki [20] present a way of placing the local frames producing an alternative parametrization of the transformation matrices. We adopt this modeling formalization in Section 6.1 for casting the distance constraints as algebraic equations. One axis, say $\hat{w}_i$, of frame F_i coincides with the bond connecting the two neighboring atom groups, while the other two axes, $\hat{u}_i$ and $\hat{v}_i$, are arbitrarily chosen so as to define an orthonormal system. This construction yields the following transformation matrix. Vectors Q_i and Q_{i-1} appearing in the formula define the positions of the frame origins:

$$R_i = \begin{pmatrix} \hat{u}_{i-1}\cdot\hat{u}_i & \hat{u}_{i-1}\cdot\hat{v}_i & \hat{u}_{i-1}\cdot\hat{w}_i & \hat{u}_{i-1}\cdot(Q_i - Q_{i-1}) \\ \hat{v}_{i-1}\cdot\hat{u}_i & \hat{v}_{i-1}\cdot\hat{v}_i & \hat{v}_{i-1}\cdot\hat{w}_i & \hat{v}_{i-1}\cdot(Q_i - Q_{i-1}) \\ \hat{w}_{i-1}\cdot\hat{u}_i & \hat{w}_{i-1}\cdot\hat{v}_i & \hat{w}_{i-1}\cdot\hat{w}_i & \hat{w}_{i-1}\cdot(Q_i - Q_{i-1}) \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos t_i & -\sin t_i & 0 & 0 \\ \sin t_i & \cos t_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

4. Algebraic and Numerical Algorithms

This section presents a brief overview of algebraic and numerical techniques used for studying and solving algebraic systems.

4.1. ROOT COUNTING

A first step in the analysis of a polynomial system consists of studying the geometry of common roots and in finding an accurate bound for the number of common solutions. We consider the algebraic closure of the given coefficient field, which is typically the field of complex numbers. Let us use the following system, to be encountered in Section 5, as an illustrative example:

$$\begin{aligned} f_1 &= \beta_{11} + \beta_{12}x_2^2 + \beta_{13}x_3^2 + \beta_{14}x_2x_3 + \beta_{15}x_2^2x_3^2 = 0 \\ f_2 &= \beta_{21} + \beta_{22}x_3^2 + \beta_{23}x_1^2 + \beta_{24}x_3x_1 + \beta_{25}x_3^2x_1^2 = 0 \\ f_3 &= \beta_{31} + \beta_{32}x_1^2 + \beta_{33}x_2^2 + \beta_{34}x_1x_2 + \beta_{35}x_1^2x_2^2 = 0. \end{aligned} \tag{2}$$

The classical theory provides the bound by Bézout's theorem on the number of isolated roots in the projective complex space, which is simply the product of the polynomials' total degrees [27]. The bound given by Bézout's theorem here is $4 \times 4 \times 4 = 64$.

This bound is too large, for the system has a very special shape. Most importantly, each f_i does not contain x_i. In addition, not all monomials of degree 4 in the remaining two variables are present, since the degree of each equation in each variable is bounded by 2. In other words, regarding only the total degree overestimates the total number of solutions. The phenomenon of "sparse" polynomials arises in several real-world applications, including those related to molecular structure, but also in robot kinematics and vision, in computer-aided design, and in geometric modeling. The fact that certain terms are missing can be exploited by the theory of sparse (or toric) variable elimination.

Sparse elimination theory, starting with Bernstein's theorem (or BKK bound) [27, 28], exploits the monomial (or term) structure of the polynomials, when roots at projective infinity are of no interest. In this model of sparseness, let us consider each three-dimensional exponent vector, corresponding to a nonvanishing monomial in x_1, x_2, x_3, as a point in $\mathbb{Z}^3$. The convex hull of all points is the Newton polytope of the polynomial. The *mixed volume* of 3 polytopes C_1, C_2, $C_3 \in \mathbb{R}^3$ is a real-valued function that generalizes Euclidean volume. A complete account of sparse elimination, including equivalent definitions for mixed volume in general dimension, and an efficient algorithm for its computation, can be found in Refs. [1, 27, 29] and references therein.

In our case the polytopes C_i are squares of size 2. Bernstein's theorem bounds the number of common roots with no zero coordinates by the mixed volume of the Newton polytopes. The mixed volume of the present system is 16, so Bernstein's theorem states that the number of complex isolated roots of system (2), counting multiplicities, with $x_1 \neq 0$, $x_2 \neq 0$, $x_3 \neq 0$ is at most 16. In fact, this bound is optimal, as shown in Section 5.2 by exhibiting an example of system (2) with 16 solutions, which are all real.

This bound is, in our case, equivalent to a variant of Bézout's theorem for multi-homogeneous polynomials, known as the *m*-Bézout bound (see Ref. [27]). If we homogenize each equation with respect to each variable x_1, x_2, x_3, we obtain a system of 3 multi-homogeneous equations with roots in $\mathbb{P}^1 \times \mathbb{P}^1 \times \mathbb{P}^1$, where $\mathbb{P}^1$ is the projective space of dimension 1. The degree vectors of these equations are (0, 2, 2), (2, 0, 2), (2, 2, 0). We can bound the number of isolated roots by the coefficient of $h_1h_2h_3$ in the polynomial $(2h_1 + 2h_2)(2h_1 + 2h_3)(2h_2 + 2h_3)$, which also yields 16. In general, the bound provided by the mixed volume is tighter than, or equal to, the *m*-Bézout bound, which is never looser than the classical Bézout bound.

4.2. SYSTEM SOLVING BY RESULTANTS

Resultants reduce the nonlinear problem, such as the one in the previous section, to a question in linear algebra. Matrix computations are then sufficient. Hence we also refer to this procedure as a linearization of the input problem in terms of matrices and determinants. An important feature of our method is precisely that it reduces to matrix operations for which powerful and accurate implementations already exist in the public domain. A more detailed introduction to these methods can be found in Refs. [27, 28, 30].

Recall that the resultant expresses the solvability of an overconstrained system of $n + 1$ polynomials in n unknowns. The resultant is itself a polynomial in the coefficients of the system such that it vanishes exactly when the coefficients are specialized so as

for the overconstrained system to have a common root. More information on different kinds of resultants can be found in Refs. [27, 28].

Let us now concentrate on well-constrained systems of n polynomial equations in n unknowns, which is the main object of our study and, in particular, on zero-dimensional sets of solutions. The same formulation is extendible to higher dimensional sets. The overall approach has the following stages:

1. Given a well-constrained polynomial system, define an overconstrained system such that the resultant of the latter will allow us to compute all roots of the original system. This shall be examined in depth in Section 4.3.
2. Define a matrix expressing the resultant of the overconstrained system. Ideally, we would like to have a matrix whose determinant equals the resultant, just as in the univariate or linear case. Otherwise, we may express the resultant as the ratio of two determinants, or, in general, obtain a matrix whose determinant is a nontrivial multiple of the resultant.
3. Given the resultant matrix, obtain a square matrix whose eigenvalues correspond to a coordinate in every solution vector. The rest of the coordinates are obtained by the eigenvectors of the matrix, because they contain the values of certain monomials at the roots. There must be sufficiently many such monomials so as to derive all root coordinates. Root finding is therefore reduced to an eigendecomposition and then existing algorithms are employed from numerical linear algebra; see Section 4.4.

Let us first survey methods for constructing the resultant matrix of an overconstrained system (cf. Refs. [27, 28]). A first class of methods is based on Sylvester, or Macaulay-type, matrices, which rely on the idea that the system is generic in some sense and whose entries are linear in the polynomial coefficients. In the classical context, the resultant's degree is a function of the classical Bézout bound, i.e., the product of total degrees. In particular, the specific matrix named after Macaulay is defined for homogenized polynomials (obtained by introducing a homogenization variable), and its size depends again on Bézout's bound. This formula expresses the resultant as the ratio of two determinants, where the numerator is the Macaulay matrix itself, and the denominator a well-defined submatrix. This rational formula has been extended to the case of sparse resultants. For more details, see Ref. [28] and the references therein.

