April 2014

Critical Materials Institute

Progress Report

HostDesigner 3.0: Enhancements via Advanced Molecular Building Algorithms

Project 1.4.2, Task 2: Improving HostDesigner

Oak Ridge National Lab

Oak Ridge, TN

Billy Wayne McCann

mccannbw@ornl.gov

Table of Contents

Definitions

1. class– a family of rotors which posses similar molecular topologies, having qualitatively similar torsional potential energy surfaces.
2. connect atom – HostDesigner operates by creating a new bonds between molecular fragments. The atoms that form the new bond are referred to as connect atoms.
3. CONSTANTS – a plain text computer file that contains data regarding atomic mass, atomic van der Wal, and covalent radii, diheral angle minima and relative energies, as well as data used internally by HostDesigner.
4. FORTRAN – (FORmula TRANslation) a computer programming language designed to efficiently perform floating-point arithmetic and to have a syntax that resembles standard algebra.
5. gfortran – computer program that translates FORTRAN source into binary machine language; compiler
6. git – a source revision control (or version control) program. All changes to the source code are able to be visualized on a line-by-line, version-by-version basis and are able to be non-destructively reversed.
7. GUEST – a monoatomic or polyatomic anion or cation, possessing divergent binding forces, which complexes to the HOST(s).
8. HOST – the molecular fragment(s), possessing convergent binding forces, that complex the GUEST. Because the HOSTs can be non-identical, a distinction between HOSTA and HOSTB is sometimes made.
9. HostDesigner – fast, flexible, and powerful software package that identifies LINKs that give HOSTs optimally converging binding sites upon the GUEST.
10. LIBRARY – a plain text computer file that contains LINKs used by HostDesigner.
11. LINK – a molecular fragment, having no binding sites, used to connect HOST fragments.

12. rotor – one half of a rotatable bond. Classified according to molecular topology, electronic domain, and, where necessary, constituent identity. Two bonded rotors creates a rotatable bond, which must be set to optimal dihedral angles.
13. source (source code) – plain text, human readable computer instructions.
14. subclass – a distinction in a class of rotors based upon quantitatively distinct torsional potential energy surfaces; usually distinguished by degree of substitution about the connect atom.
15. subroutine – a unit of source code, typically placed into a separate plain-text computer file, which performs a single function.
16. tests – a collection of nearly one hundred HostDesigner inputs representing all rotor classes and subclasses and many variations upon them. Used for quality control.
17. PROJECT – CMI Project 1.2.4 Task 2: Improving HostDesigner

Goal

This PROJECT incorporates advanced building algorithms into the de novo structure-based molecular design software HostDesigner. Algorithms currently generate molecules by linking molecular fragments through the formation of single bonds. One limitation is that only certain types of single bonds can be formed. Another is that the builder does not build molecules by fusing bonds, missing an important class of molecular architectures. Throughout their development, the advanced building algorithms were tested on systems of interest to the CMI, such as phosphate and phosphine oxide donor systems.

Quality Control

Quality control has been part of every phase of HostDesigner source modification. Copious debugging, testing, and source versioning enabled early detection and correction of syntactic and logic errors. A version control system allows a developer to maintain previous versions of code indefinitely. Any change can be nondestructively reverted on a line-by-line basis to a working state should errors arise. Git* has emerged as the version control system of choice and has been used judiciously throughout the modification of HostDesigner.

The so-called "Test Driven Development" paradigm has been followed. For every change, debugging code was added and tests were designed to ensure the correctness of the code. This resulted in 86 test cases demonstrating the correct assignment of class and subclass. Rubber duck debugging ensured that the code is comprehensible line-by-line.

Also, HostDesigner, before this work, came with six test cases that cover nearly all of HostDesigner's capabilities. At the completion of this work, HostDesigner3 produced identical results to HostDesigner2.

HostDesigner: Background

HostDesigner was written by Dr. Benjamin Hay to facilitate "the discovery of host structures with binding sites that complement targeted metal ion guests."[1] HostDesigner

* http://git-scm.com

takes as input user-specified molecular fragments. These fragments, composed of the binding HOST and the bound GUEST, are then linked together with LINK fragments to produce a host-guest complex. Both HOSTs and LINK complexes possess specified connect atoms that HostDesigner uses to create new bonds between the HOSTs and the LINK. Creating new bonds requires two pieces of information: the bond length and the optimal dihedral angles. Bond lengths are unique for every connection type and are determined using the Schomaker-Stevenson equation (*vide infra*).

Optimal dihedral angles are pre-computed and stored in a lookup table, e.g. a plain text CONSTANTS file that is read into RAM memory arrays. They are defined in terms of their four constituent atoms and designated by a class and a subclass. In the first release of HostDesigner, only primary and secondary information was stored. In version 2, the dihedral classes were expanded to include ethers, thioethers, amines, and amides. Version 3, the latest version that includes the modifications indicated in this report, greatly expands the chemical space covered by HostDesigner. Details are given below.

Bond Lengths

Bond lengths are a key descriptor when forming a link between the HOSTs and the LINK. A 0.10 Å error in bond length was found to lead to radically different results when compared to original test cases provided with HostDesigner2. Moving from a constrained chemical space to a nearly universal chemical space required a general yet accurate method for calculating bond lengths.

With the removal of the reliance upon atom types, bond length became a function of covalent radii and bond order. The subroutine *getbondlength* now uses a method first proposed by Schomaker and Stevenson (SS) (equation 1)

$$d(A-B) = r_A + r_B - c|\chi_A - \chi_B| \qquad (1)$$

where $d(A-B)$ is the distance between atoms A and B, and r_A and r_B are the covalent radii of atoms A and B, respectively, c is the Schomaker-Stevenson coefficient, and χ_A and χ_B are the electronegativites of atoms A and B, respectively.[2] The covalent radii, Schomaker-Stevenson coefficents, and electronegativities are taken from Allinger's compilation.[3]

The radii had to be slightly adjusted to account for the difference between how covalent radii are defined and how HostDesigner is programmed. Covalent radii are defined as one-half the bond length between two bonded homonuclear elements of a given bond order. For example, the covalent double bond radius of carbon is one-half the carbon-carbon bond length in ethene (Figure 1(a), red bond). However, HostDesigner creates only single bonds (Figure 1(b), red bond). The use of the double bond covalent radii of carbon to construct a single bond between two sp^2 carbons would lead to gross errors in the geometry of the ligand, producing many false positives and false negatives.

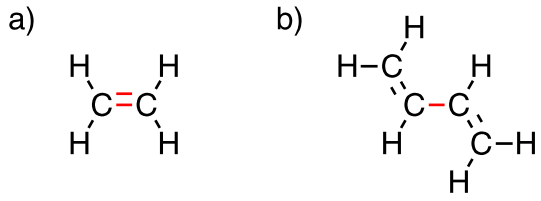


Figure 1. The double bond covalent radius is defined as ½ the red bond length in (a) but is used in HostDesigner as ½ the red bond length of (b).

The solution was a systematic lengthening* of the given covalent radii, such that the radius of the red bond in Figure 1(a) became the radius of the red bond in Figure 1(b). Type (b) bond lengths were acquired from the Cambridge Structure Database[4] (CSD) by taking an average of all of the bond lengths of a given type. When the bond length in the CSD had a large standard deviation, the bond length was calculated using the ωB97X-D[5] density functional with the 6-31+G(d,p)[6] basis set. In the calculations, all free valences were substituted with methyl groups. The correction to the covalent radius was calculated according to equation 2. Note that, for homoatomic bonds, equation 1 simplifies to $r(A-A) = 2r_A$.

$$Correction = (2r_A - Experimental\ Single\ Bond\ Length) \div 2 \qquad (2)$$

* Boron's covalent radius was decreased.

This correction simply represents the difference in equation 1 value and the experimental value. The division by two makes it applicable to the radius of a single atom. This correction is applied to the covalent radius to yield the radius stored in the CONSTANTS file. The modified radii and associated data are given in Table 1.

Table 1. To use literature values for covalent radii in constructing single bonds, the covalent radius was modified to fit experimental single bond lengths. All values are given in Å.

Element/ Bond Order	Original Radius	$2r_A$	Experimental Bond Length	Correction	Adjusted Bond Length	Modified Radius
C Single	0.760	1.520	1.523	0.000	1.520	0.760
C Double	0.669	1.338	1.480	0.071	1.480	0.740
C Triple	0.606	1.212	1.380	0.084	1.380	0.690
N Double	0.630	1.26	1.421	0.081	1.421	0.711
N Single	0.710	1.420	1.467†	0.024	1.467	0.734
O Single	0.720	1.440	1.465	0.013	1.465	0.733
P Single	1.100	2.200	2.242	0.021	2.242	1.121
P Double	1.000	2.000	2.033	0.017	2.033	1.017
P(V) Single	1.100	2.200	2.214	0.007	2.214	1.107
S Single	1.040	2.080	2.040	0.000	2.080	1.040
S Double	0.940	1.880	1.904	0.012	1.904	0.952
B Single	0.880	1.760	1.698	-0.031	1.698	0.849

† ωB97X-D/6-31+G(d,p)

The modified radii in Table 1 result in a 0.02 Å mean unsigned error for a test set of bonds most relevant to HostDesigner.

Dihedral Angles

Introduction

The term "dihedral" denotes the coming together of two planes. In the context of molecular geometry, these two planes are comprised of four atoms, ABCD, with the coordinates of ABC defining the first plane and the coordinates of BCD defining the second. The angle between these two planes, the dihedral angle Φ(ABCD), is the second key descriptor in the ligand assembly process.

Being the most time consuming area of configuration space to explore, the analysis of dihedral angles has received much attention by molecular modelers. Techniques such a Molecular Dynamics (MD), Monte Carlo (MC), and dihedral driving (DD) are the most widely used theoretical techniques to determine molecular structure and relative energy with respect to Φ.

The goal of HostDesigner is the rapid assembly of metal-coordinating ligands, some of which will have dozens of dihedral angles to explore. On-the-fly DD doesn't represent a rapid solution. MD and MC are likely to produce many conformers, only a fraction of which would be energy minima on the potential energy surface. In turn, perhaps only one or two of the MD or MC optimized structures would yield ligands with converging binding forces. All of these techniques would require geometry optimization and energy evaluation subroutines to be called approximately 10^2 to 10^5 times, resulting in unacceptably long ligand assembly time.

HostDesigner's Technique

The technique used in HostDesigner is to store pre-computed dihedral angle minima and relative energies in a look-up table, i.e a CONSTANTS file that is read into randomly accessible memory (RAM) arrays by the subroutine *readconstants*. In this way, only energy-minima conformers are constructed. This method, however, limits the chemical space to only those for which the torsional potential energy surface has been computed.

Previously, only the profiles of alkyl, alkenyl, phenyl, amine, amide, ether, thioether, and amideoxime functional groups linked to an sp^3 or an sp^2 carbon were pre-calculated (Figure2, below).

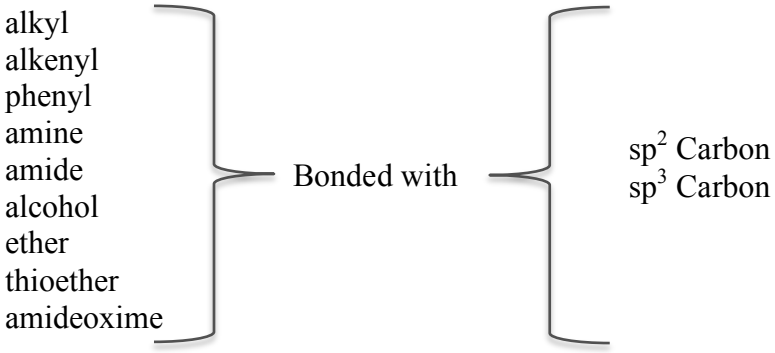


Figure 2. In previous versions of HostDesigner, only bonds from alkyls, alkenyls, phenyls, amines, amides, ethers, thioethers, and amideoximes to an sp^2 or sp^3 carbon could be created. This restriction was due to the use of a lookup table of precalculated rotational potential energy surfaces. The fragments with entries in the CONSTANTS file define the chemical space.

In order to connect the HOST to a LINK at the correct dihedral angles, a rotor is assigned to a class and a subclass. A class represents a family of rotors with similar electronic domains or molecular geometries. For example, all alkyl rotors belong to class 1. A

subclass generally denotes differing degrees of substitution within a class. A primary alkyl is classified as class 1, subclass 1 (abbreviated 1-1); a secondary would be class 1, subclass 2 (1-2) and so on. LINKs in the LIBRARY file come pre-assigned. The user-input HOST fragments, though, must be assigned a class and subclass by HostDesigner. This assignment is performed by the subroutine *assignclass*.

Previously, *assignclass* classified the HOST class and subclass based on the MM3 or MMFF molecular mechanical atom types of the connect atom and its substituents. Using atom types allowed *assignclass* to operate efficiently, but had the effect of both constraining the explorable chemical space and making the addition of new rotors tedious; every new rotor's atom types would have to be hard coded not only into *assignclass*, but throughout the entire codebase.

HostDesigner's Latest assignclass Algorithm

The goal of the improved algorithm is to cover as much chemical space as possible. To facilitate this, the dependence upon molecular mechanical atom types has been removed. *assignclass* now assigns rotors by identifying the linker's valency, electronic domain, molecular geometry, and, in some cases, the identity of the atoms involved. Prior to the most recent work, *assignclass* would identify a (1-1) rotor by seeing that the connect atom was a "type 1" atom (sp^3 carbon) and then counting the number of attached "type 5" atoms (hydrogens), yielding a count of two.

Now, *assignclass* identifies a (1-1) rotor in the following way. First, the connect atom is recognized as having a valency of four. Second, the algorithm tests the valency of every atom attached to the connect atom, i.e. the alpha atoms. If the alpha atom is monovalent (i.e. only bonded back to the connect atom), then it's element symbol is queried. If the element symbol is 'H', then the number of hydrogens is incremented by one. Once all of the alpha atoms have been inspected, *assignclass* then enters a decision tree. Having two hydrogens on a tetravalent connect atom would yield a (class-subclass) assignment of (1-1). Note that the (1-1) assignment is no longer restricted to a primary alkyl in this algorithm. Any connect atom that is tetravalent and is bonded to two hydrogens would be classified as (1-1), including the organic and metalloid elements of group 13, 14, 15, or 16 (minus oxygen). Should the primary decision tree not assign the connect atom, it is assigned a "general" class (15-1) that possesses six equienergetic dihedral minima equally spaced from 0 to 360 degrees.

New Rotors

While this algorithm increases the flexibility of *assignclass* in assigning rotors to the existing classes, these classes are not representative of a wide array of fragments. Hence, many new rotor types were added to the look-up table. These include phosphine oxide, sulfoxide, sulfone, phosphines, aldehyde, ketone, thioaldehyde, thioketone, imines, 1,2-diazines, pyridines, pyrimadine, and trigonal planar boranes. All of the rotors now included in HostDesigner are pictured in Figure 2.

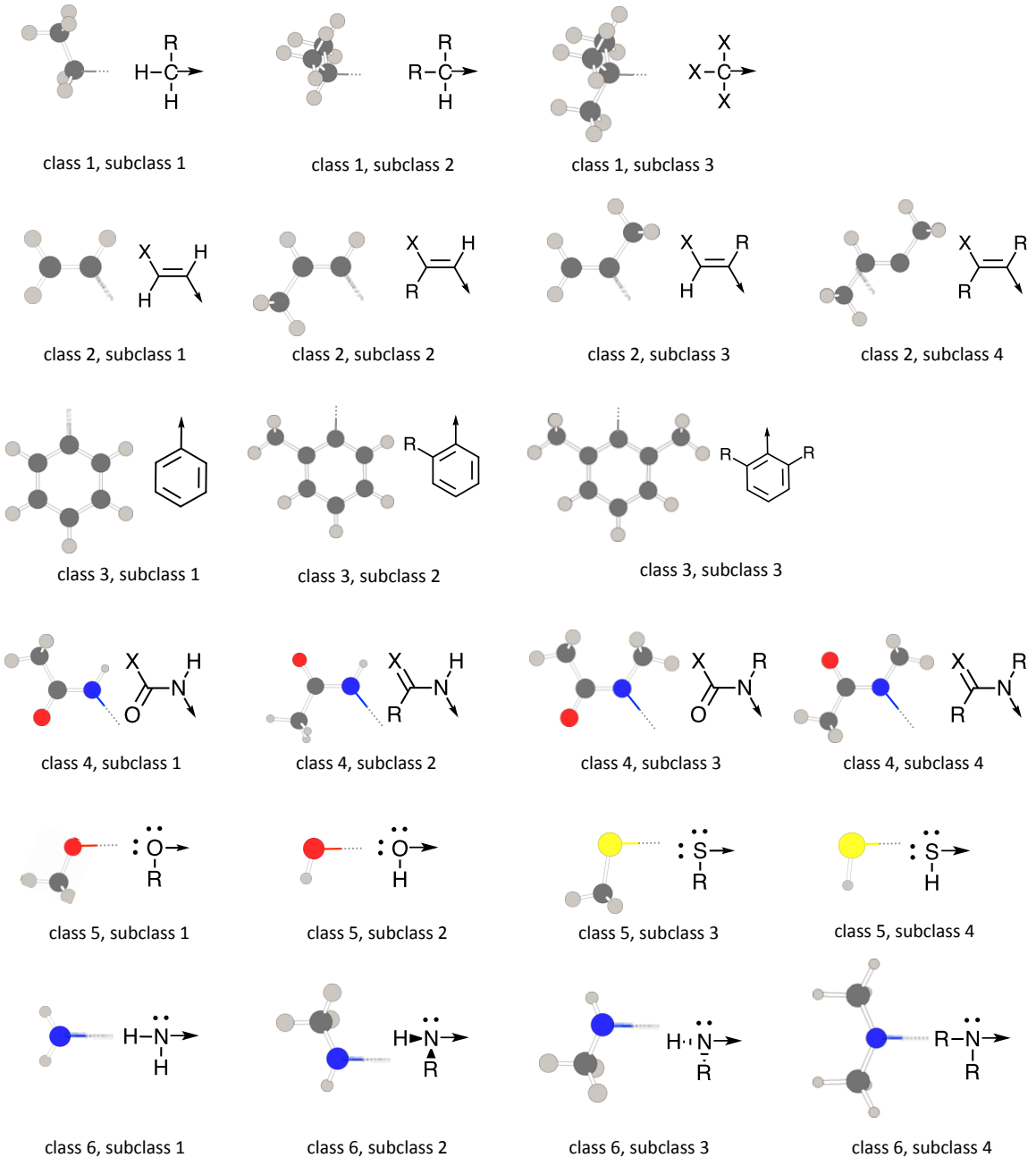
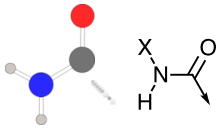
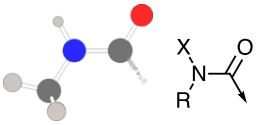


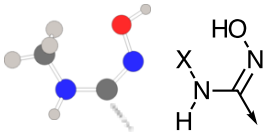
Figure 2. Ball-and-stick models and 2D schemae of the available rotors in HostDesigner. Arrows and dashed lines represent the connecting vector of the attach atom. Continued below.



