Symmetrizer: Algorithmic determination of point groups in nearly symmetric molecules

R Jeffrey Largent
William F. Polik
Hope College, polik@hope.edu
J R. Schmidt

Follow this and additional works at: http://digitalcommons.hope.edu/faculty_publications
Part of the Chemistry Commons

Recommended Citation

This Article is brought to you for free and open access by Digital Commons @ Hope College. It has been accepted for inclusion in Faculty Publications by an authorized administrator of Digital Commons @ Hope College. For more information, please contact digitalcommons@hope.edu.
SYMMETrizer: Algorithmic Determination of Point Groups in Nearly Symmetric Molecules

R. Jeffrey Largent, William F. Polik
Department of Chemistry
Hope College, 35 East 12th Street, Holland, MI 49422-9000

J. R. Schmidt
Theoretical Chemistry Institute and Department of Chemistry
University of Wisconsin, Madison, WI 53706
Abstract: Symmetry is an extremely useful and powerful tool in computational chemistry, both for predicting the properties of molecules and for simplifying calculations. Although methods for determining the point groups of perfectly-symmetric molecules are well known, finding the closest point group for a nearly-symmetric molecule is far less studied, though it presents many useful applications. For this reason, we introduce Symmetrizer, an algorithm designed to determine a molecule’s symmetry elements and closest-matching point groups based on a user-adjustable tolerance, and then to symmetrize that molecule to a given point group geometry. In contrast to conventional methods, Symmetrizer takes a bottom-up approach to symmetry detection by locating all possible symmetry elements and utilizing this set to deduce the most probable point groups. We explain this approach in detail, and assess the flexibility, robustness, and efficiency of the algorithm with respect to various input parameters on several test molecules. We also demonstrate an application of Symmetrizer by interfacing it with the WebMO web-based interface to computational chemistry packages as a showcase of its ease of integration.

Keywords: symmetry; symmetry element; point group; group theory; algorithm
1. Introduction

Consideration of symmetry is extremely useful in computational chemistry, both for predicting molecular properties and for greatly simplifying calculations. Thus it is highly advantageous to be able to automate identification of the symmetry elements and the point groups of molecules. However, finding the closest point group of nearly symmetric molecules introduces additional complexity. This is in fact a far more important and interesting problem, as molecules often do not have perfectly symmetric conformations, whether due to thermal fluctuation, Jahn-Teller distortion, external perturbation, or merely numerical imprecision. It is therefore useful for an algorithm to find the point group that a molecule most closely matches, within a user-specified tolerance.

Methods for determining the point group of a perfectly symmetric molecule are well-known. The classic approach proposed by F. A. Cotton in 1963 is a straightforward heuristic for determining the point group of a molecule based on the presence or absence of key characteristics, but it is applicable only to ideal, perfectly-symmetric structures.\textsuperscript{1} Another approach treats symmetry as a continuous property rather than a discrete one dealing solely with ideal geometries.\textsuperscript{2} Still other algorithms have been devised for use in software applications that incorporate the idea of a tolerance for error in symmetry,\textsuperscript{3,4} but these only provide the user with the highest-order suggested point group. This is undesirable in situations is which lower-order symmetries are preferred, such as accounting for the Jahn-Teller effect.

For these reasons, we adopt a bottom-up approach to symmetry detection. Our approach begins by locating all feasible symmetry elements present in a molecule, from which one can deduce plausible point groups. This approach makes it trivial to quantify the deviation
from a particular point group, and it also provides information about how well each individual symmetry element “fits” a molecule. Additionally, having located all symmetry elements of a point group using our method, identification of the symmetry-unique atoms and molecular symmetrization becomes quite straightforward.

2. Overview of Approach

We begin with a brief overview of our symmetry algorithm. The algorithm starts by searching for each type of symmetry element using an exhaustive search, utilizing rigorous theorems to narrow the search process. A detailed description of the search process is given in a subsequent section. Potential symmetry elements are tested and evaluated as follows: The $i$th symmetry operation associated with the element is applied to every atom $A_j$ in the molecule to generate atom $A_j^i$. For each $A_j^i$, the closest atom of the same type, $B_j^i$, is found, and then the error associated with that symmetry element is calculated as

$$error = \frac{1}{n} \sum_{i=1}^{n} \max_j \left| \frac{A_j^i - B_j^i}{r_j} \right|$$

