HOSTDESIGNER
Version 3.0

USER'S MANUAL

Publication Date: September 18, 2015

Dr. Benjamin P. Hay,[1] Dr. Timothy K. Firman,[2]

Dr. Vyacheslav Bryantsev,[3] and Dr. Billy Wayne McCann.[3]

[1] *Supramolecular Design Institute, 127 Chestnut Hill Rd, Oak Ridge, TN 37830. Code Administrator. Email: hayben@comcast.net*

[2] *Regional Trustee Services, Seattle, WA 98104*

[3] *Chemical Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831*

ACKNOWLEDGEMENTS

Development of HostDesigner, Version 3.0 was funded by the Critical Materials Institute, an Energy Innovation Hub funded by the U.S. Department of Energy (DOE), Office of Energy Efficiency and Renewable Energy, Advanced Manufacturing Office. Prior code development (2000-2006) was funded in part by the Chemical Sciences, Office of Basic Energy Sciences, Office of Science, DOE and in part by the Laboratory Directed Research and Development Program, Fundamental Science Division of the Pacific Northwest National Laboratory (PNNL). The authors express their appreciation to Dr. J. B. Nicholas (Neurion Pharmaceuticals) for his considerable and valuable input in the early design of this software, Prof. J. W. Ponder (Washington University Medical School) for granting permission to distribute portions of the TINKER code used in the development of the OVERLAY algorithm, Drs. T.P. Straatsma and W. deJong (PNNL) for their assistance with writing and debugging the code, and Dr. Kevin E. Gilbert (Serena Software) for the coding of the HDViewer utility program.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Oak Ridge National Laboratory, nor the Supramolecular Design Institute, nor any of their employees, makes any warranty, expressed of implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, product, or process disclosed. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Oak Ridge National Laboratory, or the Supramolecular Design Institute. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

1.0 INTRODUCTION

Because the ability to successfully design effective and selective host molecules would have a significant impact in many scientific fields, there is a large body of research focused on understanding the nature of host-guest interactions. Fundamental studies of the interactions between ions and simple coordinating groups, such as ether oxygens, amine nitrogens, or arenes, have elucidated geometric features that lead to optimal binding. It has been shown that this type of information can be incorporated into force field models to provide a rapid method of screening proposed host structures, in other words, rank-ordering a set of hosts in terms of their binding affinity for a given guest. It also has been demonstrated that more accurate, but more costly, screening can be accomplished with molecular dynamics or electronic structure calculations. However, although understanding the nature of guest-binding site interactions and screening candidate structures are important components of ligand design, we are still missing a vital piece to the puzzle – a way to efficiently generate new host candidates.

Simply put, host design is the process of choosing a set of binding sites and then choosing the connecting geometric structure that ties them together. In many instances, we have sufficient knowledge to identify the number and types of binding sites to complement a given metal, but we need an effective method to identify how to connect these groups together to form an integrated host molecule. At present, we can only generate trial structures by hand with a graphical user interface, an extremely time-consuming process. Often, it is not readily obvious which linkage structures might be best used to connect the binding sites groups.

Drug designers have developed computational approaches to address the inverse of this problem, that is, how to identify a molecular structure (guest) that will bind tightly within the binding site of a protein (host). One approach involves docking fragments into the pocket and then linking them together to yield the potential drug candidate. Software packages that perform these operations require input of the atomic coordinates of a protein binding site and are highly specialized to address protein–organic interactions. Because of this, they are not generally applicable to other types of host-guest interactions that arise in supramolecular chemistry. Our aim was to develop molecular design algorithms to achieve software that is capable of identifying the best host architctures for a specific guest. HostDesigner is the result.

The first version of the HostDesigner software, Version 1.0, was released in April 2002. The features of this software and examples of applications were published in Hay, B. P.; Firman, T. K. *Inorganic Chemistry*, **2002**, *41*, 5502-5512. Since that time, there have been significant modifications both to enhance the performance and to extend the application of this software.

With the assistance of Dr. Kevin Gilbert (Serena Software) we have developed a graphical user interface for HostDesigner named HDViewer. This utility program can be used to create input files from imported coordinates and view the output files created by HostDesigner. Further information on the HDViewer software is available in the HDViewer User's Manual provided in the download package.

The second release of HostDesigner, Version 2.0 (2006), contained improvements in several major areas. While the Version 1.0 was limited to single-atom guests, Version 2.0 treated multi-

atom guests, allowing application to a much broader range of compounds. In Version 1.0, the input structures were defined as rigid geometries, but in Version 2.0 the geometry of the input structures could be varied. The screening options within HostDesigner were improved as well, with estimates for conformational and of entropic contributions to the binding energy of potential hosts. Other improvements include an expanded library of structures to bridge binding groups and an increase in speed. A recent overview of these updated features was published in Hay, B. P.; Jia, C.; Nadas, J. *Computational and Theoretical Chemistry* **2014**, *1028*, 72-80. An archival version of HostDesigner, 2.0 is available at no cost and can be downloaded from the http://sourceforge.net/projects/hostdesigner20/ website.

In the current release, HostDesigner, Version 3.0, modifications have been made to eliminate prior limitations with respect to the types of molecules that were accepted as input. New algorithms have been developed to allow the estimation of bond lengths and rotational potential energy surfaces for any covalent bond that is formed between input and library fragments. Version 3.0 is able to handle any molecular input fragment provided by the user, making possible the general application of *de novo* structure-based design methods to a much wider range of chemistry. Version 2.0 required molecular mechanics atom types to be assigned in the input fragment. This information was previously required by the molecule building algorithms and could be included in the output structures to facilitate post-processing with molecular mechanics models. In Version 3.0, the specification of atom types is now optional, rather than mandatory. Finally, the linking fragment library has been updated to remove any high-energy linkages, in other words, those with relative confomational energies greater than 3.0 kcal/mol. HostDesigner 3.0 is available at no cost and can be downloaded from the http://sourceforge.net/projects/hostdesigner-v3-0/ website. The download package includes this User's Manual, HostDesigner source code, two data files LIBRARY and CONSTANTS, example input files, and HDViewer executables (MacOSX or Windows).

2.0 DESCRIPTION OF THE ALGORITHMS

2.1 The LINKER Algorithm

2.1.1 Overview

The LINKER algorithm is used to connect three different molecular fragments together and make a preliminary judgement on how well the resulting structure will bind a user-defined guest. Two user–defined input fragments are bridged by a linking fragment, a molecular group taken from a LIBRARY file (see Section 2.3). LINKER can generate and evaluate millions of structures per minute on a desktop computer, and writes files containing descriptions and coordinates for the most promising candidates. This process is intended to be a first step in the design of new hosts, to be followed by more detailed analysis of the output structures by other methods.

Host molecules are composed of groups of binding sites. Thus, any multidentate host can be dissected into two or more structural fragments which we can designate as host components. For example, the well known 18-crown-6 macrocycle can be broken down into two triglyme components, three diglyme components, or six dimethylether components. It is possible to define the structure of an input fragment, in other words, a piece of a host-guest complex, by combining a host component with a guest. In constructing the input fragment, the guest can be positioned relative to the host component to define a complementary geometry, that is, a geometry that would give the strongest interaction between the binding sites of the host component and the guest in an actual complex.

The LINKER algorithm is based on the following assumptions: (1) there is an optimal geometry for the interaction between each binding site of the host and the guest and (2) this optimal geometry is largely independent of the other binding sites that may be present in the host. The LINKER algorithm builds new host structures by connecting two input fragments to linking fragments that are taken from a library. Figure 1 shows input fragments for a number of common donor groups with a variety of host-guest orientations. Although it is true that the host-guest distance will increase with increasing coordination number, the preferred orientation of the host component with respect to the guest remains largely invariant. Optimal angles and dihedral angles can be deduced from examination of experimental geometries of host-guest complexes or through the careful application of electronic structure calculations. Given a targeted coordination number, the optimal host-guest distances can be deduced in the same manner. Thus, it is possible to define an optimal geometry for the interaction between a guest with any host component.

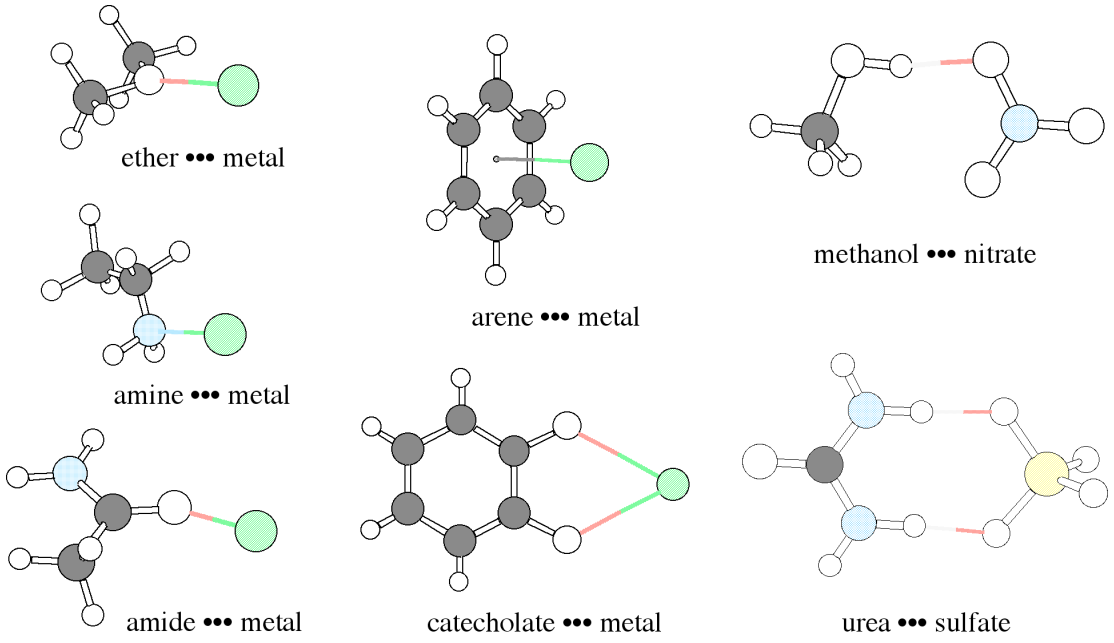


Figure 1. Examples of simple input fragments showing optimal placement of cation and anion guests with respect to different host components.

2.1.2 The Building Algorithm

LINKER builds new host structures by forming bonds between two input fragments and linking fragments. The user must provide an input fragment files that specify the coordinates for all the atoms, atom connectivity, and attachment points (see Section 3.3). Attachment points are indicated by listing hydrogen atoms that will be lost from the input fragment and replaced with carbon atoms from the linking fragment. The building process is illustrated in Figure 2. In this example, two identical lithium-dimethylether input fragments are attached to a methylene link.

Given the possibility of multiple attachment points per component structure and multiple rotational minima about each bond formed, it is possible to generate a number of structures with one link. With the example shown in Figure 2, there are three attachments to each host and there are three possible dihedral angles for each bond giving rise to a total of 81 potential structures. However, when this example is run through the code, only the 9 structures shown in Figure 3 are retained. There are two reasons for this. First, subsequent builds may lead to the identical structure or the mirror image of the structure. These degenerate results are rejected. Second, in a number of cases, rotation about the bonds leads to physically unreasonable collision or superposition of generated host atoms (see Figure 4). When this occurs, such structures are rejected.

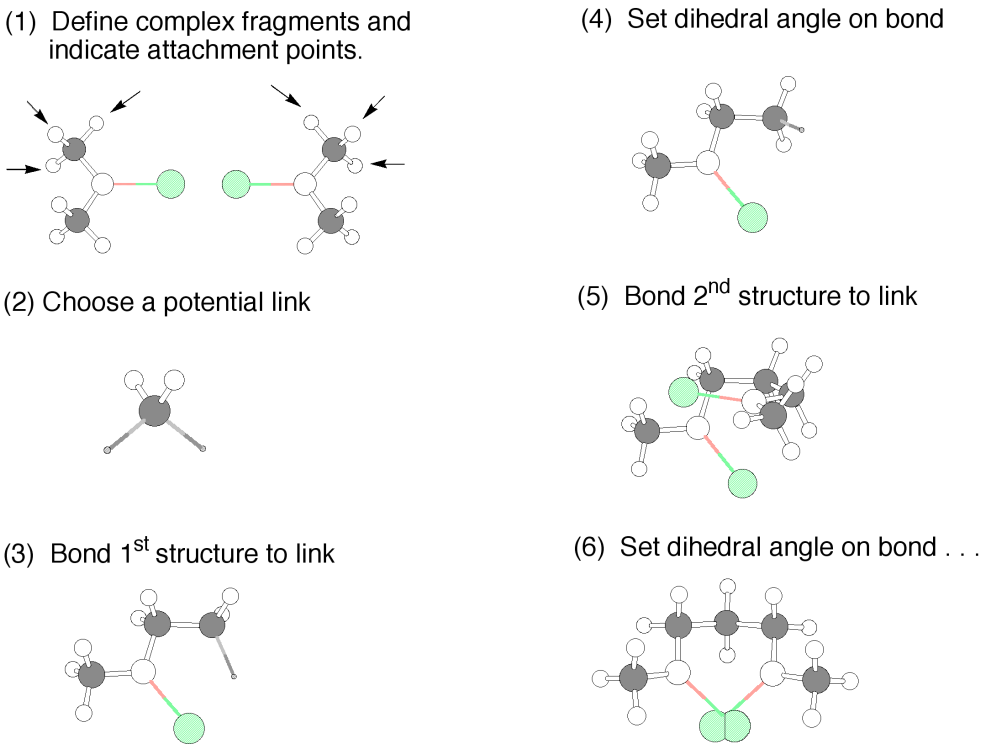


Figure 2. How the LINKER algorithm constructs a host structure from two input fragments and a linking fragment.

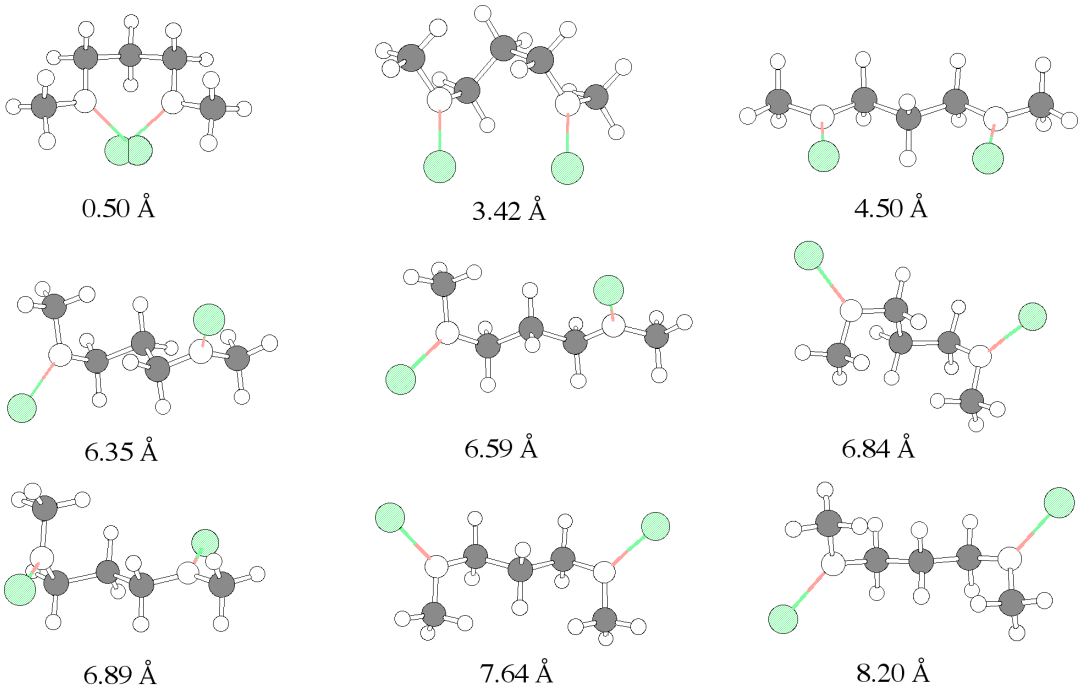


Figure 3. Host structures obtained by combining the three fragments shown in Figure 2. Degree of guest superposition, given here as Li-Li distance, is used for scoring the structures (see Section 2.1.3).

