The Fordham Undergraduate Research Journal

Volume 2 | Issue 1 Article 8

January 2014

Algebraic Detection of Flexibility of Polyhedral Structures with Applications to Robotics and Chemistry

Stephen Fox FCRH '11
Fordham University, furj19@fordham.edu

Robert Lewis
Fordham University, furj19a@fordham.edu

Stephen Fox, FCRH '11

Dr. Robert Lewis, Mathematics

Algebraic Detection of Flexibility of Polyhedral Structures with Applications to Robotics and Chemistry

Introduction

Many problems arise in biochemistry, robotics, and other fields in which flexibility of a polygonal or polyhedral structure plays an important role. In biochemistry, the flexibility and folding of molecules is an important factor in drug design and is a subject of ongoing research (Erickson et al. 2004). In robotics, stable configurations of manipulators (e.g., a mechanical arm grasping) as well as mechanical joints for locomotion (e.g., walking) must be calculated for safe, smooth movement.

The spatial configuration of a molecule or a robot's manipulator can be modeled by a *polyhedral structure*, a three dimensional figure with straight edges, such as a geodesic dome. The faces of a polyhedral structure are polygons, typically triangles. Where these edges join is known as a *vertex*. It is important to distinguish between *generic* and *nongeneric* flexibility. For example, a planar rectangle made of rigid rods but hinged at each vertex is clearly flexible: one can easily change its shape. That is generic flexibility; there are simply not enough constraints to make it rigid. In this paper we are concerned with nongeneric flexibility, which means that a configuration of hinged rods (edges) that is rigid if the lengths of the sides are arbitrarily assigned may become flexible under certain precise conditions on the edge lengths (see Figure 1). Similarly, if the bond lengths of the molecule satisfy these conditions, the polyhedral structure of the molecule becomes flexible as well.

Lewis has developed an algorithm to detect conditions under which a generically rigid polygonal or polyhedral structure becomes flexible (Lewis and Coutsias 2006). He relates the sides and angles of the figure by using basic trigonometry and the distance formula. This yields a system of multivariate polynomial equations, a classically difficult problem to solve. To solve the system efficiently, he uses the Dixon-EDF method to compute a "resultant," a single equation that encapsulates many of the important properties of the original system (Lewis 2010, 1996). The last part of the algorithm, called *Solve*, searches to find the ratios of side lengths necessary for the structure to become flexible by finding when the resultant vanishes identically.

The contribution of this paper is to report on a significant improvement to the *Solve* algorithm. The algorithm, which searches for appropriate substitutions for flexibility, battles the combinatorial explosion inherent in many tree search algorithms. Initially, on a real example coming from the cyclohexane molecule, *Solve* ran for approximately seventy hours before producing a set of 3 139 solution tables that describe the geometry of the molecule when it is flexible. We have refined the algorithm to prune the search tree of possible substitutions, reducing the total run-time of the algorithm, and eliminating subtly disguised duplicates. First, by establishing a canonical form for the solution tables, a test for equivalence can be used to identify and eliminate duplicate solutions. Furthermore, we found ways to eliminate duplicates as they arise by following a similar procedure on the fly (that is, as the algorithm runs).

The remainder of this paper is structured as follows. In the second section, we walk step-by-step through the algorithm for determining molecule flexibility using a simple "toy" quadrilateral example. The third section describes the improvement for comparing different algebraic descriptions of the same geometric figure. Finally, the fourth section summarizes the results of our improvement and its applicability to new problems.

Detecting Flexibility

Consider the quadrilateral with a bar across it in Figure 1. It is attached to the x-axis at the origin (0, 0) and (s_3, 0). The reader can imagine that each of the six connection joints is a hinge allowing the sides s_1, s_5, s_2 and the rod s_4 to pivot within the 2D plane of the page. The points *A*, *B*, *C*, and *D* can move anywhere in the plane as long as the distances between them remain constant. Note that *A* and *D* are not vertices; they are attachment points of the segment *AD*. The structure as pictured is rigid because the rod across the middle appears to brace it up.

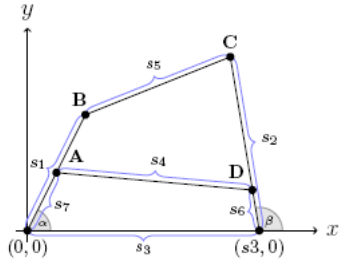

Figure 1
A simple quadrilateral with a bar across it.