where $r_j$ is the perpendicular distance between atom $A_j$ and the symmetry element and $n$ is the order of the element. The numerator is the distance that any generated atom lies from one of its symmetry equivalent counterparts, and the denominator serves to keep the error independent of a molecule’s size. If the distance between $A_j$ and the symmetry element is small (within one Å), then the denominator is set to unity to prevent the assignment of an erroneously high error to the symmetry element. This error is then compared to a user-defined threshold, and a symmetry element is only accepted if it lies within the threshold.
The next step is to determine all viable point groups given the set of accepted symmetry elements. This set is compared to the set of elements that compose each known point group, and a point group is accepted if its elements are a subset of the accepted symmetry elements. The list of accepted point groups is then ranked based on two criteria: The set difference between the two sets and the mean error of each point group’s constituent symmetry elements. Point groups are ranked higher the smaller the difference between the set of accepted symmetry elements and the set of expected symmetry elements, with tied point groups ranked higher the lower their mean element error. The end result is a list of possible point groups ordered from most-specific to least-specific. The user can choose among various plausible point groups if the symmetry of the molecule is unclear, or even to symmetrize the molecule to a less-specific point group.

Having identified a target point group, the molecule can be symmetrized. Before symmetrizing, however, the identified symmetry elements must be idealized to remove the error resulting from the imperfect geometry of the original molecule. The set of atoms are reduced to a small subset of symmetry-unique atoms, and these atoms are aligned with respect to the symmetry elements. The operations of the idealized symmetry elements are applied to these atoms to generate the symmetrized molecule.

A graphical overview of the symmetrization procedure can be seen below in Figure 1.

3. Description of Algorithms

3.A. Symmetry Element Detection Algorithm

We will briefly outline the details of the symmetry element detection algorithm. We use the phrase “testing” or “checking” symmetry elements to refer to calculating the error
associated with an element and comparing it to the user threshold as described in section 2. Here we utilize a number of theorems to simplify the search for symmetry elements, italicized in the sections below.

3.A.1. Identification of Principal Axes

First, the center of mass is calculated and all symmetry elements locations are defined relative to it. Next, the principal moments and principal axes of inertia are calculated by constructing and diagonalizing the inertial tensor. Finally, the principal moments of inertia are compared to determine their degeneracy, specifically to determine if the molecule is a spherical, symmetric, or asymmetric top.

The principal axes can be employed to dramatically speed up the subsequent search for the molecular symmetry elements, utilizing the fact that a symmetry axis or reflection plane normal must be an eigenvector of the inertial tensor. This is evident because a symmetry operation must leave a molecule (and therefore the moment of inertia tensor) unchanged. Thus the tensor must be invariant with respect to all symmetry operations. This theorem implies that a symmetry axis or plane must contain at least one of the principal axes if the moments of inertia are non-degenerate. If the moments of inertia are doubly-degenerate, then any axis or plane must either contain the non-degenerate axis or be orthogonal to it. If an element does not obey these criteria within a small tolerance, it is immediately rejected without further testing. These ideas are used to optimize the subsequent symmetry element searches.

3.A.2. Test for Inversion Center

As inversion centers are points and every symmetry element must contain the center of mass, an inversion center must be at the center of mass, if it exists.
3.A.3. Test for Proper Rotational Axes

Proper rotation axes are initially checked along the principal axes. The algorithm then searches for axes containing both the center of mass and an atom, followed by axes containing the center of mass and midpoints between like-atoms. If the point group of the molecule is cubic, determined by its moments of inertia being triply degenerate, then an additional step must be employed to look for proper rotational axes passing through the centers of polygonal faces. This is accomplished by searching along found $C_2$ axes for $O$ and $I$ groups, and along all combinations of cross-products of $C_2$ axes for $T$ groups. Finally, if the molecule is not cubic, then the highest-order proper rotational axis is selected and denoted as the primary axis.

3.A.4. Test for Improper Rotational Axes

All improper rotational axes must be coincident with existing proper rotational axes and have a degree either equal to or twice the degree of the coincident proper rotational axis.

3.A.5. Test for Reflection Planes

The first step for locating reflection planes is to search for those that are orthogonal to previously located proper rotational axes. For $O$ and $I$ point groups, this search is limited to $C_2$ axes, as all reflections are orthogonal to $C_2$ axes in these groups. For all other groups, additional checks are required: Planes with normals containing the center of mass and an atom are checked, followed by planes with normals containing the center of mass and a midpoint between like-atoms.

See Figure 2 for a flowchart overview of the symmetry element detection algorithm.