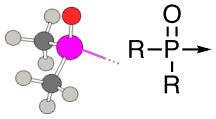
class 7, subclass 1



class 7, subclass 2

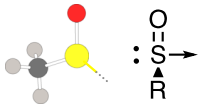


class 7, subclass 3



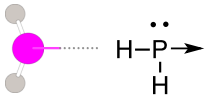
class 8, subclass 1

class 8, subclass 2

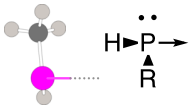


class 8, subclass 3

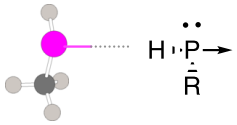
class 8, subclass 4



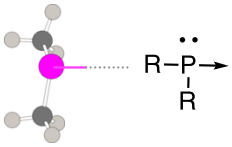
class 9, subclass 1



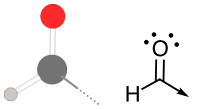
class 9, subclass 2



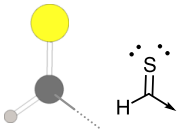
class 9, subclass 3



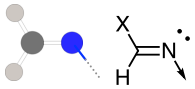
class 9, subclass 4



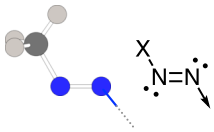
class 10, subclass 1



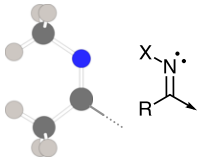
class 10, subclass 2



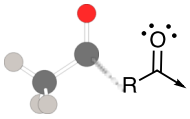
class 10, subclass 3



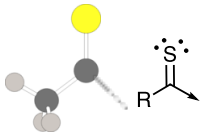
class 10, subclass 4



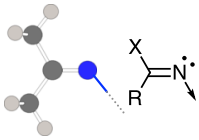
class 11, subclass 1



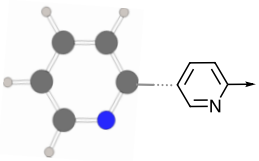
class 11, subclass 2



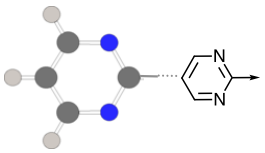
class 11, subclass 3



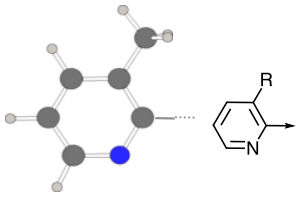
class 12, subclass 1



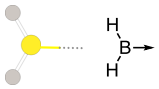
class 13, subclass 1



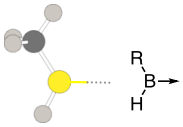
class 13, subclass 2



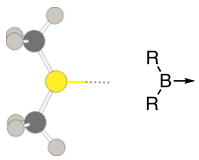
class 13, subclass 3



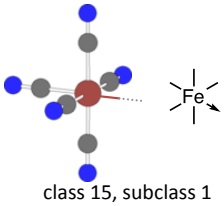
class 14, subclass 1



class 14, subclass 2



class 14, subclass 3



class 15, subclass 1

class 16, subclass 1

Figure 2, cont. Ball-and-stick models and 2D schemae of the available rotors in HostDesigner. Arrows and dashed lines represent the connecting vector of the attach atom.

Newly added rotor types include classes 8 through 16. These functional groups were chosen based upon the geometry of the connect atoms and the topology of their substituents. The key descriptors within the *assignclass* subroutine are the valency of the connect atom, the molecular shape of the connect atom, the nature of the α substituents, and, where applicable, the orientation of the β atoms or lone pair (LP) of the α substituent in relation to the bonding vector. The components that are *cis* to the bonding vector are categorized as LP, hydrogen, or non-hydrogen. The identity of these components distinguishes a subclass. Non-hydrogens were simulated using a methyl group. This is an approximation that will break down in some cases. *Trans* identities are assumed not to qualitatively alter the torsional potential energy surface. See Figure 3 for an example.

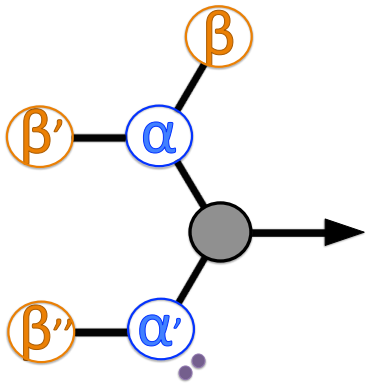


Figure 3. The electronic domain, geometry, alpha atoms, and the orientation of the beta atoms determine a rotor's class and subclass. In the example above, the algorithm first recognizes the connect atom (grey) is trigonal planar. The α's aren't monovalent, so the orientation of the β atoms is determined. β′ and β″ are *trans* to the bonding vector and

wouldn't contribute to the classification. Since β on α and α′'s LP are *cis* to the bonding vector, *assignclass* will return 13-1 or 13-3, if β is a hydrogen or non-hydrogen, respectively.

Cases are listed below. Note that for legacy reasons the number of subclasses in a class is restricted to four. Cases that appear to cover the same chemical space, e.g. 11-1 through 11-3, were deemed necessary after finding that the effect of atomic radius of the α atom was an easily addressable factor. The newly added rotor types include:

- tetrahedra possesing:
  - a non-hydrogen α possessing LP *cis* to the bonding vector (8-1)
  - two non-hydrogen α's possessing LP *cis* to the bonding vector (8-4)
- trigonal pyramids having:
  - lone pair on the connect atom and a non-hydrogen α possessing LP *cis* to the bonding vector (8-2, 8-3)
  - more acute bond angles than amines (class 9)
- trigonal planars with:
  - a hydrogen α and a non-hydrogen α possessing LP *cis* to the bonding vector (10-1, 10-2)
  - a LP on the connect atom and a β hydrogen *cis* to the bonding vector (10-3)
  - a LP on the connect atom and an α having a LP *cis* to the bonding vector (10-4)
  - a LP on the connect atom and a non-hydrogen α (11-1, 11-2,11-3)
  - a LP on the connect atom and a non-hydrogen β *cis* to the bonding vector (12-1)
  - an α having a LP *cis* to the bonding vector and α′ having a hydrogen *cis* to the bonding vector (13-1)
  - both α's having LP *cis* to the bonding vector (13-2)

    - α having a LP *cis* to the bonding vector and α′ having a non-hydrogen *cis* to the bonding vector. (13-3)
    - either hydrogen or tetravalent α's (class 14)
- metals; pentavalent or higher valencies; linear cases; other rotors not handled by the existing classes and subclasses (15-1)
- single atoms (16-1)

With the addition of these cases, the only illegal inputs are those that do not represent a host-guest complex, i.e. no host and/or no guest. Previously, an illegal input was any for which the force field atom types had no bond length or dihedral angle definitions in the CONSTANTS file. The PROJECT can now use HostDesigner to test a variety of ligands without the need of adding new rotor classes.

Analysis of New Rotors

The primary goal in the analysis of the new rotors is the location of the minima on the rotational potential energy surface and the relative energies of these minima. These have been determined using theoretical calculations. Experimental data from the Cambridge Structure Database (CSD)[4] has been used to verify the calculated minima. An estimation of dihedral relative energies from crystal statistical population analysis is an area for further research.

Organic functional groups representative of a great majority molecular fragments have been determined based on details discussed above. Rotor profiles were characterized bonded to rotors in classes one through three. Molecular mechanics (MM) and density

function theory (DFT) were used to identify minima and calculate their relative energies. MM force fields used include MM3[7], MMFF[8] and MMX[9]. MM3 was preferred. If there were no MM3 parameters for a given dihedral, MMFF was used. If there were no MMFF parameters, MMX was used. In some cases no MM parameters existed. These dihedrals were calculated using DFT, specifically the ωB97X-D[5] functional with a 6-31+G(d,p)[6] basis set. This functional and basis set combination was tested and gave quantitavely similar profiles as MP2/cc-pVTZ. With MM, the dihedral surface was scanned in 5 degree increments. Fifteen degree increments were used with DFT.

The CSD was queried for every new dihedral created. A query comprised a 2D schema of the dihedral of interest with an additional piece of 3D geometric information, namely the dihedral angle of specified atoms, being requested. Once the schema is given and dihedral is defined, search filters are specified and the CSD returns the crystal structures that possess the dihedral as well as the value of the angle. The number of hits, a histogram of the dihedral angles, and the average bond length between the two connect atoms was recorded. See Data.

Search filters apply constraints upon the results return by the CSD. Those imposed for this study included "3D Coordinates Defined", "R Factor <= 0.05", "Not Disordered", "No Errors", "Not Polymeric", "No Ions", "No powerder structures", "Only Organics." These options represent very strict filters that ensure the quality of the returned crystal structures. The "No Ions" filter was removed in appropriate cases. Also, other restrictions, such as the acylicity of the bond between the two connect atoms and their

valency, were applied. The valency restriction is necessary to prevent the CSD returning, for example, ammonium when an amine is queried. Also, if the linking bond were part of a ring, the dihedral angle would not represent ligands that HostDesigner creates.

All data created is stored in easily accessible formats. Microsoft Office formatted files may be accessed via the free Google Docs[*] or the free, open-source[†] LibreOffice package[‡]. See the Data section for details.

References

(1) Hay, B.P.; Firman, T. *Inogr. Chem.* **2002**, *41*, 5502-5512.

(2) Schomaker, V.; Stevenson, D.P. *J. Am. Chem. Soc.* **1941**, *63*, 37.

(3) Allinger, N.L; Zhou, X.; Bergsma, J. *J. Mol. Struct. (THEOCHEM)*, **1994**, *312*, 69-83.

(4) Allen, F. H. *Acta Crystallogr., Sect. B* **2002**, *58*, 380-388.

(5) Chai, J.; Head-Gordon, M. *Phys. Chem. Chem. Phys.* **2008**, *10*, 6615-6620.

(6) Hehre, W.J.; Radom, L.; Schleyer, P. v. R.; Pople, J.A. *Ab Initio Molecular Orbital Theory*; Wiley: New York, 1986.

(7) Allinger, N.; Yuh, Y.; Lii, J. *J. Am. Chem. Soc.* 1989, *111*, 8551-8566.

(8) Halgren, T. A. *J. Comp. Chem.*, **1996**, *17*, 490-519.

(9) Gilbert, K. E. PCMODEL, Serena Software.

[*] https://docs.google.com

[†] http://opensource.org/licenses/lgpl-3.0.html

[‡] https://www.libreoffice.org

Data

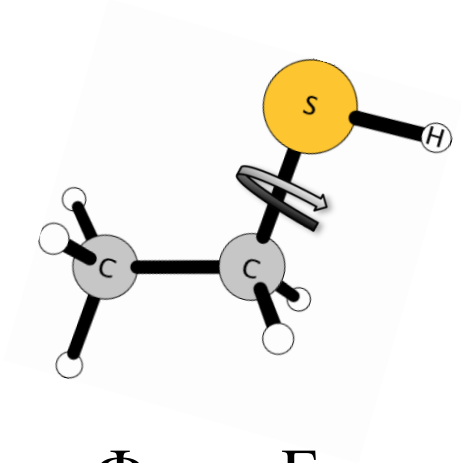
All data has been stored locally as plain text files, tiff images, and PDF files. These are meant to serve archival purposes. PDF files were created using Mac OS X's built-in `postscript` functionality.

Microsoft Office pptx, docx, and xlsx files serve as intermediate formats in the creation of PDFs.

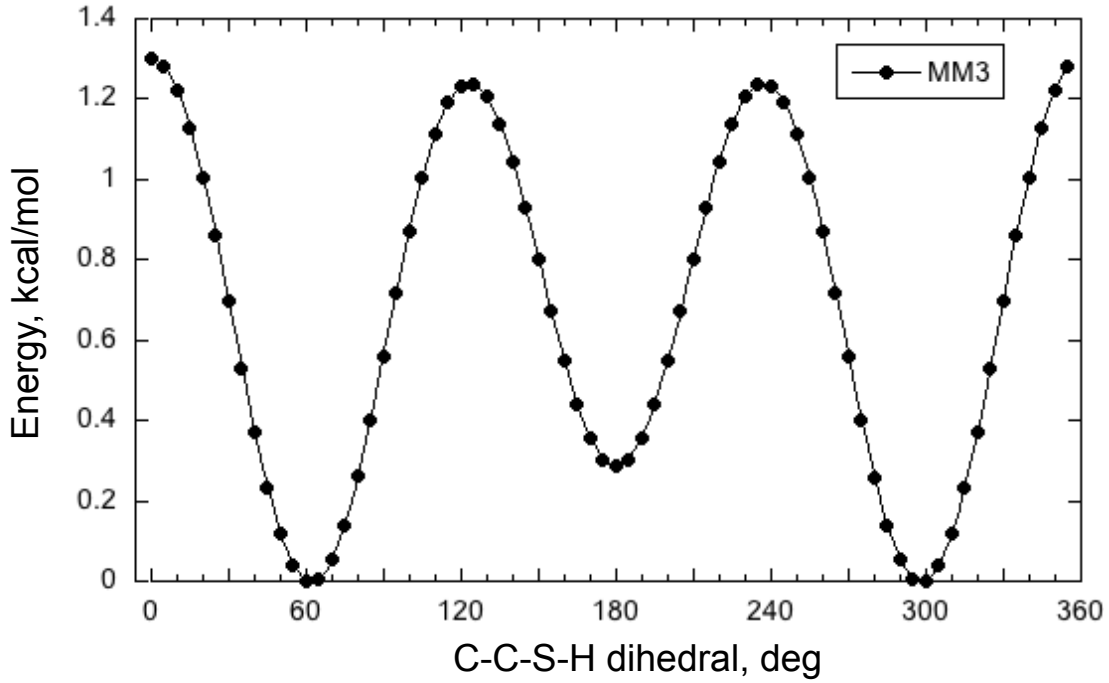
All data is available upon request.