On the other hand, if this quadrilateral is arranged as a parallelogram with the bar across the middle parallel to the bases as in Figure 2, the figure becomes flexible. This means that if the plane were vertical, under the force of gravity, the figure would "fall" to the *x*-axis, flexing at all four of its corners while the segment *AD* moves along smoothly.

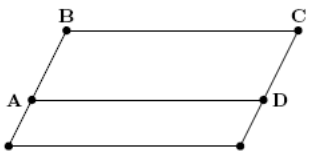

Figure 2
A flexible configuration of Figure 1.

The variables, which determine the shape and configuration of the quadrilateral are the locations of points *A*, *B*, *C*, and *D*. By plac-

ing this figure in the Cartesian plane, with the bottom-left vertex coincident with the origin and the base coincident with the x-axis, the coordinates of the unknown points can be specified using basic trigonometry:

$$A = (s_7 \cos\alpha, s_7 \sin\alpha)$$
$$B = (s_1 \cos\alpha, s_1 \sin\alpha)$$
$$C = (s_3 + s_2 \cos\beta, s_2 \sin\beta)$$
$$D = (s_3 + s_6 \cos\beta, s_6 \sin\beta)$$

Since there are four unknowns, four equations are necessary to completely describe the system. The first two equations below are expressions for the lengths of s_5 and s_4 using the distance formula, i.e., $s_5 = dist(B, C)$ and $s_4 = dist(A, D)$. The final two equations are elementary trigonometric identities. In this system of equations (set each to 0), we write $\cos\alpha$ as ca and $\sin\alpha$ as sa for simplicity of notation and to emphasize that these equations are polynomials:

$$(s_1 \cdot ca - s_3 - s_2 \cdot cb)^2 + (s_1 \cdot sa - s_2 \cdot sb)^2 - s_5^2$$
$$(s_7 \cdot ca - s_3 - s_6 \cdot cb)^2 + (s_7 \cdot ca - s_6 \cdot sb)^2 - s_4^2$$
$$ca^2 + sa^2 - 1$$
$$cb^2 + sb^2 - 1$$

The same method of writing a system of multivariate polynomial equations is used for more complex geometric figures, such as molecules (Lewis and Coutsias 2006).

The next step in the analysis, the Dixon-EDF method, transforms this system of equations into a single equation (the resultant) in one variable with seven parameters that encapsulates the most important information about the system (Lewis 2010, 1996). The process is analogous to the determinant of a matrix. When the determinant of a homogeneous linear system vanishes, the system of equations is said to be *singular*; this means there are an infinite number of solutions to the system. If the resultant of a nonlinear system describing the geometry of a polygon vanishes identically, that system, too, has an infinite number of solutions (Coutsias et al. 2005). Therefore, the polygon is flexible.

The resultant, which is the determinant of the Dixon matrix, is often difficult to compute, and may not even be defined (Dixon 1909). However, Lewis uses the Dixon-KSY idea first proposed by Kapur to overcome some of these problems (Lewis 2008; Kapur et al. 1994). Lewis's implementation in his computer algebra system, *Ferman*, uses his Early Detection of Factors (EDF) algorithm to accelerate the calculation of the resultant (Lewis [date unknown]). In the second phase of the process, his algorithm, *Solve*, determines the conditions under which the resultant is identically zero; i.e., all coefficients (relative to the one remaining variable) of the resultant polynomial must be zero. Even simple figures can give rise to very complicated resultants.

We return to the example pictured in Figure 1. The resultant that arises is a degree three polynomial in the variable ca that contains 162 terms in seven parameter variables (s_1 to s_7) and the reference variable, ca. The first few terms are:

$$8 \cdot s_1^2 \cdot s_4^3 \cdot s_6 \cdot s_7^2 \cdot ca^3 - 8 \cdot s_1 \cdot s_2 \cdot s_4^3 \cdot s_6^2 \cdot s_7 \cdot ca^3 -$$
$$8 \cdot s_1^2 \cdot s_2 \cdot s_4^3 \cdot s_6 \cdot s_7 \cdot ca^3 + 8 \cdot s_1 \cdot s_2^2 \cdot s_4^3 \cdot s_6^2 \cdot ca^3 +$$
$$4 \cdot s_1 \cdot s_2 \cdot s_4^2 \cdot s_6 \cdot s_7^3 \cdot ca^2 + \cdots$$