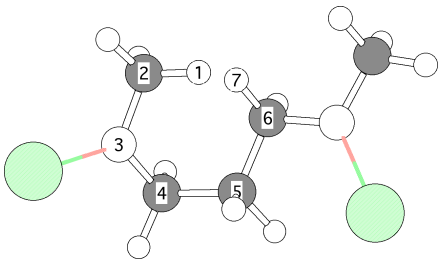


Figure 4. Example of a structure rejected due to a close H--H contact.

Every time a bond is formed, the new structure is checked for unacceptably close atom contacts between the fragments. The distance between each atom from the first fragment and each atom from the second fragment may not fall below a limit when the pair of atoms is separated by four or more bonds. When the pair of atoms is separated by three or fewer bonds, the distances are not checked. Therefore, atom pairs that are allowed to be close to one another are those connected to form the bond (atoms 5 and 6 in Figure 4), connected through two bonds (for example, atoms 7 and 5), or connected indirectly through three bonds (for example, atoms 7 and 4).

During a typical run, LINKER examines every possible connectivity and bond rotation with many different link structures. In addition, the ability to vary the geometry of the input fragments (see Section 3.5) can increase the number of starting structures dramatically. Therefore, it is possible to generate large numbers of structures. The resulting hosts are prioritized and only the best structures are written to the output files (see Section 2.1.3). The link library currently contains 8,266 structures (see Section 2.3), however the user has some control over which links will be used for building. Usage can be limited by specifying a minimum or maximum length of the linker, the number of rotatable bonds in the link, the valence of the bonding atoms, the maximum conformational energy of the linker, and other descriptors pertaining to chirality and symmetry. Sections 3.2 and 3.3 provide further detail on how to control this application.

Each bond formed by LINKER requires the assignment of a length and a dihedral angle. The parameters used to make these assignments is stored in a file named CONSTANTS and loaded when HostDesigner is initialized. The way that these parameters are applied is discussed briefly below.

Bond lengths are assigned using the following algorithm:

(1) Obtain element names for the two bonded atoms
(2) Obtain the covalent radius and electronegativity associated with each atom from data in the CONSTANTS file
(3) If both of the atoms have non-zero electronegativity, then use the Blom-Haaland relationship to compute the bond length (Blom, R.; Haaland, A. *J. Mol. Struct.* **1985**, *128*, 21-27):

$$R_{ij} = r_i + r_j - 0.085 \bullet |X_i - X_j|^{1.4}$$

where the bond length, R_{ij}, between atoms i and j is given in units of Å as the sum of their covalent radii, r_i and r_j, minus a constant times the absolute difference in the (Allred-Rochow) electronegativities of the two atoms, X_i and X_j.

(4) Else, simply sum the effective covalent radii to compute the bond length:

$$R_{ij} = r_i + r_j$$

(5) Because the linking fragment library consists of hydrocarbon molecules (See Section 2.3), the bonds that are formed to linking fragments will involve covalent attachment to a C atom. If one of the bonding atoms is C, adjust the Blom-Haaland bond length to account for differences due to the number and size of substituents on each bonded atom.

The CONSTANTS file contains an effective covalent radius for each element in the Periodic Table. A few lines from this file show how element data is formatted:

```
A    Li    3    6.9400    0.00    1.4060    1.489
A    Be    4    9.0122    0.00    0.8325    0.980
A    B     5   10.8100    2.01    2.1500    0.810
```

The A indicates that the line provides atom information. This is followed by the element name, the periodic number, the molecular weight, the Allred-Rochow electronegativity, the MM3 van der Waals radius in Å, and the effective covalent radius in Å. The electronegativity is assigned a non-zero value for hydrogen and all main group elements (Groups IIIA – VIIA) and a value of 0.00 for all other elements.

The performance of the bond length algorithm has been extensively tested. Calculated lengths for all 48 possible bonds between the main group elements of rows 1 and 2 were compared to average values observed in x-ray structures. With the exception of Al, the Blom-Haaland relationship gives the experimental average values to within ≤ 0.100 Å. After adding corrections for the number and size of substituents on each bonded atom to the algorithm, evaluation of 177 C–X bonds (X = C, N, O, P, S) verifies that MM3 bond lengths are reproduced with an average deviation of -0.002 Å. The largest deviation is 0.025 Å and 157/177 bond lengths were predicted to an accuracy better than ≤ 0.010 Å. Evaluation of M–C bonds for non-main group metal ions verify that the algorithm returns the correct experimental average values to within an average absolute deviation of 0.00 Å for C(sp^3) and 0.05Å for C(sp^2).

To assign preferred dihedral angles for a bond, the bond is viewed as a combination of two rotor types. In the initial version of HostDesigner, molecule building algorithms were limited to the formation of C–C bonds and consideration was restricted to combinations of the 10 possible rotor types shown in Figure 5. MM3 potential surfaces for each of the 55 possible bonds that can be formed by connecting these 10 rotor types were generated using the prototype fragments shown in Figure 5 and the results were stored in the CONSTANTS file. A rotor type

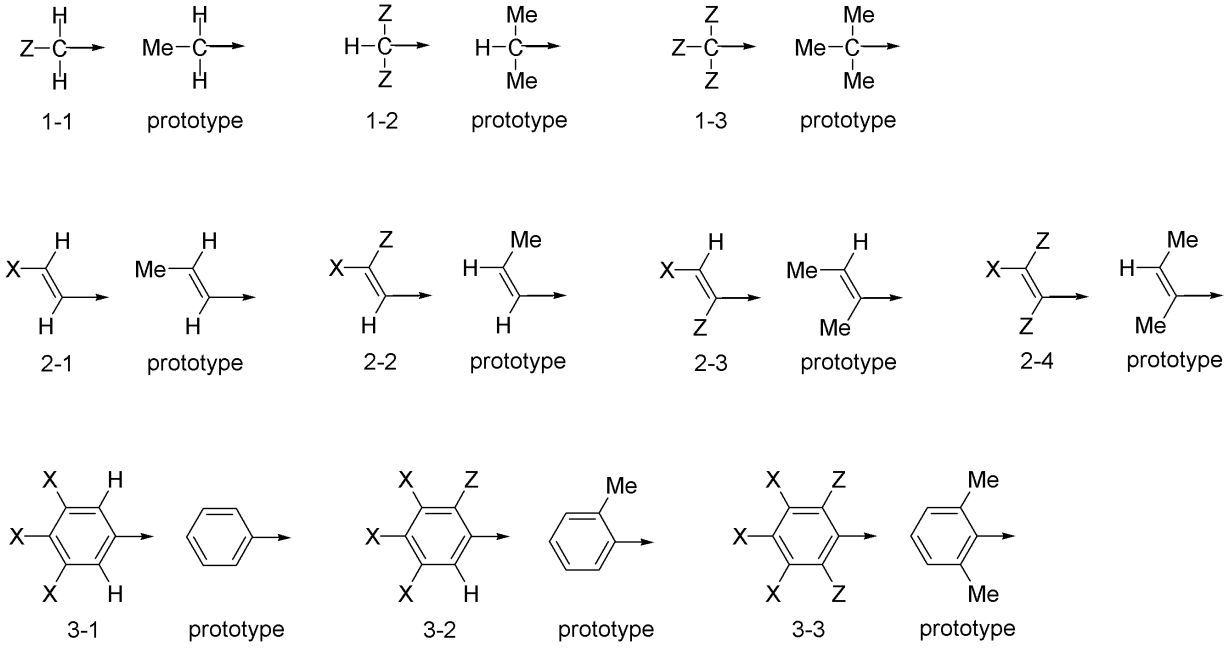


Figure 5. Rotor types recognized by the initial version of HostDesigner. The arrow indicates the bond vector used to attach the group to another rotor. Z indicates any non-hydrogen atom and X indicates any atom. Class and subclass are given below each type. Prototype cases were used to create molecules to generate the MM3 potential surfaces.

is specified by two integers, referred to as the class and subclass. For example, a primary alkyl group is designated as class = 1 and subclass = 1. When a primary alkyl rotor is connected to another primary alkyl rotor, dihedral data for the rotor combination is found in the CONSTANTS file as follows:

```
D 1 1 1 1 3
 65. 0.81
180. 0.00
295. 0.81
```

The first line indicates that this is dihedral angle information for connecting a rotor of type 1-1 to a rotor of type 1-1 and there are three rotamers. The next three lines give the dihedral angle in degrees and the relative energy in kcal/mol for each rotamer. Allowed rotor types are limited to those defined in the code (Figure 5 and 6). For any rotor combination, the maximum number of rotamers is currently limited to 6.

The linking fragment database used by HostDesigner contains molecular fragments made from alkanes, alkenes, and arenes (see Figure 9, Section 2.3). Each fragment has two bond vectors and each bond vector is pre-assigned to one of the 10 rotor types shown in Figure 5. Two things are required in order for HostDesigner to assign dihedral angles to any bond that is made to one of these linking fragments. First the code must be able to assign a rotor type to the attachment point in any input fragment. Second, the CONSTANTS file must contain dihedral

angle data for any combination of the input fragment rotor type with the 10 possible rotor types present in the linking fragment database (Figure 5).

The following strategy was adopted to meet these requirements. The number of rotor types identified by HostDesigner was expanded from the original 10 types to 53 possible types. The new rotor types, shown in Figure 6, were selected to represent the range of possible valency, geometry, and steric hindrance that is typically encountered in molecules. In many cases, the prototype examples for each rotor were chosen to represent functional groups commonly used to make connections in synthetic chemistry such as ethers, amines, amides, imines, and esters. Dihedral angle data for combination of new prototype rotors, Figure 6, with prototype rotors representing the linking fragment database, Figure 5, were obtained by evaluation of 370 additional rotational potential energy surfaces generated where possible with the MM3 model. The resulting data were used to expand the number of dihedral angle entries in the CONSTANTS file from the original 55 rotor combinations to a total of 475 possible rotor combinations.

Following this tabulation of dihedral angle data, the remaining task was to develop an algorithm to classify any attachment point in an input fragment as belonging to one of the 53 possible rotor types. This was accomplished by consideration of several descriptors. The first classification was based on the valency of the bonding atom, in other words, number of substituents attached to the rotor. Second, the geometry of the bonding atom was considered. Next, α substituents are classified as either small (H), medium (heteroatom), or large ($\geq$ 3 β substituents). When an α substituent is trigonal planar, then further analysis is performed to identify the identity (small, medium, or large as above) and orientation (*cis* vs. *trans*) of the β substituents. This general classification scheme assigns rotor types based on the similarity of steric and electronic characteristics. In cases where more than one rotor type exhibits the same characteristics, then element labels are used to make more specific assignments. The possible assignments for each valency are summarized below.

Valency = 1 This is a terminal atom where the only bond is the one made to the linking fragment. Because there is no other atom attached to a terminal atom, it is not possible to define a dihedral angle for a bond made to this atom. A terminal atom is assigned type 16-1.

Valency = 2 Two geometries are possible for this case, linear or bent. If the group is linear, then the rotor is assigned a type of 15-1. This assignment is used when the dihedral is free to take any value. If the group is bent, the rotor is assigned a type of 5-1, 5-2, 5-3, 5-4, 11-1, 11-2, 11-3, 14-3, or 14-4.

Valency = 3 Two geometries are possible for this case, pyramidal or planar. If the group is pyramidal, then the rotor is assigned a type of 6-1, 6-2, 6-3, 6-4, 8-2, 8-3, 9-1, 9-2, 9-3, or 9-4. The planar geometry, which exhibits the greatest number of variants, is assigned a type of 2-1, 2-2, 2-3, 2-4, 3-1, 3-2, 3-3, 4-1, 4-2, 4-3, 4-4, 7-1, 7-2, 10-1, 10-2, 10-3, 10-4, 12-1, 12-2, 12-3, 12-4, 13-1, 13-2, 13-3, 14-1, or 14-2.

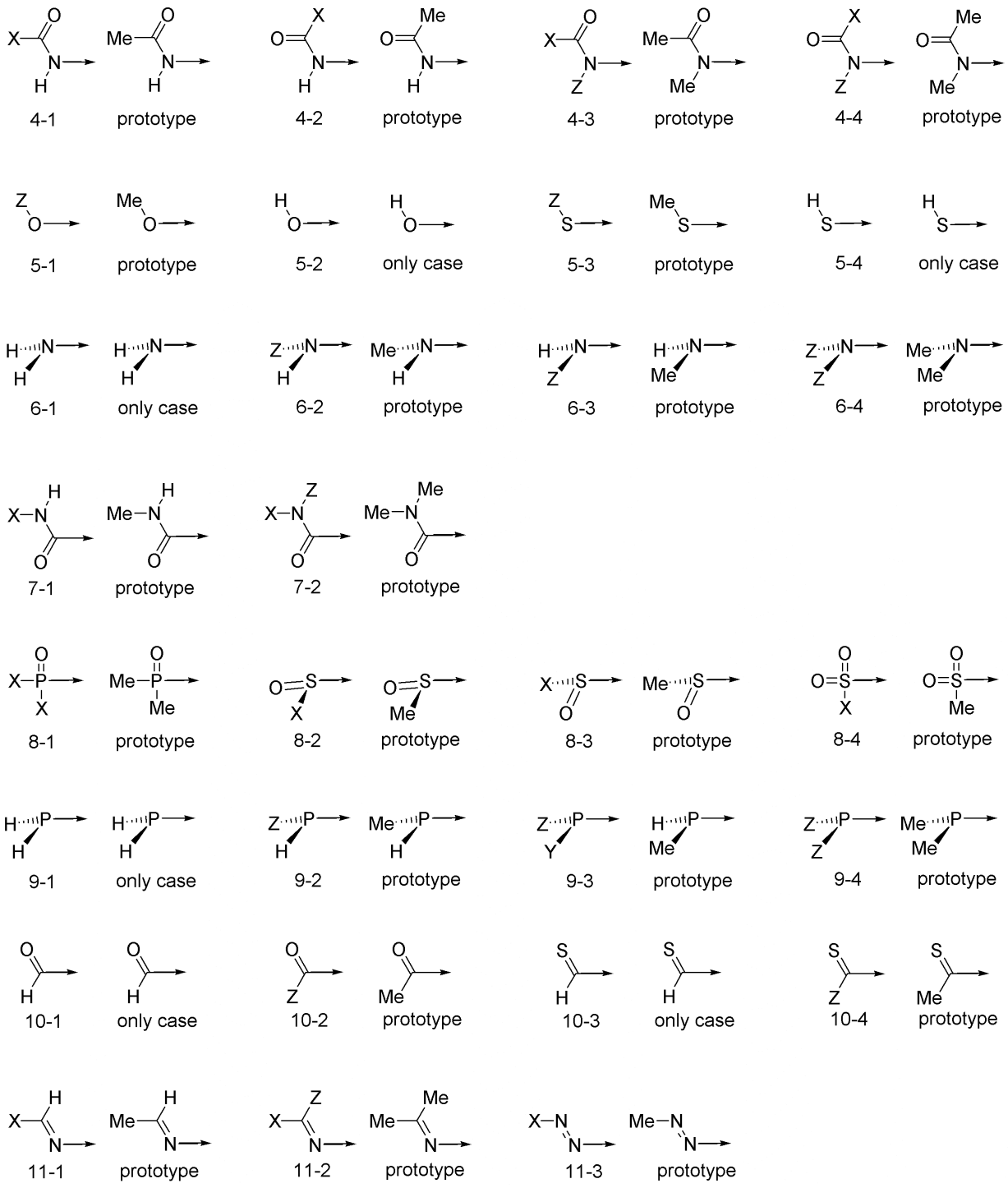


Figure 6 (continued on next page). Additional rotor types added to supplement the original alkane, alkene, and arene rotor types shown in Figure 5. The arrow indicates the bond vector used to attach the group to another rotor. Z is any non-hydrogen atom and X is any atom. Class and subclass are given below each type. Prototype cases were used to create molecules for the generation of MM3 potential surfaces.