In sparse elimination, the sparse resultant has a degree dependent on the mixed volume of the Newton polytopes of the equations. The sparse resultant approach generalizes the well-known Sylvester resultant for two univariate polynomials into several multivariate polynomials as well as Macaulay's algorithm for dense polynomials. Several algorithms have been proposed for constructing sparse resultant matrices whose determinant is a nontrivial multiple of the sparse resultant and which express multiplication in the quotient ring. The size of these matrices scales with mixed volume. In fact, the optimal size of a Sylvester-type matrix equals the sum of the mixed volumes of all well-constrained $n \times n$ systems, where n is the number of variables in the overconstrained system.

The second class of methods makes fewer genericity assumptions. The constructed matrices generalize the constructions proposed by Bézout for the resultant of two polynomials in one variable as early as 1779 and are typically smaller than sparse resultant matrices but the coefficients are no longer linear in the coefficients of the input polynomials. This technique can be used for any polynomials, either homogeneous or not, including sparse polynomials.

The matrices we obtain from the Sylvester and Macaulay-type formulae are typically sparse; moreover, they exhibit a structure that generalizes Toeplitz structure, hence they are called quasi-Toeplitz [31]. This reduces the complexity of the linear algebra by almost one order of magnitude in terms of matrix dimension. Matrices coming from the Bézout, or Dixon, formulae are dense but are also characterized by some type of structure [32].

4.3. DEFINING AN OVERCONSTRAINED SYSTEM

Let us now sketch the two existing approaches in order to define an overconstrained algebraic system from a well-constrained one. We focus on Sylvester-type resultant matrices; we also indicate how to recover all common roots of the well-constrained system, given an algorithm to construct such matrices:

$$M = \begin{bmatrix}
c_{11} & c_{12} & c_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{11} & c_{12} & c_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & c_{11} & c_{12} & c_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & c_{11} & c_{12} & c_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_{31} & 0 & c_{32} & 0 & 0 & c_{33} & 0 & 0 & c_{34} & 0 & c_{35} & 0 & 0 & 0 & 0 & 0 \\
0 & c_{31} & 0 & c_{32} & 0 & 0 & c_{33} & 0 & 0 & c_{34} & 0 & c_{35} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & c_{31} & 0 & c_{32} & 0 & 0 & c_{33} & 0 & 0 & c_{34} & 0 & c_{35} & 0 \\
0 & 0 & 0 & 0 & 0 & c_{31} & 0 & c_{32} & 0 & 0 & c_{33} & 0 & 0 & c_{34} & 0 & c_{35} \\
c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{21} & 0 & 0 & 0 & c_{22} & 0 & 0 & 0 & c_{23}
\end{bmatrix}.$$

Given is a system of n equations in n unknowns,

$$\begin{aligned}
F_1(x_1, x_2, \ldots, x_n) &= 0 \\
F_2(x_1, x_2, \ldots, x_n) &= 0 \\
&\vdots \\
F_n(x_1, x_2, \ldots, x_n) &= 0.
\end{aligned} \tag{3}$$

The first approach "hides" one variable of the well-constrained system in the coefficient field; let the hidden variable be x_1. The resultant, $R(x_1)$, is obtained by eliminating the variables $x_2, x_3, \ldots, x_n$ from the above equations. This yields a polynomial in x_1, whose roots correspond to the x_1 coordinate of each solution of the given multivariate system. The degree of the resultant is equal the total number of nontrivial solutions of the given system of equations and in general, corresponds to the Bernstein (or BKK) bound for sparse systems and Bézout's bound for dense polynomial systems.

In principle, the hidden variable must take discrete values among the different solutions, for reasons of numerical accuracy. For example, if we consider variable x_3 as part of the coefficient field, the three equations of system (2) can be viewed as bivariate:

$$\begin{aligned}
f_1 &= c_{11} + c_{12}x_2 + c_{13}x_2^2 = 0, \\
f_2 &= c_{21} + c_{22}x_1 + c_{23}x_1^2 = 0, \\
f_3 &= c_{31} + c_{32}x_2^2 + c_{33}x_1x_2 + c_{34}x_1^2 + c_{35}x_1^2x_2^2 = 0.
\end{aligned} \tag{4}$$

The resultant matrix of this system has entries c_{ij}, which are quadratic polynomials in x_3. The sparse resultant has total degree in the coefficients equal to the sum of the mixed volumes of the bivariate well-constrained subsystems. Each of the three mixed volumes (in x_1, x_2) is equal to 4; therefore, the sparse resultant degree is 12 in the $c_{i,j}$. This equals the dimension of the optimal Sylvester-type matrix. The 16×16 matrix M is shown above; it is the resultant matrix constructed by both greedy and incremental algorithms, described in Refs. [29,b]. Matrix M expresses a quadratic matrix polynomial in x_3, and its determinant is a multiple of the sparse resultant. Solving this determinant for x_3 gives us the values of x_3 at the roots, but determinant calculation is numerically ill-conditioned and is therefore to be avoided.

The key observation is that each kernel vector of M expresses the evaluation of the following vector of monomials at the roots of x_1, x_2. The monomial vector is precisely the sequence of monomials indexing the columns of M:

$$[1, x_2, x_2^2, x_2^3, x_1, x_1x_2, x_1x_2^2, x_1x_2^3, x_1^2, x_1^2x_2, x_1^2x_2^2, x_1^2x_2^3, x_1^3, x_1^3x_2, x_1^3x_2^2, x_1^3x_2^3].$$

To recover the value of x_1, x_2 at the roots, it suffices to divide certain vector entries. For instance, to find the value of x_2, we divide the second entry by the first. However innocuous this operation may seem, the choice of which entries to use is an important numerical issue that has to be studied further. Results vary significantly according to this choice.

But how do we recover such a vector? It can be shown (see Section 4.4) that it can be computed via a matrix eigendecomposition, which is well conditioned numerically. For the particular example, the standard procedure constructs a 32×32 companion matrix whose eigenvalues and eigenvectors are the x_3 values and the kernel vectors of M. Since there are 32 eigenvalue and eigenvector pairs, exactly one-half of them will not correspond to solutions.

The second way to define an overconstrained system is to add a polynomial of our choice to system (3). This is the u-resultant approach, where the new polynomial is linear in the variables. Specifically, we append polynomial

$$F_{n+1}(x_1, x_2, \ldots, x_n) = u_0 + u_1x_1 + \cdots + u_nx_n$$

to the given system of equations. The resultant is obtained by eliminating the variables $x_1, \ldots, x_n$ from the $n + 1$ equations and is a polynomial in $u_0, u_1, \ldots, u_n$. Given resultant matrix M, whose entries are polynomials in the u_i's, the resultant corresponding to its determinant can be factored into linear factors of the form [27, 28]:

$$\det(M) = \prod_{i=1}^{k} (\alpha_{i0}u_0 + \alpha_{i1}u_1 + \cdots + \alpha_{in}u_n),$$

where k is the total number of nontrivial solutions and $(\alpha_{i0}, \alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{in})$ are the projective coordinates of a solution of the given system (3) of equations.

As an illustrative example, we keep using system (2). If we choose polynomial $u_0 + u_1x_1$, with u_0, u_1 indeterminate, the mixed volumes of the three-polynomial systems are 4, 4, 4, 16; hence the sparse resultant degree is 4 in the variables $\beta_{i,1}, \ldots, \beta_{i,5}$, 16 in the variables u_0, u_1 and its total degree is 28. The incremental algorithm was applied and the smallest matrix found is of size 48. Typically, u_1 will take some random value; then, finding all common roots of the original system is reduced to factoring the resultant as a polynomial in u. Alternatively, system solving can again be reduced to an eigendecomposition, or to univariate polynomial solving by means of the primitive element method [33], also known as rational univariate representation.

4.4. SYSTEM SOLVING BY EIGENDECOMPOSITIONS

In this section, we show how system solving reduces to computing eigenvalues and eigenvectors, given a resultant matrix; for more information see Refs. [27, 28, 34–37, a, c]. In general, resultants linearize a nonlinear polynomial system.