3.B. Point Group Assignment Algorithm
The point group assignment algorithm matches the accepted symmetry elements to the elements contained in each known point group. However, since we want users to be able to symmetrize molecules to lower-order point groups in addition to the highest-order point group found, this process must be done in a specific fashion. This involves ensuring that rotations around the primary axis in axial groups are the same angle from each other, and that reflections in the plane of the molecule are prioritized over dihedral reflections. The accepted symmetry elements are compared to those of plausible point groups, and any point group that did not have all of its symmetry elements matched is immediately discarded. Remaining point groups are ranked based on how many accepted elements could not be matched to them, and secondarily on the mean error of the elements that were matched to them.

3.C Symmetrization Algorithm

Once a point group has been designated for a molecule, symmetrization to that point group can occur. The symmetry elements found to match that point group must be first be idealized with respect to each other. This entails, for example, rotating the existing elements to ensure that all peripheral axes are orthogonal to the primary axis and that the reflection planes are appropriately oriented with respect to each other. The symmetrization routine also determines the least set of atoms that generates the original atoms in the molecule when all symmetry operations are applied to them, which are referred to as the symmetry-unique atoms. This set of atoms must be oriented so that they lie exactly on the idealized symmetry elements. As such, atoms within 0.1 Å are moved onto them, while those further away are assumed to not lie on symmetry elements and are left unchanged since the positions of
symmetry-unique atoms that do not coincide with symmetry elements are arbitrary with respect to the point group. Finally, the full set of symmetry operations is applied to every symmetry-unique atom, generating the full set of atoms in their symmetric configuration.

4. Structure of Symmetrizer

Our implementation, Symmetrizer, was written in Java and makes use of the object-oriented programming paradigm to partition the code into various classes. These classes can be easily integrated into Java applets or applications as a simple library for performing symmetry-related tasks.

The library is centrally oriented around the Symmetry class, which does the bulk of the work of symmetry detection and point group matching. The PointGroup class contains the symmetry elements, symmetry-unique atoms, and point group name based on a potential point group for a molecule, and takes care of symmetrizing the molecule’s atoms to its point group symmetry. The Element superclass defines an interface for the Inversion, Reflection, ProperRotation, and ImproperRotation classes, which contain the position and direction information of each type of element, as well as its associated symmetry operation. The Atom, Molecule, Point3D, and Matrix4D classes serve as data structures and do mathematical processing, and the Profiler class provides some very basic runtime-profiling functionality.

Symmetrizer also includes a Main class that allows it to run as a stand-alone command-line application. For example,

```
Java -jar symmetrizer.jar H2O.xyz
```
produces output shown in Listing 1. Symmetrizer contains basic functionality for reading and
writing XYZ files, as well as a comprehensive test suite of example molecules and their intended
symmetries at various tolerances. Symmetrizer source code and binary are being distributed as
tools for both developers and researchers under the GNU Lesser license, which allows users to
modify the code for their own personally-tailored requirements if they so desire.  

5. Integration with Other Programs

While Symmetrizer can be run from the command line, it is also exposes a
programmatic interface for easy integration with other Java-based programs.
The core of the Symmetrizer API consists of the guessPointGroup and findAllPointGroups
methods in Symmetry and the symmetrizeMolecule and getSymmetryElements methods in
PointGroup. The findAllPointGroups method initiates the symmetry and point group-detection
method described earlier in the paper to build up a list of all potential point groups and their
symmetry elements, while the guessPointGroup method implements a fast, decision-tree point
group determination algorithm for basic on-the-fly point group information. All of the desired
functionality for symmetry-related tasks can be performed using these four methods. However,
other methods exist for obtaining additional detailed symmetry data if needed.

As a case study, this library was integrated with a web-based graphical interface for
computational chemistry programs called WebMO.  
WebMO uses Symmetrizer to provide real-
time symmetry information about a molecule being modeled, as well as the option to
symmetrize a molecule prior to submission to a computational engine such as Gaussian,
Gamess, or MOPAC. In addition, WebMO uses the Element data structures from
Symmetrizer to visualize symmetry elements on any modeled molecule. This feature provides a very intuitive picture of molecular symmetry and is useful for educational purposes. Screenshots of WebMO’s symmetry capabilities can be viewed in Figures 3 and 4.

6. Results and Discussion

This symmetry detection algorithm has proved to be a flexible, robust, and relatively fast method of determining molecular symmetry properties. Some sample statistics of the algorithm for several large molecules with low, intermediate, and high symmetry as given in Table 1.