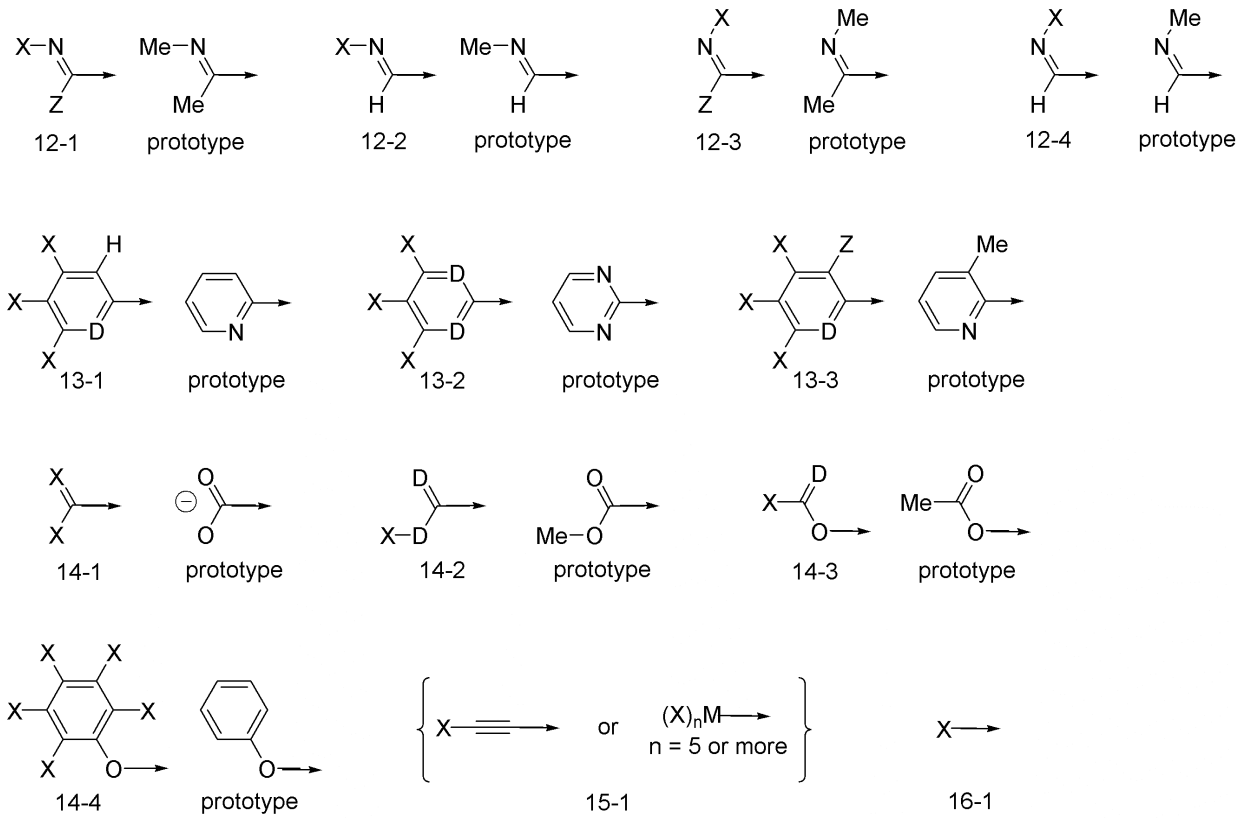


Figure 6 (continued). Additional rotor types added to supplement the original alkane, alkene, and arene rotor types shown in Figure 5. The arrow indicates the bond vector used to attach the group to another rotor. Z is any non-hydrogen atom, D is any heteroatom, and X is any atom. Class and subclass are given below each type. Prototype cases were used to create molecules for the generation of MM3 potential surfaces.

Valency = 4 Two geometries are possible for this case, tetrahedral or square planar. If the group is tetrahedral, then the rotor is assigned a type of 1-1, 1-2, 1-3, 8-1, or 8-4. If the group is square planar, the rotor is assigned type 3-3 based on comparison to dihedral angle distributions exhibited in the Cambridge Structural Database by M–C bonds in square planar complexes.

Valency ≥ 5 As with the linear divalent case, dihedral angles about bonds to atoms with coordination numbers ≥ 5 do not generally exhibit a significant preference for specific rotamer locations due to a high periodicity and low barrier heights. Such rotors are assigned type 15-1.

2.1.3 Scoring Structures Built by LINKER

Two scoring methods are used to prioritize the structures produced from a LINKER run. The first method ranks the structures in terms of complementarity estimated using geometric

parameters. The second method ranks the structures in terms of preorganization, based on an estimate of the conformational energy of the host structure.

During the construction of each input fragment, the guest was positioned relative to a host component to define a complementary geometry with the binding sites in that host component. When two input fragments are linked, the degree of superposition of the two guests provides a simple criterion for the rapid evaluation of the degree of complementarity in the new host. Optimal complementarity would be obtained when the root mean squared deviation, RMSD, of the distances between equivalent pairs of atoms in the two guests is zero, in other words, when the two guests representing the optimal bonding orientation with respect to each host component are exactly superimposed.

In the example given in Figure 3 each guest is a single lithium ion and the RMSD is simply the distance between them. These distances range from 0.50 to 8.20 Å, the host structure with the shortest Li--Li distance clearly gives the most complementary placement of the two ether binding sites. LINKER would use the Li--Li distance to score the generated host structures and output Cartesian coordinates for each structure in the order of increasing distance, in other words, in order of decreasing complementarity for the guest.

In the case of single atom guests, often the case with metal ions, there is an additional geometric feature that can be addressed when scoring the structures. Assuming that the geometries of the input fragments were accurately defined, then host structures that superimpose the two metal ions will, by definition, have complementary binding sites with respect to the M-L distances, the M-L-X bond angles, and the M-L-X-X dihedral angles (M = metal, L = donor atom, X = any atom). This definition of structural complementarity is sufficient for metal ion guests that do not exhibit bonding directionality such as the Group 1A and 2A metals and the trivalent lanthanides and actinides. However, some metal ions, such as Cu(II), Pt(II), and Pd(II), exhibit distinct metal-centered bonding directionalities. In these cases, a complementary host architecture must also provide an array of donor atoms to produce L-M-L angles that correspond to the preferred bonding directionality of the metal ion. To address this issue, an optional screening capability is available in LINKER to retain only those hosts in which the donor atoms are located near the verticies of idealized polyhedra. Any one of the following stereochemistries may be specified: tetrahedral, square planar, square pyramidal, trigonal bipyramidal, or octahedral (see Section 3.2).

In the case of multi-atom guests, the determination of RMSD requires HostDesigner to decide how the atoms from one guest should be paired with the atoms from the other guest for the superposition. This is done automatically by the code. If the guest has symmetry, there may be more than one way to pair the atoms. In such cases, LINKER will try every symmetry equivalent pairing of the atoms and report the minimal RMSD result.

In addition to ranking the structures by RMSD, a second method is used to rank the structures in terms of preorganization, based on an estimate of the relative conformational free energy of the host structure. The conformational free energy is estimated using the following equation:

$$\Delta G_{conf} = \Delta H_{link} + \Delta H_{bondA} + \Delta H_{bondB} + N_{rot} \bullet \Delta G_{rot}$$

The first three terms are enthalpic. A ΔH_{link} value is stored in the LIBRARY for each linking fragment (see Section 2.3). When the linking fragment has only one conformer, this value is zero. However, when the linking fragment has more than one conformer, the ΔH_{link} value is the relative enthalpy for that conformer with two methyl groups attached to the binding sites, obtained from MM3 calculations. The ΔH_{bondA} and ΔH_{bondB} terms are the relative rotamer energy associated with the first and second bond formed during the building process. The values assigned to each rotamer are based on MM3 potential surfaces for the groups shown in Figure 5. The final term, $N_{rot} \bullet \Delta G_{rot}$, is an estimate of the entropic penalty associated with restricted rotation of single bonds. The N_{rot} value is the sum of the rotatable bonds in the link, read from the LIBRARY, plus user-defined values for the attachment points in the input fragments. The free energy per rotatable bond is set to a default value of 0.31 kcal/mol per restricted rotation. After LINKER sorts the output by RMSD, the list of structures is sorted again, to yield a second output file prioritized by ΔG_{conf}.

The authors would like to caution the user from over interpreting the results produced from the LINKER algorithm. This algorithm is designed to examine large numbers of structures in a short period of time, identifying candidate architectures that are potentially complementary and/or preorganized for guest complexation by the definitions given above. Every effort has been made to provide accurate linking fragments and dihedral angle assignments. However, please recognize that small changes in individual structural parameters, which often do not cost much in terms of energy, can have a significant impact on the resulting structure. Because the algorithm is connecting rigid components and using generic dihedral angles, the ranking of the structural complementarity based on the superposition of guest atoms should be regarded as somewhat crude. When these distances differ by angstroms, then the difference between convergent and divergent binding sites is clear (see, for example, Figure 3). However, when these distances differ by less than an angstrom, the ranking becomes more uncertain. Similarly, the ranking of preorganization, based on conformational energies derived from simple alkane and alkene potential surfaces, may not accurately reflect the conformational stability of the host architectures. To obtain a more accurate prioritizations, the list of structures initially generated by LINKER can be subjected to further analyses using more expensive methods such as force field or electronic structure calculations.

2.2 The OVERLAY algorithm.

2.2.1 Overview

The OVERLAY algorithm represents a different approach than that used in the LINKER algorithm. It forms host molecules by combining one user–defined input fragment with a linking fragment. Two examples where OVERLAY could be applied are shown in Figure 7. In the first example (left), U(IV) forms a stable tetrakis-catecholate complex with a dodecahedral geometry. Is it possible to find a connecting structure to that will bridge two catecholates at the *ortho* carbons of the catecholate fragments? In the second example (right), two N-methylureas are coordinated to nitrate. Is it possible to find a connecting structure that will bridge the two ureas through the methyl groups?

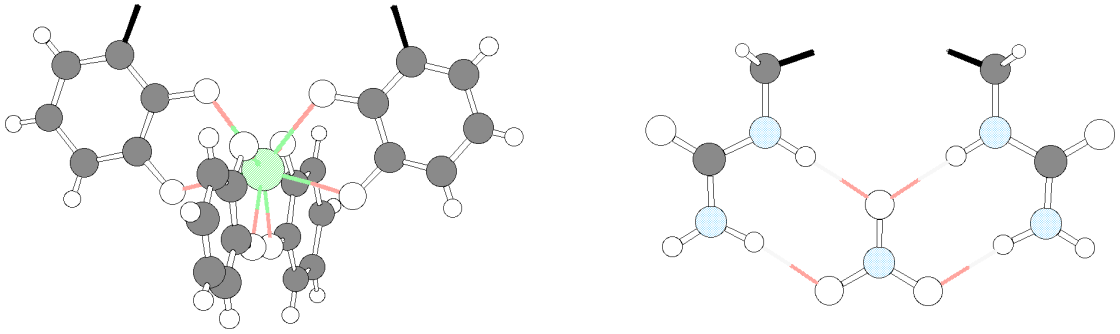
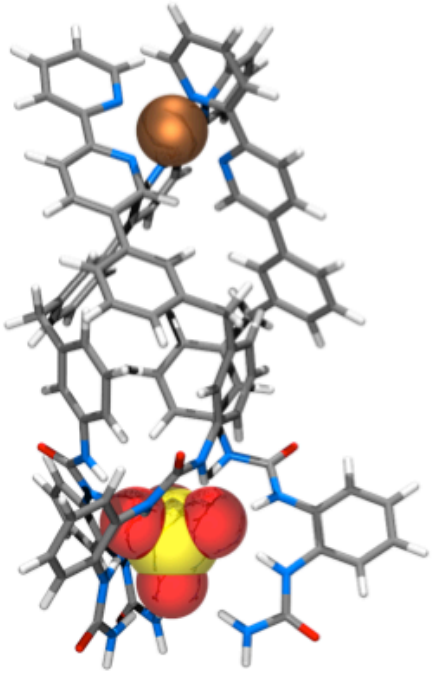
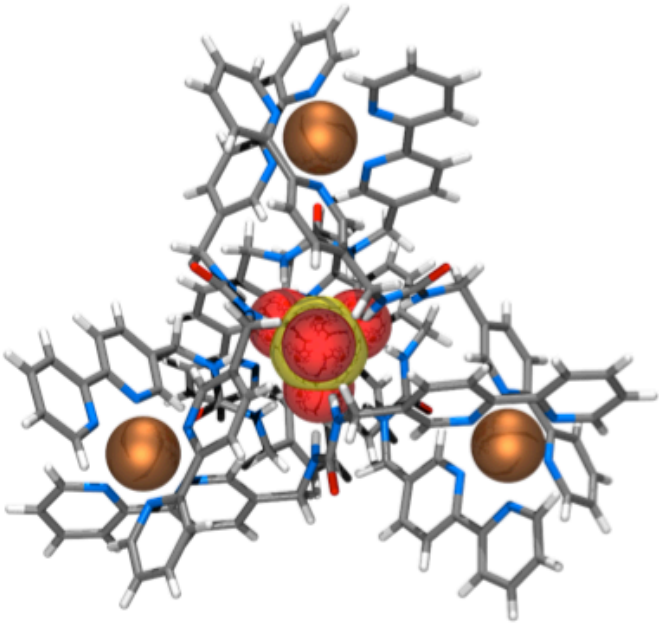


Figure 7. Examples of cases where the OVERLAY algorithm would be used. Possible bonding vectors, created by removal of hydrogen atoms are indicated by the black lines.

As outlined in a recent article (Young, N. J.; Hay, B. P. *Chem. Commun.* **2013**, *49*, 1354-1379), another application where OVERLAY has proven useful is the design of ligand shapes that direct the formation of self-assembled polygons and polyhedra. The high degree of symmetry in such assemblies coupled with the requirement that all edges and vertices have an identical shape allows the definition of input fragments with constrained bonding vector orientations. This approach has allowed the design of ligands that yield assemblies such as those shown below when mixed with the appropriate salts.



Jia, C.; Hay, B. P.; Custelcean, R. "De Novo Structure-Based Design of Ion-Pair Triple-Stranded Helicates" *Inorg. Chem.* **2014**, 53, 3893-3898.



Custelcean, R.; Bosano, J.; Bonnesen, P. V.; Kertesz, V.; Hay, B. P. "Computer-Aided Design of a Sulfate-Encapsulating Receptor" *Angew. Chem. Intl. Ed.* **2009**, *48*, 4025-4029.

2.2.2 The Building Algorithm

OVERLAY builds new host structures by forming two bonds between one input fragment and one linking fragment. The user must provide an input fragment file that specifies the coordinates for all the atoms, atom connectivity, and pairs of attachment points (see Section 3.4). Attachment points are indicated by listing hydrogen atoms that will be lost from the input fragment. The steps used by OVERLAY to construct a new structure are as follows: (1) select a linking fragment from the LIBRARY, (2) adjust the lengths of the bonding vectors on the input fragment and the linking fragment to the ideal lengths for the type of bonds that would be formed, (3) compare the geometries of the linking fragment bonding vectors and the input fragment bonding vectors, (4) if the vector geometry is similar, then superimpose the bonding vectors of the linking fragment on the bonding vectors of the input fragment to achieve the best overlay, and (5) form bonds between the input fragment and linking fragment. As with LINKER, OVERLAY is now able to handle attachments to any type of bonding atom.

Comparison of vector geometries in step (3) involves taking the difference in three geometric parameters shown in Figure 8. These are the distances, d_1 and d_2, and the dihedral angle, Φ. All three differences must be within defined tolerance limits. Smaller tolerance values give fewer results of higher quality. Larger tolerance limits give more results, but more of the structures may have distorted geometries. As a compromise, we use default tolerance limits of 1.00 Å for the distances and 90° for the dihedral angles. These variables, toldist and tolphi, are read from the CONSTANTS file and can be modified by the user simply by editing the CONSTANTS file. In addition, the user can control the quality of the structures that are accepted by specifying a maximum RMSD value for the bond overlay using the **maxrmsd** keyword in the control file (see Section 3.2).

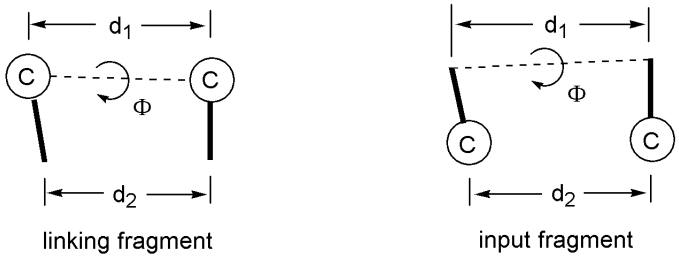


Figure 8. Parameters d_1, d_2, and Φ used to compare the bonding vectors in the linking fragment and the input fragment.

The resulting structure may still be rejected even if the bonding vectors of the linking fragment are perfectly superimposed on the bonding vectors of the input fragment. Although a perfect superposition ensures optimal bond distances and bond angles between the two fragments, the dihedral angles about each of the new bonds could have any value. Thus, after a new structure has been built, OVERLAY checks the difference between the actual dihedral angles and the dihedral angle corresponding to the nearest local minimum. As described above (see Section 2.1.2), data for the position of rotational minima have been derived from

examination of MM3 potential surfaces. The structure will be rejected if the difference in dihedral angles is greater than a default threshold value 20°. This threshold value can be altered through the use of the **tightness** keyword (see Section 3.2). Finally, as with LINKER, the structure will be rejected if there are any disallowed close contacts between non-bonded atoms in the linking fragment and the input fragment.