In practice, matrix operations will be carried out by efficient numerical linear algebra software such as LAPACK, which is used in the sparse-resultant solver in Ref. [1]. One of the advantages of LAPACK is that it implements state-of-the-art algorithms, offering the choice of a tradeoff between speed and accuracy, and provides efficient ways for computing estimates on the condition numbers and error bounds [38].

Resultant matrices take a well-constrained system of nonlinear polynomial equations, and reduce it to a linear system of the form

$$M(x_1)[1, x_2, \ldots, x_n, \ldots, x_2^d, x_3^d, \ldots, x_n^d]^T = [0, 0, \ldots, 0]^T, \quad (5)$$

where matrix $M(x_1)$ is a square matrix and its entries are polynomials in x_1; the latter can be some hidden variable or parameter u_0. The entries of the vector consist of power products of $x_2, x_3, \ldots, x_n$. The actual arrangement of the power products of these variables is a function of the degrees of the polynomials and the formulation of resultant being used. The above example refers to the case of Macaulay matrices, where all products up to a certain degree are present. In the case of sparse resultants, the monomials correspond to integer points inside some convex polytope in n-dimensional space [b].

In any case, the linearization of the problem has the property that for any given solution $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, of the given system, $M(\alpha_1)$ is a singular matrix and the vector in its kernel is obtained by substituting $x_1 = \alpha_1, x_2 = \alpha_2, \ldots, x_n = \alpha_n$ in the vector consisting of power products in Eq. (5). We use this property along with those of matrix polynomials to compute the common solution of the polynomial equations.

The resultant matrix $M(x_1)$ can be expressed as a matrix polynomial:

$$M(x_1) = M_0 + M_1x_1 + M_2x_2 + \cdots + M_lx_1^l, \quad (6)$$

where the M_i are $m \times m$ numeric matrices and l is the maximum degree of x_1 in any term of $M(x_1)$. All

M_i have the same order, say $m \times m$. The determinant of $M(x_1)$ corresponds exactly to the resultant of the given equations.

Expression (6) is a matrix polynomial, and the problem of computing the roots of the original system of polynomial equations corresponds to computing generalized eigenvalues and eigenvectors of a matrix polynomial. The given matrix polynomial is regular, if the univariate polynomial $\det[M(x_1)]$ is not identically zero. Otherwise, it is called singular.

The eigenvalues of a matrix polynomial consists of finite and infinite eigenvalues. In particular, when rank $M_l = m$, the determinant of $M(x_1)$ is a polynomial of degree ml, and all the eigenvalues of $M(x_1)$ are finite. When rank $M_l < m$, the degree of $\det[M(x_1)]$, say k, is less than ml. In this case, $M(x_1)$ has k finite eigenvalues and $(ml - k)$ infinite eigenvalues. In most applications we are only interested in computing the affine solutions of the original system of polynomial equations. Therefore, we are interested in computing the finite eigenvalues of a matrix polynomial. It is possible that the resultant formulation for a (sparse) polynomial system can result in a singular matrix polynomial. In such cases, we are still interested in computing the finite eigenvalues only.

Let us consider the case, when the matrix polynomial is regular. Furthermore, we start our analysis by assuming that the leading matrix of the matrix polynomial, M_l is nonsingular and well conditioned. As a result, computation of M_l^{-1} does not introduce severe numerical errors. Let

$$\bar{M}(x_1) = M_l^{-1} M(x_1), \quad \text{and} \quad \bar{M}_i = M_l^{-1} M_i, \qquad 0 \le i < l.$$

Now, $\bar{M}(x_1)$ is a monic matrix polynomial. Its determinant has the same roots as does the determinant of $M(x_1)$. Let $x_1 = \alpha_1$ be a root of the equation, $\det[\bar{M}(x_1)] = 0$. As a result $\bar{M}(\alpha_1)$ is a singular matrix and there is at least one nontrivial vector in its kernel. Let us denote that $m \times 1$ vector by v. Therefore,

$$\bar{M}(\alpha_1) v = 0, \tag{7}$$

where 0 is a $m \times 1$ null vector. The roots of the determinant of $M(x_1)$ correspond to the eigenvalues of matrix C, defined as follows [31] (cf. also Refs. [1, 27]):

$$C = \begin{bmatrix} 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & I_m & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I_m \\ -\bar{M}_0 & -\bar{M}_1 & -\bar{M}_2 & \cdots & -\bar{M}_{l-1} \end{bmatrix},$$

where 0 and I_m are $m \times m$ null and identity matrices, respectively. Furthermore, the eigenvector of C corresponding to the eigenvalue $x_1 = \alpha_1$ has the form

$$[v \quad \alpha_1 v \quad \alpha_1^2 v \ldots \alpha_1^{l-1} v]^T,$$

where v is the vector in the kernel of $\bar{M}(\alpha_1)$ as highlighted in expression (7).

Many times the leading matrix M_l is singular or close to being singular (due to high condition number). Some techniques based on linear transformations are highlighted in Ref. [30], such that the problem of finding roots of the determinant of a matrix polynomial can be reduced to an eigenvalue problem. However, there are cases in which they may not work. For example, when the matrices have singular pencils. In such cases, we reduce the intersection problem to a generalized eigenvalue problem as follows.

Given matrix polynomial, $M(x_1)$ the roots of the polynomial corresponding to its determinant are the eigenvalues of the generalized system $C_1 x_1 - C_2$, where [30]:

$$C_1 = \begin{bmatrix} I_m & 0 & 0 & \cdots & 0 \\ 0 & I_m & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & I_m & 0 \\ 0 & 0 & \cdots & 0 & M_l \end{bmatrix},$$

$$C_2 = \begin{bmatrix} 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & I_m & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I_m \\ -M_0 & -M_1 & -M_2 & \cdots & -M_{l-1} \end{bmatrix}.$$

In many applications, we are interested only in the real solutions or solutions lying in a particular domain. The QR or QZ algorithm for eigenvalue computation returns all the eigenvalues of a given matrix, and it is difficult to restrict it to eigenvalues in a particular domain [39]. Algorithms to compute selected eigenvalues of the these matrices based on power iterations and their structure are given in Ref. [36].

Let us assume that α_1 is a simple eigenvalue of C. In the rest of the section, we carry out the analysis on the eigenvalues of C and the resulting algorithm is similar for the finite eigenvalues of the pencil $C_1x_1 - C_2$. Since α_1 is a simple eigenvalue, the kernel of $C - \alpha_1 I$ has dimension one, represented as

$$V = [v \quad \alpha_1 v \quad \alpha_1^2 v \ldots \alpha_1^{l-1} v]^T.$$

Furthermore, we know that $v = [v_1 \quad v_2 \ldots v_m]^T$ corresponds to the vector in the kernel of $M(\alpha_1)$. Given v, it is possible to compute the $x_2, \ldots, x_n$ coordinates.

In many cases α_1 may correspond to an eigenvalue of multiplicity greater than one. There are two possibilities:

$(\alpha_1, \alpha_2, \ldots, \alpha_n)$ is a solution of multiplicity greater than one of the given system of equations.

There may be two solutions of the given equations of the form $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $(\alpha_1, \alpha_2', \ldots, \alpha_n')$. As a result the kernel of $C - \alpha_1 I$ has dimension greater than one.

The problem of computing higher multiplicity roots can be numerically ill-conditioned. However, in many cases it is possible to identify higher multiplicity eigenvalues of a matrix by identifying clusters of eigenvalues and using the knowledge of the condition number of the clusters. More details of its application to finding solutions of polynomial equations are given in Ref. [30]. Such analysis is well developed for eigenvalues of a matrix.

Given a higher multiplicity eigenvalue, α_1, we compute its geometric multiplicity by computing the SVD of $C - \alpha_1 I$. The geometric multiplicity corresponds to the number of singular values equal to zero. In case, the geometric multiplicity is one, the relationship in expression (5) is used to compute $\alpha_2, \alpha_3, \ldots, \alpha_n$ for each α_1. Otherwise there are two or more vectors in the kernel of $M(\alpha_1)$. The vectors computed using linear algebra routines may correspond to any two vectors in the vector space corresponding to the kernel. Hence, to solve the problem, we substitute $x_1 = \alpha_1$ in the n equations (3) and solve them for the rest of the unknowns. This procedure is applied recursively.