As the data indicates, the performance of the algorithm is a complex factor of several parameters, including number of atoms, “uniqueness” of atoms, type of point group, and the various tolerance thresholds employed through the program. Obviously, the greater the number of atoms in a molecule, the more positions need to be checked for possible symmetry elements, so computational time increases with the size of the input molecule. Additionally, the more of one type of element there is within the molecule, the harder it is for the algorithm to rule out possible symmetry elements on the basis of atomic properties. So a very homogeneous molecule like Buckminsterfullerene takes considerably more time to analyze than a more heterogeneous molecule like hexapyridine iron. Additionally, while some symmetry elements of axial groups can be ruled out based on inertial constraints, cubic groups have triply degenerate moments of inertia and so all of their symmetry elements are inertially valid. Similarly, while Taxol has almost twice as many atoms as C\textsubscript{60}, it has no symmetry, and so only elements that coincide with the principal axes of inertia must be searched. Finally, the error tolerances for
various calculations also affect the speed of the algorithm profoundly. For example, lowering the tolerance used for angle comparisons when checking if a symmetry element is coincident with the principal axes of inertia can dramatically speedup the calculation, but at the expense of possibly missing a symmetry element if the molecule is severely distorted.

7. Conclusions

Symmetrizer provides a straightforward, flexible, and robust algorithm for determining a molecule’s point group and symmetrizing its geometry. While similar in aim to past approaches, this approach is unique in that it utilizes nothing but spatial geometry and incorporates a high degree of control over the detection and symmetrization process. Symmetrizer is able to handle nearly-symmetric molecules as well as ideal ones, can suggest all plausibly matching point groups within a user-set tolerance, and both locates actual symmetry elements on a molecule and outputs information about them in an organized format. Additionally, it is implemented in Java and so can be easily incorporated into web applets as well as standalone software packages.

8. Acknowledgments

Computing resources used in this research was supported by National Science Foundation awards CHE-0520704 and CHE-1039925. RJL and WFP gratefully acknowledge summer stipends from National Science Foundation award CHE-0624602.
References


6. Symmetrizer is available as supplemental material or at www.webmo.net/download/symmetrizer.zip.


Principal Axes are non-degenerate

+ ELEMENTS FOUND:

C2 with a rating of 0.098488
sigma with a rating of 0.000000
sigma with a rating of 0.098488

+ POINT GROUP FOUND:

{ E, 2sigma, C2 }

***************************
1. C2v:  0.06565896714675919
***************************
2. Cs:  0.0
3. C2:  0.09848845072013875
4. C1:  0.0

Search took 0.018000 seconds

Listing 1: Symmetrizer command line interface output for nearly symmetric water.
<table>
<thead>
<tr>
<th></th>
<th>Taxol</th>
<th>Dimanganese Decacarbonyl</th>
<th>Hexapyridine Iron</th>
<th>Buckyball</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical Formula</td>
<td>$C_{47}H_{51}NO_{14}$</td>
<td>$Mn_2(CO)_{10}$</td>
<td>$Fe(C_3H_5N)_6^{2+}$</td>
<td>$C_{60}$</td>
</tr>
<tr>
<td># of Atoms</td>
<td>113</td>
<td>22</td>
<td>67</td>
<td>60</td>
</tr>
<tr>
<td>Point Group</td>
<td>$C_1$</td>
<td>$D_{5d}$</td>
<td>$T_h$</td>
<td>$I_h$</td>
</tr>
<tr>
<td>Inversion (s)</td>
<td>0.000</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Proper Rotation (s)</td>
<td>0.052</td>
<td>0.045</td>
<td>0.186</td>
<td>0.358</td>
</tr>
<tr>
<td>Improper Rotation (s)</td>
<td>0.000</td>
<td>0.004</td>
<td>0.012</td>
<td>0.040</td>
</tr>
<tr>
<td>Reflection (s)</td>
<td>0.019</td>
<td>0.008</td>
<td>0.002</td>
<td>0.007</td>
</tr>
<tr>
<td>Symmetrization (s)</td>
<td>0.000</td>
<td>0.016</td>
<td>0.090</td>
<td>0.097</td>
</tr>
<tr>
<td>Total Time (s)</td>
<td>0.071</td>
<td>0.075</td>
<td>0.290</td>
<td>0.502</td>
</tr>
</tbody>
</table>

**Table 1**: Comparison of Symmetrizer algorithm runtimes.
Figure Captions

**Figure 1.** Overview of symmetrization process.

**Figure 2.** Flowchart of point symmetry element detection algorithm.

**Figure 3.** Screenshot of ammonia with reflection planes.

**Figure 4.** Screenshot of hexapyridineiron with $C_2$ axes and reflection plane.
Step 1. Identify symmetry elements and suggest point groups

Step 2. Adjust symmetry elements of selected point group

Step 3. Determine symmetry-unique atoms and align with respect to symmetry elements

Step 4. Generate symmetric molecule by applying all symmetry operations to symmetry-unique atoms

Figure 1
Figure 2