Figure 1 becomes flexible when the bar is parallel to the bases, and the outer quadrilateral is a parallelogram. *Solve* also discovers a *degenerate* case in which the bar coincides with one of the bases (this table is not shown). (Degenerate means that some of the vertices coincide; i.e., they lie on top of each other.) The former can be represented algebraically with the following system of substitutions that causes the 162 term resultant to be equal to zero:

$$s_6 = s_7$$
$$s_1 = s_2$$
$$s_3 = s_5$$
$$s_4 = s_5$$

Figure 3
Solve finds this *table* of substitutions which algebraically describes Figure 2.

When all four of these substitutions are plugged into the resultant, the resultant is equal to zero, and the condition of flexibility is satisfied.

Solve is a recursive algorithm that searches for substitutions in the variables corresponding to geometric ratios of sidelengths that cause the resultant to vanish identically. The algorithm generates a set of tables of substitutions for variables that correspond to sides of the structure. These substitutions can be quite complicated, or very simple. Each solution table describes a geometric configuration of the structure. For example, $s_1 = s_2$ means the segment labeled s_1 (in Figure 1), and the segment labeled s_2 must be of the same length. The *Solve* algorithm takes as input a multivariate polynomial f in a primary variable x with N parameters s_i. The output is a list of solution tables, as defined above. The steps of *Solve* are outlined below.

1) Factor the leading coefficient in $f(x)$.

2) Use the factors to produce a list of parameters s_j.

3) Within each factor, find all linear parameters in the list of s_j.

4) For all elements in the list of linear parameters:

5) Solve for each linear parameter as a function of the remaining parameters; i.e., solve for $s_j = g(s_{i1}, s_{i2}, \ldots)$.

6) Use the relation g to replace s_j in f.

7) This yields $f_j(x)$, of lower degree.

8) *Recursively* call *Solve* on new f_j with original x as primary variable and append valid $s_j = g$ to the solution tables.

9) For each parameter that was not detected as a linear factor, recursively call *Solve* on the leading coefficient with that parameter as the primary variable.

a) Substitute valid solutions of *Solve* into the coefficient.

b) *Recursively* call *Solve* on the reduced original polynomial with the original primary variable and append all valid solutions to the solution tables.

10) Look for duplicates in the solution tables.

Canonical Form for Solution Tables

Solve is a recursive algorithm which calls itself from the body of its own code. If the first part of the algorithm fails on the input polynomial, it calls *Solve* on the multivariate coefficient, which is also a polynomial. As a consequence of the recursive search tree in *Solve*, the algorithm finds a very large set of solution tables, many of them redundant. As partial substitution tables are discovered, the recursive anture of the algorithm causes even more potentially

redundant tables to be found, undetected until the end of the algorithm or not at all. Removing these redundancies is therefore likely to significantly accelerate the algorithm.

The key example in this research is a configuration of three quadrilaterals that was first discussed by the French mathematician R. Bricard in 1987. It is mathematically, though not superficially, equivalent to the cyclohexane molecule (Coutsias et al. 2005).

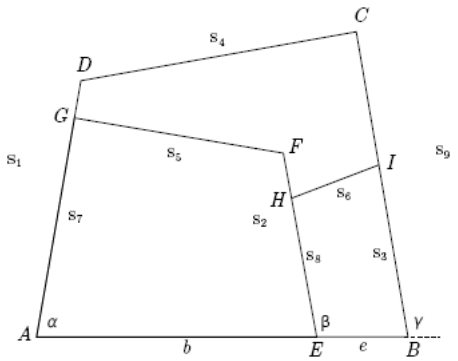

Figure 4
This configuration of three quadrilaterals is mathematically equivalent to the model of the cyclohexane molecule.

Without the rod *HI* to brace it, it is clearly flexible. With *HI*, it is generically rigid. Bricard showed that there are three nondegenerate ways this becomes flexible: all three quadrilaterals are parallelograms; two are similar and the third is a parallelogram; and an arrangement involving a certain series of ratios that is too technical to include here.