4.5. MATRIX COMPUTATIONS

In this section, we briefly review some techniques from linear algebra and numerical analysis, which have been used in our algorithms. More details can be found in Refs. [39, 40].

Given an $n \times n$ matrix A, its eigenvalues and eigenvectors are the solutions to the equation

$$Ax = sx,$$

where s is the eigenvalue and $x \neq 0$ is the eigenvector. The eigenvalues of a matrix are the roots of its characteristic polynomial, corresponding to $\det(A - sI)$. As a result, the eigenvalues of a diagonal matrix, upper triangular matrix or lower triangular matrix correspond to the elements on its diagonal. Efficient algorithms for computing eigenvalues and eigenvectors are well known, and their implementations are available as part of packages EISPACK and LAPACK [38].

Given $n \times n$ matrices, A and B, the generalized eigenvalue problem corresponds to solving

$$Ax = sBx.$$

We represent this problem as eigenvalues of $A - sB$. The vectors $x \neq 0$ correspond to the eigenvectors of this equation. If B is nonsingular and its condition number is low, the problem can be reduced to an eigenvalue problem by multiplying both sides of the equation by B^{-1} and thereby obtaining

$$B^{-1}Ax = sx.$$

However, B may have a high condition number and such a reduction may be numerically unstable. A better algorithm is the QZ algorithm, which is in EISPACK and in LAPACK [38].

The singular value decomposition (SVD) is a powerful tool that gives us accurate information about matrix rank in the presence of round off errors. The rank of a matrix can also be computed by Gauss elimination. However, many situations arise in which near rank deficiency prevails. Rounding errors and fuzzy data make rank determination a nontrivial exercise. In these situations, the numerical rank is easily characterized in terms of the SVD. Given A, an $m \times n$ real matrix, matrices U and V exist such that

$$A = U\Sigma V^T,$$

where U is an $m \times n$ orthogonal matrix, V is an $n \times n$ orthogonal matrix and Σ is a $n \times n$ diagonal matrix of the form $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$. More-

over, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. The σ_i values are called the singular values and the columns of U and V, denoted as u_i and v_j values are known as the left and right singular vectors, respectively.

The largest or the smallest eigenvalue of a matrix (and the corresponding eigenvector) can be computed using the power method. The power method involves multiplication of a matrix by a vector, and after a few steps it converges to the largest eigenvalue. Given a matrix, A, the technique starts with a vector q_0 and performs computation of the form

$$z_i = Aq_{i-1}, \qquad q_i = z_i/\|z_i\|, \qquad \lambda_i = q_i^T A q_i.$$

After a few iterations, λ_k corresponds to the eigenvalue of maximum magnitude, and q_k is the corresponding eigenvector.

The formulation of resultant matrices of Sylvester (or Macaulay) type leads to sparse matrices [30]. In such cases we want to make use of the sparsity of the matrix in computing its eigendecomposition. The sparsity of the matrix increases with the degrees of the polynomials or the number of equations. Algorithms for sparse matrix computations are based on matrix–vector multiplication, as highlighted in the power method. For our applications, we use the algorithm highlighted in Ref. [41] for computing the invariant subspaces and thereby the eigendecomposition of a sparse matrix.

5. Molecular Conformations

We apply our algebraic tools to computing all conformations of ring molecules. First, we model the problem in terms of polynomial equations.

5.1. MODELING RING CLOSURE

Two different modeling approaches are sketched here, following Ref. [2]. The first is the direct approach based on Ref. [9]. The molecule has a cyclic backbone of 6 atoms, typically of carbon. Carbon-hydrogen or other bonds outside the backbone are ignored. Backbone atoms are points $p_1, \ldots, p_6 \in \mathbb{R}^3$; the unknown dihedrals are about axes (p_6, p_1) and (p_{i-1}, p_i) for $i = 2, \ldots, 6$, respectively. The triangles (p_1, p_2, p_6), (p_2, p_3, p_4) and (p_4, p_5, p_6) are fixed for constant bond lengths L_i and bond angles ϕ_1, ϕ_3, ϕ_5. Then, base triangle (p_2, p_4, p_6) is fixed in space, defining the xy-plane of a coordinate frame. Let θ_1 be the (dihedral) angle between the plane of triangle (p_1, p_2, p_6) and that xy-plane. For any conformation, θ_1 is well defined and, similarly, we define angles θ_2 and θ_3. We call them flap (dihedral) angles. Conversely, given the L_i, ϕ_i and θ_i, the coordinates of all p_i are uniquely determined, and hence the bond dihedral angles and the associated conformation are all well defined. We have therefore reduced the problem to computing the three θ_i which satisfy the constraints on ϕ_2, ϕ_4, ϕ_6. Hence we obtain a polynomial system:

$$\begin{aligned}
&\alpha_{11} + \alpha_{12}\cos\theta_2 + \alpha_{13}\cos\theta_3 + \alpha_{14}\cos\theta_2\cos\theta_3 \\
&\qquad + \alpha_{15}\sin\theta_2\sin\theta_3 = 0, \\
&\alpha_{21} + \alpha_{22}\cos\theta_3 + \alpha_{23}\cos\theta_1 + \alpha_{24}\cos\theta_3\cos\theta_1 \\
&\qquad + \alpha_{25}\sin\theta_3\sin\theta_1 = 0, \\
&\alpha_{31} + \alpha_{32}\cos\theta_1 + \alpha_{33}\cos\theta_2 + \alpha_{34}\cos\theta_1\cos\theta_2 \\
&\qquad + \alpha_{35}\sin\theta_1\sin\theta_2 = 0, \\
&\cos^2\theta_i + \sin^2\theta_i - 1 = 0, \qquad i = 1, 2, 3, \qquad (8)
\end{aligned}$$

where the α_{ij} are input coefficients. For our solvers we prefer an equivalent formulation with a smaller number of polynomials, obtained by applying transformation (1) to the tangents of the half-angles, which leads to rational equations in the new unknowns $x_i = \tan(\theta_i/2)$, where $i = 1, 2, 3$. This transformation automatically captures the last three equations in (8). By multiplying both sides of the ith equation by $(1 + x_j^2)(1 + x_k^2)$, where (i, j, k) is a permutation of $\{1, 2, 3\}$, we arrive at polynomial system (2):

$$\begin{aligned}
f_1 &= \beta_{11} + \beta_{12}x_2^2 + \beta_{13}x_3^2 + \beta_{14}x_2x_3 + \beta_{15}x_2^2x_3^2 = 0 \\
f_2 &= \beta_{21} + \beta_{22}x_3^2 + \beta_{23}x_1^2 + \beta_{24}x_3x_1 + \beta_{25}x_3^2x_1^2 = 0 \\
f_3 &= \beta_{31} + \beta_{32}x_1^2 + \beta_{33}x_2^2 + \beta_{34}x_1x_2 + \beta_{35}x_1^2x_2^2 = 0,
\end{aligned}$$

where β_{ij} are input coefficients.

Another formulation of the problem relying on the geometry of distances is presented. This is a rather general methodology with numerous applications. Consider the configurations of points $p_1, \ldots, p_6$, and consider an additional artificial point p_0, such that all distances to p_0 equal 1. Now define a 7×7 symmetric matrix whose entry $(i + 1, j + 1)$ is the squared distance between p_i and p_j. This is the Cayley–Menger matrix of points $p_1, \ldots, p_6$; see, for instance, Refs. [10, 11]:

$$\begin{array}{c} \\ p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} p_0 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & d(p_1, p_2)^2 & d(p_1, p_3)^2 & u_1 & d(p_1, p_5)^2 & d(p_1, p_6)^2 \\ 1 & d(p_1, p_2)^2 & 0 & d(p_2, p_3)^2 & d(p_2, p_4)^2 & u_2 & d(p_2, p_6)^2 \\ 1 & d(p_1, p_3)^2 & d(p_2, p_3)^2 & 0 & d(p_3, p_4)^2 & d(p_3, p_5)^2 & u_3 \\ 1 & u_1 & d(p_2, p_4)^2 & d(p_3, p_4)^2 & 0 & d(p_4, p_5)^2 & d(p_4, p_6)^2 \\ 1 & d(p_1, p_5)^2 & u_2 & d(p_3, p_5)^2 & d(p_4, p_5)^2 & 0 & d(p_5, p_6)^2 \\ 1 & d(p_1, p_6)^2 & d(p_2, p_6)^2 & u_3 & d(p_4, p_6)^2 & d(p_5, p_6)^2 & 0 \end{bmatrix} \end{array},$$

where $d(p_i, p_j)$ stands for the Euclidean distance between points p_i, p_j. There are only three unknowns u_1, u_2, u_2. Once these unknowns are computed, we can recover the geometry of the molecule, up to global translations and rotations.