Rotor Profiles

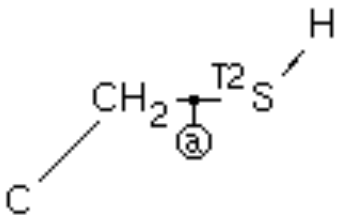
TYPE 1-1:5-4



Φ	E
60.0	0.00
180.0	0.29
300.0	0.00

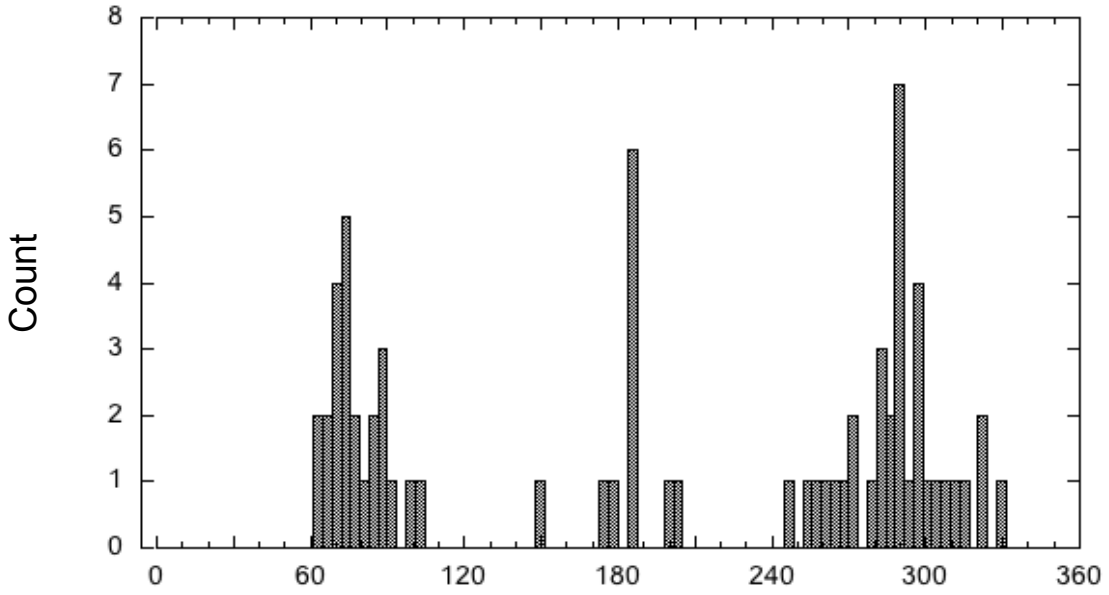


CSD comparison

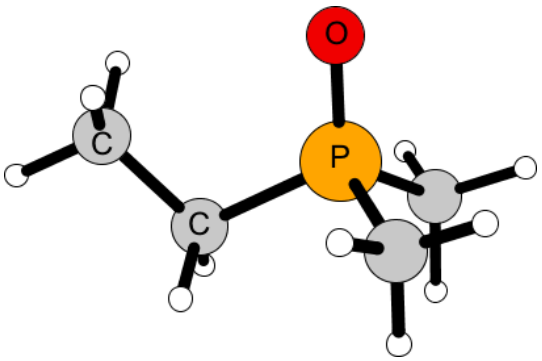


59 hits

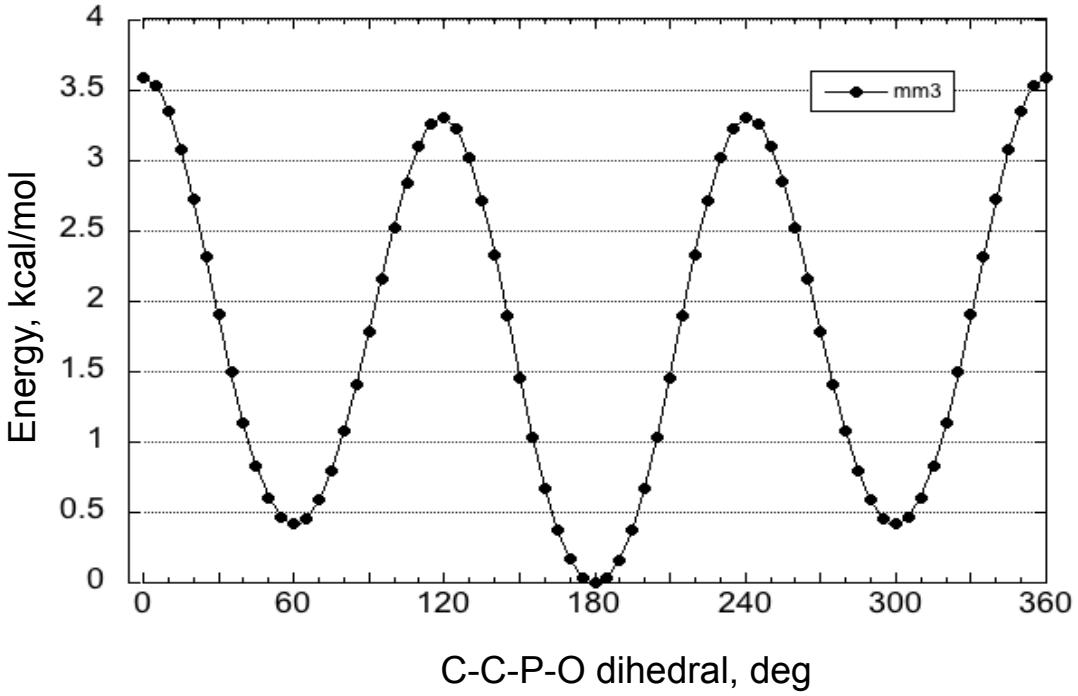
C–S = 1.81 ± 0.01 Å



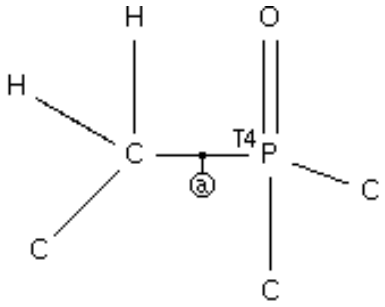
TYPE 1-1:8-1



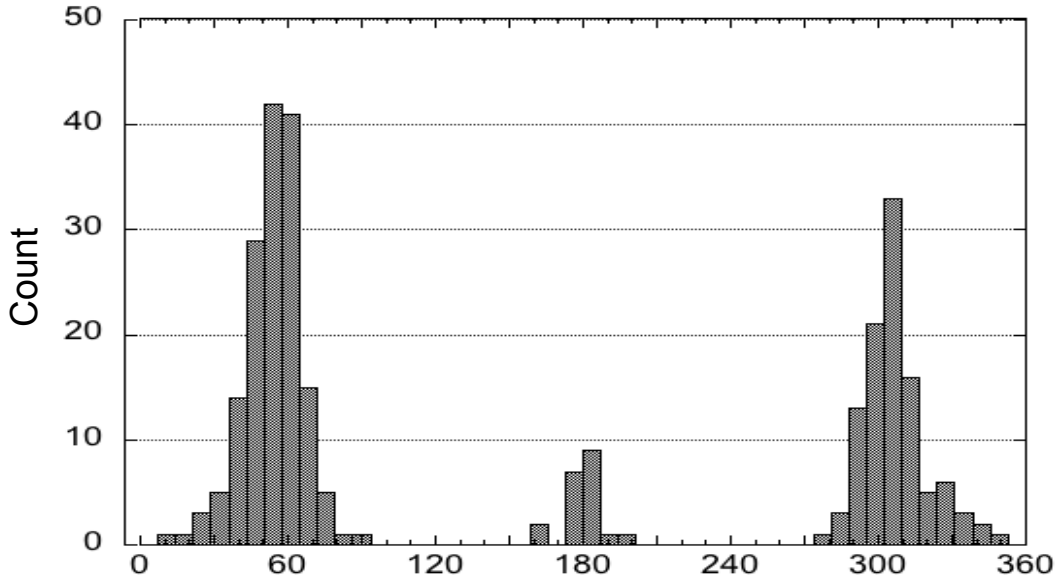
Φ	E
60.0	0.42
180.0	0.00
300.0	0.42



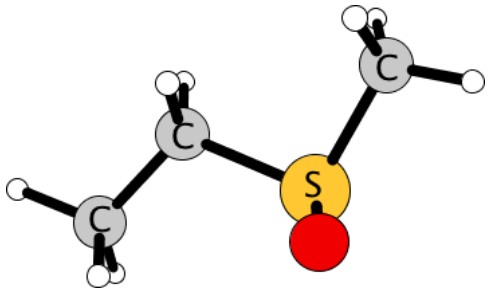
CSD comparison



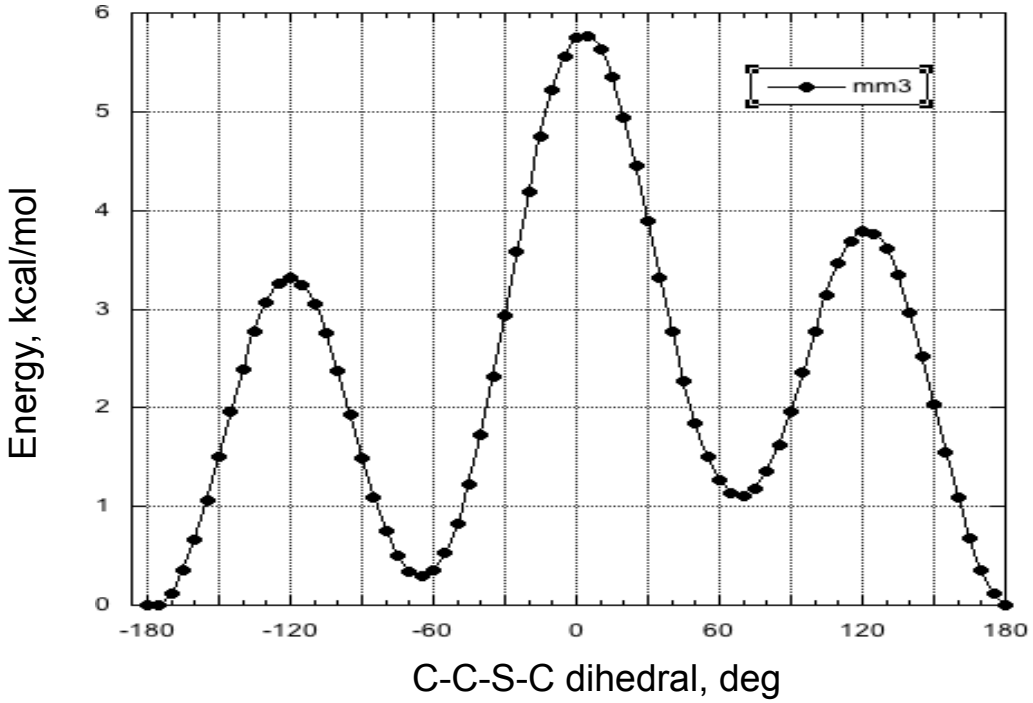
C–P = 1.81 ± 0.01 Å



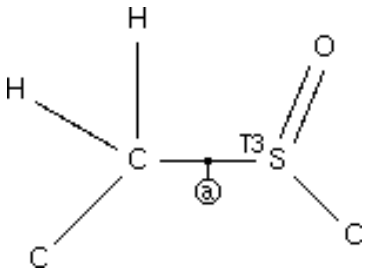
TYPE 11-T2b



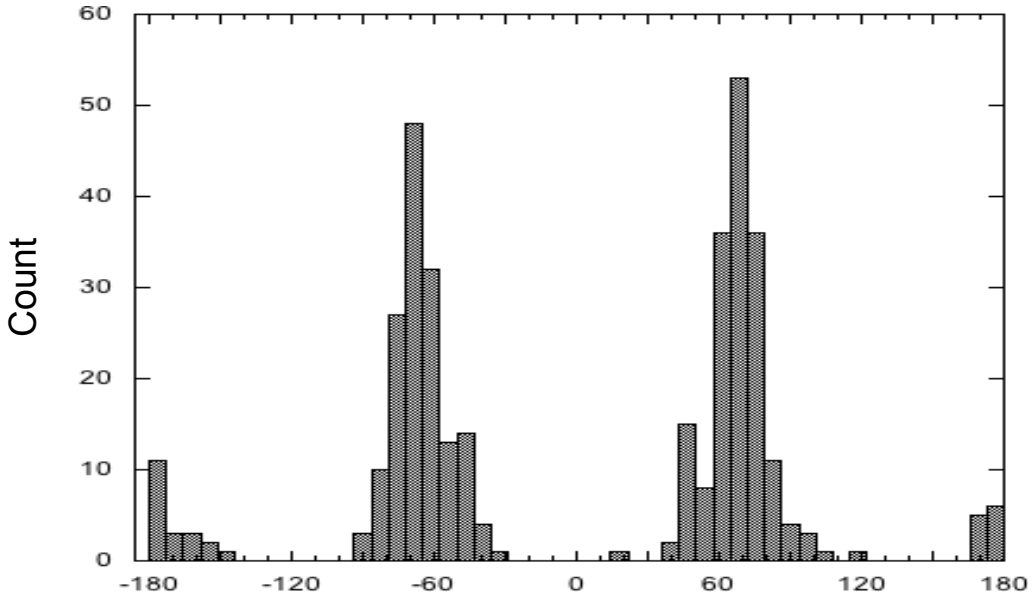
Φ	E
70.0	1.11
180.0	0.00
305.0	0.30



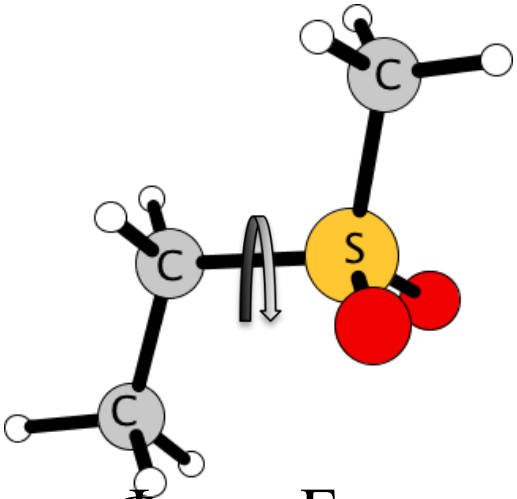
CSD comparison



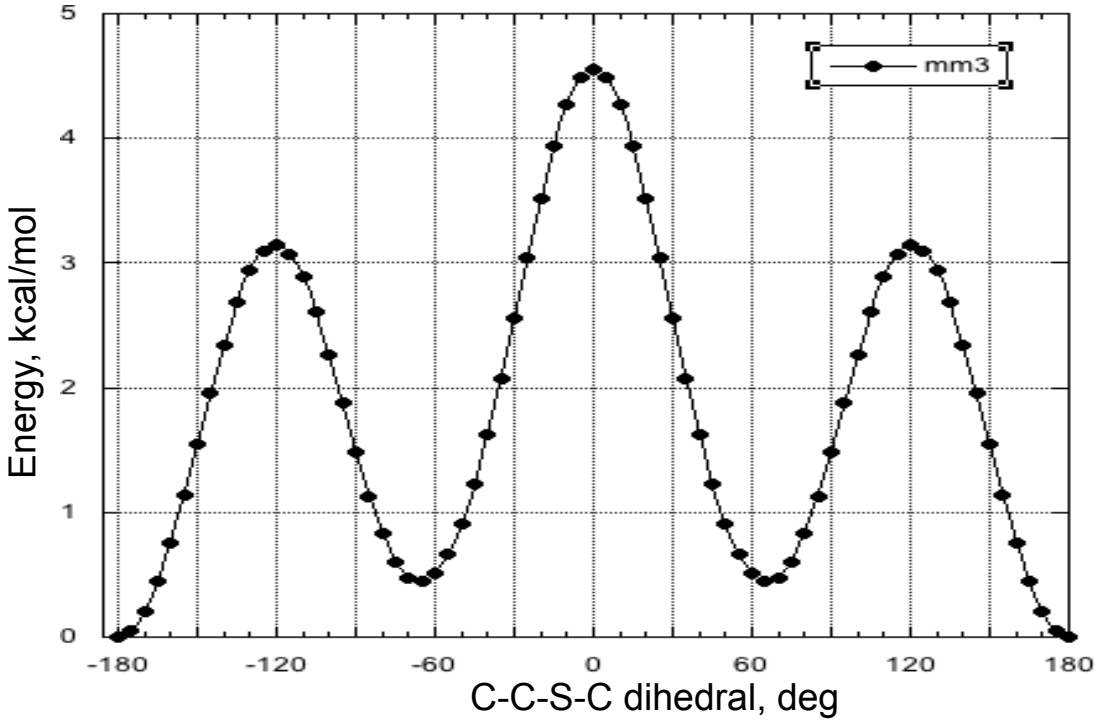
C–N = 1.81 ± 0.02 Å



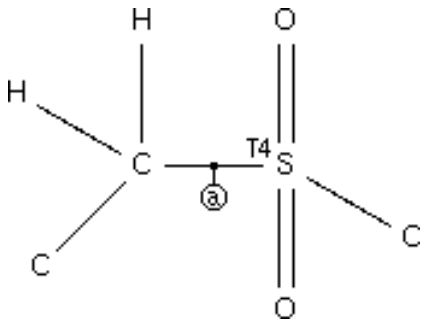
TYPE 1-1:8-4



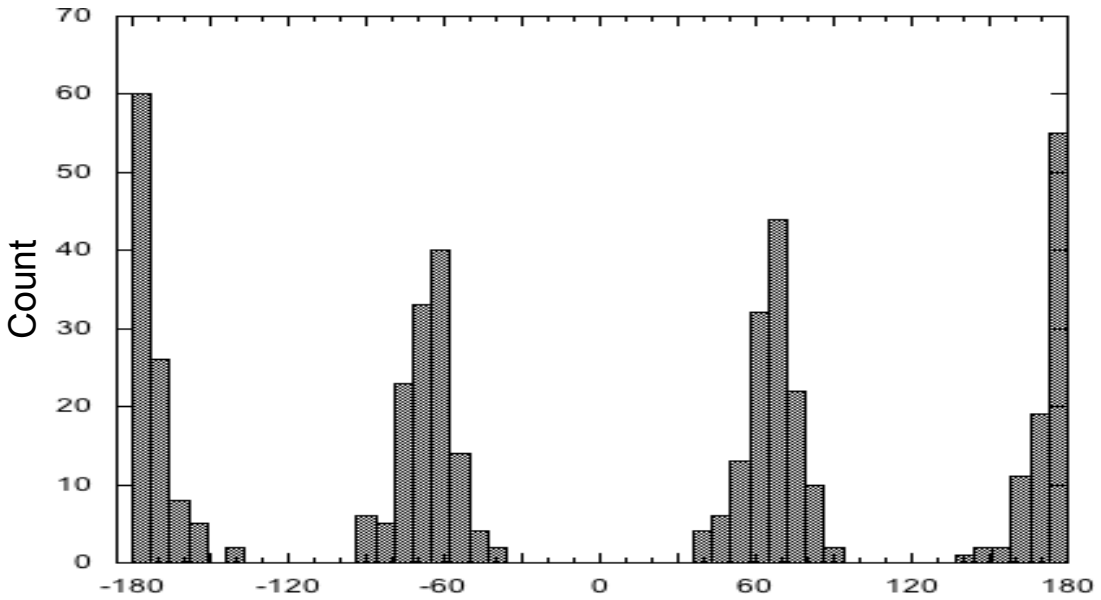
Φ	E
65.0	0.45