During a typical run, OVERLAY will examine many different link structures, but produce relatively few candidates that exhibit a good superposition of bond vectors. The ability to vary the geometry of the input fragment (see Section 3.4) can dramatically increase the number of bond vector poses, and therefore, generate larger numbers of candidates. The resulting hosts are prioritized and only the best structures are written to the output files (see Section 2.2.3). As with LINKER, the user has some control over which links will be used for building. Usage can be limited by specifying a minimum or maximum length of the linker, the number of rotatable bonds in the link, the valence of the bonding atoms, the maximum conformational energy of the linker, and other descriptors pertaining to chirality and symmetry. Sections 3.2 and 3.3 provide further detail on how to control this application.

2.2.3 Scoring Structures Built by OVERLAY

Two scoring methods are used to prioritize the structures produced from an OVERLAY run. The first method ranks the structures in terms of how well the linking fragment fits onto the input fragment. The second method ranks the structures in terms of preorganization, based on an estimate of the conformational energy of the host structure. The degree–of–fit is measured by the root mean squared displacement, RMSD, of four points on the input fragment with four points on the linking fragment, where in each case, the four points are the ends of the bonding vectors. In addition to ranking the structures by RMSD, a second method is used to rank the structures in terms of preorganization, based on an estimate of the relative conformational free energy of the host structure. The estimation algorithm is identical to that used with LINKER (see Section 2.1.3 for a description).

2.3 The Linking Fragment LIBRARY

A linking fragment library, named LIBRARY, is provided with the HostDesigner software. This section of the manual describes how it was developed. In building the initial library, it was decided (1) to limit the entries to molecules containing hydrogen and up to six carbon atoms, (2) to limit carbon hybridization to C(sp^2) and C(sp^3), and (3) to exclude three- and four-membered rings. With these limitations, a systematic evaluation yielded a total of 86 connectivities. Subsequently, 66 additional connectivities representing dimethylated 5 and 6 membered rings and 59 additional bi- and tri-cyclic connectivites have been added. These molecules, shown in Figure 9, have been used to create the current LIBRARY containing a total of 8,266 links.

2.3.1 Library Development.

The process used to create the linking fragments is summarized in Figure 10. The process starts by selecting a hydrocarbon molecule. The molecule is conformer searched with the MM3 force field. A conformer is selected and a pair of hydrogens are replaced by methyl groups and the structure is reoptimized with the MM3 program. When the link structure is used by HostDesigner, connections to input fragments will add steric bulk at these sites and by optimizing with methyl groups at these sites the link structure is pretreated for future attachment to carbon substituents. After optimization, the added methyl groups are then replaced by dummy atoms with C-dummy bond lengths of 1.0 Å. This generation process locates all conformations for a given connectivity. When multiple conformations for the same connectivity exist, the higher energy forms were added to the library only if their relative energy is ≤ 3.0 kcal/mol above the global minimum. The new link is checked for degeneracy, that is, to make sure that it neither identical to or the mirror image of an existing link. It is then added to the library. In constructing the library, this process was repeated for every possible pair of hydrogen atoms on the molecule and for every conformer of the molecule. Given the above process, note that the library does not contain both enantiomers of a chiral linking fragment. However, if a link is chiral, then both the LINKER and OVERLAY algorithms will, by default, consider the mirror image during the building process (see Section 3.2).

Initially, all new links were simply entered into the library in order of increasing molecular weight. However, as the size of the library increased, we encountered a problem with this approach. This problem has to do with the fact that many links have very similar geometry. For example, the 1,1-ethane link has bonding vectors that are almost the same as the 1,1-propane, 1,1-butane, 1,1-pentane, and 1,1-hexane links. Thus, if a 1,1-alkane linkage gives the best result, the code would generate an entire family of 1,1-alkane linked structures. To overcome this problem, we adopted a method of grouping link structures into families based on geometric features of the bonding vectors. At the same time, we developed a classification scheme to organize the library with respect to the connectivity, the hybridization of the bonding atoms, and the molecular weight of the link. Now the library is organized by class, as described in Section 2.3.3, with a maximum of 5 linking fragments per class. During a run, the user has the option to examine every link in the library or to limit examination to the first member of each class. The current library of 8,266 links represents a total of 5,094 classes.

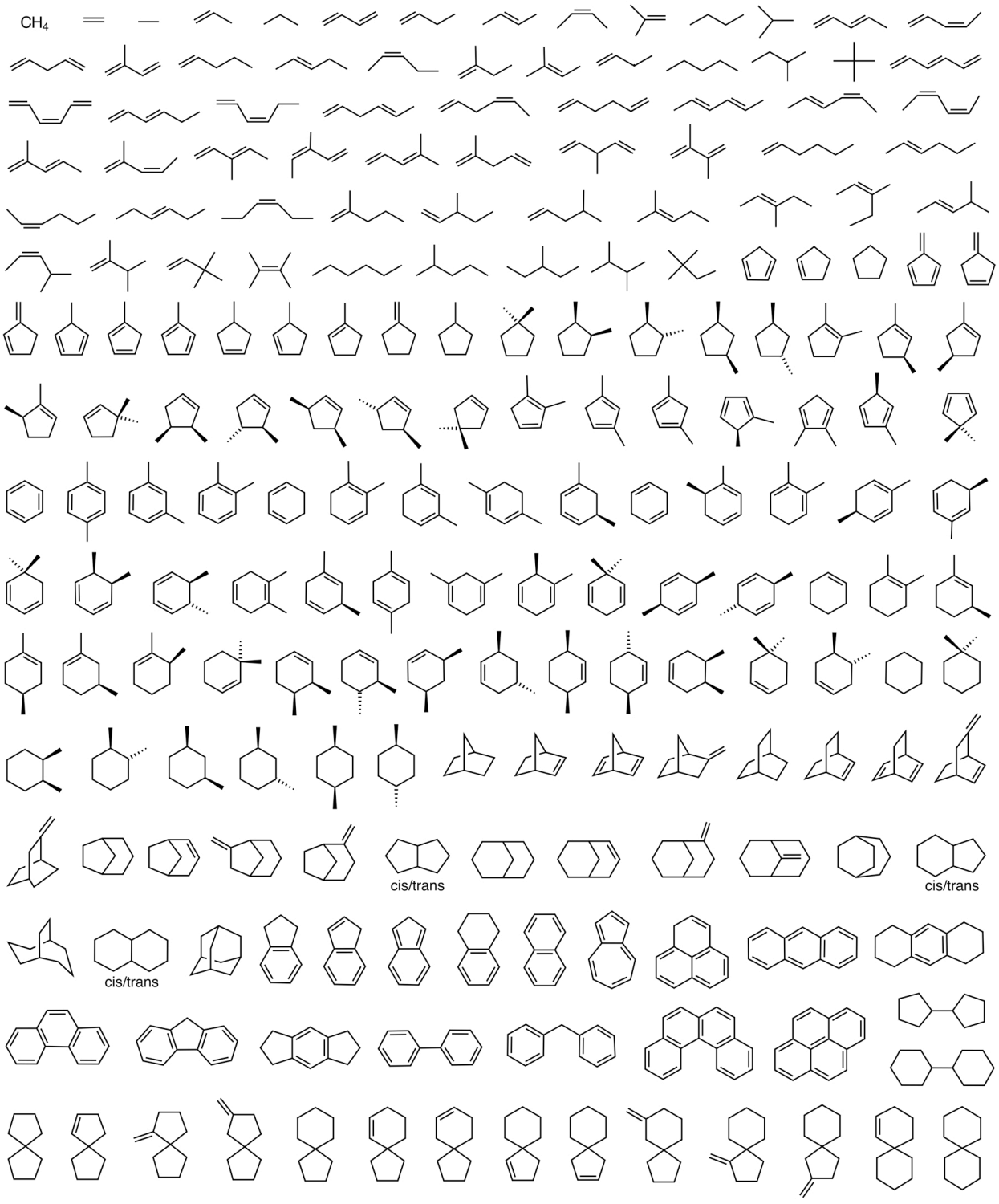
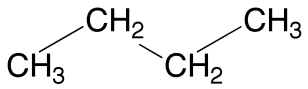
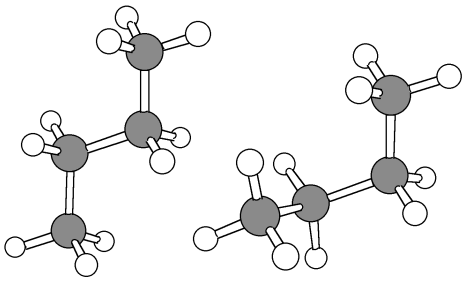


Figure 9. The 206 hydrocarbons used to derive the linking fragment library represent all possible alkanes and alkenes containing up to six C atoms (excluding 3- and 4-membered rings), all dimethylated 5- and 6-membered rings, and selected polycylic systems.

(1) Select a hydrocarbon. Here we are using n-butane.

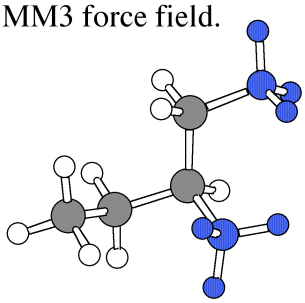


(2) Search to identify all stable conformers. Here there are two.

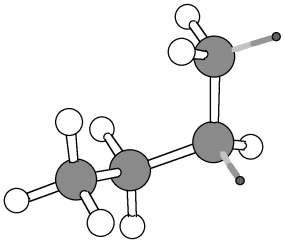


(3) For each conformer, identify all possible hydrogen pairs. With 10 hydrogens, these conformers each have 45 hydrogen pairs, in other words, n-butane could yield as many as 90 linkages.

(4) For each hydrogen pair, replace hydrogens with methyl groups and optimize the structure with the MM3 force field.



(5) Replace the methyl groups with dummy atoms and set the dummy atom to carbon distances to 1.0 Å.



(6) Check for degeneracy, that is, the link is neither identical or the mirror image of an existing link.

(7) Add the new link to the library.

Figure 10. A summary of the process used to create linking fragments.

2.3.2 Format of a Linking Fragment LIBRARY Entry

The LIBRARY is an ascii file consisting of a sequential series of entries. Each entry contains information that completely describes a linking fragment. Below is an example of the entry for a linking fragment made from butane. Except for the first line, each entry is read in a free format fashion, which means that it is not critical how many spaces are left between each entry on a line. The blue numbers on the right of this sample library entry are line numbers for refererence in this manual, and are not used in practice.

```
LINK     4278 C( 0),H( 0),N( 0),O( 0) propane                                  1
 11    3 C     3    1    1    2                                               2
 10    1 C     3    1    1    2                                               3
  3 T F F F T   0.000    4                                                     4
   2.55629 146.29149 146.29583    0.00465                                     5
 11                                                                           6
  1    C       1.453300     0.029300     2.092200    1    2  10    4    5     7
  2    C       1.425400     0.000700     0.555700    1    1   3    6    7     8
  3    C      -0.004900    -0.028100    -0.006600    1    2  11    8    9     9
  4    H       0.897300     0.923400     2.457900    5    1                  10
  5    H       0.920300    -0.864800     2.490800    5    1                  11
  6    H       1.959800     0.894900     0.159900    5    2                  12
  7    H       1.982900    -0.893400     0.192800    5    2                  13
  8    H      -0.541500    -0.922300     0.386800    5    3                  14
  9    H      -0.564500     0.865900     0.353900    5    3                  15
 10    X       2.383564     0.048067     2.458610    0    1                  16
 11    X      -0.023668    -0.046738    -1.006250    0    3                  17
```

Line 1: Always begins with 'LINK'. Characters 5-12 are a serial number that should be unique. The example given is the 4278th entry in the LIBRARY file. Characters 38 through 75 are the name of the molecule from which the link was derived, and will be printed in the output along with the serial number.

Lines 2 and 3: These two lines concern the attachment points. Each link has two attachment points, which are indicated by the dummy atoms that will be lost when bonds are made (in this case, 'atoms' 10 and 11.) There are seven variables per line: the serial number of the dummy atom, the serial number, atom label, and hybridization (2 = sp^2, 3 = sp^3) of the atom it is bonded to, the rotor class and subclass (Figure 5) and finally the serial number of one of the other atoms bound to the binding atom for use as a reference point for setting dihedral angles.

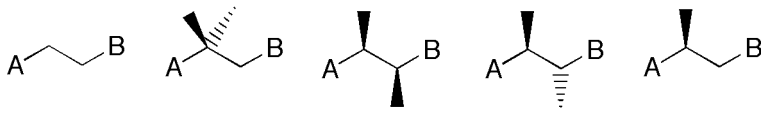
Line 4: Contains eight variables: the number of atoms in the shortest path between the two dummy atoms, attachment symmetry based on 3D structure, superimposable mirror image, chiral based on connectivity, prochiral based on connectivity, attachment symmetry based on connectivity, relative energy of the conformer, and the number of rotatable bonds in the link.

The number of atoms in the shortest path between the two dummy atoms is an integer between 0 and 999.

The 3D attachment symmetry is a logical variable. A value of T indicates that switching the order of attachments would yield either the same 3D structure or its mirror image.

The superimposable mirror image is a logical variable. A value of T indicates that the mirror image of the 3D link structure is superimposable on the original 3D link structure.

The next three logical variables refer strictly to the connectivity of the link, independent of its 3D conformation. Chiral is T if the link is chiral, prochiral is T if the link is prochiral, and the attachment symmetry is T if switching the attachment points for two structurally different groups yields a structure with identical connectivity. There are five possible settings for these three variables. Examples are illustrated below where groups A and B have been attached to various linking fragments:

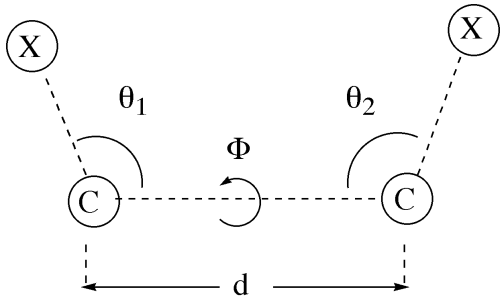


chiral	F	F	F	T	T
prochiral	F	F	T	F	F
attach sym	T	F	F	T	F

The next variable gives the relative conformational energy of the link. This number is one of the components used to estimate the conformational energy of the host (see Section 2.1.3).

The final number is the number of rotatable bonds in the link. With one exception, this value is defined as number of single bonds between the dummy atoms that are not within a cyclic structure. The exception is the null link, used by LINKER to connect the two input fragments directly to each other. In this case, there is one rotatable bond.

Line 5: Contains four variables that describe the geometry of the bonding vectors: distance (d), angle1 (θ_1), angle2 (θ_2), and dihedral angle, (Φ) . These values are used to classify the link and also used in the OVERLAY algorithm to decide whether to attempt to superimpose bonding vectors. The parameters are defined as shown in the scheme below where d is the distance between the two carbon atoms, θ_1 is the angle between the first C-X vector and the C-C vector, θ_2 is the angle between the second C-X vector and the C-C vector, and Φ is the dihedral angle between the two C-X vectors about the C-C axis. In the existing library, the molecule is numbered such that θ_1 is always $\leq \theta_2$. In addition, the chirality of the molecule is chosen such that Φ is always in the range of 0 and 180°. In the special case when both dummy atoms are attached to the same carbon, then $\theta_1 = \theta_2 = 90° + 1/2$ the angle between the two C-X vectors and $\Phi = 0°$.



Line 6: The number of atoms in the link including the two dummy atoms X.

Lines 7 to end: There is one line per atom. Each line contains the following informaton: The serial number, the atom label, x, y, and z coordinates, the MM3 atom type number, and a connectivity list by serial number. The atom label is a two character string. It is assigned a value of X for the dummy atoms. The MM3 atom type numbers used in the current library are limited to 1 (sp^3 carbon), 2 (alkene carbon or arene carbon), 5 (hydrogen) and 0 (dummy atom). The connectivity list is simply a list of all serial numbers to which this atom is attached. The

connectivity list can contain up to ten atoms. Note that the serial order of the atoms is important. The carbons (and, in principle, any other heavy atom would) always go first, the hydrogens always go next, and the two dummy atoms must always go last.

2.3.3 How to Add a New Link to the LIBRARY.

The user can create additional links to expand the existing LIBRARY or they can define their own personal linking fragment library. Assuming that the user has created a new link and formatted a link entry as described above, the entry should be added to the existing library as follows.