It is known that this matrix is at most of rank 5, if and only if the points $p_1, \ldots, p_6$ with the respective distances can be embedded in $\mathbb{R}^3$. Since every 6×6 minor of the matrix vanishes, we derive several constraints on the parameters u_i. For instance, taking the principal minors of column (respective row) indices (1, 2, 3, 4, 5, 6) of this matrix, we obtain an equation $P_1(u_1, u_2) = 0$ in the variables u_1, u_2, which is of degree 2 in u_1 and u_2. In a similar way, we can derive constraints $P_2(u_1, u_3) = 0$ and $P_3(u_2, u_3) = 0$ by considering the principal minor of indices (1, 2, 3, 4, 5, 7) and (1, 2, 3, 5, 6, 7). This leads to a system identical to (2).

5.2. CYCLOHEXANE CONFORMATIONS

This section examines resultant methods from Ref. [4], as well as matrix formulae of the sparse resultant from Refs. [1, 2, 42], and how they were used to compute all conformations of ring molecules. Here, we limit our discussion to cyclic molecules of 6 degrees of freedom, which are modeled by system (2). Problems with fewer and more degrees of freedom are examined in Refs. [2, 4].

We have chosen to hide x_3 in the coefficient field hence arriving at system (4). Our procedure constructs the matrix by the incremental algorithm of Ref. [29] with generic coefficients, therefore independent of the input instance and the arithmetic precision. This is matrix M from Section 4.2. The program then applies the standard eigendecomposition routines of LAPACK to the corresponding companion matrix; more information is included in Refs. [1, 41]. For every system the coefficients of system (4) are specialized to integer polynomials in the hidden variable x_3. Recall that the inputs are the β_{ij}, defined in Section 5.1.

The first instance refers to the symmetric cyclohexane molecule, which has 6 carbon atoms at equal distances and equal bond angles. Usually noise enters in the process that produces the coefficients. To illustrate this phenomenon, starting with the pure cyclohexane, we randomly perturb the data by about 10% to obtain β_{ij} as the entries of matrix:

$$\begin{bmatrix} -310 & 959 & 774 & 1389 & 1313 \\ -365 & 755 & 917 & 1451 & 1269 \\ -413 & 837 & 838 & 1655 & 1352 \end{bmatrix}.$$

In this case, we find the four solutions of Table I (see Ref. [2]). Each root contains at least 8 correct digits, which is largely sufficient in chemical applications. A chemist will immediately recognize two chair and two twisted boat, or crown, conformations, shown in Figure 4.

Now let the coefficients be $(\beta_{i1}, \ldots, \beta_{i5}) = -13, -1, -1, 24, -1$. This has the maximum number of roots, namely 16, and furthermore, they are all real (see Table II). The roots are correct to at least 7 decimal digits, which is sufficient for chemical applications. Certain roots correspond to a triple eigenvalue, so we recover the values of x_1, x_2 by substituting x_3 in the original system and by solving the resulting univariate equations.

TABLE I

Four possible real values of the tangents in the case of the symmetric cyclohexane.

x_1	x_2	x_3
0.3684363946	0.3197251270	0.2969559367
−0.3684363946	−0.3197251270	−0.2969559367
0.7126464332	−0.01038413185	−0.6234532743
−0.7126464332	0.01038413185	0.6234532743

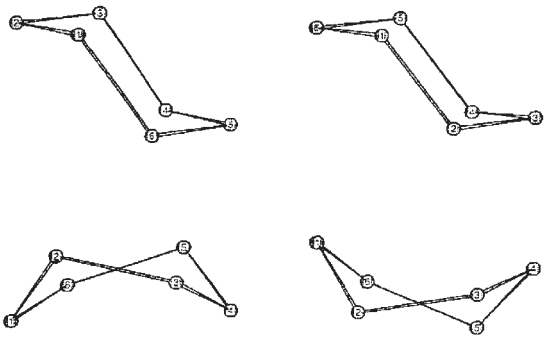

FIGURE 4. The four conformations of the symmetric cyclohexane.

The last system is defined by $(\beta_{i1}, \ldots, \beta_{i5}) = (-\sqrt{3}/2, 1/2, 1/2, 2, \sqrt{3}/2)$. The system that we actually solve is obtained by taking the floating-point approximations of the coefficients β_{ij} with 5 digits. There are 16 complex roots, of which the real ones are in Table III. The first two correspond to isolated points, whereas the last 6 points are perturbations of points that lie on a one-dimensional component of the solution set, when taking rational inputs [2].

6. Pharmacophore Matching

A fundamental assumption in structural biology is that the power of interaction between two molecules strongly depends on the existence of geometrical complementarity between them. The same assumption holds true in pharmacology, where the most common interaction model involves docking a (small) ligand in the pocket of a (large) receptor, thereby triggering or hindering a biochemical process. Furthermore, it is often assumed that only certain parts of the ligand play a significant role in the interaction with the receptor, while the rest of the molecule serves mainly as a scaffold, allowing the active groups to be properly positioned in space. Thus, every ligand that interacts with a certain receptor according to a certain model of interaction, has to contain certain chemical groups, properly positioned in space. This gives rise to an abstract concept of a 3D pattern that characterizes the potential of a ligand to show chemical activity. This 3D pattern, called a pharmacophore [43], can be equivalently regarded as a set of geometric constraints imposed on the structure of the ligand. The latter has to be able to satisfy the constraints, in order to be characterized as potentially active.

We present an algebraic methodology for the solution of such kinds of geometric constraints on molecules, based on Ref. [3]. We focus on constraints involving only distances between certain points of the molecular structure. We should mention that a set of predefined intermolecular distances is a common, though not standardized, way to quantitatively describe a pharmacophore. Apart from a set of distances, a pharmacophore definition can also include other features, such as angles or planarity constraints. The problem can be formulated as follows: given the 3D structure of a molecule and a set of predefined distances between points of the molecule, compute all conformations that satisfy the distance constraints. Thereby, we adopt the assumption of fixed bond lengths and

TABLE II
All solutions for the example with 16 real roots.

x_1	x_2	x_3
10.85770360	0.7795480449	0.7795480451
−10.85770360	−0.7795480449	−0.7795480451
0.3320730984	4.625181601	4.625181601
−0.3320730984	−4.625181601	−4.625181601
0.7795480449	0.7795480449	0.7795480451
0.7795480449	10.85770360	0.7795480451
0.7795480440	0.7795480457	10.85770360
4.625181601	4.625181601	4.625181601
4.625181601	0.3320730984	4.625181601
4.625181600	4.625181613	0.3320730984
−0.7795480440	−0.7795480457	−10.85770360
−0.7795480449	−0.7795480449	−0.7795480451
−0.7795480449	−10.85770360	−0.7795480451
−4.625181600	−4.625181613	−0.3320730984
−4.625181601	−4.625181601	−4.625181601
−4.625181601	−0.3320730984	−4.625181601

TABLE III
All solutions for the third example.

x_1	x_2	x_3
0.5176444559	0.5176444559	0.5176444563
−0.5176444567	−0.5176444567	−0.5176444555
0.5176444559	−1.931851652	0.5176444563
−0.5176444567	1.931851652	−0.5176444555
1.931851652	−0.5176444567	−0.5176444555
−1.931851652	0.5176444559	0.5176444563
0.5176444561	0.5176444561	−1.931851652
−0.5176444561	−0.5176444561	1.931851652

angles and model the problem in the space of torsional angles.