180.0	0.00
295.0	0.45



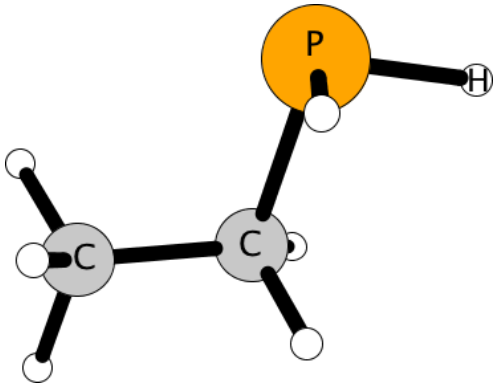
CSD comparison



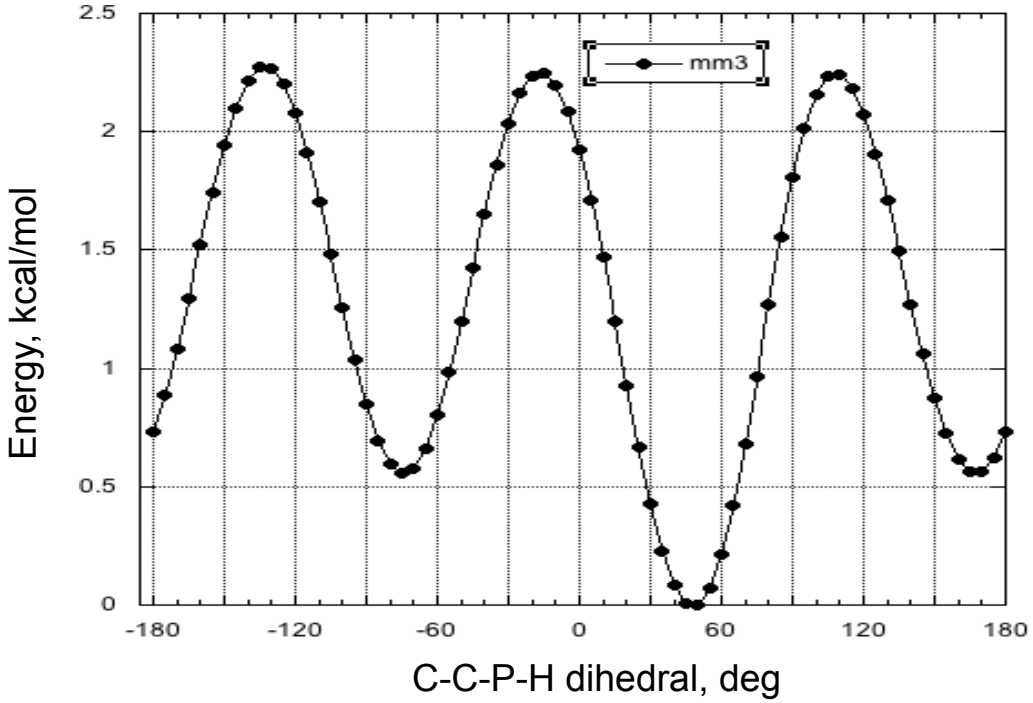
C–S = 1.78 ± 0.01 Å



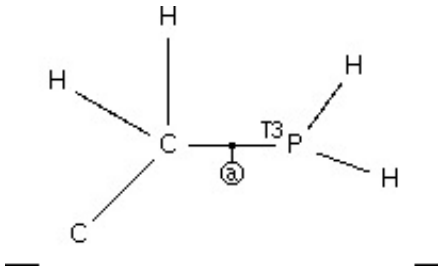
TYPE 11-T0d



Φ	E
50.0	0.00
165.0	0.56
285.0	0.56

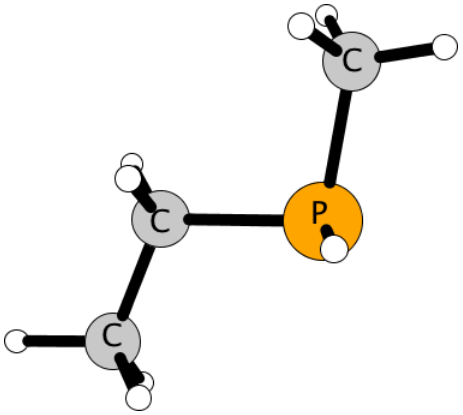


CSD comparison

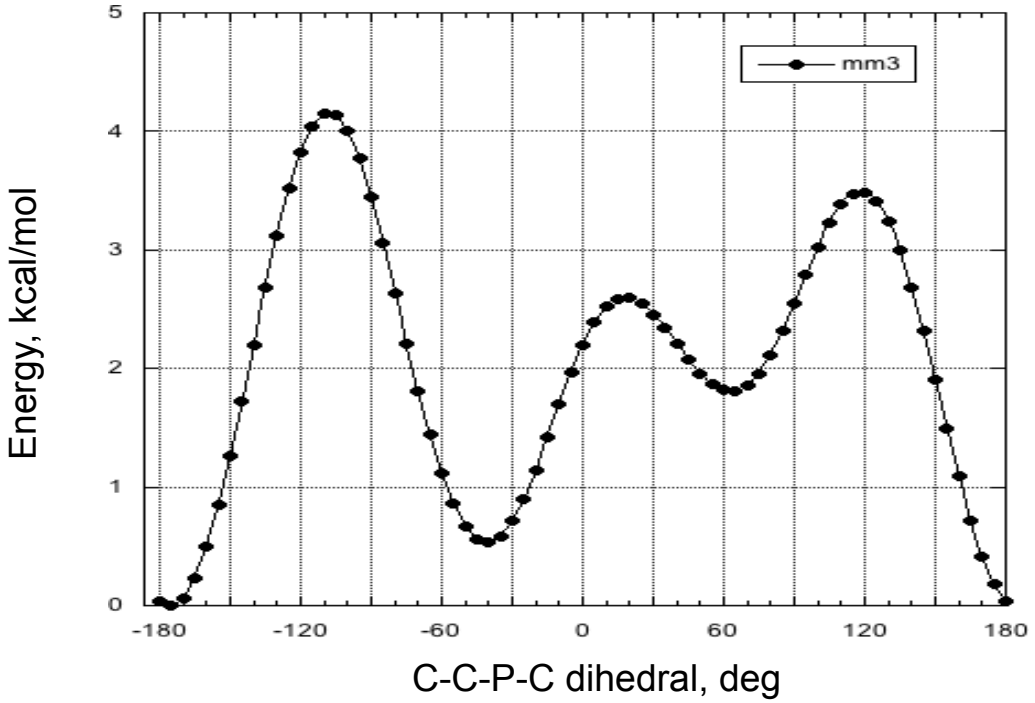


NO CSD HITS

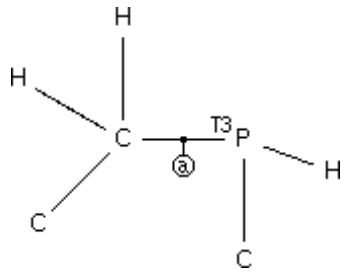
TYPE 1-1:9-2



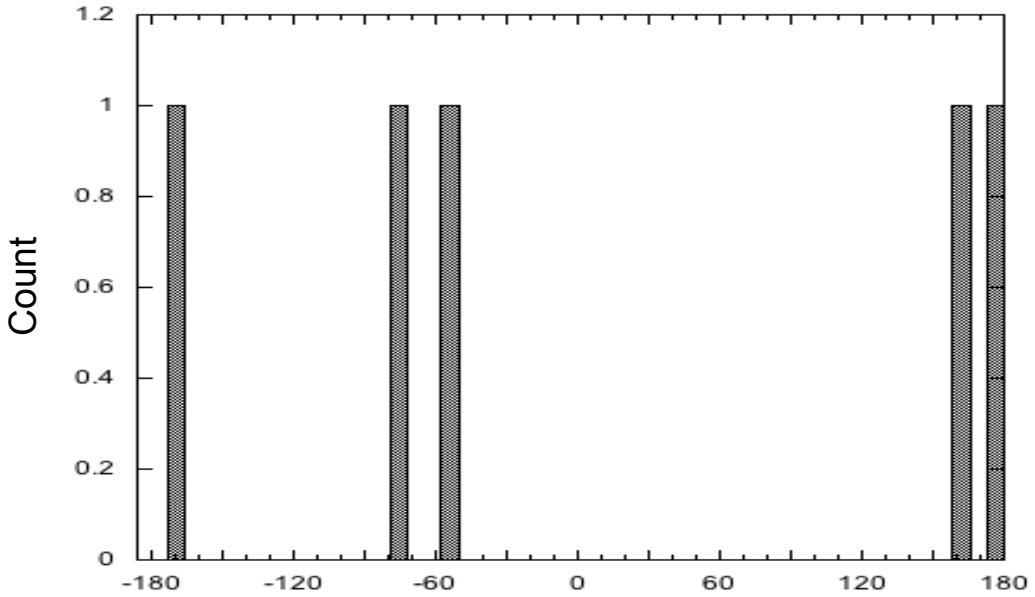
Φ	E
65.0	1.81
185.0	0.00
320.0	0.53



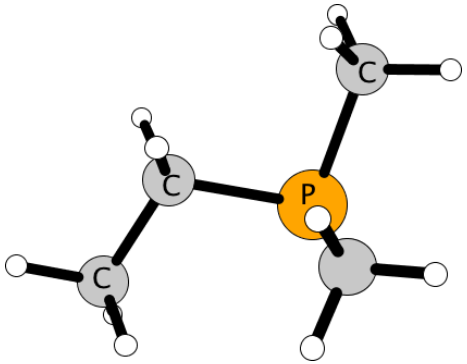
CSD comparison



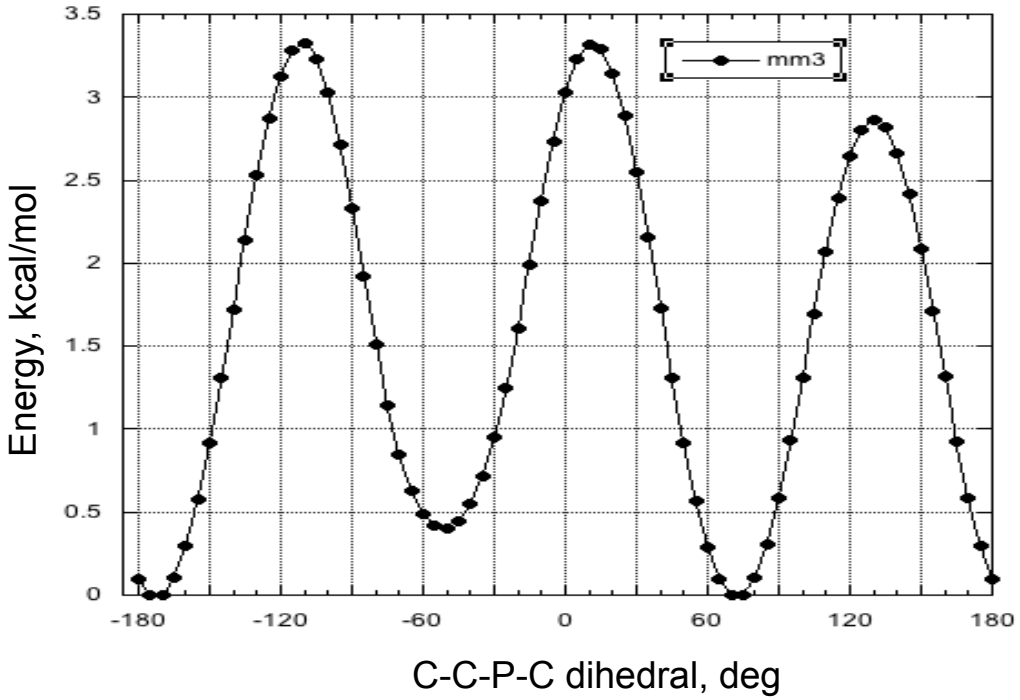
C–P = 1.86 ± 0.01 Å



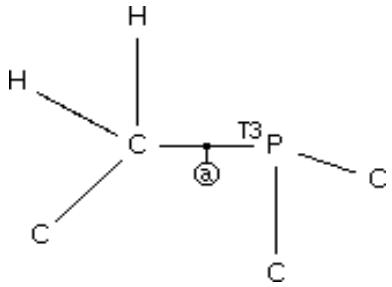
TYPE 1-1_9-4



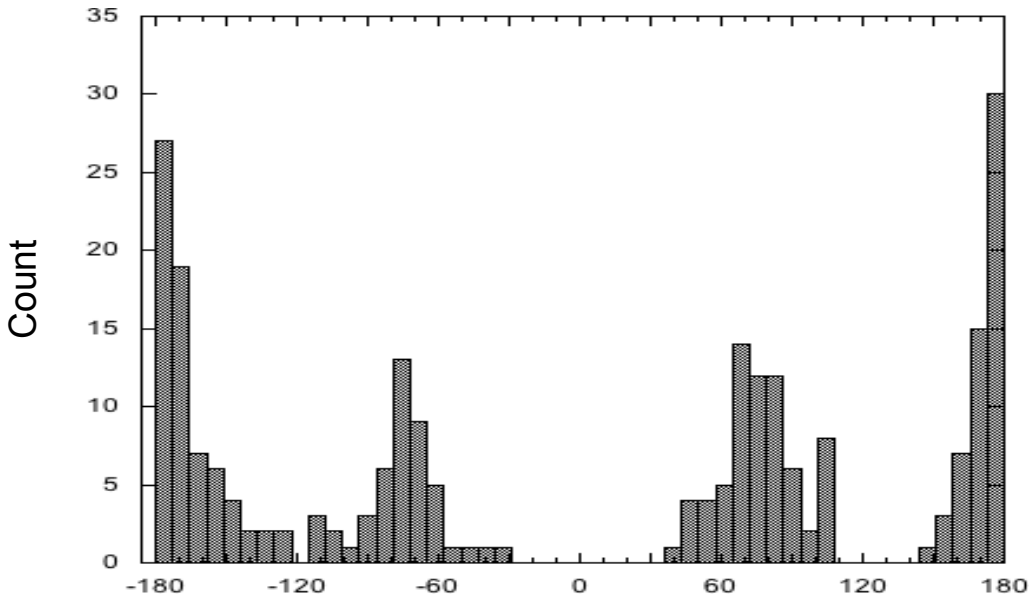
Φ	E
70.0	0.00
185.0	0.00
310.0	0.40



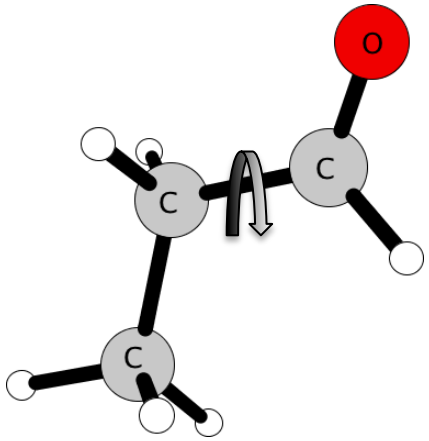
CSD comparison



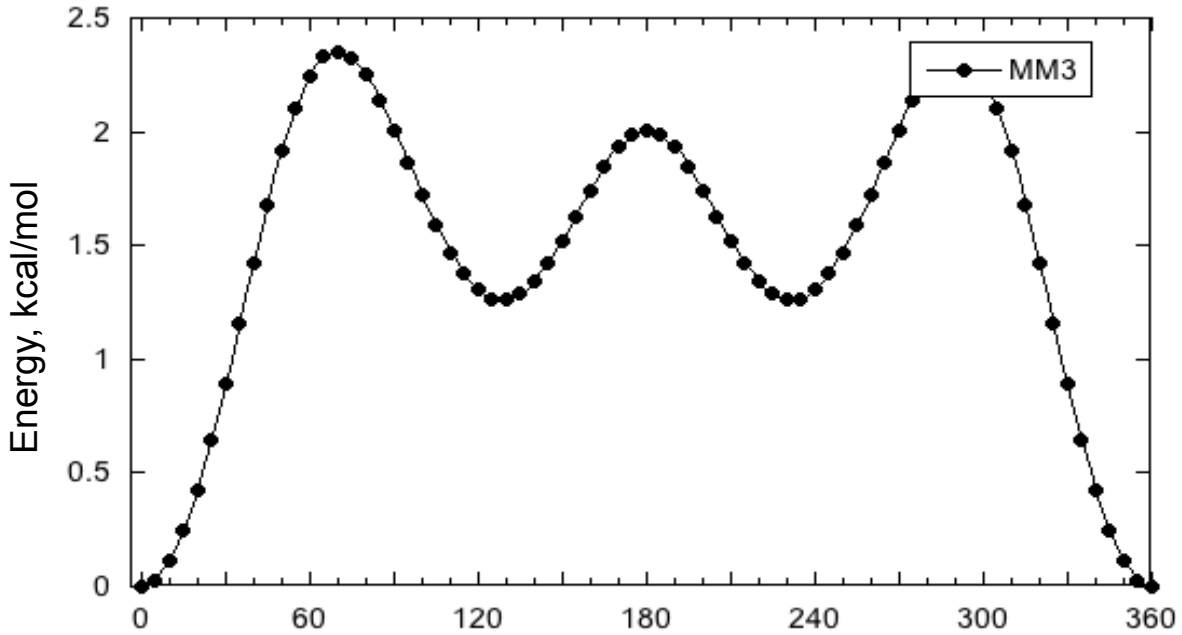
C–P = 1.85 ± 0.01 Å



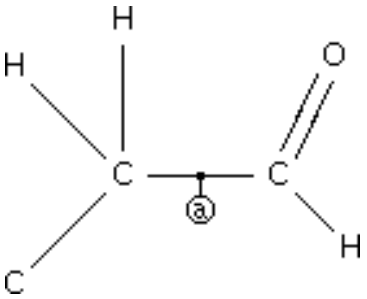