The library is organized by class and each class section begins with a header line. An example of a class header line used in the library is shown below:

```
CLASS  connectivity =  1  nrot =  2  rot1 = 1,1  rot2 = 1,1  type(    1)
```

Classes are defined by the connectivity, in other words, the number of atoms that would be between two groups connected to this link, by the number of rotatable bonds, by the type of rotor represented by each bonding vector (see Figure 5), and by the geometry of the bonding vectors. Classes are arranged first in order of increasing connectivity (number of atoms in the shortest path between connection points), next in the order of increasing number of rotatable bonds, by rotor type, and finally by difference in the bonding vector geometry. If a new link has the same characteristics of an existing class, and the geometric parameters are the same as the first member of that class, to within 0.1 Å for distance, 3° for angles, and 5° for dihedrals, then it is a member of that class. Otherwise, it becomes the first member of a new type for a class with the same characteristics.

Before adding a link to the library, it is necessary to identify the class to which the new link belongs. If the new link does not belong to an existing class, then you should create a new class header line and insert it in the proper position in the library. Other than the initial four characters, CLAS, the contents of this line are not used by the code. However, note that when inserting a new class within the library, it is important that the connectivity of the linking fragment, in other words, the number of atoms in the shorts path between the two dummy atoms, be in ascending order. If the user specifies a maximum connectivity, then the code will skip the remainder of the library once it encounters a link that exceeds that connectivity.

2.3.4 Using an Alternative Fragment Library

In this release of HostDesigner a second fragment library named LIBRARY_syn has been included with the data files. This alternative library was developed with the intent of producing more synthetically tractable molecules. It consists of a subset of 4756 fragments present in the default LIBRARY file where links derived from linear dienes and trienes, spiro compounds, and other selected bicyclic compounds have been excluded. Invoking the use of an alternate fragment library is specified with the **linklib** keyword in the control file (see Section 3.2).

3.0 HOW TO USE HOSTDESIGNER

3.1 Installation

The HostDesigner download package contains the following directories:

00_Documentation: HostDesigner User's Manual

01_Source: HostDesigner source code

02_DataFiles: LIBRARY and CONSTANTS data files used by HostDesigner

03_Examples: Example HostDesigner input files

04_HDViewer: A graphical user interface for HostDesigner

Before you can start using the software, you must compile the HostDesigner executable and properly configure your system.

3.1.1 Compiling the Code

HostDesigner is provided as Fortran source code. The user must compile the code before it can be used. This means that the user must have the capability to compile Fortran source code. Over the past 8 years, the HostDesigner executable has been created using the GCC Fortran compiler named 'gfortran'. Links to download 'gfortran' at no cost are available at https://gcc.gnu.org

The downloaded directory 01_Source contains the files needed to create the HostDesigner executable. Assuming that that both 'gfortran' and 'make' executables are available in the user's path, simply open a terminal window, enter the 01_Source directory, and run make. This should produce the executable named 'hd3.0' in less than a minute.

Over the years a variety of different Fortran compilers have been used successfully to make HostDesigner for LINUX, MacOS, and Windows operating systems. Thus, it is expected that alternative Fortran compilers will also work after making appropriate changes to the makefile (name of compiler, compiler directives, location of runtime libraries, etc).

Parameters controlling the size of important arrays within the code are found in an include file named 'params.i'. Examples include the maximum number of atoms in the input fragments, maximum number of atoms in a guest, maximum number of atoms in a linking fragment, maximum valence, maximum number of geometry drives, etc. If the default settings do not accommodate the needs of the user, it is often possible to address the deficiency by modifying these parameters and recompiling the code.

One situation that has frequently arisen when using the code in the OVERLAY mode is the need to increase the number of drives. A sample of the settings used by the authors for this purpose is provided in a file named 'params_over.i'. A warning here is that it is possible to set the array sizes too large such that the code will either not compile or will not run. If this happens, then reduce the array size and try again.

3.1.2 Configuring the Environment

HostDesigner is a command line executable and, once the system is properly configured, the user should be able to invoke it from any directory. After hd3.0 has been successfully compiled, there are two additional things that must be done for this to happen. First, the executable must be placed in a location that is in the user's path. Second, the executable must be provided the location of two required data files – CONSTANTS and LIBRARY.

To satisfy the first requirement, put the executable hd3.0 in a directory that is part of the user's default path. For example, user *hay* keeps his executables in a directory with the path /Users/hay/bin. So, *hay* added the executable hd3.0 to this directory. Being a tcsh user, *hay* added this directory to his default path by including the following line in his .tcshrc file:

```
set path = ( $path /Users/hay/bin )
```

If *hay* had been a bash user, then he would have accomplished this by adding the following line to his .bashrc file:

```
export PATH="PATH:/Users/hay/bin"
```

The second requirement is accomplished by setting an environmental variable named 'HD_DIR' to the path of a directory containing the two data files CONSTANTS and LIBRARY. These two files are in the directory 02_Datafiles provided in the download package Continuing with the above example, user *hay* created a new directory named hd_dir in the same location as the executable. He put the files CONSTANTS and LIBRARY in this new directory. He then assigned the environmental variable HD_DIR by including the following line in his .tcshrc file:

```
setenv HD_DIR /Users/hay/bin/hd_dir
```

If *hay* had been a bash user, then he would have accomplished this by adding the following line to his .bashrc file:

```
export HD_DIR="/Users/hay/bin/hd_dir"
```

After completing the above steps, the configuration should be tested by opening a new terminal window and issuing a couple simple commands. Issuing the command:

```
which hd3.0
```

Should return the response:

```
path_to_the_executable/hd3.0
```

In the example above, user *hay* should get the response:

```
/Users/hay/bin/hd3.0
```

To verify that the environmental variable has been properly set, issue the command:

```
echo $HD_DIR
```

This should return the value that HD_DIR has been set to. In the above example, user *hay* should get the response:

```
/Users/hay/bin/hd_dir
```

3.1.3 Running test cases

Directory 03_Examples in the download package contains eight test cases that can be run to verify that HostDesigner has been correctly compiled and that the system has been correctly configured. Each test case consists of a control file, input fragment(s), and a prior summary file obtained when the test was last run.

To run any of the test cases, open a terminal window, enter one of the subdirectories containing the input files (named case#), type hd3.0 on the command line, and hit return. The code should execute and begin printing information to the screen. All the test cases should complete within a couple of minutes and produce several output files. To verify expected performance, compare the output file with the .summ extension to the provided file named prior.summ, which is output obtained when the authors previously ran the example. The timings will be machine specific, but the number of links read, the number of links used, the total number of structures examined, and the number of structures stored should be identical.

3.1.4 HDViewer

In 2004, there was a successful attempt at creating a graphic user interface for HostDesigner. The result was the software named HDViewer that can import structures from several molecular file formats, generate host fragment input files and control files needed to run HostDesigner, view HostDesigner output files, and export structures in several file formats. The download package directory 04_HDViewer contains a User Manual and executables that allowed this code to be used on MacOS or Windows operating systems. The provided executables were last known to work with MacOS 10.7 and Windows 7. However, given the continuous evolution of these operating systems over time, there is no guarantee whether HDViewer will continue to function going forward.

3.2 The File Named 'control'

The control file is a mandatory input file that tells HostDesigner what to do. It must be named 'control'. The control file can be prepared with any plain text editor. Alternatively, it is possible to generate a control file with the HDViewer utility (see Section 6.0 of the HDViewer User's Guide).

A 'control' file is composed of a list of keywords. These keywords are entered on lines that are 80 characters in length. You may put as many keywords on one line that will fit within this length. If more than one line is used, then you must specify that another line will be read by placing the keyword 'AND' in the preceding line.

Here is an example of acceptable format for the 'control' file:

```
keyword1  keyword2  keyword3  keyword4  keyword5
```

And this is also an acceptable way to format the same information:

```
keyword1          AND
keyword2          AND
keyword3          AND
keyword4
```

Keywords can be given in any order. Keywords must be separated from one another by at least one space. Some keywords are used to specify additional numeric or string input. In such cases, no spaces may be present within the keyword. For example, '`numview = 200`' will result in an error while '`numview=200`' will not. Note also that keywords are case sensitive. The following are a list of keywords that are allowed by HostDesigner.

Mandatory Keyword:

One, and only one, of these two keywords must be present in the 'control' file.

LINK Tells the code to do a LINKER run

OVER Tells the code to do an OVERLAY run

Optional Keywords:

AND Used when more than one line is needed in the control file. Instructs the code to continue reading input from the next line. If the line being read does not contain the **AND** keyword, then all subsequent lines are not checked for keywords.

hosta=name Specifies the filename containing the input fragment used in OVERLAY, or containing the first input fragment used in LINKER (See Sections 3.3 and 3.4). The name may be a maximum of 120 characters in length. If no name is specified, then the default filename is 'hosta'.

hostb=name Specifies the filename containing the second input fragment used in LINKER (See Section 3.3). The name may be a maximum of 120 characters in length. If no name is specified, then the default filename is 'hostb'.

notype By default, the code expects to read force field atom type information in the input fragments. If the **notype** keyword is present, this eliminates the need to include atom type information (see Section 3.3 and 3.4).

out=string A string with a maximum length of 20 characters. The default setting is 'out'. The string specified by this keyword is used as the prefix for the output files, which will thus be named 'string.summ', 'string_1.hdo' and 'string_2.hdo' (see Section 3.6).

numview=# An integer equal to the maximum number of structures to be printed to output. The maximum possible value is 1000. The default is set to 100.

xyz Limits the information provided in the output files. By default, each atom line contains the label, x, y, z Cartesian coordinates, atom type, and list of connected atoms. If the **xyz** keyword is present, each atom line contains only the label and the Cartesian coordinates (format used with earlier HostDesigner Version 1.0). The HDViewer utility will read either output format.

linklib=name Specifies the path and name of the link library. The name may be no more than 120 characters in length. If no name is specified, then the default filename is 'LIBRARY'. Here is an example showing how to invoke the use an alternative library: linklib=/Users/hay/bin/hd_dir/LIBRARY_syn

mirroraoff By default, LINKER uses both the 'hosta' structure and its mirror image for building structures. This keyword turns this feature off and uses only the geometry specified in 'hosta'.

mirrorboff By default, LINKER uses both the 'hostb' structure and its mirror image for building structures. This keyword turns this feature off and uses only the structure specified in 'hostb'.

drivea This keyword applies geometry drives specified for 'hosta' (see Section 3.5).

driveb This keyword applies geometry drives specified for 'hostb' (see Section 3.5).

testdrive This keyword is used to test the geometry drive specifications. When used, the code will write out all sets of coordinates for the input fragments *after drives have been applied* and stop before any host structures are built. The results are written to files named 'string_testa.hdo' and/or 'string_testb.hdo', where 'string' is defined by the out=string keyword.

metshape=name Used to reject host structures when binding sites do not fit a specified topology about a single atom guest. Reads a 4 character string that may be set equal to NONE, TETR (tetrahedral), SQPL (square planar), TBPY

(trigonal bipyramid), SQPY (square pyramid), or OCTA (octahedral). The default setting is NONE. This keyword can be applied to partial occupations of a target polyhedron, keeping only those structures in which the binding sites are on contiguous verticies. For example, with tridentate structures, the OCTA specification would keep only structures with either *fac* or *mer* topologies.

useclass This keyword makes HostDesigner only use the first structure from each class in the link library.

nochiral All chiral links are discarded.

noprochiral All prochiral links are discarded.

noasym All links that would give a different connectivity when the attachment points are switched are discarded.

minconn=# An integer equal to the minimum number of atoms in the minimal path through the link, this defaults to 0. If set higher, the shorter path structures in the link library will be skipped.

maxconn=# An integer equal to the maximum number of atoms in the minimal path through the link, this defaults to 20. If set lower, the longer path structures in the link library will be skipped.

maxconfe=# A real number which defaults to 100.0. A potential hit will be discarded if the estimate of its relative conformational energy (see Section 2.1.3) exceeds **maxconfe** kcal/mol.

maxnrot=# An integer which defaults to 20. A potential hit will be discarded if the total number of rotatable bonds, the sum of rotatable bonds in the linking fragment (see Section 2.3.2) plus rotatable bonds in the input fragment (see Section 3.3 and 3.4), exceeds **maxnrot**.

maxrmsd=# A real number which is used to control whether a hit will be stored in memory during a LINKER or OVERLAY run. Default value is 100. When the root-mean-squared deviation, RMSD, for the hit exceeds **maxrmsd**, the hit will be rejected. RMSD measures the degree of guest superposition in LINKER runs (see Section 2.1.3) or the degree of bond vector superposition in OVERLAY runs (see Section 2.2.3).

tightness=# A real number which is used to control rejection of hits in OVERLAY due to bad torsion angles on the two bonds that are formed. The default value of 1.0 allows a deviation of 20° from the possible rotational minima read in from CONSTANTS. The allowed deviation = 20°/**tightness**. Thus, setting **tightness** to a larger number will give fewer hits, but of higher quality.

Here is an example of a 'control' file that presents the minimal amount of input:

```
OVER
```

This control file tells the code to use the OVERLAY algorithm. It will look for the default names of the other input files, in other words, the link library must be named LIBRARY and the input fragment must be named 'hosta'. Output files will be named 'out_1.hdo', 'out_2.hdo' and 'out.summ'. All other options will be at their default settings.

Here is an example of a 'control' file that uses many keywords:

```
LINK hosta=one hostb=two linklib=short mirroraoff mirrorboff AND
metshape=OCTA drivea driveb useclass minconn=1 maxconn=3 AND maxnrot=3
maxconfe=0.0 maxrmsd=2.0 numview=20 out=three
```

This control file tells the code to use the LINKER algorithm with input files named 'one' and 'two', links from a library named 'short', not to consider mirror images of either input fragment, to screen for octahedra, to consider only the first links in each class in the library, to screen out links with connectivities < 1 or > 3 and with at most 3 rotatable bonds, to use only links derived from minimum energy conformers, to name output files 'three_1.hdo', 'three_2.hdo', and 'three.summ', store only structures with RMSD values less than or equal to 2.0 Å, and write at most 20 structures each to 'three_1.hdo' and 'three_2.hdo'.

3.3 Input Fragment Files for the LINKER mode.

In addition to the 'control' file, the LINKER mode requires the user to prepare input files that define the structure of the fragments that will be connected to the link structures. The two input fragment files, 'hosta' and 'hostb' (see previous section and Section 5.1 of the HDViewer User's Manual), can be identical or two different input fragments can be used. The following is an example of the input fragment file for the dimethylether lithium complex shown in Figure 2 above. All lines are free format so that it does not matter how many spaces are left between the entries.

```
Lithium dimethylether, M-O = 2.26 Å                                          1
 10    1                                                                     2
  C    1   -0.796860   -1.176666   -0.008575   1   2   4   5   6            3
  O    2    0.000000    0.000000    0.000000   6   1   3  10                4
  C    3   -0.795944    1.177567   -0.008606   1   2   7   8   9            5
  H    4   -0.092346   -2.040680   -0.009293   5   1                        6
  H    5   -1.374283   -1.179657    0.945770   5   1                        7
  H    6   -1.366318   -1.171310   -0.968475   5   1                        8
  H    7   -1.368774    1.184189    0.948334   5   3                        9
  H    8   -1.369934    1.169693   -0.965530   5   3                       10
  H    9   -0.090622    2.041031   -0.015411   5   3                       11
 Li   10    2.259933    0.002136   -0.007889 301   2                       12
  3                                                                         13
  4   C  0   1                                                             14
  5   C  0   1                                                             15
  6   C  0   1                                                             16
```

Line 1: A title that can be up to 60 characters long.

Line 2: Two integers. The first integer is the number of atoms in the structure, n. The maximum allowed value is 200. The second integer, ng, is the number of atoms in the guest. The maximum allowed value is 50.

Lines 3 to 2 + n: Each line describes an atom in the structure and contains the following information: the atom label, the serial number, the x, y, and z coordinates, the atom type number (optional), and a connectivity list. **IMPORTANT: The atoms must be in serial order with the non-hydrogen host component atoms first, the hydrogens next, and the guest last. If dummy atoms are used, they must be numbered before the guest**.