6.1. MODELING DISTANCE CONSTRAINTS

Using Figure 1, we show how we can express the distance $d(X, Y)$ between the points X and Y as a function of the torsional angles $t_1, \ldots, t_n$. Let the position of X with respect to F_X be x and the position of Y with respect to F_Y be y, both of which are known and fixed vectors. Then, the position of Y with respect to F_X will be $y' = R(t_1, \ldots, t_n)y$. In order to express the distance $d(X, Y)$ in terms of the torsional angles, all we have to do is compute the length of the vector $\overline{XY}$ with respect to F_X.

$$d(X, Y) = \|\overline{XY}\| = \|y' - x\| = \|R(t_1, \ldots, t_n)y - x\|.$$

In the above equation, $d(X, Y)$ is expressed as a trigonometric function of $t_1, \ldots, t_n$. In Lemma 1 we will show that d^2 is multilinear with respect to the sines and cosines of the angles. Having expressed the distance $d(X, Y)$ as a function of the angles, a constraint of the form $d(X, Y) = d_o$ can be readily formulated as a trigonometric equation, simply by squaring both sides of the equation to eliminate the square root. As a last step, we apply the half-tangent substitution and eliminate the denominators. What remains is a polynomial equation, containing the half-tangents as unknowns. Repeating the same procedure for all distance constraints leaves us with an algebraic system.

We will close the current section by proving a result that characterizes the form of the equations. As we explained above, a distance constraint of the form $d(X, Y) = d_o$ is reduced to the trigonometric equation $\|R(t_1, \ldots, t_n)y - x\|^2 = d_o^2$. For the left-hand side of this equation, the following property is valid.

Lemma 1 The expression $\|R(t_1, \ldots, t_n)y - x\|^2$ is multilinear with respect to the sines and cosines of angles $t_1, \ldots, t_n$.

Proof:

$$\|Ry - x\|^2 = (Ry - x)^T(Ry - x) = y^TR^TRy - 2x^TRy + x^Tx.$$

Suffice it to prove that the elements of matrices R and R^TR are multilinear with respect to the sines and cosines. Matrix R is a product: $R = R_n \cdots R_2R_1$. Each R_i consists of two nontrivial parts: a 3×3 matrix $\Theta_i(t_i)$ expressing rotation, and a 3×1 column c_i expressing translation:

$$R_i = \begin{bmatrix} \Theta_i & c_i \\ 0 & 1 \end{bmatrix}.$$

We compute matrix R directly as a product:

$$R = \begin{bmatrix} \Theta_n & c_n \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} \Theta_2 & c_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Theta_1 & c_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \Theta_n \cdots \Theta_1 & c_n + \sum_{k=1}^{n-1} \Theta_n \cdots \Theta_{k+1} c_k \\ 0 & 1 \end{bmatrix}.$$

With substitution we also directly compute the product R^TR. In this computation, we use the orthonormality of the rotation matrices: $\Theta_i\Theta_i^T = \Theta_i^T\Theta_i = I$. This property makes the upper left submatrix of the product R^TR equal to the 3×3 identity matrix:

$$R^TR = \begin{bmatrix} I & \sum_{k=1}^{n} \Theta_1^T \cdots \Theta_k^T c_k \\ \sum_{k=1}^{n} c_k \Theta_k \cdots \Theta_1 & s \end{bmatrix}$$

$$s = 1 + \sum_{k=1}^{n} \|c_k\|^2 + 2 \sum_{k=1}^{n-1} \sum_{m=k+1}^{n} c_k^T \Theta_{k+1}^T \cdots \Theta_m^T c_m.$$

From the above expressions, it is clear that R and R^TR contain elements that are multilinear with respect to the sines and cosines of the angles.

We generally expect the elements of R, R^TR to contain all possible products of the form $\Pi_{i=1}^n f_i(t_i)$, where function f_i can be one of sine, cosine, or constant.

6.2. APPLICATION TO ACE INHIBITORS

In this section, we combine the modeling methodology described above with the available resultant-based tools for the solution of a concrete molecular problem. We would like to emphasize that both the modeling and the solution stages do not contain any ad hoc constructions, but are based instead on generic theoretical results and algorith-

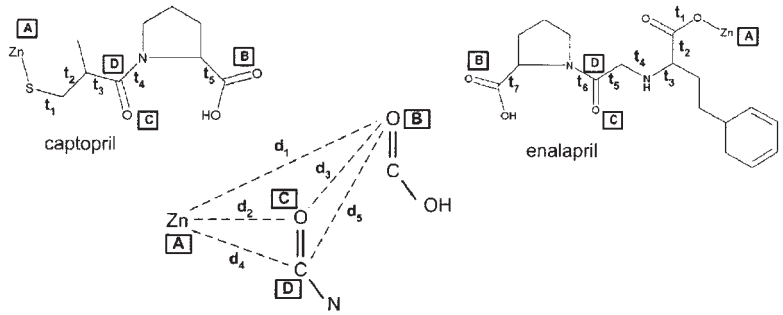

FIGURE 5. A pharmacophore for ACE inhibition.

mic implementations. Therefore, the procedure described below can be applied unchanged to other cases. For the modeling stage we have implemented a library of MAPLE routines, which take as input the 3D structure of a molecule, its connectivity table, and a set of distance constraints, and return the algebraic formulation of the constraints in terms of polynomial equations. For the solution of the resulting system, we use the public-domain standalone solver of Ref. [42].

We focus on a pharmacophore matching problem, because a pharmacophore readily provides us with the necessary geometric constraints for the formulation of our equations. The two molecules, captopril and enalapril, that serve as the material of our study, belong to the ACE inhibitors. The latter constitute a well-studied group of molecules, several of which are nowadays used as drugs against hypertension. We chose these two particular molecules, as we found them suitable for our purposes: they have few degrees of freedom, which makes it possible to apply the algebraic methods and to inspect the results visually, their formulae can be retrieved from publicly available chemical databases, and finally PDB also contains recently published reference structures of them, complexed with ACE [44].

We should note that the pharmacophore we are using below may be somewhat obsolete, since it was proposed before the determination of the actual structure of the ACE receptor. However, since our purpose is the demonstration of the algebraic methodology, this is not of major importance. The pharmacophore depicted in Figure 5 was proposed in Ref. [45] as characteristic of ACE inhibition. It contains three features: a carboxyl group, an amidocarbonyl group, and a zinc atom, which is actually part of the receptor. Five interatomic distances are specified, so as to characterize the relative positions of the pharmacophoric features. Using the information from the available structures, we calculated the five constrained distances as the averages of the values observed in the three human ACE complexes found in PDB (codes: 1UZE, 1UZF, 1O86).

For both molecules, captopril and enalapril, we retrieved their chemical formulae from ChemBank [46] as a SMILES string and generated their 3D structures using the Web interface of Corina [47]. We finally added to each one a zinc pseudo-atom, connected to the rest of the molecule via a pseudobond. In the analysis below, we treat the zinc atom as part of the ligand. We apply the algebraic methodology on each of the two modeled ligands, in order to match its structure to the requirements of the pharmacophore. Starting from the modeled molecules, which do not satisfy the distance constraints a priori, we seek to determine whether they can fold so as to satisfy the constraints and, if so, we wish to compute all appropriate conformations.

Let us first concentrate on captopril. It should be obvious that the resulting polynomial system is square, i.e. there are as many equations as unknowns. This is a fortunate coincidence in this particular case, since the molecule has five degrees of freedom and the pharmacophore imposes exactly five constraints. Most problems of this type should be underconstrained.