TYPE 1-1:10-1



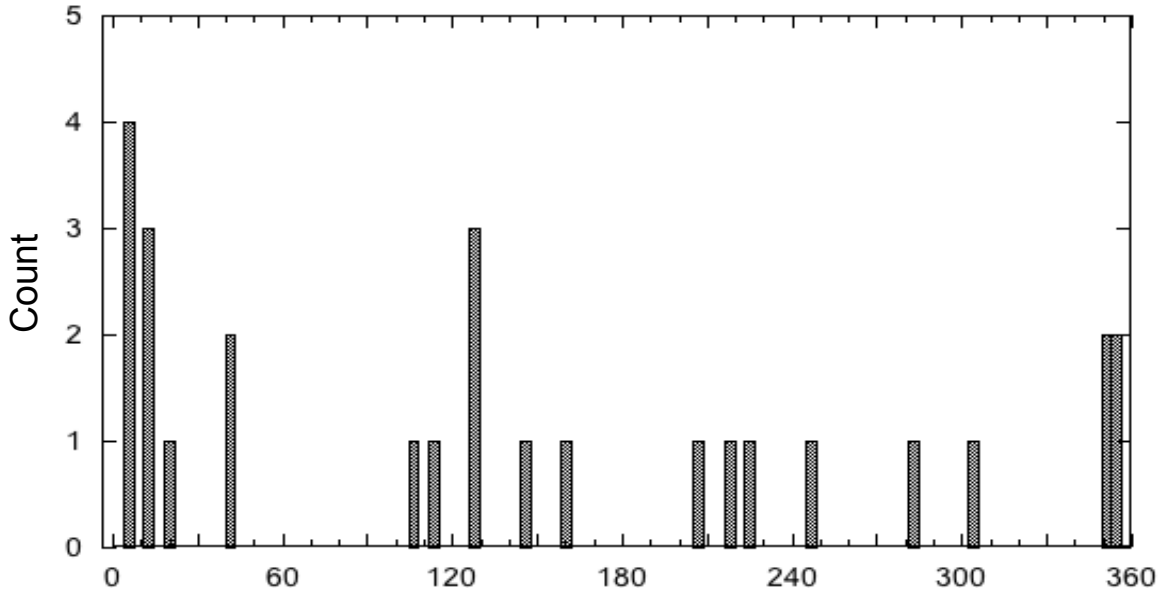
Φ	E
0.0	0.00
130.0	1.26
230.0	1.26



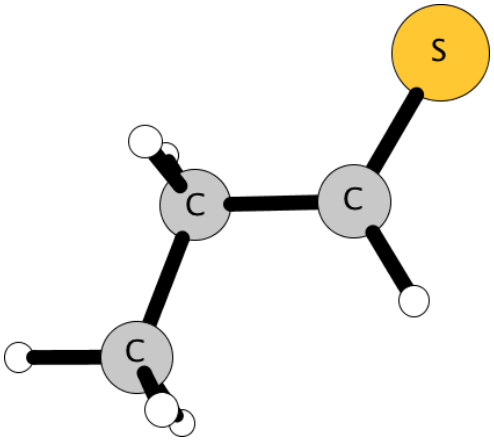
CSD comparison



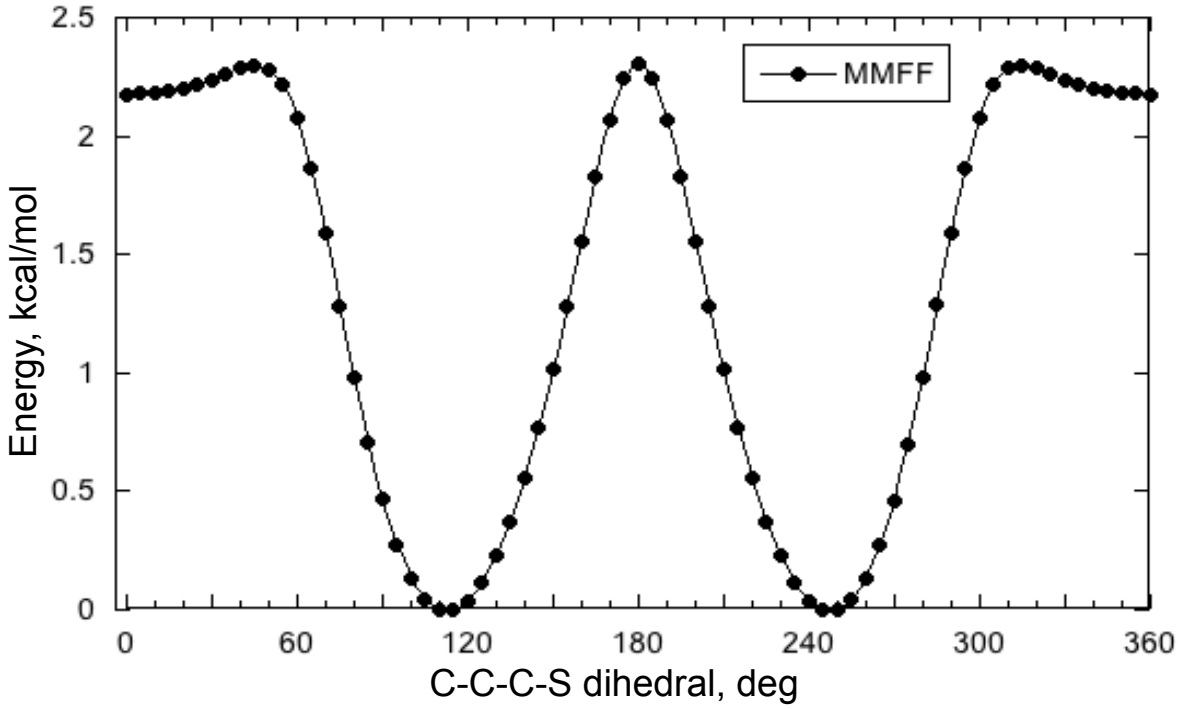
26 hits
C–C = 1.48 ± 0.03 Å



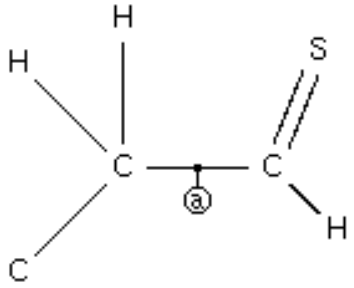
TYPE 1-1:10-2



Φ	E
0.0	2.18
115.0	0.00
245.0	0.00

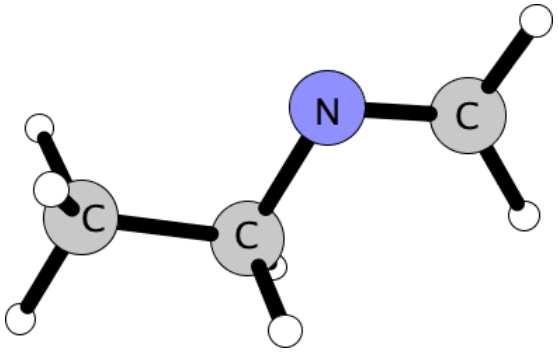


CSD comparison

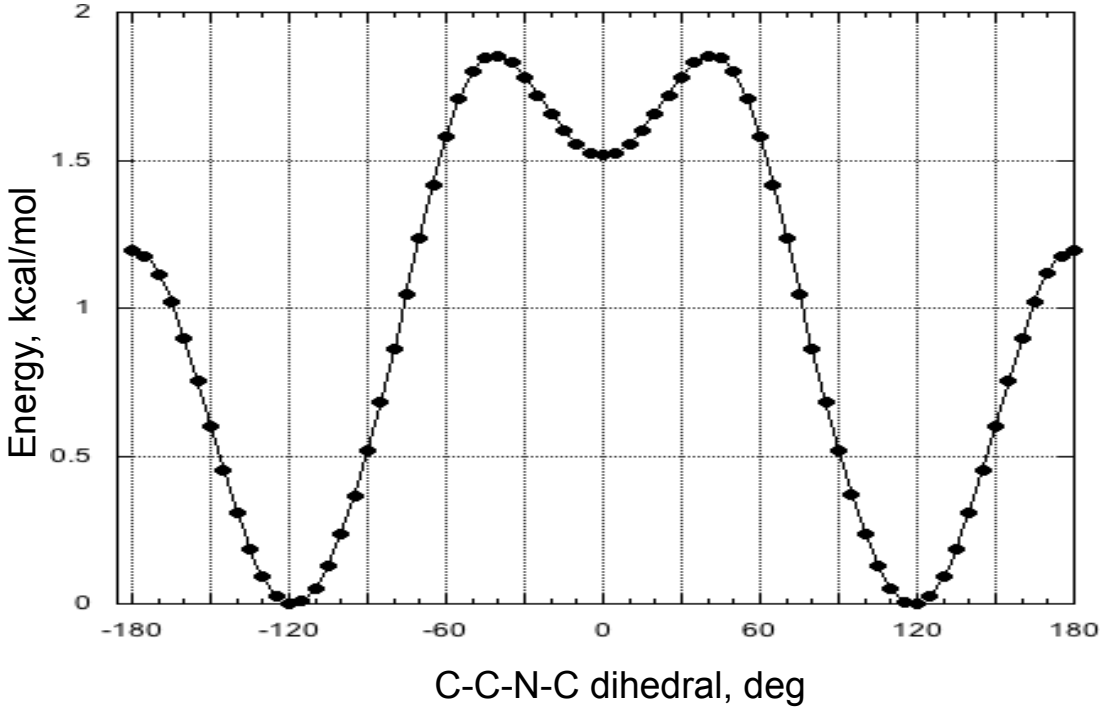


ZERO CSD HITS

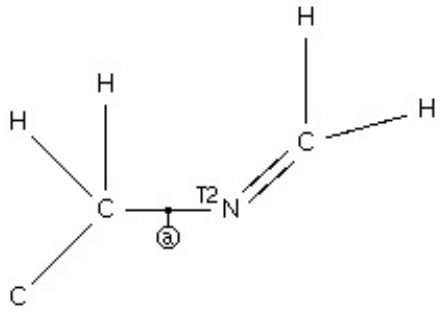
TYPE 1-11:10-3



Φ	E
0.0	1.52
120.0	0.00
240.0	0.00

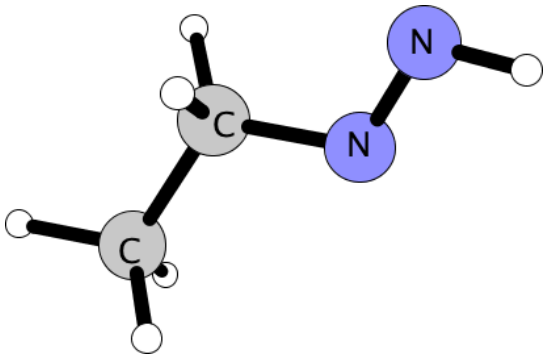


CSD comparison

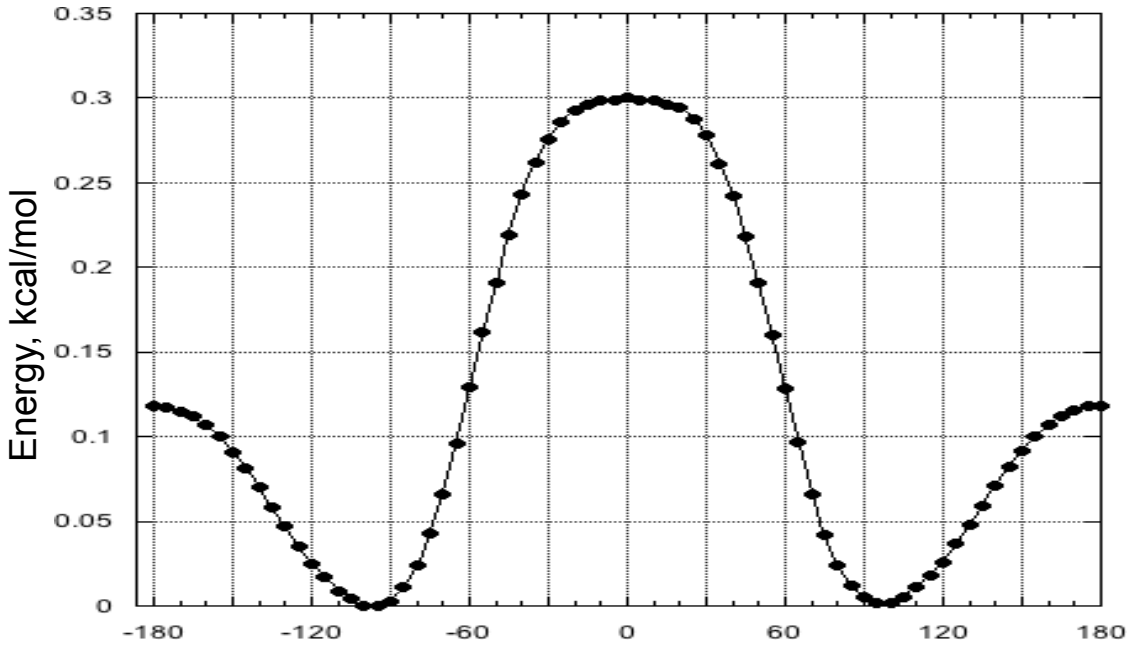


NO CSD HITS

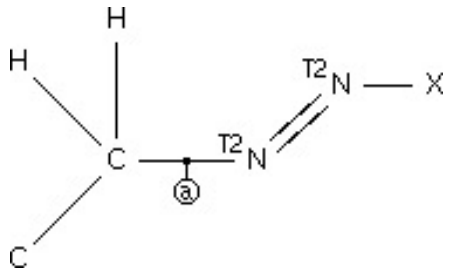
TYPE 1-1:10-4



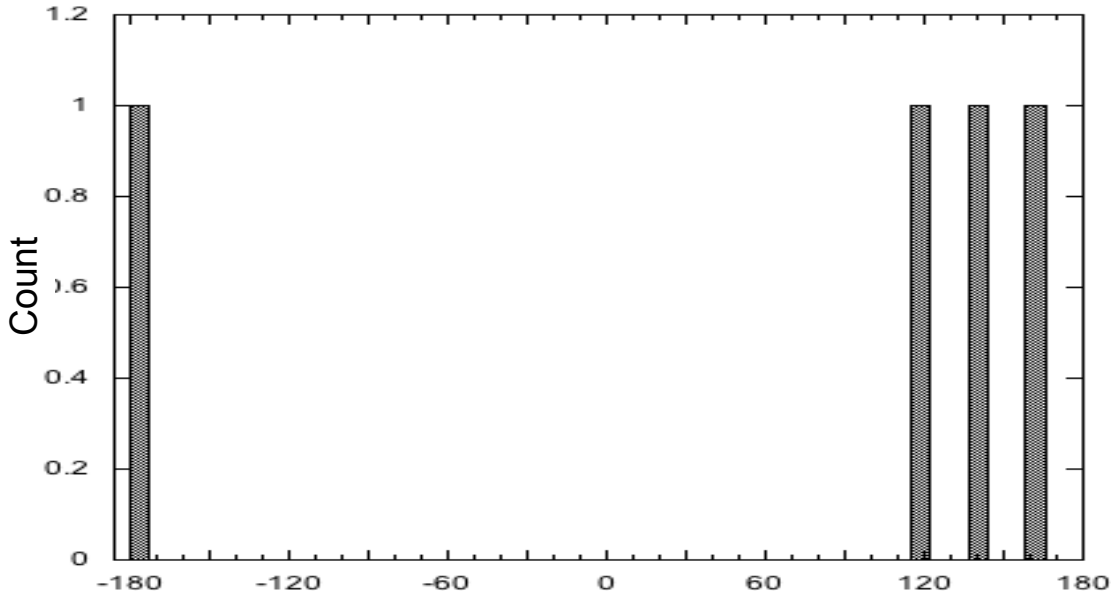
Φ	E
100.0	0.00
260.0	0.00



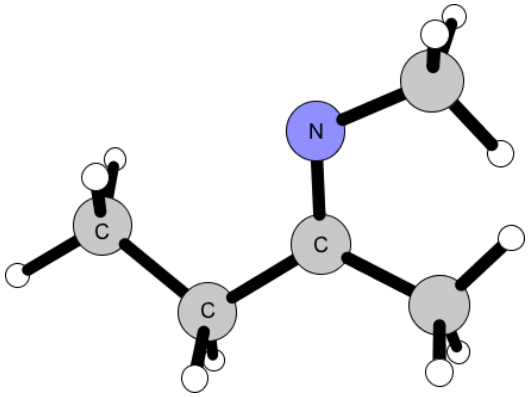
CSD comparison



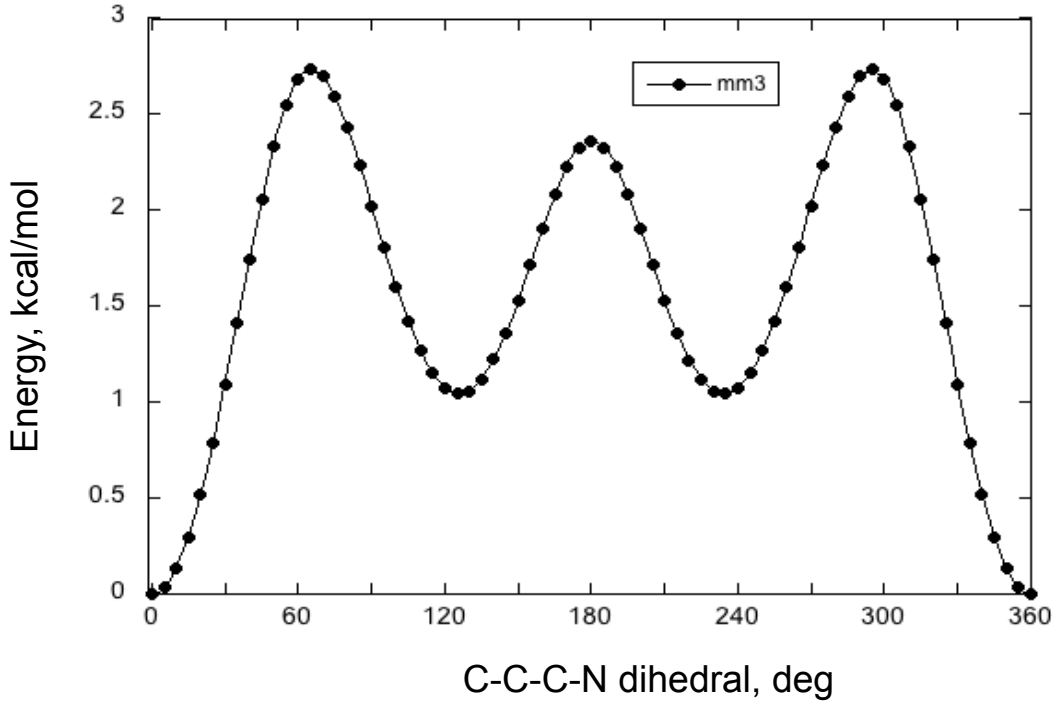
C–N = 1.47 ± 0.00 Å