> *atom label* is a two character string, with one exception corresponding to the normal atom abbreviations used in the periodic table. The exception occurs with dummy atoms, used in optional geometry drives, which must be named either 'Du' or 'DU'. The atom label is used to set atom radii that are used in collision checks.
>
> *serial number* is an integer. It is important that the atoms be sequentially numbered from 1 to n.
>
> *x, y, and z coordinates* are real numbers specifying the atom positions in Cartesian space in units of Å.
>
> *atom type number* is an optional integer that can be used to specify the force field atom

type. In this example, we are using MM3 atom types. By default, the code expects an atom type will be read in as an integer. This information is not used by HostDesigner, but by default is written to output files to facilitate molecular mechanics post-processing. If you do not wish to include atom types, then add the **notype** keyword to the control file.

connectivity list is a series of no more than 10 integers that specify the serial numbers of the atoms attached to this atom.

Line 3+n: An integer, na, giving the number of attachment points to the structure. Minimum value is one. Maximum number of attachments is 20 for 'hosta' and 10 for 'hostb'.

Line 4+n to 3+n+na: Each line contains up to four variables to describe one attachment site. These are the serial number of a hydrogen atom that will be lost, atom label specifying the kind of atoms that can attach, the hybridization of atoms that can attach, and the number of rotatable bonds (optional).

serial number of hydrogen atom is an integer from 1 to n that specifies which hydrogen atom will be lost to generate an attachment point. Currently hydrogen atoms may be lost from alkane carbons, alkene carbons, arene carbons, aliphatic alcohol oxygens, phenol oxygens, amine nitrogens, and amide nitrogens. In other words, bonds can be made only to these types of atoms.

atom label is a two character string, corresponding to the normal atom abbreviations used in the periodic table, which must match the atom label of the link atom that will replace the hydrogen. At this time, all links in the library bond to the input fragment via carbon only. Thus, 'C ' is appropriate. Alternatively, the user can specify 'XX', which indicates any atom type.

hybridization is an integer that limits the hybridization of the link atom that will replace the hydrogen. Allowed values are: 0 = any hybridization, 2 = sp^2 hybridization, or 3 = sp^3 hybridization.

rotatable bonds (optional) is an integer that is added to the number of rotatable bonds in the linking fragment to obtain the total number of rotatable bonds in the molecule, a descriptor that can be used to screen out hits. It is meant to represent the number of single bonds within the input fragment that would experience restricted rotation on complexation with the guest. If no value is provided, the variable is assigned a value of zero.

The type of information described in the above example is required for all input fragments used by LINKER. In addition, there are two instances where the user may want to add further information by appending additional lines to the bottom of the input file. This occurs when the user invokes the option to drive the geometry of the input fragment (keywords **drivea** and/or **driveb** present in the control file).

If using **drivea**, then the 'hosta' file must contain extra lines. If using **driveb**, then the 'hostb' file must contain extra lines:

Line 4+n+na: A 'D ' followed by an integer, ndrives, equal to the number of drive lines to be read in. The maximum value is 10.

Lines 5+n+na to 4+n+na+ndrives: Drive lines. This will be described in more detail below (see Section 3.5 for format and examples).

3.4 Input Fragment File for the OVERLAY mode.

In addition to the 'control' file, OVERLAY mode requires the user to prepare one input file, 'hosta' (see Section 3.2 of this manual and Section 5.2 of the HDViewer User's Manual) that defines the structure of an input fragment. The following is an example of the input fragment made from the uranium complex shown in Figure 7 above. The format is similar, but in several instances differs from the format used to define input fragments used by LINKER.

```
Uranium Tetrachatecholate - dodecahedral                                    1
49  1                                                                       2
O    1    -1.71600  -1.47600   0.65700    6     9    49                    3
O    2    -0.91400  -1.15300  -1.85100    6    14    49                    4
O    3    -1.71600   1.47600  -0.65700    6    15    49                    5
O    4    -0.91400   1.15300   1.85100    6    20    49                    6
O    5     0.91400   1.15300  -1.85100    6    21    49                    7
O    6     1.71600   1.47600   0.65700    6    26    49                    8
O    7     0.91400  -1.15300   1.85100    6    27    49                    9
O    8     1.71600  -1.47600  -0.65700    6    32    49                   10
C    9    -2.33500  -2.21500  -0.30000    2     1    10    14              11
C   10    -3.35700  -3.11600   0.00100    2     9    11    33              12
C   11    -3.95600  -3.84800  -1.02800    2    10    12    34              13
C   12    -3.53300  -3.67900  -2.35000    2    11    13    35              14
C   13    -2.50900  -2.77600  -2.65100    2    12    14    36              15
C   14    -1.91200  -2.04500  -1.62200    2     2     9    13              16
C   15    -2.33500   2.21500   0.30000    2     3    16    20              17
C   16    -3.35700   3.11600  -0.00100    2    15    17    37              18
C   17    -3.95600   3.84800   1.02800    2    16    18    38              19
C   18    -3.53300   3.67900   2.35000    2    17    19    39              20
C   19    -2.50900   2.77600   2.65100    2    18    20    40              21
C   20    -1.91200   2.04500   1.62200    2     4    15    19              22
C   21     1.91200   2.04500  -1.62200    2     5    22    26              23
C   22     2.50900   2.77600  -2.65100    2    21    23    41              24
C   23     3.53300   3.67900  -2.35000    2    22    24    42              25
C   24     3.95600   3.84800  -1.02800    2    23    25    43              26
C   25     3.35700   3.11600   0.00100    2    24    26    44              27
C   26     2.33500   2.21500  -0.30000    2     6    21    25              28
C   27     1.91200  -2.04500   1.62200    2     7    28    32              29
C   28     2.50900  -2.77600   2.65100    2    27    29    45              30
C   29     3.53300  -3.67900   2.35000    2    28    30    46              31
C   30     3.95600  -3.84800   1.02800    2    29    31    47              32
C   31     3.35700  -3.11600  -0.00100    2    30    32    48              33
C   32     2.33500  -2.21500   0.30000    2     8    27    31              34
H   33    -3.69500  -3.25400   1.04200    5    10                          35
H   34    -4.76400  -4.56200  -0.79700    5    11                          36
H   35    -4.00900  -4.25900  -3.15900    5    12                          37
H   36    -2.18000  -2.64600  -3.69600    5    13                          38
H   37    -3.69500   3.25400  -1.04200    5    16                          39
H   38    -4.76400   4.56200   0.79700    5    17                          40
H   39    -4.00900   4.25900   3.15900    5    18                          41
H   40    -2.18000   2.64600   3.69600    5    19                          42
H   41     2.18000   2.64600  -3.69600    5    22                          43
H   42     4.00900   4.25900  -3.15900    5    23                          44
H   43     4.76400   4.56200  -0.79700    5    24                          45
H   44     3.69500   3.25400   1.04200    5    25                          46
H   45     2.18000  -2.64600   3.69600    5    28                          47
H   46     4.00900  -4.25900   3.15900    5    29                          48
H   47     4.76400  -4.56200   0.79700    5    30                          49
H   48     3.69500  -3.25400  -1.04200    5    31                          50
U   49     0.00000   0.00000   0.00000  372  1  2  3  4  5  6  7  8        51
 6  0                                                                       52
40  C  0  0                                                                53
45  C  0  0                                                                54
40  C  0  0                                                                55
33  C  0  0                                                                56
40  C  0  0                                                                57
44  C  0  0                                                                58
```

Line 1: Title string that can be up to 60 characters long.

Line 2: Two integers. The first integer is the number of atoms in the structure, n. The maximum allowed value is 200. The second integer, ng, is the number of atoms in the guest. The maximum allowed value is 50.

Lines 3 to 2+n: The same information as described above for LINKER.

Line 3+n: Two integers. The first integer is the number of attachment points in the structure, na. Unlike LINKER, OVERLAY uses pairs of attachment points. Therefore, na must be an even number. A maximum of 20 attachment points, in other words, a maximum of 10 pairs, can be specified. The second integer (optional) is the number of symmetric attachment points, *nsym*. In this example, there are no symmetric attachment points (see next example for a case where there are attachment points related by symmetry). If specified, *nsym* can be any integer between 1 and 5. If not specified, the default value for *nsym* is 0.

Line 4+n to 3+n+na: The attachment points list. When *nsym* is 0, the information used to specify each attachment point is identical to that described for LINKER (see above). In this example, there are three pairs of attachment points: 40 and 45, 40 and 33, 40 and 44. During the building process, OVERLAY will attempt to superimpose linking fragments across each of these three pairs of attachment points.

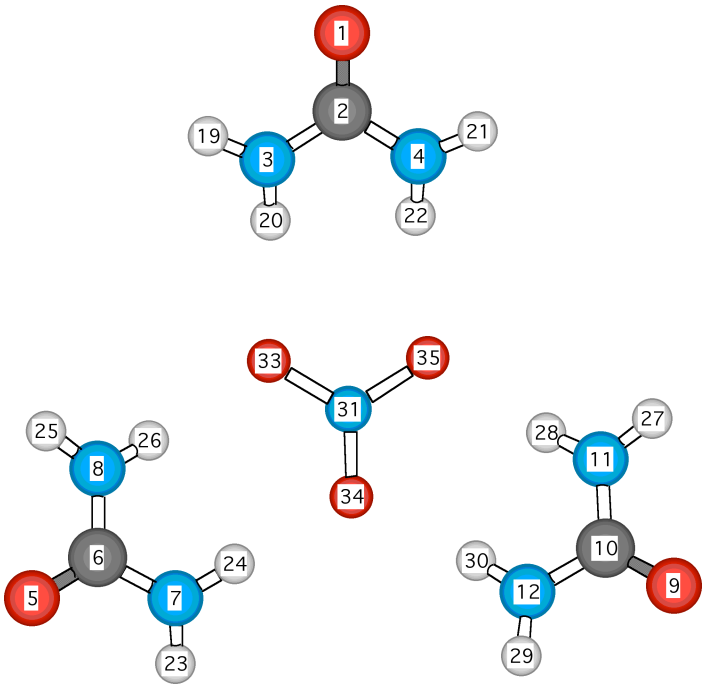
The information described in the above example is required for all input fragments used by OVERLAY. In addition, there is one instance where the user may need to add further information by appending additional lines to the bottom of the input file. This occurs when the user invokes the option to drive the geometry of the input fragment (keyword **drivea** present in the control file).

If the **drivea** keyword is in the control file, then:

Line 4+n+na: A 'D ' followed by an integer, ndrives, equal to the number of drive lines to be read in. The maximum value is 10.

Lines 5+n+na to 4+n+na+ndrives: Drive lines. This will be described in more detail below (see Section 3.5 for format and examples).

The optional *nsym* variable described above was added to allow OVERLAY to connect more than one link to a structure in cases where pairs of attachment points are related by symmetry. This feature is useful for generating host molecules such as tripods, tetrapods, and macrocycles. An example would be forming a macrocycle by linking three urea groups around a nitrate anion to give the orientation shown below.



This hosta structure has three-fold symmetry such that if a link gives a good fit to one set of attachment vectors, given by atoms 19 and 25, then it will also give a good fit to two other sets of attachment vectors, 23 and 29, and 27 and 21. The code is instructed to add the 2nd and 3rd links by indicating a number of symmetric positions with the *nsym* variable. In this example, *nsym* = 2. After each atom in the attachment list, the serial numbers of nsym equivalent atoms are assigned. This is illustrated in the input file for the tris-urea structure shown below (see next page).

When the first link is added, the code tests to see whether there are disallowed close contacts between the link and the hosta atoms (see Section 2.2.2). When the next link is added, this test is repeated to ensure that there are not any close contacts between the next link and prior added links.

```
tris-urea macrocycle for nitrate with symmetric attachment points
 28    4
  O    1   -0.170578    4.687164    0.065094    7    2
  C    2   -0.159409    3.522934   -0.306335    3    1    3    4
  N    3   -1.281967    2.804245   -0.512405    9    2   19   20
  N    4    0.974900    2.851089   -0.591278    9    2   21   22
  O    5   -4.800049   -3.735123    0.226868    7    6
  C    6   -3.817352   -3.132156   -0.178589    3    5    7    8
  N    7   -2.647018   -3.738022   -0.464767    9    6   23   24
  N    8   -3.815994   -1.805923   -0.423798    9    6   25   26
  O    9    4.804169   -3.535721   -0.108810    7   10
  C   10    3.771484   -2.974442   -0.443756    3    9   11   12
  N   11    3.697906   -1.649765   -0.686371    9   10   27   28
  N   12    2.610214   -3.628876   -0.648453    9   10   29   30
  H   19   -2.123672    3.223083   -0.145737   28    3
  H   20   -1.226990    1.795181   -0.394424   28    3
  H   21    1.821991    3.305161   -0.283676   28    4
  H   22    0.970139    1.840836   -0.471252   28    4
  H   23   -2.571014   -4.687164   -0.130295   28    7
  H   24   -1.794861   -3.189514   -0.374695   28    7
  H   25   -4.614792   -1.309006   -0.058746   28    8
  H   26   -2.932831   -1.308472   -0.334900   28    8
  H   27    4.498962   -1.119690   -0.377258   28   11
  H   28    2.803314   -1.189316   -0.535294   28   11
  H   29    2.597031   -4.579788   -0.310745   28   12
  H   30    1.744293   -3.115921   -0.498322   28   12
  N   31   -0.059326   -0.905624    0.070175   88   33   34   35
  O   33   -1.248886   -0.204285    0.108856   32   31
  O   34   -0.026900   -2.224014    0.066162   32   31
  O   35    1.110046   -0.155365    0.026459   32   31
  2   2
  19  C  0  0
  23
  27
  25  C  0  0
  29
  21
```

The specification of attachment points given here indicates the following. There is one primary pair of attachment points, na = 2. These attachment points and their characteristics are specified by serial numbers 19 and 25. There are two pairs of symmetric attachment points, nsym = 2. When a link is found that spans 19 --- 25, the same link will be placed across 23 --- 29 and 27 --- 21. Thus, a total of three links will be added to give a macrocyclic structure.

If instead *nsym* = 0, then the attachment specification for this structure would be:

```
  2   0
  19  C  0  0
  25  C  0  0
```

yielding output structures with a single link added across 19 --- 25.

3.5 Geometry Drives

In Version 1.0, the input fragments were defined as rigid geometries. This approach allowed the user to specify a single orientation of the host component with respect to the guest, one that was meant to represent the most complementary orientation. However, there are number of cases where it is either not desireable to use a single optimal geometry for the input fragment. Since Version 2.0, the geometry of each input fragment can be varied during the building process by driving specified degrees of freedom. This optional feature has been found to be very useful in cases where the input structures are known to be flexible allowing the code to generate more host structures of better quality.

To use the optional geometry drive feature, it is necessary to include the appropriate keyword in the control file and to include additional information at the bottom of the input fragment file. The keyword **drivea** instructs the code to read drive information from the first input fragment file. The keyword **driveb** instructs the code to read drive informaton from the second input fragment file. Either one or both of these keywords can be used in a run.

Drive information is appended to the bottom of an input fragment file as follows:

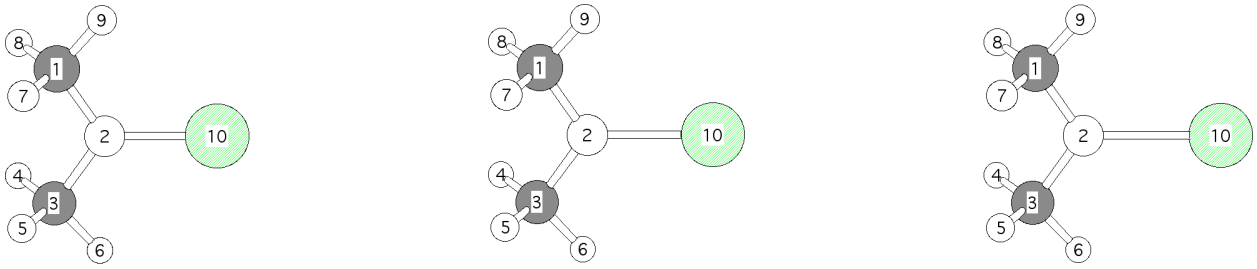
Line 4+n+na: A 'D ' followed by an integer, ndrives, specifying the number of drive lines to be read in. Maximum value is 10.

Lines 5+n+na to 4+n+na+ndrives: The drive lines. Each line contains the following information: a single character to specify the type of interaction to be driven (B = bond distance, A = angle, D = dihedral angle, + = linked), a single character specifying which atoms in the input fragment will move (G = guest, I = internal), a real number specifying the starting value for the driven interaction relative to the input geometry, a real number specifying the ending value for the driven interaction relative to the input geometry, the drive interval, and either two or three integers that specify the serial numbers defining the interaction. As will be illustrated by the examples given below, bond distances and dihedral angles are specified by two serial numbers, and angles are specified by three serial numbers. Each drive line will generate a number of geometries that are stored in an array during execution of the code. Space in this array is currently allocated such that the product of the number of steps generated by each drive may not exceed 2000 for a hosta input fragment or 200 for a hostb input fragment.