We study the redundancies that arise in this example. Lewis's first successful run of *Solve* on the resultant arising from the Bricard quadrilaterals, a polynomial in 5 685 terms with 15 parameters, ran for approximately 70 h, ultimately returning 3 139 solution tables. The 15 parameters, labeled a_1, b_1, c_1, ... , d_3, e_3, are certain combinations of the sides in Figure 4. A close examination revealed that many of these solution sets are slightly different algebraically, but describe the same geometric configuration.

A simple case of redundancy can be fabricated by permuting rows of the table (e.g., Table 1 to Table 2 in Figure 5). Table 3 in Figure 5 reveals a more subtle equivalent variation.

1	2	3
$a_1 = a_2$	$a_3 = a_2$	$a_3 = a_1$
$a_3 = a_2$	$a_1 = a_2$	$a_2 = a_1$

Figure 5
Equivalent tables.

The following is a more sophisticated example of equivalent tables discovered by *Solve*. The first three rows of the table are the same. However, the fourth row is different. Geometrically, the tables describe the same configuration of the structure, yet by inspection they appear algebraically different.

Table 1

$$e_3 = 0$$
$$d_3 = \frac{(-c_1^2 \cdot b_3 \cdot a_2^2 - 2 \cdot c_1 \cdot c_2 \cdot c_3 \cdot a_1 \cdot a_2)}{(c_2^2 \cdot a_1^2)}$$
$$a_3 = 0$$
$$e_2 = \frac{(b_1 \cdot b_2)}{a_1}$$
$$d_2 = \frac{(b_1 \cdot a_2)}{a_1}$$
$$e_1 = \frac{(b_1 \cdot c_1^2 \cdot b_2 \cdot a_2)}{(c_2^2 \cdot a_1^2)}$$
$$d_1 = \frac{(c_1^2 \cdot b_2 \cdot a_2)}{(c_2^2 \cdot a_1)}$$

Table 2

$$e_3 = 0$$
$$d_3 = \frac{(-c_1^2 \cdot b_3 \cdot a_2^2 - 2 \cdot c_1 \cdot c_2 \cdot c_3 \cdot a_1 \cdot a_2)}{(c_2^2 \cdot a_1^2)}$$
$$a_3 = 0$$
$$e_2 = \frac{(b_1 \cdot c_2^2 \cdot d_1)}{(c_1^2 \cdot a_2)}$$
$$d_2 = \frac{(b_1 \cdot a_2)}{a_1}$$
$$e_1 = \frac{(b_1 \cdot d_1)}{a_1}$$
$$b_2 = \frac{(c_2^2 \cdot a_1 \cdot d_1)}{(c_1^2 \cdot a_2)}$$

Table 3

$$e_3 = 0$$
$$d_3 = \frac{(-c_1^2 \cdot b_3 \cdot a_2^2 - 2 \cdot c_1 \cdot c_2 \cdot c_3 \cdot a_1 \cdot a_2)}{(c_2^2 \cdot a_1^2)}$$
$$a_3 = 0$$
$$e_2 = \frac{(b_1 \cdot b_2)}{a_1}$$
$$d_2 = \frac{(b_1 \cdot a_2)}{a_1}$$
$$e_1 = \frac{(b_1 \cdot c_1^2 \cdot b_2 \cdot a_2)}{c_2^2 \cdot a_1^2}$$
$$d_1 = \frac{(c_1^2 \cdot b_2 \cdot a_2)}{(c_2^2 \cdot a_1)}$$

Variables in the solution tables are partitioned by the equal sign: no variables that appear on the left hand side (LHS) of the equalities appear on the right hand side (RHS) and vice versa. However, b_2 is on the RHS in Table 1, but on the LHS in Table 2. The rational functions can be resolved in terms of a desired variable to move that variable from the RHS to the LHS. This is a more difficult type of redundancy to detect visually or algorithmically.

Swapping rows (i.e., rearranging the order of equations) produces equivalent geometric descriptions in different algebraic ways. Furthermore, swapping variables from the LHS to RHS gives rise to more redundant representations. As the algorithm proceeds, duplicates that arise early on in the algorithm lead to exponentially more duplicates later. For this reason, we have developed an algorithm to detect and eliminate duplicates both on the fly (i.e., running as a step in *Solve*) and in post-processing as a separate algorithm that runs once *Solve* has finished.