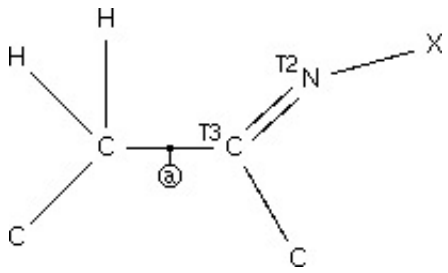
TYPE 1-1:11-1



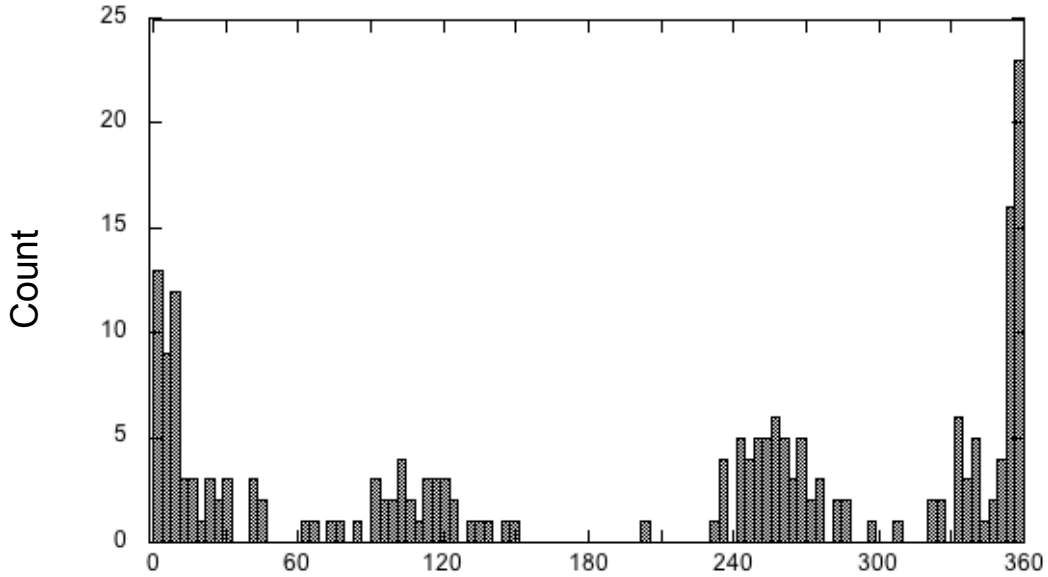
Φ	E
0.0	0.00
125.0	1.04
235.0	1.04



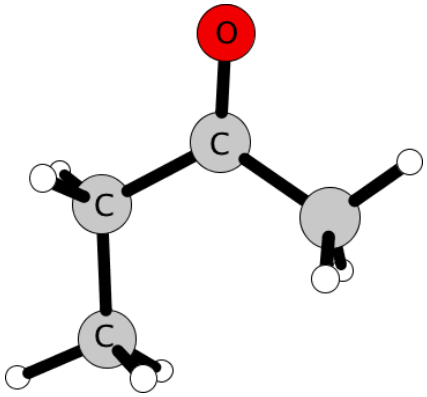
CSD comparison



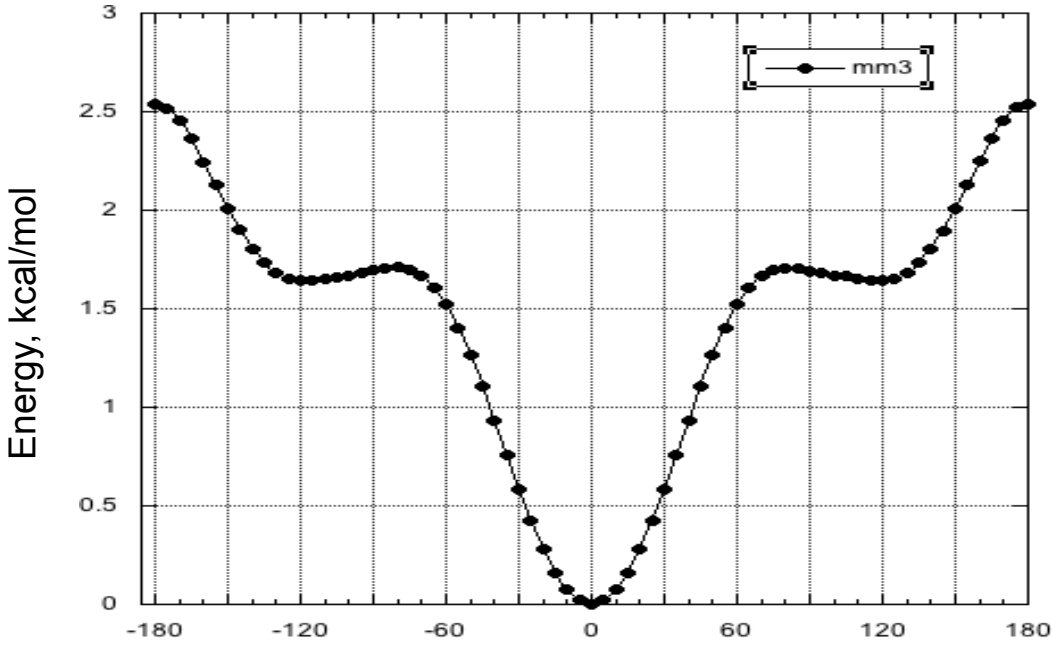
C–C = 1.50 ± 0.02 Å



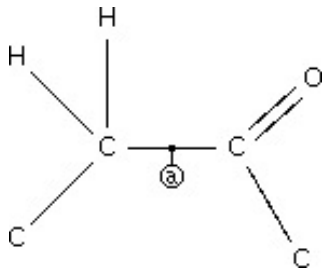
TYPE 1-1:11-2



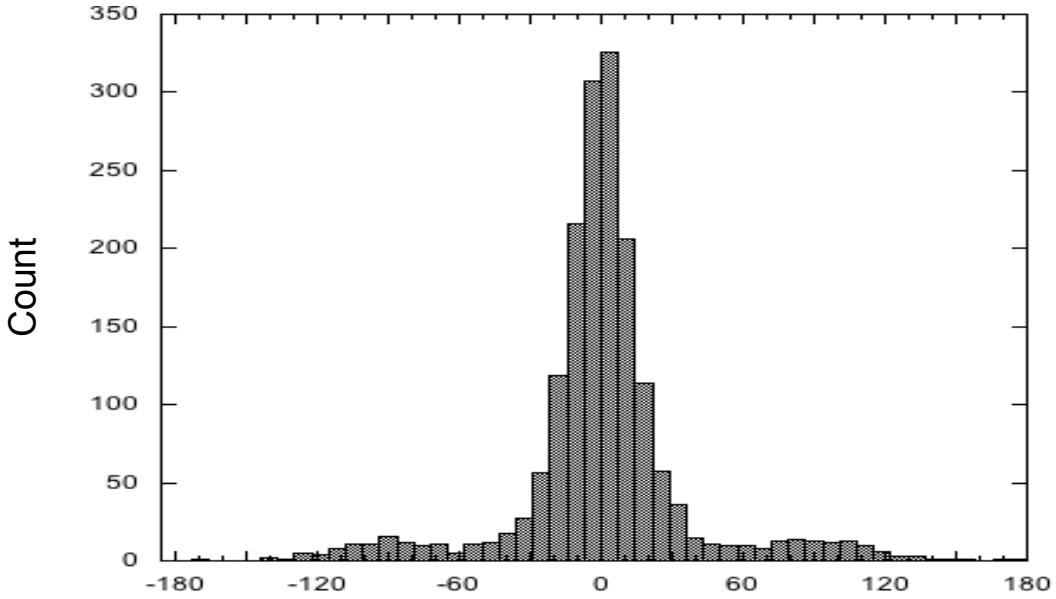
Φ	E
0.0	0.00
120.0	1.64
240.0	1.64



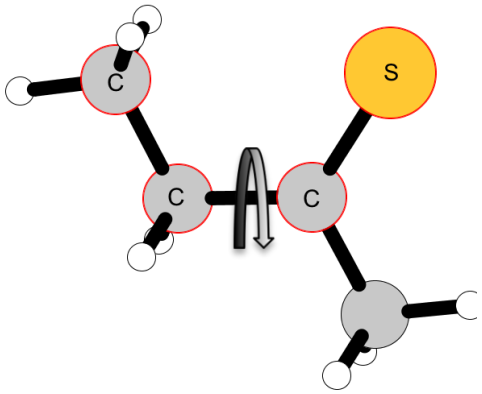
CSD comparison



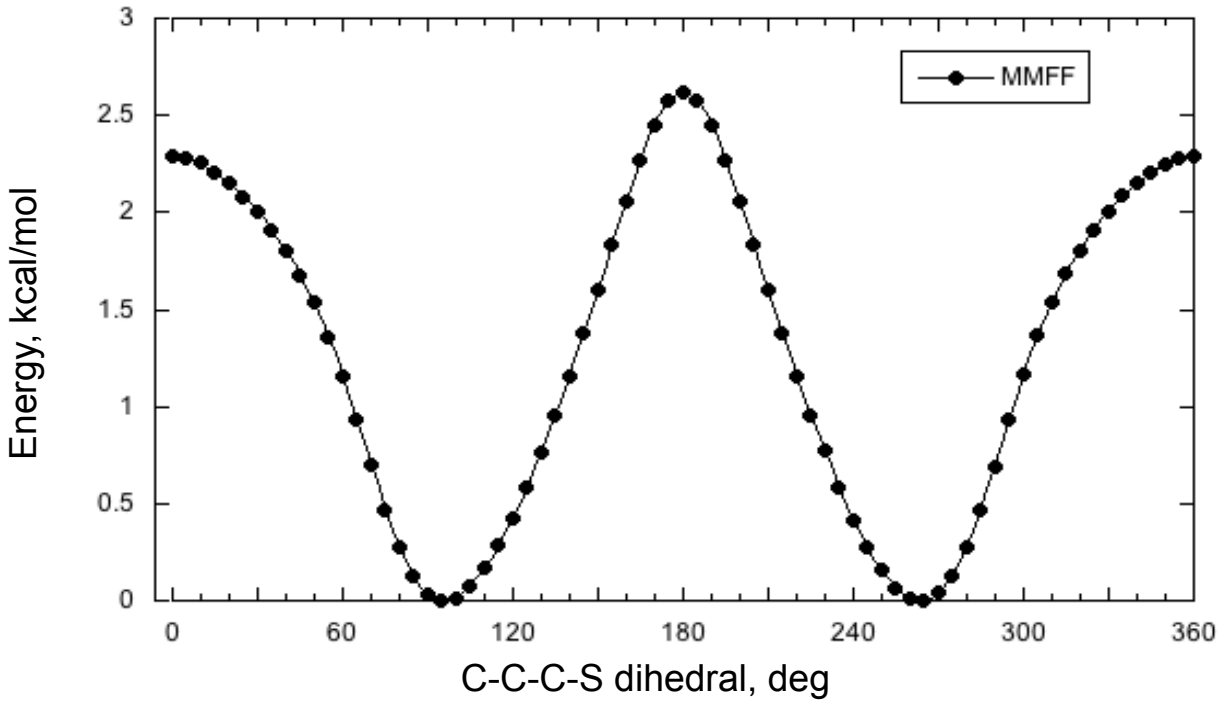
C–C = 1.50 ± 0.02 Å



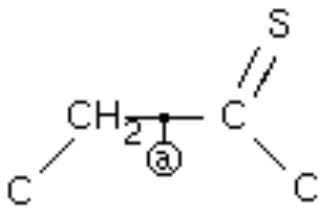
TYPE 1-1:11-3



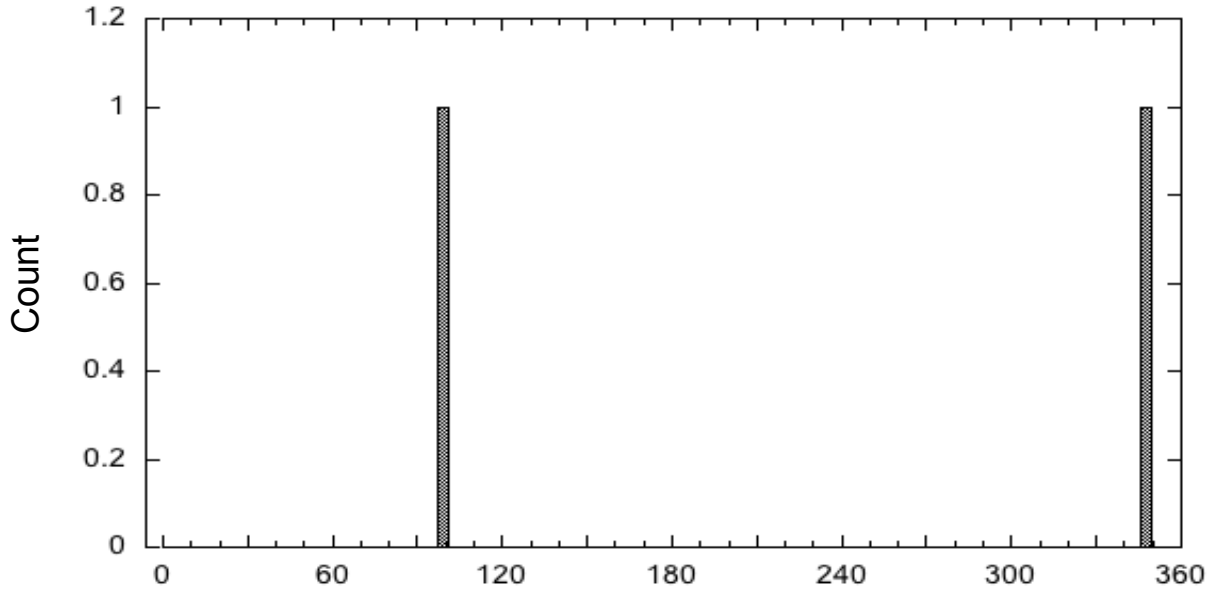
Φ	E
95.0	0.00
265.0	0.00



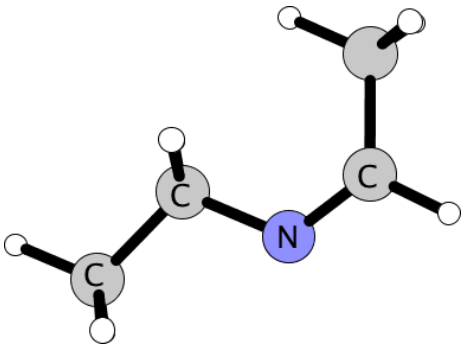
CSD comparison



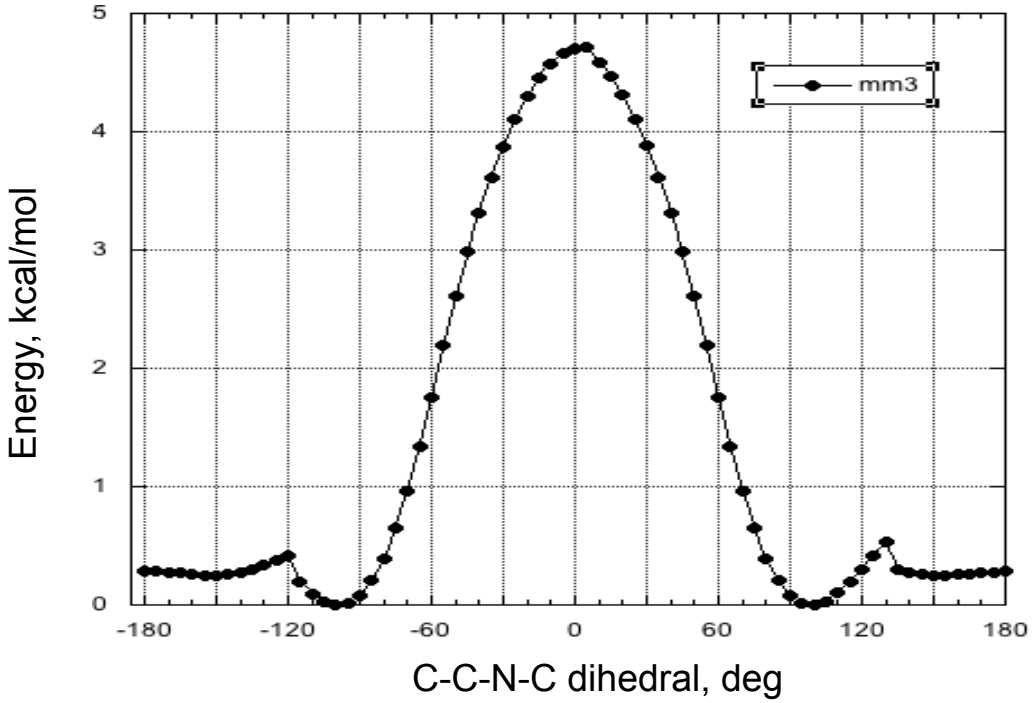
2 hits
C–C = 1.48 ± 0.02 Å



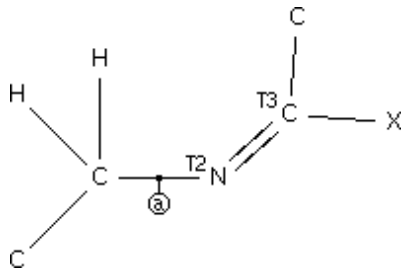
TYPE 1-1:12-1



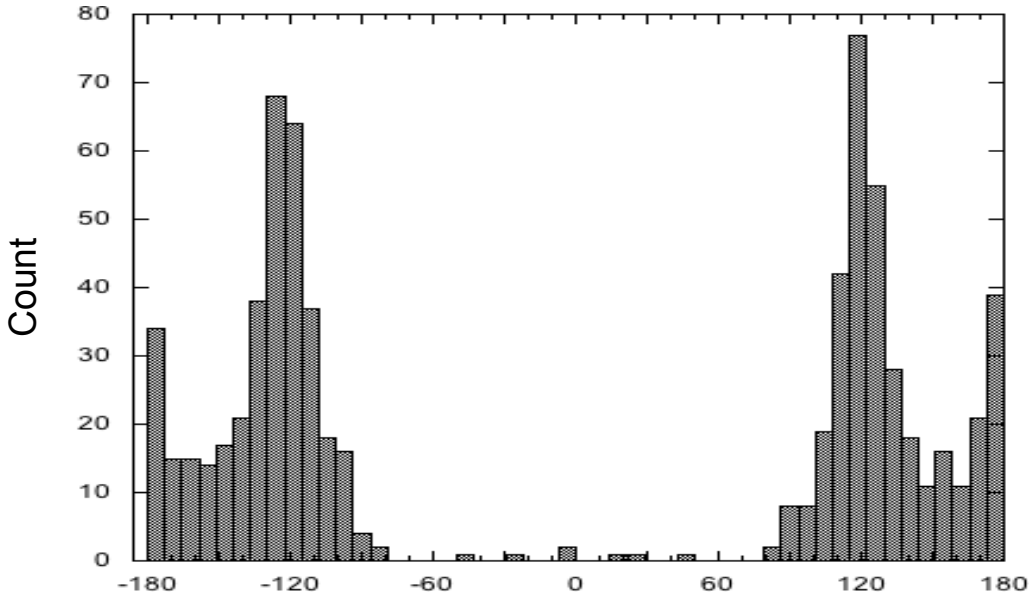
Φ	E
100.0	0.00
150.0	0.25
210.0	0.25
260.0	0.00