Applying the modeling algorithm, we derive for each distance constraint a polynomial equation, which depends on the torsional angles of all, but two, rotating bonds connecting the corresponding atoms. The bonds at the two ends of the path, obviously, do not appear in the equation, even if they can rotate, because their rotation does not affect the relative position of the constrained atoms. Thus, the resulting system for captopril has the following structure, where each polynomial p_i corresponds to distance constraint d_i. Each variable x_i

equals the tangent of one-half of the corresponding torsional angle t_i:

$$\begin{aligned} p_1(x_1, x_2, x_3, x_4, x_5) &= 0 \\ p_2(x_1, x_2, x_3) &= 0 \\ p_3(x_4, x_5) &= 0 \\ p_4(x_1, x_2) &= 0 \\ p_5(x_5) &= 0. \end{aligned}$$

Furthermore, as immediately follows from Lemma 1, each polynomial p_i is complete multiquadratic with respect to its variables. Because of space limitations, we give below an expanded expression for the third polynomial only:

$$\begin{aligned} p_3(x_4, x_5) = {} & 17.699x_4^2x_5^2 + 3.4532x_4^2x_5 + 6.0548x_4x_5^2 \\ & + 3.393x_4^2 + 4.1946x_4x_5 + 4.991x_5^2 + 8.6680x_4 \\ & + 8.0294x_5 - 3.4886. \end{aligned}$$

Each polynomial contains 3^n monomials, where n is the number of variables appearing in it. However, the overall system, considered in the five dimensional space (x_1, x_2, x_3, x_4, x_5), is clearly very sparse, which makes the resultant approach particularly well suited for its solution. We generally expect problems of this type to lead to sparse systems, especially when the pharmacophoric features are uniformly distributed along the molecular chain. In particular, they may lead to an almost triangular system like the one above, or to systems containing some decoupled subsystem.

This is what happens in this particular case. As a first step, we exploit the fact that the third and fifth equations are decoupled from the rest and, even better, that the fifth is a simple quadratic. So, we can directly solve it for x_5, substitute the solutions into the third, which then becomes a simple quadratic, and finally solve the latter for x_4. From this procedure, we expect at most four real solution pairs (x_4, x_5). We proceed by substituting each one into the first equation. Each substitution yields a 3×3 subsystem:

$$\begin{aligned} p_1(x_1, x_2, x_3) &= 0 \\ p_2(x_1, x_2, x_3) &= 0 \\ p_4(x_1, x_2) &= 0. \end{aligned}$$

For the latter, we use the tools of sparse elimination. Since the system is sparse, we expect the mixed volume to provide a tighter bound than the Bézout theorem. Indeed, the mixed volume is 32, while the Bézout bound is 144. The next step is to construct a resultant matrix that will reduce the solution of the system to an eigenproblem. Therefore, we use the algorithm described in Ref. [29], which is implemented in the software we are using. Since the resultant is defined for systems with one more equation than unknowns, we hide one of the variables in the coefficient field; i.e., we regard the system as containing three equations in two unknowns. In our case, we choose to hide x_2; the size of the optimal sparse-resultant matrix of Sylvester type equals the sum of the pairwise mixed volumes, which is 16. The algorithm used for the resultant construction returns, here, an optimal 16×16 matrix.

Having constructed the resultant matrix, the solution of the system in the field of complex numbers has been reduced to an eigenproblem for the companion matrix. In our case, as the hidden variable appears in the resultant matrix in quadratic terms, the size of the companion matrix will be 32×32. In summary, we have reduced the initial geometric problem to the solution of quadratic equations and eigenproblems of size 32. From this procedure, we obtain 24 discrete, real solutions to the overall polynomial system. It is straightforward to compute the corresponding values of the torsional angles. Table IV lists all 24 real solutions for captopril. We show the rounded values of the five torsional angles, measured in degrees. As already explained, the solutions come in four groups, each one corresponding to a different pair of (t_4, t_5). The members of each group correspond to the triplets t_1, t_2, t_3 that are solutions of the 3×3 subsystem. We observe that within each group there exist strong arithmetical symmetries. These symmetries are easy to justify. The distance constraints imposed on the molecular structure describe the relative positions of the corresponding atoms, without involving any definition of direction. Consequently, the solutions we obtain correspond to several pairs of enantiomers, i.e., conformers that are related to each other like mirror images. In fact, the structure of this particular molecule and the particular set of constraints define several planes of symmetry, with respect to which conformational enantiomers are generated, all of them satisfying the distance constraints. We believe that the generation of enantiomers as solutions may be inherent to problems

TABLE IV

The 24 real solutions for captopril, rounded to the nearest integer, containing several symmetries.

$t_5 = -99$		$t_4 = -79$	$t_5 = -99$		$t_4 = 34$
t_3	t_2	t_1	t_3	t_2	t_1
−151	43	−69	151	−43	108
−151	−61	108	151	61	−69
−103	−43	108	103	43	−69
−103	61	−69	103	−61	108
$t_5 = 55$		$t_4 = -94$	$t_5 = 55$		$t_4 = -22$
t_3	t_2	t_1	t_3	t_2	t_1
−103	125	−151	103	−125	−170
−103	83	−170	103	−82	−151
−132	−125	−170	132	125	−151
−132	−83	−151	132	82	−170
110	45	−69	−110	−45	108
110	−58	108	−110	58	−69
147	−45	108	−147	45	−69
147	58	−69	−147	−58	108

involving only distance constraints and no definition of direction. Finally, we should note that in this case the symmetry is not perfect. For two pairs of (t_4, t_5) the corresponding 3×3 subsystem has four real solutions, while for the other two pairs there are eight solutions. Figure 6 depicts the eight conformers corresponding to $(t_5, t_4) = (-99, -79)$ and $(t_5, t_4) = (-99, 34)$, in order to illustrate the existing symmetries.

Let us now focus on enalapril. The resulting system is underconstrained, because the equations contain seven unknowns, while there are only five constraints. The system has the structure shown below:

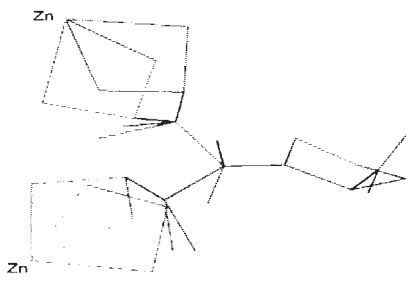

FIGURE 6. Eight conformers satisfying the distance constraints for captopril.

$$p_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = 0$$
$$p_2(x_1, x_2, x_3, x_4, x_5) = 0$$
$$p_3(x_6, x_7) = 0$$
$$p_4(x_1, x_2, x_3, x_4) = 0$$
$$p_5(x_7) = 0.$$

Two equations are still decoupled and consequently can be immediately solved for (x_6, x_7). Obviously, this is a convenient feature characterizing this group of molecules, due to the relative position of the carboxyl and carbonyl groups. Following the procedure described above, for each pair (x_6, x_7) we obtain a 3×5 subsystem:

$$p_1(x_1, x_2, x_3, x_4, x_5) = 0$$
$$p_2(x_1, x_2, x_3, x_4, x_5) = 0$$
$$p_4(x_1, x_2, x_3, x_4) = 0.$$

However, this time we cannot solve the system directly, because it is not square, which is a fundamental requirement for the elimination tools to be applicable. So we apply an exhaustive grid search for two of the variables and solve the resulting square systems. If we choose x_5 as one of the vari-

TABLE V

Number of real solutions for different values of t_2 and t_3.

	−180	−150	−120	−90	−60	−30	0	30	60	90	120	150
−180	0	0	8	16	16	16	16	16	16	8	0	0
−150	4	16	16	24	24	24	24	32	16	16	4	0
−120	12	32	24	16	16	16	24	32	24	12	12	8
−90	20	16	16	8	0	8	24	24	20	12	16	8
−60	24	8	8	0	0	16	16	24	20	12	12	20
−30	16	16	0	0	8	16	24	20	12	12	12	20
0	24	8	0	0	16	16	24	8	12	12	20	24
30	24	8	0	16	16	24	12	12	12	12	24	32
60	20	16	16	24	24	16	12	0	12	12	32	32
90	24	24	24	24	16	8	0	0	8	16	16	32
120	16	16	16	16	8	0	0	0	0	8	16	16
150	8	8	8	0	16	16	8	8	8	8	8	8

ables to be sampled, the resulting systems will contain three complete triquadratic equations. The theory guarantees, that for such a system, an optimal resultant matrix of size 24×24 can be constructed. However, if we do not choose x_5 as a sampling variable, the resulting systems will be like the one discussed in the case of captopril, for which we already have a smaller 16×16 resultant matrix. Therefore, we chose to use x_2 and x_3 as our sampling variables.