Lewis's original implementation of *Solve* sorts tables by the variable that appears in the LHS of each row to identify duplicate tables from row permutations. The sorting process is more subtle than simply permuting rows: when two rows are swapped, the right hand side (RHS) of the variable moved up in the table must be substituted down the right hand side of the lower rows. Using this process of sorting tables and doing a simple difference of the respective left and right sides of the tables for comparison (i.e., if the difference is 0, they are equivalent tables) still leaves many redundant solutions undetected. Table 1 and Table 2 are examples of undetected duplicate tables from the original implementation.

In order to identify and detect these lingering redundant solutions, we establish a *canonical form* for the tables. A canonical form is a standard expression for the arrangement of the solution sets. Our process of standardizing the form of the table follows these steps:

1) Establish an order of variables from highest to lowest, a step already required by the *Solve* algorithm.

2) Search each table for the row with the lowest linear variable (i.e., linear in the sense that it is raised only to the power of one). This variable will be either on the LHS of the table or the RHS, but never both.

3) Re-solve that row in terms of the lowest variable.

4) Before committing to the rearrangement, check that this new expression does not cause any denominator to become zero.

5) Permute the row to the top of the table, arranging the LHS from lowest variable to highest variable (top to bottom), substituting appropriate variables where necessary.

6) Sort the table and repeat until the table does not change.

Once the tables have been arranged into canonical form, they can be compared using a simple and efficient algorithm for comparison:

1) If the tables do not have the same number of entries, they are not equal.

2) Otherwise, subtract the RHS of each row from the LHS within the same table.

3) Compare corresponding rows from the two tables up to sign.

4) If all rows return positive correspondence, the tables are equal and one can be discarded.

The resultant arising from the cyclohexane molecule has 5 685 terms with 1 variable and 16 parameter variable of the coefficients. If the order of the parameter variables is assigned to be: t_1, e_3, d_3, a_3, e_2, d_2, a_2, b_1, e_1, d_1, a_1, c_3, b_3, c_2, b_2, c_1, then both tables have the same canonical form given above.

Conclusion

Although these algorithms were developed for post-processing, we are also able to compare tables on the fly to eliminate duplicates as they arise. After each level of recursion, all partial tables of substitutions that have been discovered are sorted and compared using these algorithms. By discarding the duplicates, we have greatly decreased the overall run time of the *Solve* algorithm. For the cyclohexane molecule, the run time for *Solve* was reduced from approximately 70 h, to just 3 min and 14 s. The set of 3 139 solution tables was condensed to 62 tables. All three of the nondegenerate solutions of the cyclohexane configuration appear in the list of 62. The results of this work could lead to the ability to analyze more complex and larger molecules and geometric structures and have been used by Lewis to further explore the solution space of flexible polyhedra.

References

Bricard R. 1897. Memoire sur la theorie de l'octaedre articule. *Journal de Mathématiques pures et appliquées, 5*(3):113–148.

Coutsias EA, Seok C, Wester MJ, Dill KA. Resultants and loop closure. *International Journal of Quantum Chemistry. 106*(1):176–189.

Dixon AL. 1909. The eliminant of three quantics in two independent variables. *Proceedings of The London Mathematical Society. 2–7*(1): 49–96.

Erickson JA, Jalaie M, Robertson DH, Lewis RA, Vieth M. 2004. Lessons in molecular recognition: the effects of ligand and protein flexibility on molecular docking accuracy. *Journal of Medical Chemistry. 47*(1):45–55.

Kapur D, Saxena T, Yang L. 1994. Algebraic and geometric reasoning using Dixon resultants. *Proceedings from The International Symposium on Symbolic and Algebraic Computation*; 1994 July 20–22 Oxford (UK).

Lewis RH. 2010. Comparing acceleration techniques for the Dixon and Macaulay resultants. *Mathematics and Computers in Simulation. 80*(6):1146–1152.

———. 2008. Heuristics to accelerate the Dixon resultant. *Mathematics and Computers in Simulation. 77*(4):400–407.

———. 1996. The Dixon resultant following Kapur-Saxena-Yang. Retrieved from http://fordham.academia.edu/RobertLewis/Papers.

———. Fermat computer algebra system. Retrieved from http://home.bway.net/lewis.

Lewis RH, Coutsias EA. 2006. Algorithmic search for flexibility using resultants of polynomial systems. In *Automated deduction in geometry, Lecture Notes in Computer Science. 4869*: 68–79.