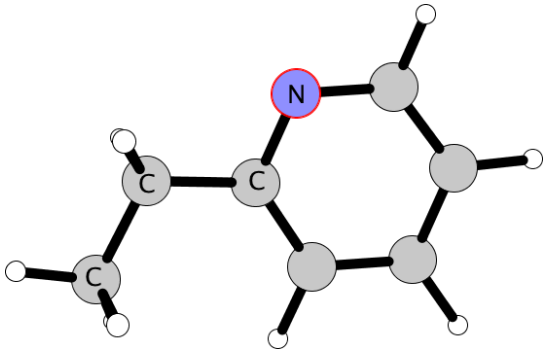
CSD comparison



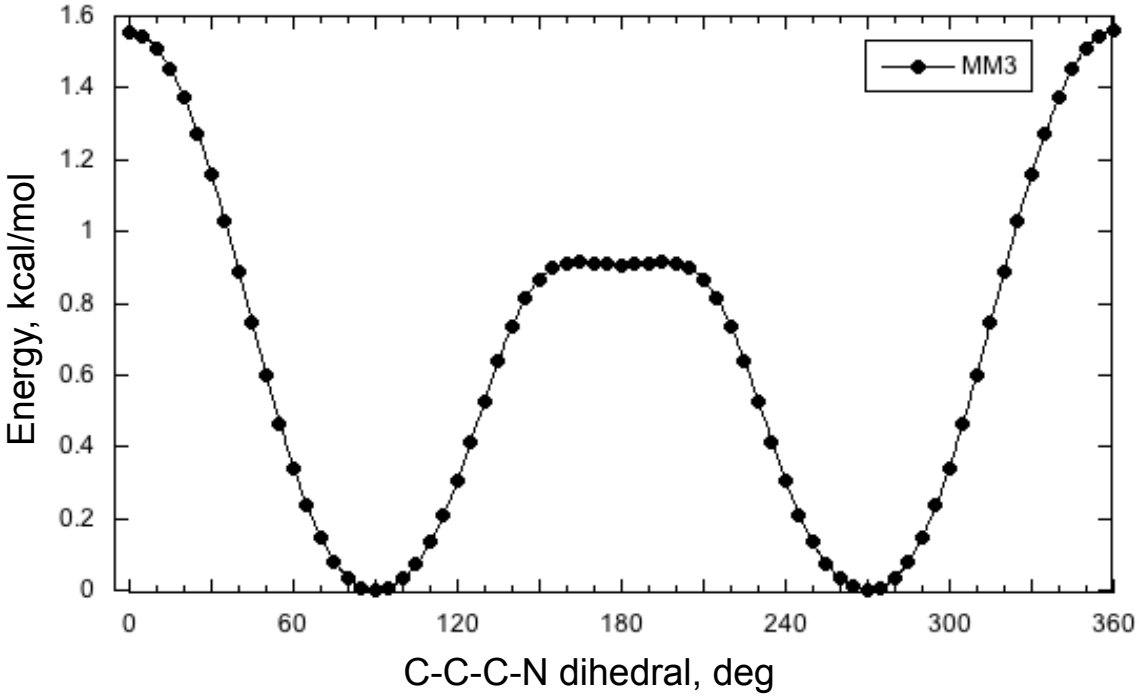
C–N = 1.46 ± 0.02 Å



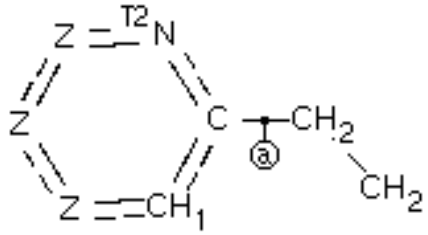
TYPE 1-1:13-1



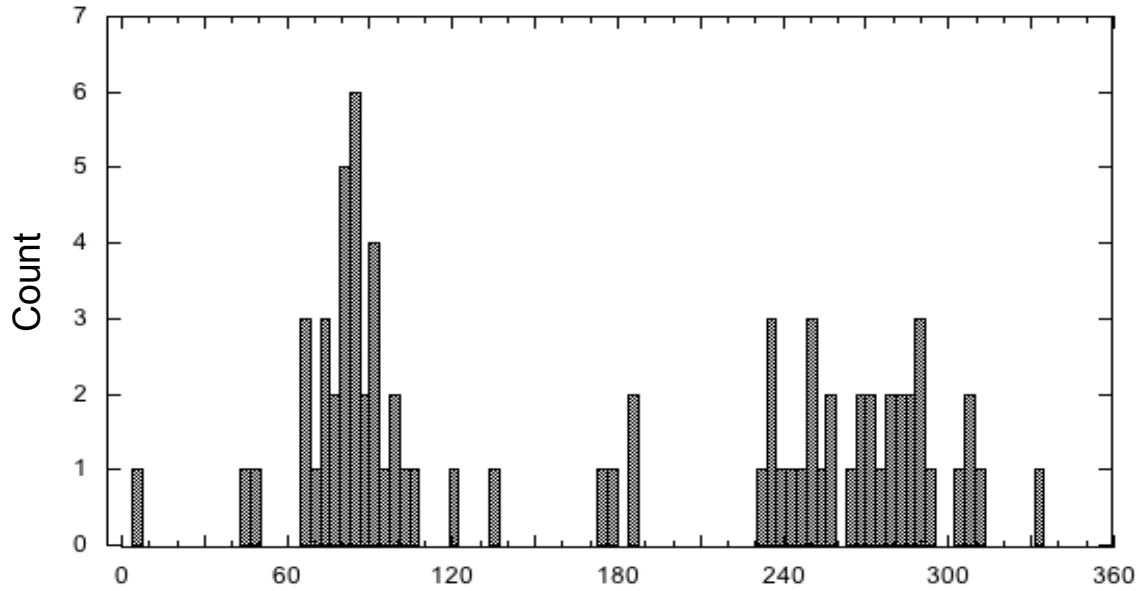
Φ	E
90.0	0.00
180.0	0.90
270.0	0.00



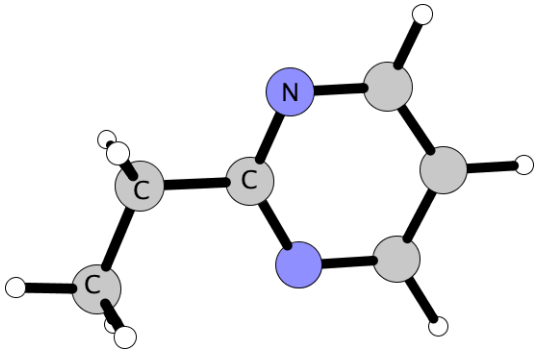
CSD comparison



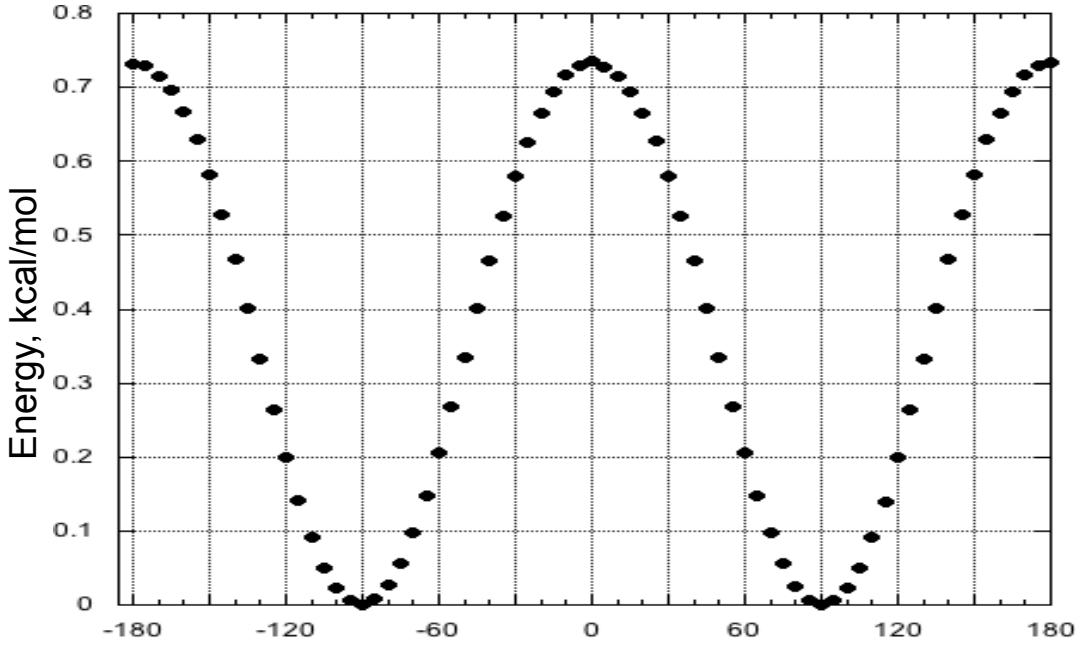
62 hits
C–C = 1.51 ± 0.01 Å



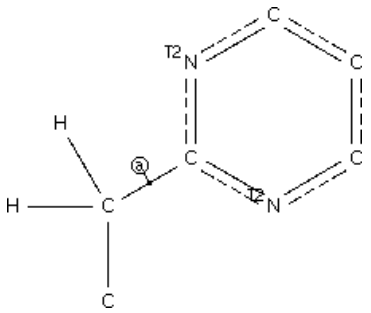
TYPE 1-1:13-2



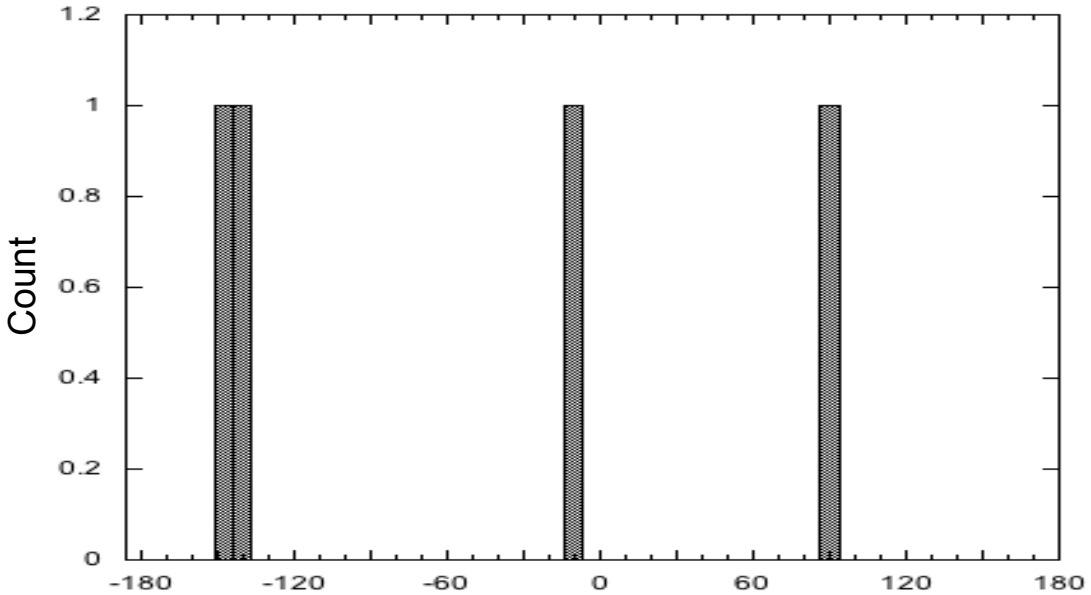
Φ	E
90.0	0.00
270.0	0.00



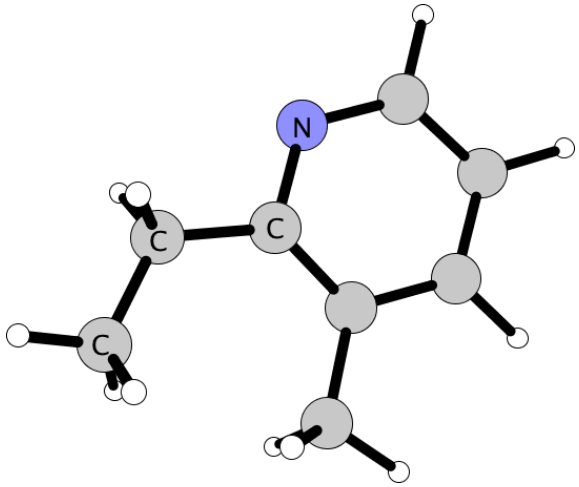
CSD comparison



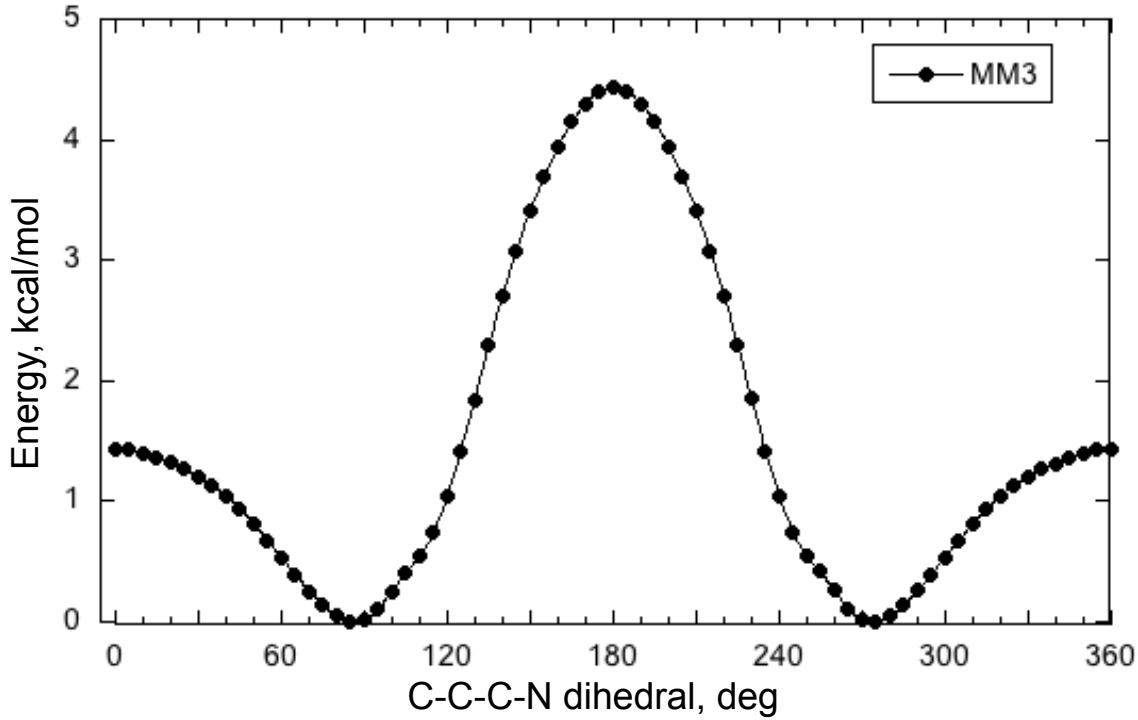
C–C = 1.50 ± 0.01 Å



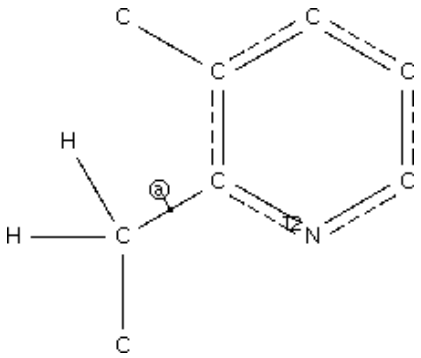
TYPE 1-1:13-3



Φ	E
85.0	0.00
275.0	0.00

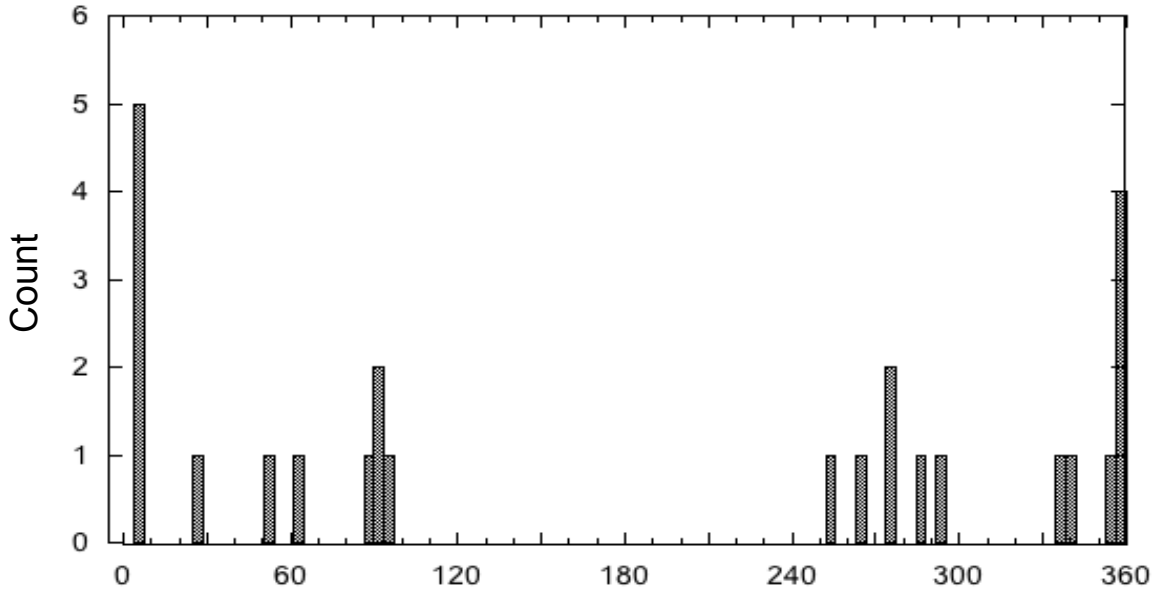


CSD comparison

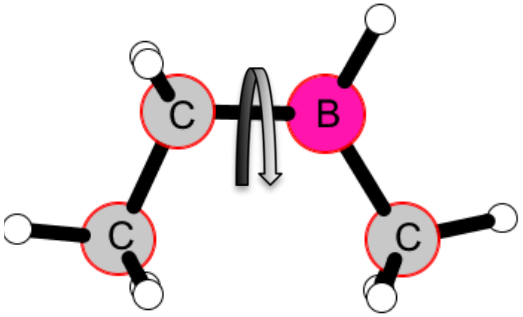


24 hits

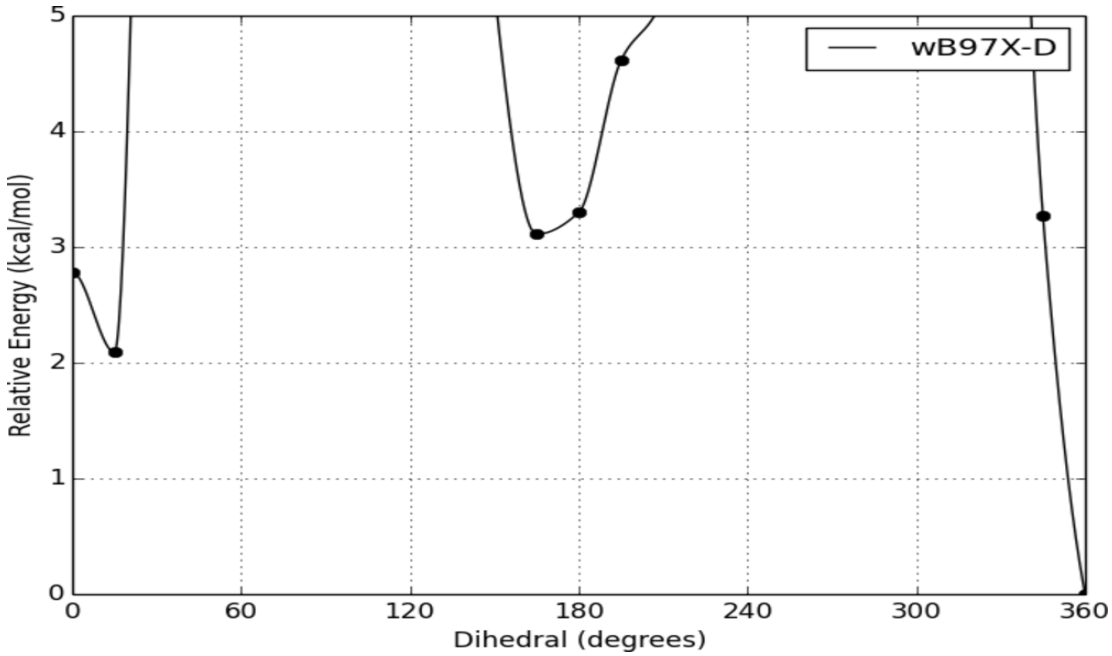
C–C = 1.51 ± 0.01 Å



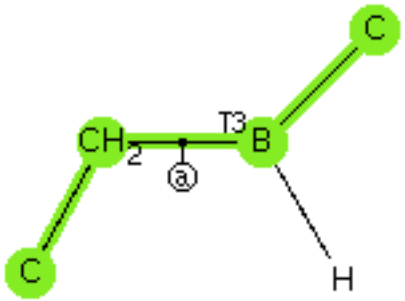
TYPE 1-1:14-2



Φ	E
0.0	0.00
15.0	2.09
165.0	3.11