During the drive process, some atoms in the input fragment are fixed and others move. The second character in a drive line, either 'G' or 'I', specifies how the input fragment is divided and which atoms will move. When the 'G' designation is used, then the guest will move relative to the rest of the atoms in the input fragment. When the 'I' designation is used, then care must be taken in defining the drive line. The division of atoms depends on the connectivity within the input fragment and which atoms will move depends on the serial number order in the drive line. Input fragments can be defined as one fully connected group of atoms or as several atom groups that are not formally connected to one another. If the input fragment is composed of n non-connected groups, then the 'I' designation will divide the input fragment into n + 1 non-connected groups. In a bond distance drive, the new group is created by splitting the bond to be driven. In a angle drive, the new group is created by splitting the first bond specified. In a dihedral angle drive, the new group is created by splitting the bond about which the rotation

occurs. In each case, the group containing the second serial number defining the split bond will move relative to the other n groups.

Example 1. The user has defined a M-dimethylether input fragment and wants to drive the initial 2.0 Å M-O distance from 1.9 to 2.1 Å in 0.1 Å intervals.



Here is a drive line that yields the desired series of geometries. The 'B' indicates a bond distance will be varied. The 'G' indicates that the guest, in this case a single M atom, will move relative to the rest of the atoms. The next three numbers give the starting and ending point of the drive relative to the input geometry, and the interval. Note that the starting point must be less than the ending point and the interval must be a positive number. Finally, the bond in question is defined by atoms 2 and 10.

```
B  G  -0.1  0.1  0.1  2  10
```

When using the 'G' designation, where the guest always moves relative to the host, it does not matter in which order the serial numbers are specified. In this case, the following drive line would give identical results.

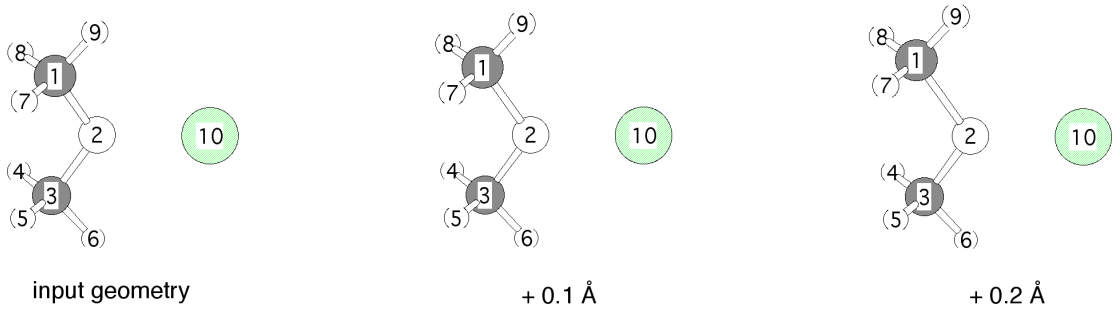
```
B  G  -0.1  0.1  0.1  10  2
```

Alternatively, this drive could be accomplished using the 'I', or internal coordinates, designation. To use the 'I' designation, splitting the input fragment at the specified bond must give two non-connected molecular fragments. The second fragment, everything that is attached to the second atom, will move relative to the first fragment, everything attached to the first atom. In this example, the geometries generated do not depend on which order the serial numbers are presented, in other words, both of the following drive lines also perform the desired drive.

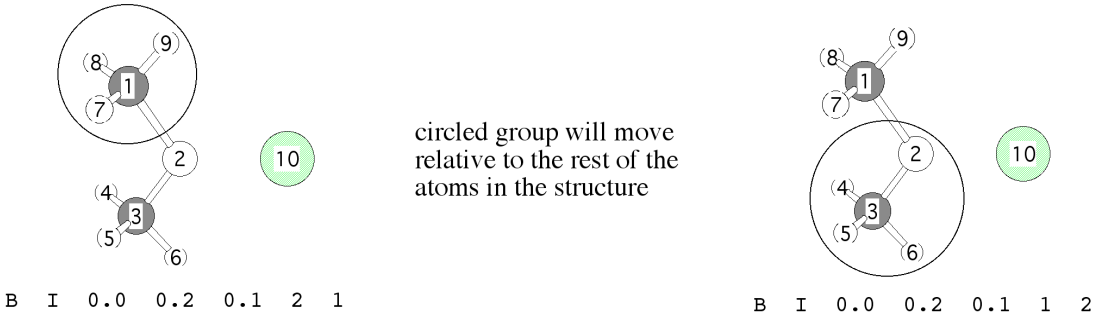
```
B  I  -0.1  0.1  0.1  2  10
```

```
B  I  -0.1  0.1  0.1  10  2
```

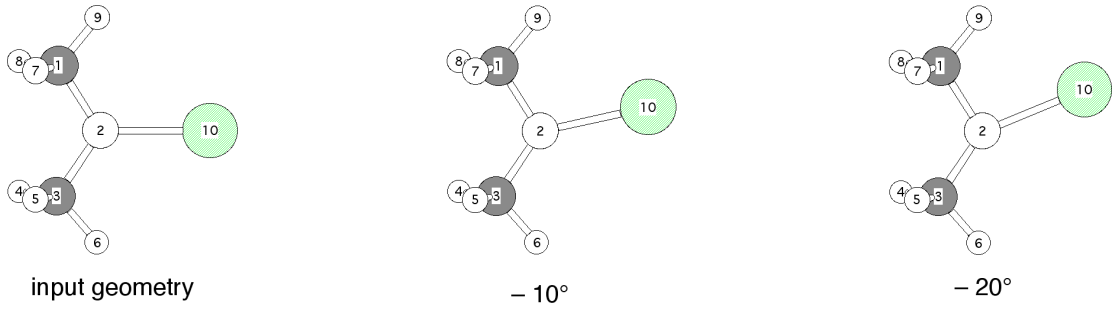
Example 2. The user has defined a Li-dimethylether input fragment in which the Li ion is not bonded to the oxygen. The user wants to drive one C-O bond distance from an initial input value of 1.4 Å to 1.6 Å in 0.1 Å intervals.



This drive, which cannot be achieved by moving the guest alone, must be done with the 'I' designation. As above, there are two possible orders to specify the bond, either '1 2' or '2 1'. In this example, however, only one of these drive specifications gives the desired result. As in the previous example, the 'I' method splits the group of atoms containing the C-O bond into two groups. Any part of the input structure, which is not connected to either of these groups, will be associated with the first group, and the second group will move. If the user wants the methyl group fragment to move relative to the rest of the structure, including the metal ion, then the '2 1' specification must be used.



Example 3. The user has defined a M-dimethylether input fragment and wants to drive the initial 120° M-O-C angle to 100° in 10° increments by moving the metal ion relative to the dimethylether group and within the C-O-C plane.



This drive can be achieved with any of the following three drive lines:

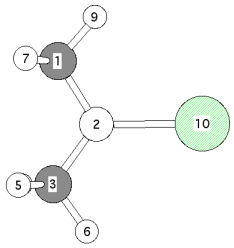
```
A  G  -20.0  0.0  10.0  1  2  10

A  G  -20.0  0.0  10.0  10  2  1

A  I  -20.0  0.0  10.0  10  2  1
```

In the first two drive lines, the guest moves relative to the rest of the molecule and, as in Example 1 above, the serial number order defining the angle does not matter. In the third drive line, the group of atoms will be split at the 10–2 bond, the metal ion will remain fixed, and the entire dimethyl ether group will move, yielding the desired result.

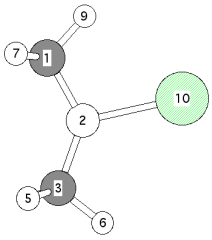
Another potential drive line using the 'I' designation is:

```
A  I  -20.0  0.0  10.0  1  2  10
```

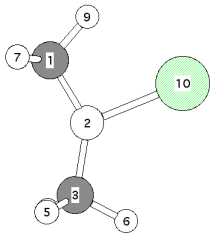
However, this specification fails to give the desired result. In this case, the group of atoms is split at the 1–2 bond, the methyl group containing atom 1 remains stationary, and the remaining atoms move, yielding the set of geometries shown below.



input geometry

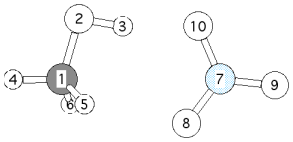


– 10°

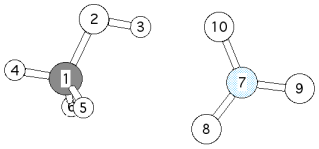


– 20°

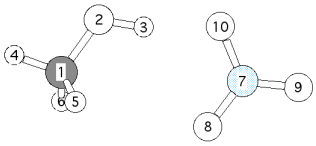
Example 4. The user has defined a methanol-nitrate input fragment in which the two groups are not formally connected. The user wants to drive the initial 107° C–O–H angle from 97 to 117° in 10° intervals.



– 10°



input geometry



+ 10°

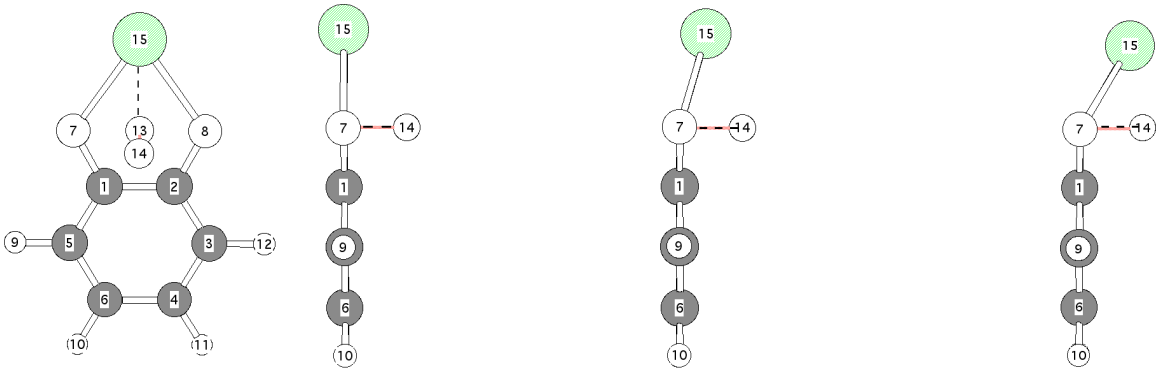
This drive is achieved using the 'I' designation:

```
A  I  -10.0  10.0  10.0  3  2  1
```

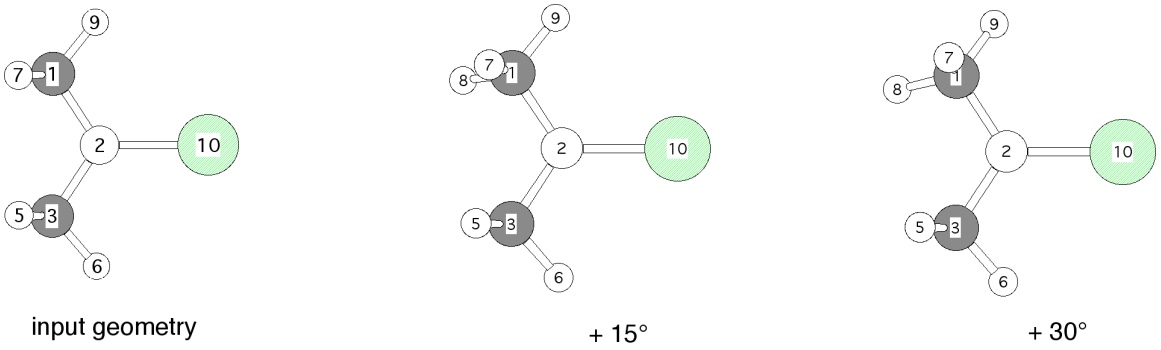
The alternate incorrect drive line `(A  I  -10.0  10.0  10.0  1  2  3)` would split the methanol at the 1–2 bond, fix the nitrate with the methyl group, and move the OH group.

Example 5. The user has defined a metal-catecholate input fragment and, keeping the metal ion on the mirror plane, wants to move the metal ion out of the plane of the catecholate group by 30° in 15° increments keeping the metal-oxygen distances constant. This can be accomplished with the use of dummy atoms. Two dummy atoms are placed on the mirror plane, the first one centered between the two oxygen atoms and the second one 1 Å above the first dummy atom. The drive can now be specified using the angle defined by the dummy atoms and the metal ion:

```
A  G  -30.0  0.0  15.0  14  13  15
```



Example 6. The user has defined a metal-dimethylether input fragment and wants to rotate one of the methyl groups by 30° from the starting geometry in 15° increments.



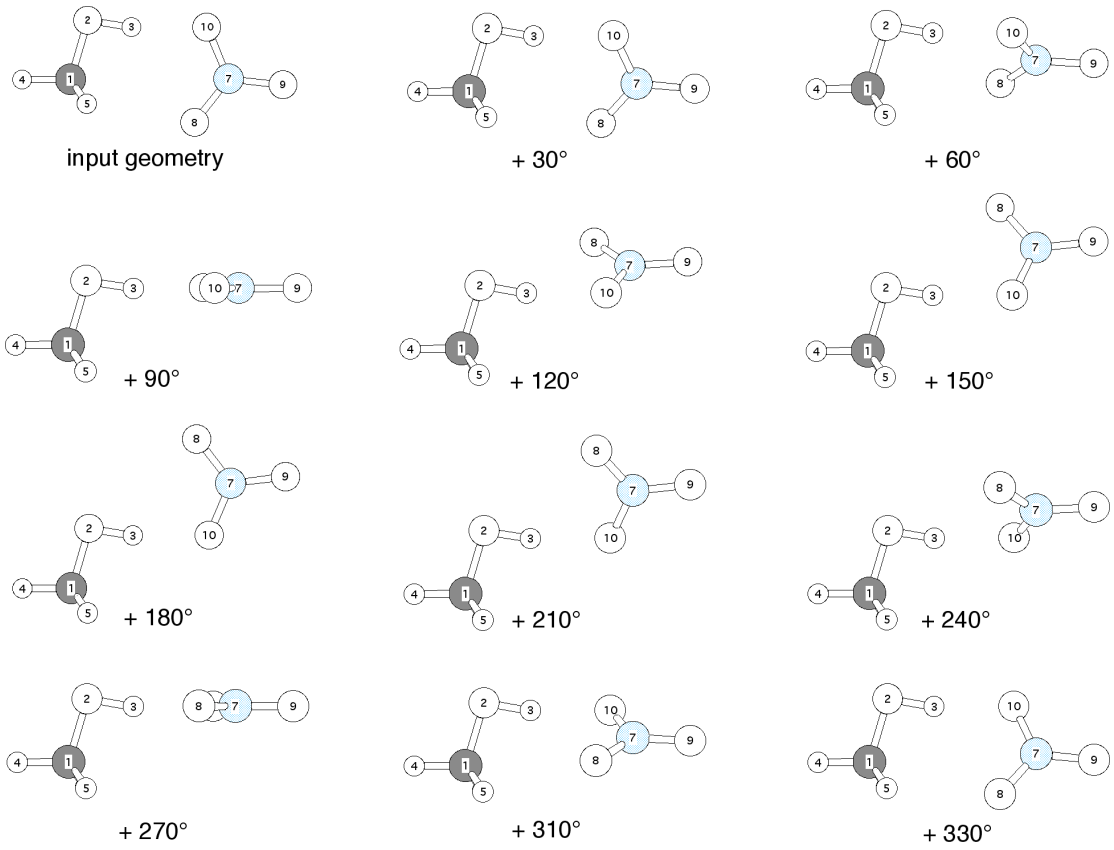
This drive is achieved using the 'I' designation:

```
D  I  0.0  30.0  15.0  2  1
```

Because the guest is formally connected to the oxygen atom, the alternative drive line also works in this case:

```
D  I  0.0  30.0  15.0  1  2
```

Example 7. The user has defined a methanol-nitrate input fragment in which the two groups are not formally connected. The user wants to drive the N–O•••H–O dihedral angle, initially at 0°, to 330° in 30° increments.



This drive is achieved with either of the following lines:

```
D   G   0.0   330.0   30.0   3   10

D   G   0.0   330.0   30.0   10   3
```

An Example of Input File with Drive Lines Present.