We perform the sampling for a grid size of 30 degrees. Table V lists the number of real solutions for different combinations of the sampling angles, the rows and columns corresponding to the angles t_2 and t_3, respectively, measured in degrees. We observe that the numbers are even, so the solutions should again appear in pairs, perhaps symmetric to each other.

7. Conclusions

This study surveys symbolic-numerical methods for various problems in computational chemistry and molecular biology. Several questions, which can be answered off-line or in some comfortable time frame, can best be faced with computational algebra. Moreover, we demonstrated that the veracity of symbolic methods can be combined with the speed of numerical or approximate techniques to yield important algorithms, even for problems requiring real-time solutions.

ACKNOWLEDGMENT

Ioannis Emiris is partially supported by Pythagoras project 70/3/7392 under the EPEAEK program of the Greek Ministry of Educational Affairs and the European Union, and by Kapodistrias project 70/4/6452 of the Research Council of National University of Athens, Greece. Ioannis Emiris also acknowledges insightful discussions with Bruce Donald and the other participants of the 2005 Workshop on Geometry in NMR Protein Structure Determination and NMR Structural Biology, organized by Ileana Streinu at the Bellairs Research Institute of McGill University (Barbados) in January 2005. Dinesh Manocha is supported in part by ARO contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134 and 0118743, ONR contract N00014-01-1-0496, DARPA/RDECOM contract N61339-04-C-0043, and Intel.

References

1. Emiris, I. Z. PhD thesis; Computer Science Division, University of California at Berkeley, 1994.
2. Emiris, I. Z.; Mourrain, B. Algorithmica Spec Issue Algorithms Comput Biol 1999, 25, 372.
3. Fritzilas, E. D. Master's thesis; National University of Athens: Greece, 2005.
4. Manocha, D.; Zhu, Y.; Wright, W. Comput Appl Biol Sci 1995, 11, 71.
5. Bailey-Kellogg, C.; Widge, A.; Kelley, J. J., III; Berardi, M. J.; Bushweller, J. H.; Donald, B. R. J Comp Biol 2000, 7, 537.
6. Clark, D. E.; Jones, G.; Willett, P.; Kenny, P. W.; Glen, R. C. J Chem Inform Comput Sci 1994, 34, 197.

7. Finn, P. W.; Halperin, D.; Kavraki, L. E.; Latombe, J.-C.; Motwani, R.; Shelton, C.; Venkatasubramanian, S. In Applied Computational Geometry; Vol 1148 of LNCS; Springer: Berlin, 1996; p 67.
8. Leach, A. R.; Kuntz, I. D. J Comput Chem 1992, 13, 730.
9. Parsons, D.; Canny, J. In Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology; Palo Alto, CA, 1994; p 322.
10. Havel, T. F. J Symb Comput 1991, 11, 579.
11. Havel, T. F.; Najfeld, I. In Computer Algebra in Science and Engineering; Fleischer, J.; Grabmeier, J.; Hehl, W.; Küchlin, W. Eds.; World Scientific: Singapore, 1995; p 243.
12. Havel, T. F.; Najfeld, I. Technical Report 04-97; Center for Research in Computing Technology; Harvard University: Cambridge, MA, 1997.
13. Gō, N.; Scheraga, H. A. Macromolecules 1970, 3, 178.
14. Gō, N.; Scheraga, H. A. Macromolecules 1973, 6, 273.
15. Manocha, D.; Canny, J. In Proceedings of the IEEE International Conference on Robotics and Automation; Nice, May 1992; p 383.
16. Wedemeyer, W. J.; Scheraga, H. A. J Comput Chem 1999, 20, 819.
17. Cortes, J.; Simeon, T.; Remaud-Simeon, M.; Tran, V. J Comp Chem 2004, 25, 956.
18. Coutsias, E. A.; Seok, C.; Jacobson, M. P.; Dill, K. A. J Comp Chem 2004, 25, 510.
19. Zhang, M.; White, R. A.; Wang, L.; Goldman, R.; Kavraki, L. E.; Hassett, B. Bioinformatics 2005, 21, 624.
20. Zhang, M.; Kavraki, L. E. J Chem Inform Model 2002, 42, 64.
21. Howard, A. E.; Kollman, P. A. J Med Chem 1988, 31, 1669.
22. LaValle, S.; Finn, P.; Kavraki, L.; Latombe, J. J Comput Chem 2000, 21, 731.
23. Craig, J. J. Introduction to Robotics: Mechanics and Control; 2nd Ed.; Addison-Wesley: Reading, MA, 1989.
24. Leach, A. R. Molecular Modelling: Principles and Applications; Prentice-Hall: Englewood Cliffs, NJ, 2001.
25. Lee, H. Y.; Liang, C. G. Mech Mach Theory 1988, 23, 209.
26. Raghavan, M.; Roth, B. In International Symposium on Robotics Research, Tokyo, 1989; p 314.
27. Cox, D.; Little, J.; O'Shea, D. Using Algebraic Geometry; 2nd Ed.; No. 185, Graduate Texts in Mathematics; Springer-Verlag: New York, 2005.
28. Dickenstein, A.; Emiris, I. Z., Eds. Solving Polynomial Equations: Foundations, Algorithms and Applications; Vol 14 of Algorithms and Computation in Mathematics; Springer-Verlag: Berlin, 2005.
29. Emiris, I. Z.; Canny, J. F. J Symbolic Comput 1995, 20, 117.
30. Manocha, D. PhD thesis; Computer Science Division; University of California at Berkeley; 1992.
31. Emiris, I. Z.; Pan, V. Y. J Symb Comput 2002, 33, 393.
32. Mourrain, B.; Pan, V. Y. J Complex 2000, 16, 110.
33. Canny, J. F. In Proceedings of the Symposium on the Theory of Computation; 1988, p 460.
34. Emiris, I. Z. J Complex 1996, 12, 134.
35. Manocha, D. Comput J 1993, 36, 485.
36. Manocha, D. In Proceedings of the ACM International Symposium on Symbolic Algebraic Computation, 1994; p 1.
37. Nielsen, J.; Roth, B. In Computational Kinematics '95; Merlet, J.-P.; Ravani, B., Eds.; Kluwer: The Netherlands, 1995; p 51.
38. Anderson, E.; Bai, Z.; Bischof, C.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; Ostrouchov, S.; Sorensen, D. LAPACK Users' Guide, 2nd Ed.; SIAM: Philadelphia, 1995.
39. Golub, G. H.; Van Loan, C. F. Matrix Computations; 3rd Ed.; Johns Hopkins University Press: Baltimore, MD, 1996.
40. Wilkinson, J. H. The Algebraic Eigenvalue Problem; Oxford University Press: London, 1965.
41. Stewart, G. W. Num Math 1976, 25, 123.
42. Emiris, I. Z. Technical Report 3110; Project SAFIR, INRIA Sophia–Antipolis, France, 1997; also in Proceedings of PoSSo (Polynomial System Solving) Workshop on Software; Paris, 1995; p 35.
43. Dror, O.; Shulman-Peleg, A.; Nussinov, R.; Wolfson, H. J. Curr Med Chem 2004, 11, 71.
44. Natesh, R.; Schwager, S. L. U.; Evans, H. R.; Sturrock, E. D.; Ravi Acharya, K. Biochemistry 2004, 43, 8718.
45. Dammkoehler, R. A.; Karasek, S. F.; Shands, E. F. B.; Marshall, G. R. J Comp Aided Mol Design 1989, 3, 3.
46. http://chembank.broad.harvard.edu.
47. http://www2.ccc.uni-erlangen.de/software/corina.

[a] Canny, J. F. The Complexity of Robot Motion Planning; ACM doctoral dissertation award; MIT Press: Cambridge, MA, 1988.

[b] Canny, J.; Emiris, I. In Proceedings of the International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correction Codes, Puerto Rico, No. 263; in Lecture Notes in Comp. Science; Cohen, G.; Mora, T.; Moreno, O., Eds.; Springer-Verlag: Berlin, 1993; p 89.

[c] Manocha, D.; Canny, J. J Symb Comput 1993, 15, 99.