As stated above, the drive lines are appended to the bottom of the input fragment. There can be up to 10 different drives per structure, provided that the product of the number of structures generated from each drive does not exceed 2000 for host a and 200 for host b. If the **drivea** or **driveb** keyword is included in the control file, then there must be at least one drive line present in the 'hosta' or 'hostb' file, respectively. When geometry drives are used during a LINKER run, all geometries derived from 'hosta' are pair–wise linked with all geometries derived from 'hostb'. This operation can significantly increase the number of structures that will be evaluated during a run (in the extreme case there would be 2000 x 200 = 400,000 possible starting geometries). Therefore, the run times can increase very significantly when large drives are used.

The following is an example of the input fragment file for the dimethylether lithium complex given above (see Section 3.3), but this time with one dummy atom and three geometry drive lines added. The first drive line varies the Li–O–C angle within the C–O–C plane to three positions, the second drive line uses a dummy atom to move the Li out of the C–O–C plane to three positions, and the third drive line rotates one of the methyl groups to three positions. In this example, the product of the number of structures generated from the drives would be 3 x 3 x 3 = 27.

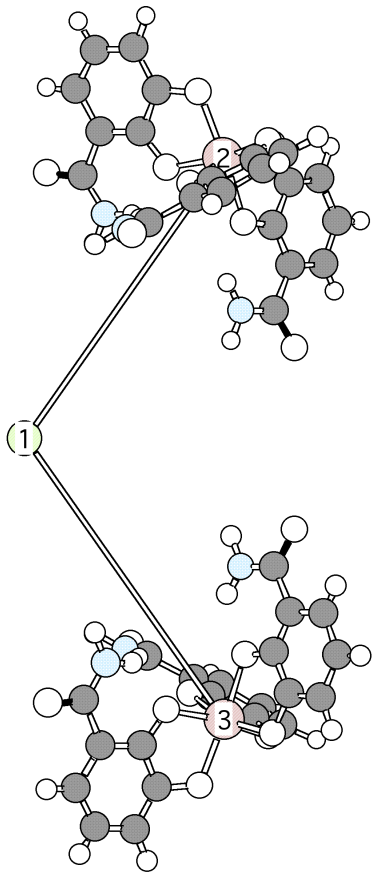
```
Lithium dimethylether, M-O = 2.26 Å, with drives                                1
 11    1                                                                        2
  C    1   -0.796860   -1.176666   -0.008575   1   2   4   5   6               3
  O    2    0.000000    0.000000    0.000000   6   1   3  10  11               4
  C    3   -0.795944    1.177567   -0.008606   1   2   7   8   9               5
  H    4   -0.092346   -2.040680   -0.009293   5   1                           6
  H    5   -1.374283   -1.179657    0.945770   5   1                           7
  H    6   -1.366318   -1.171310   -0.968475   5   1                           8
  H    7   -1.368774    1.184189    0.948334   5   3                           9
  H    8   -1.369934    1.169693   -0.965530   5   3                          10
  H    9   -0.090622    2.041031   -0.015411   5   3                          11
 Du   10   -0.001358    0.000351    0.972885   0   2                          12
 Li   11    2.259933    0.002136   -0.007889 301   2                          13
  3                                                                            14
  4   C  0   1                                                                 15
  5   C  0   1                                                                 16
  6   C  0   1                                                                 17
 D 3                                                                           18
  A   G  -10.0  10.0  10.0   1   2  11                                         19
  A   G    0.0  20.0  10.0  10   2  11                                         20
  D   I    0.0  30.0  15.0   1   2                                             21
```

When more than one drive is specified, multiple geometries are generated in a certain sequence. The structure is driven to the first geometry specified in drive1. This geometry is, in turn, used as input to generate the first geometry from drive2. This geometry is, in turn, used as input to generate all geometries from drive3. Then, drive2 is incremented, still using the first geometry from drive1. All geometries are again made from drive3. And so on. Once drive2 and drive3 have been completed, drive1 is incremented and the process starts over. Therefore, when the drives are not linearly independent, different ordering of the drive lines may yield different results. This is true in the above example as both the first and second drive will alter the Li–O–C angles.

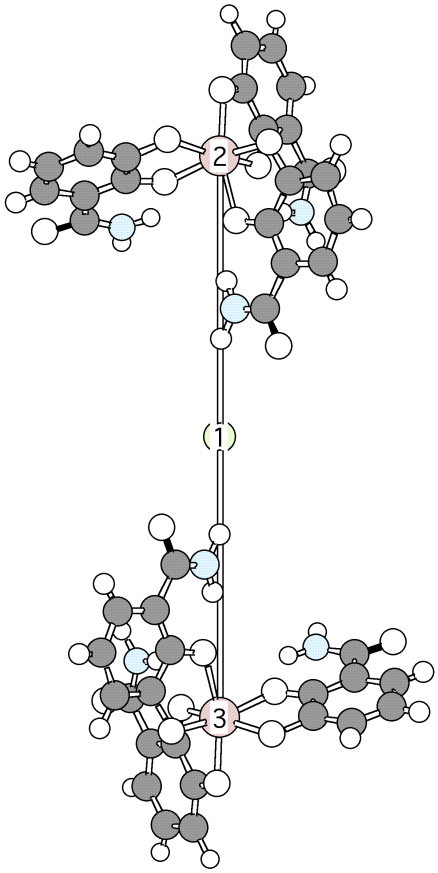
Because the results from a series of dependent drives may not be obvious, we have provided a facility for the user to view all the geometries that are produced. When the keyword **testdrive**, is included in the control file, HostDesigner will generate all the geometries and the results are written to files named ‘string_testa.hdo’ and/or ‘string_testb.hdo’, where ‘string’ is defined by the **out=string** keyword (see Section 3.2).

There may be occasions during an OVERLAY run where you desire two drives to be linked such that symmetry is maintained in the input fragment. An example of such a case is shown in Example 8. In this example, HostDesigner is being used to search for a link that will yield a ligand structure organized for the formation of a tetrahedral metal cluster. A single input fragment is constructed by orienting two $[Fe(catechol\text{-}amide)_3]^{3-}$ complexes such that their C_3 axes are aligned with the C_3 axes of a tetrahedron. The intent is to locate links that will connect the amide nitrogen from one complex to the amide nitrogen of the other one. To obtain the starting geometry shown, each complex was rotated about its respective C_3 axis such that the final structure has C_2 symmetry. The drives are set such that each complex is allowed to translate along and rotate about its C_3 axis. These drives must be linked in order to maintain the C_2 symmetry. This is accomplished by use of the '+' designator.

Example 8. The user has defined input fragment in which the two Fe(III) complexes are attached to a dummy atom, 1. The user wants to drive both Fe-dummy distances and rotate about both Fe-dummy bonds such that the symmetry of the input fragment is maintained.



looking at plane containing two C_3 axes of a tetrahedron



looking down the C_2 axis

The first translation drive is given by:

```
B  I  0.0  1.0  0.5  1   2
```

The second translation drive is linked to it using:

```
+  I  0.0  1.0  0.5  1   3
```

Similarly, linking the rotational drives would be accomplished with the following:

```
D  I  0.0  120.0   3.0   1  2
+  I  0.0  120.0   3.0   1  3
```

When a drive line beginning with '+' follows a preceding drive line, it means that the second drive will be incremented in step with the first one. In the example above, the two sets of linked geometric parameters being driven will have the identical values at all times. Currently linked drives are implemented for use with OVERLAY applications, in other words, they can be defined only for the 'hosta' fragment.

3.6 Description of the Output Files.

In either mode of operation, LINKER or OVERLAY, the code will write three output files with suffixes appended to a root name specified in the control file using the **out=string** keyword.

One of these files, 'string.summ' provides a summary that tells the user how many links were examined, how many structures were built, how many structures were retained, and timings for various aspects of the run. It also reproduces the control file and prints the title line from each of 'hosta' and 'hostb' files.

The other two files, 'string_1.hdo' and 'string_2.hdo', contain lists of Cartesian coordinates for the structures that were generated. The number of structures that will be written to these files is controlled using the **numview=#** keyword. During code operation, the best hits, in other words those with the lowest RMSD values, are stored in memory. At the end of the run, these hits are written to the 'string_1.hdo' file in ascending RMSD order. Next, the hits are sorted by conformational energy and written to the 'string_2.hdo' file in ascending conformational energy order. Because the number of structures stored in memory may be larger than the number to be written, each of these two output files may contain structures that are not found in the other one.

Example of the format used to output structures:

```
35                                                                                1
____6:RMSD=0.136,nrot=2,E=_2.740_1,4-dimethylcyclopentene____(_1345)              2
O      1.98381  -2.02329   0.13148   6   2   3   9                               3
C      3.15826  -2.82259   0.08920   1   1   6   7   8                           4
C      2.22468  -0.70451  -0.34001   1   1   4   5  14                           5
H      2.96202  -0.24349   0.35979   5   3                                       6
H      2.50580  -0.79508  -1.41560   5   3                                       7
H      3.43840  -2.92407  -0.98552   5   2                                       8
H      3.89350  -2.34908   0.78240   5   2                                       9
H      2.87769  -3.82542   0.48744   5   2                                      10
Li    -0.01549  -2.75385   0.89081 301   1                                      11
C      0.32271   0.08025   1.17579   2  11  14  15                              12
C     -1.00559  -0.11144   1.15647   2  10  12  17                              13
C     -1.53869  -0.29679  -0.25611   1  11  13  18  29                          14
C     -0.23692  -0.66577  -1.02003   1  12  14  16  19                          15
C      0.90242   0.04721  -0.23397   1  10  13  20   3                          16
C      1.17243   0.41142   2.38449   1  10  21  22  23                          17
C     -0.27863  -0.29612  -2.50876   1  13  24  25  26                          18
H     -1.66350  -0.07697   2.04065   5  11                                      19
H     -1.94617   0.67681  -0.61595   5  12                                      20
H     -0.08317  -1.76739  -0.93062   5  13                                      21
H      1.02800   1.09574  -0.59191   5  14                                      22
H      1.67370   1.39668   2.26012   5  15                                      23
H      1.96094  -0.35649   2.54427   5  15                                      24
H      0.56180   0.45976   3.31323   5  15                                      25
H     -0.42879   0.79659  -2.65666   5  16                                      26
H      0.66687  -0.57415  -3.02538   5  16                                      27
H     -1.10741  -0.81934  -3.03562   5  16                                      28
O     -2.01160  -2.58156   0.12612   6  28  29  35                              29
C     -2.92389  -3.67101   0.09727   1  27  32  33  34                          30
C     -2.60666  -1.38112  -0.34752   1  27  30  31  12                          31
H     -3.43620  -1.13461   0.35748   5  29                                      32
H     -2.86095  -1.55103  -1.42019   5  29                                      33
```

```
H      -3.17425  -3.85114  -0.97449    5  28                                  34
H      -3.75487  -3.41194   0.79560    5  28                                  35
H      -2.37736  -4.55707   0.49624    5  28                                  36
Li      0.11710  -2.73502   0.86950  301  27                                  37
```

Line 1: An integer, n, giving the number of atoms in the structure.

Line 2: A string reporting the serial number, the RMSD of the hit, the number of rotatable bonds, the estimate of the conformational energy, the source of the link, and finally the serial number of the link from the library. When the file comes from a LINKER run, the RMSD refers to the guest-guest distance. When the file comes from an OVERLAY run, the RMSD refers to the superposition of the bonding vectors.

Lines 3 to 2+n: Each line describes an atom in the structure with the atom label and the x, y, and z coordinates of that atom, the atom type, and a connectivity list.

The user has some control over the output file format. By default, each atom line contains the label, x, y, and z coordinates, atom type, and list of connected atoms as shown above. The behavior can be modified by adding keywords to the control file. If the **notype** keyword is present, then the atom types will be omitted. If the **xyz** keyword is present, each atom line will contain only the atom label and the Cartesian coordinates.

Output files with the default format or that generated by **xyz** can be viewed using the HDViewer utility provided with the download. Alternatively, the files can be viewed using molecule viewers such as XMOL or MOLDEN. The format for a single structure entry in these files is illustrated in the example shown on the next page.

It is recommended that if the user is applying geometry drive feature, the input should be checked to ensure that the drives are working as the user intended. When the **testdrive** keyword is used, the normal sequence of the program is not followed. Instead one or two files, named 'string_testa.hdo' and/or 'string_testb.hdo' are generated. These contain all the geometries generated for 'hosta' and/or or 'hostb', respectively. The series of structures in these files is presented with the same format used in the normal output files and can be viewed in the same fashion.

4.0 HOW TO CITE HOSTDESIGNER IN THE LITERATURE

In publishing results obtained either in part of in full from use of HostDesigner, the user should use the following citation:

Citation:

Hay, B. P.; Firman, T. K. "HostDesigner: A Program for the *de Novo* Structure-Based Design of Molecular Receptors with Binding Sites that Complement Metal Ion Guests" *Inorg. Chem.* **2002**, *41*, 5502-5512. (b) Hay, B. P., Jia, C.; Nadas, J. "Computer-Aided Design of Host Molecules for Recognition of Organic Guests" *Comp. Theor. Chem.* **2014**, *1028*, 72-80. (c) *HostDesigner, Version 3.0* can be downloaded from the website http://hostdesigner-v3-0.sourceforge.net

Example Applications:

Bryan, J. C.; Hay, B. P.; Sachleben, R. A.; Eagle, C. T.; Zhang, C. G.; Bonnesen, P. V. "Design, Synthesis, and Structure of Novel Cesium Receptors" *J. Chem. Cryst.* **2003**, *33*, 349-355.

Hay, B. P.; Oliferenko, A. A.; Uddin, J.; Zhang, C; Firman, T. K. "Search for Improved Host Architectures: Application of *De Novo* Structure-Based Design and High Throughput Screening Methods to Identify Optimal Building Blocks for Multidentate Ethers." *J. Am. Chem. Soc.* **2005**, 127, 17043-17053.

Bryantsev, V. S.; Hay, B. P. "*De Novo* Structure-Based Design of Bis-Urea Hosts for Tetrahedral Oxyanion Guests." *J. Am. Chem. Soc.* **2006**, *128*, 2035-2042.

Reyheller, C.; Hay, B. P.; Kubik, S. "Influence of Linker Structure on the Anion Binding Affinity of Biscyclopeptides" *New. J. Chem.* **2007**, *31*, 2095-2102.

Custelcean, R.; Bosano, J.; Bonnesen, P. V.; Kertesz, V.; Hay, B. P. "Computer-Aided Design of a Sulfate-Encapsulating Receptor" *Angew. Chem. Intl. Ed.* **2009**, *48*, 4025-4029.

Hay, B. P. "De Novo Structure-Based Design of Anion Receptors" *Chem. Soc. Rev.* **2010**, *39*, 3700-3708.

Szigethy, G.; Raymond, K. N. "Influence of Linker Geometry on Uranyl Complexation by Rigidly Linked Bis(3-hydroxy-N-methyl-pyridin-2-one" *Inorg. Chem.* **2010**, *49*, 6755-6765.

Duncan, N. C.; Hay, B. P.; Hagaman, E. W.; Custelcean, R. "Thermodynamic, Kinetic, and Structural Factors in the Synthesis of Imine-Linked Covalent Organic Frameworks" *Tetrahedron* **2012**, *68*, 53-64.

Young, N. J.; Hay, B. P. "Structural Design Principles for Self-Assembled Coordination Polygons and Polyhedra." *Chem. Commun.* **2013**, *49*, 1354-1379.

Vukovic, S.; Hay, B. P. "De Novo Structure-Based Design of bis-Amidoxime Uranophiles" *Inorg. Chem.* **2013**, *52*, 7805-7810.

Hay, B. P.; Jia, C.; Nadas, J. "Computer-Aided Design of Host Molecules for Recognition of Organic Guests" *Comp. Theor. Chem.* **2014**, *1028*, 72-80.

Jia, C.; Hay, B. P.; Custelcean, R. "De Novo Structure-Based Design of Ion-Pair Triple-Stranded Helicates" *Inorg. Chem.* **2014**, 53, 3893-3898.

Semenov, S. A.; Drobot, D. V.; Masatova, V. Yu.; Pomogailo, A. D.; Dzhardimalieva, G. I.; Kalinina, K. S. "Calculation of Energetic Characteristics for the Complexation of Unsaturated Dicarboxylic Acids with Cobalt(III)" *Russ. J. Inorg. Chem.* **2014**, *59*, 345-348.

5.0 TERMS OF USE

Copyright © 2015 Benjamin Hay
Supramolecular Design Institute
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

HostDesigner is provided by the authors "as is" and any express or implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the authors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use; data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.