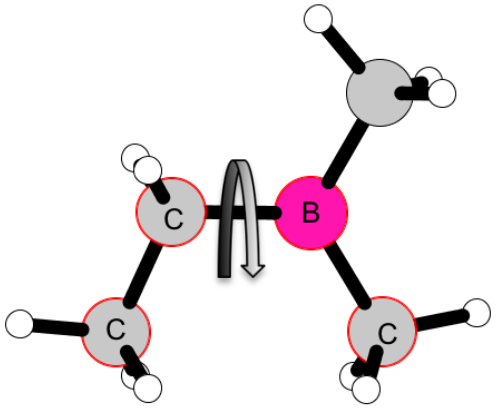
CSD comparison



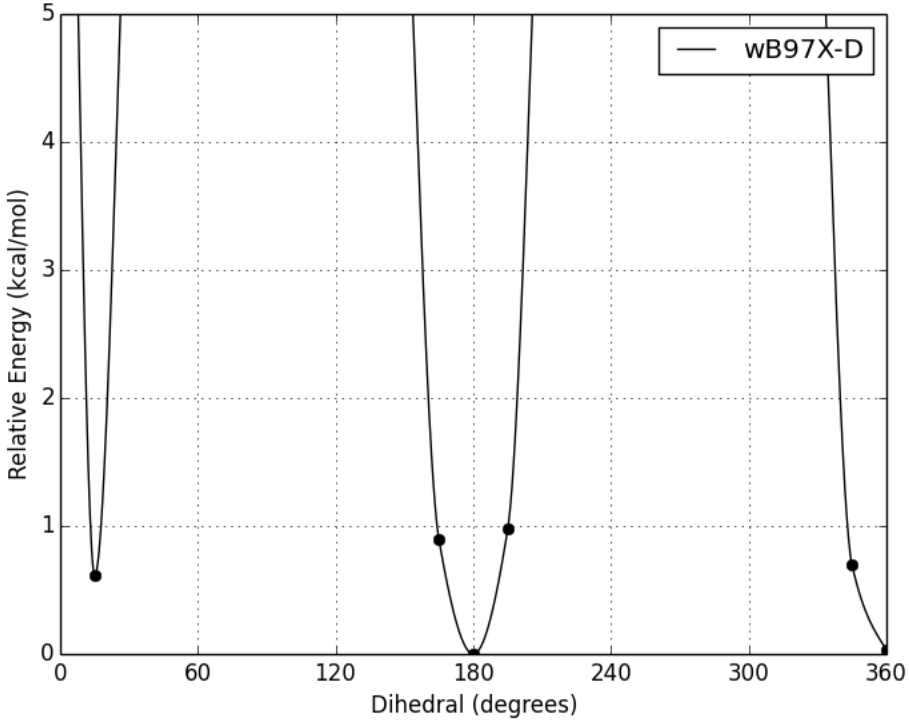
0 hits

0 CSD HITS

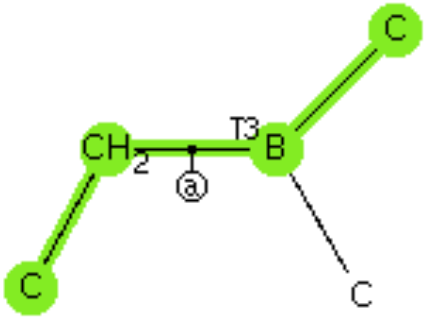
TYPE 1-1:14-3



Φ	E
0.0	0.00
180.0	0.00

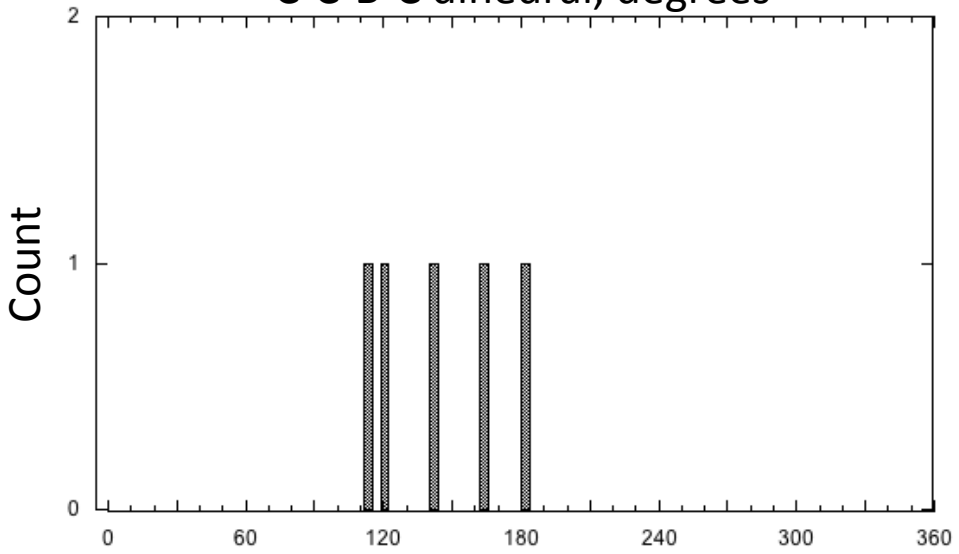


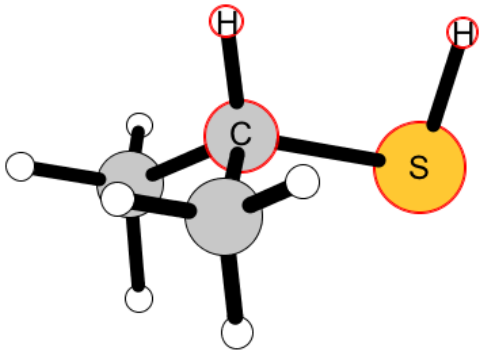
CSD comparison



5 hits

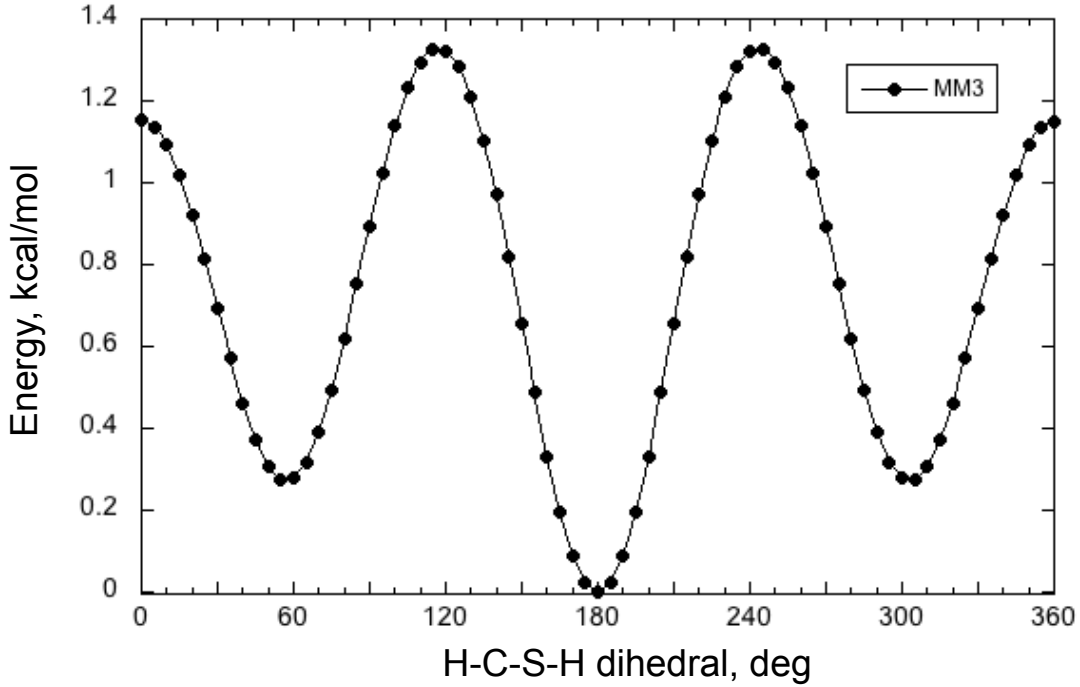
C-C-B-C dihedral, degrees



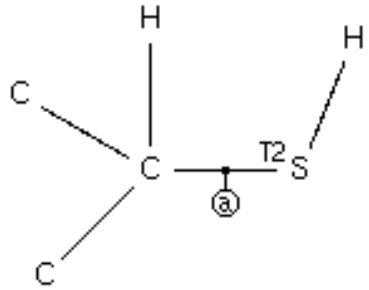


TYPE 1-2:5-4

Φ	E
55.0	0.28
180.0	0.00
305.0	0.28

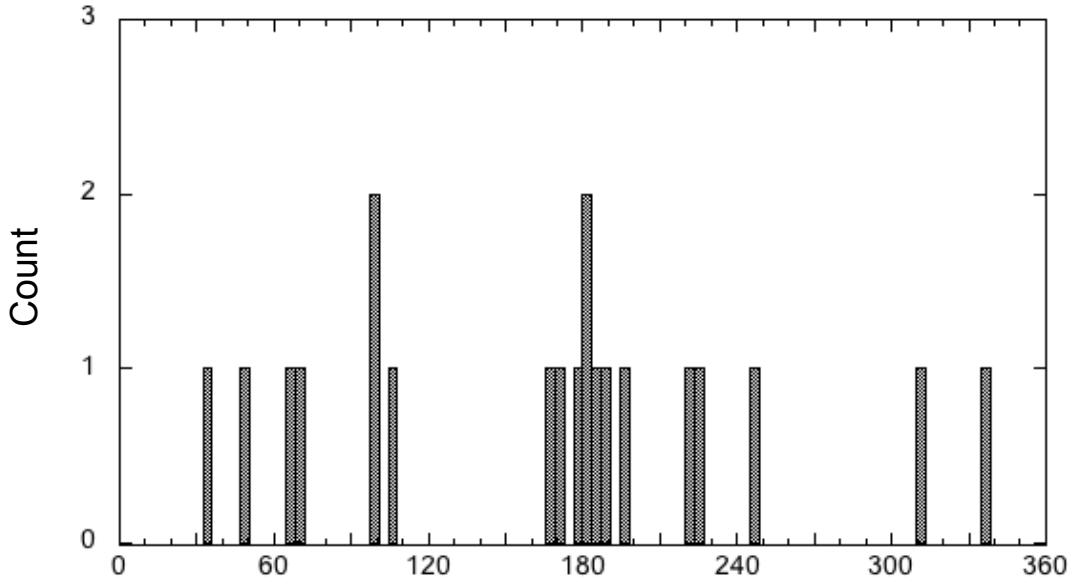


CSD comparison

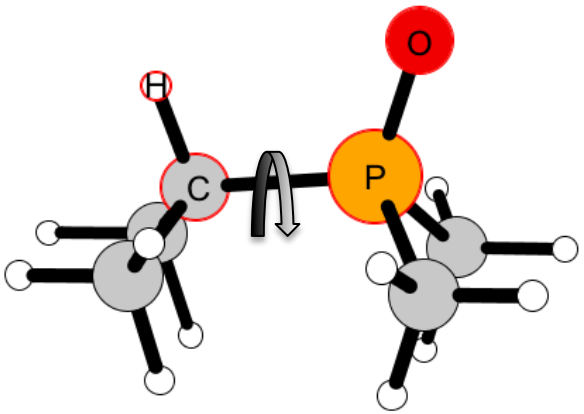


18 hits

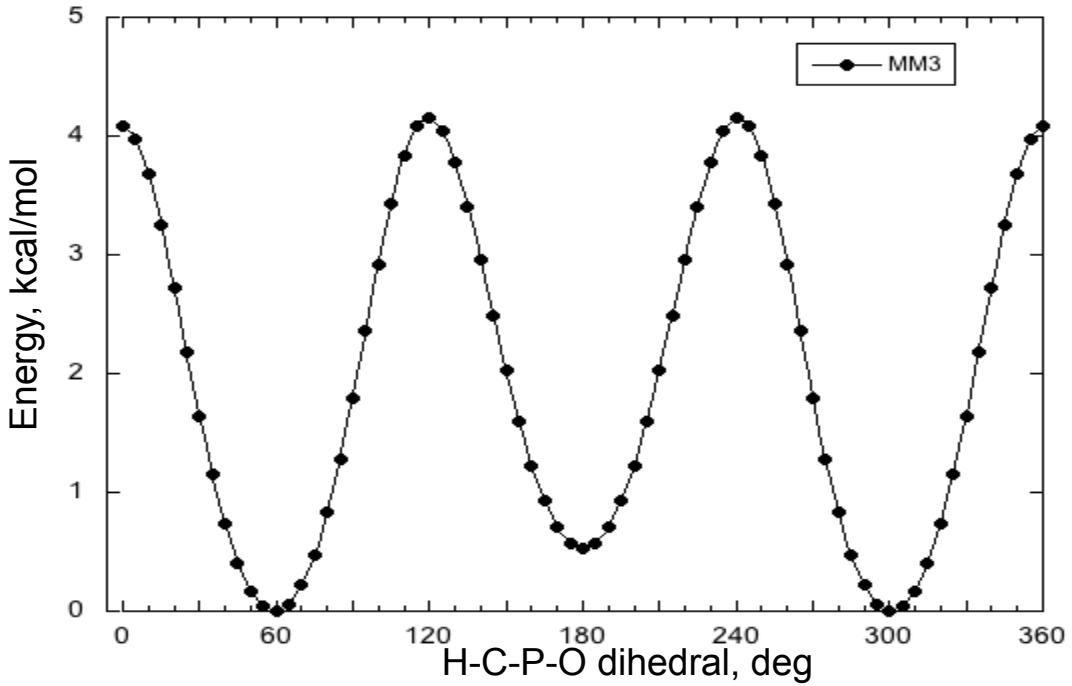
C–S = 1.82 ± 0.01 Å



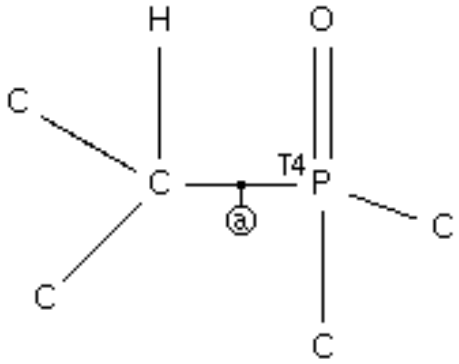
TYPE 1-2:8-1



Φ	E
60.0	0.00
180.0	0.52
300.0	0.00

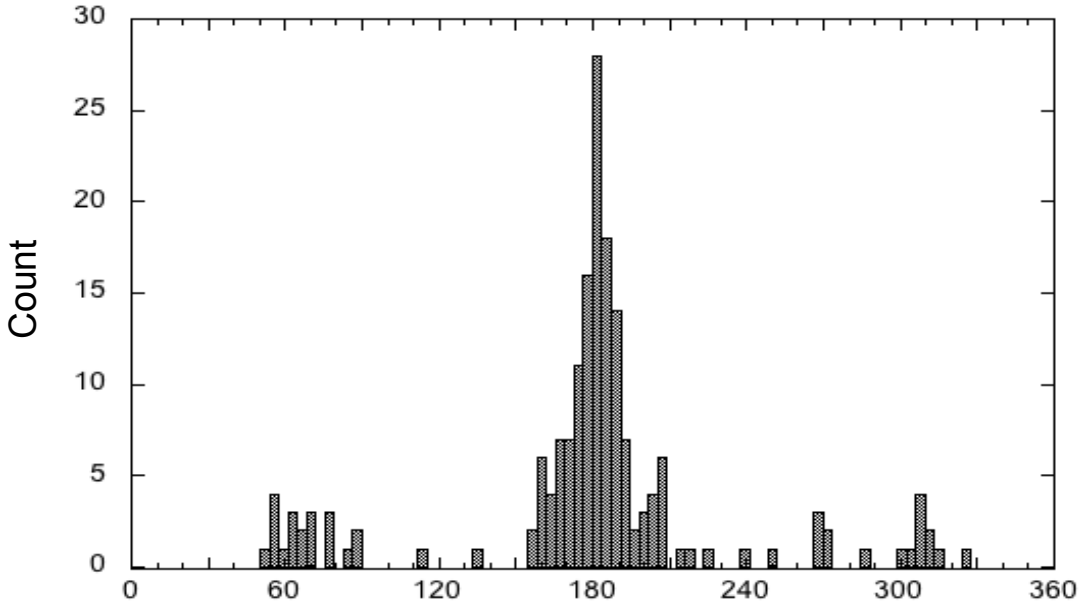


CSD comparison

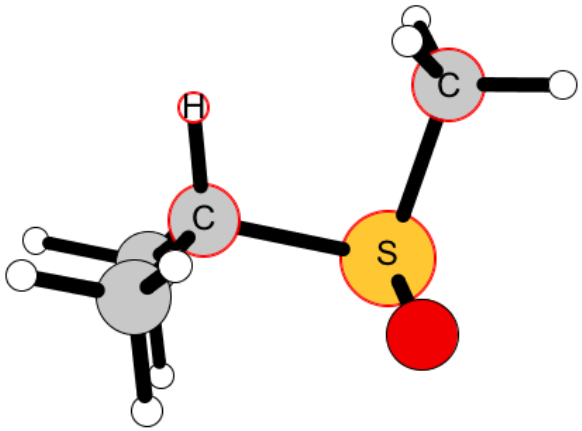


106 hits

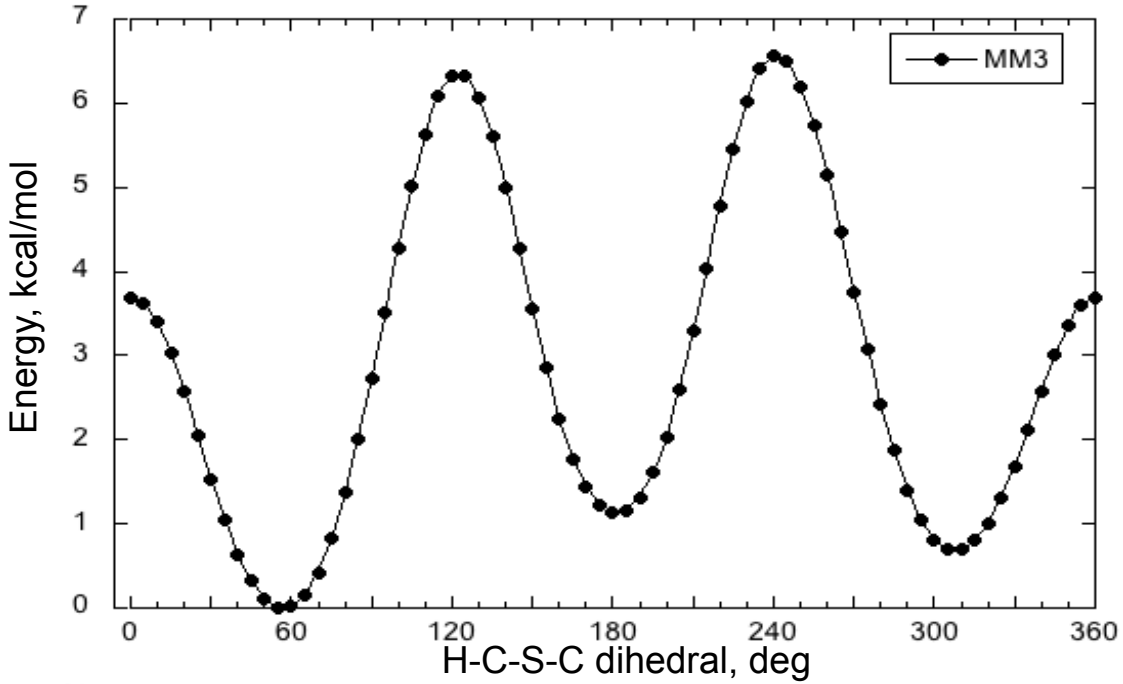
C–P = 1.82 ± 0.02 Å



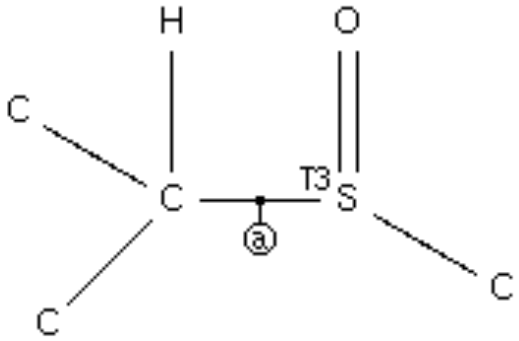
TYPE 1-2:8-2



Φ	E
55.0	0.00
180.0	1.14
305.0	0.70

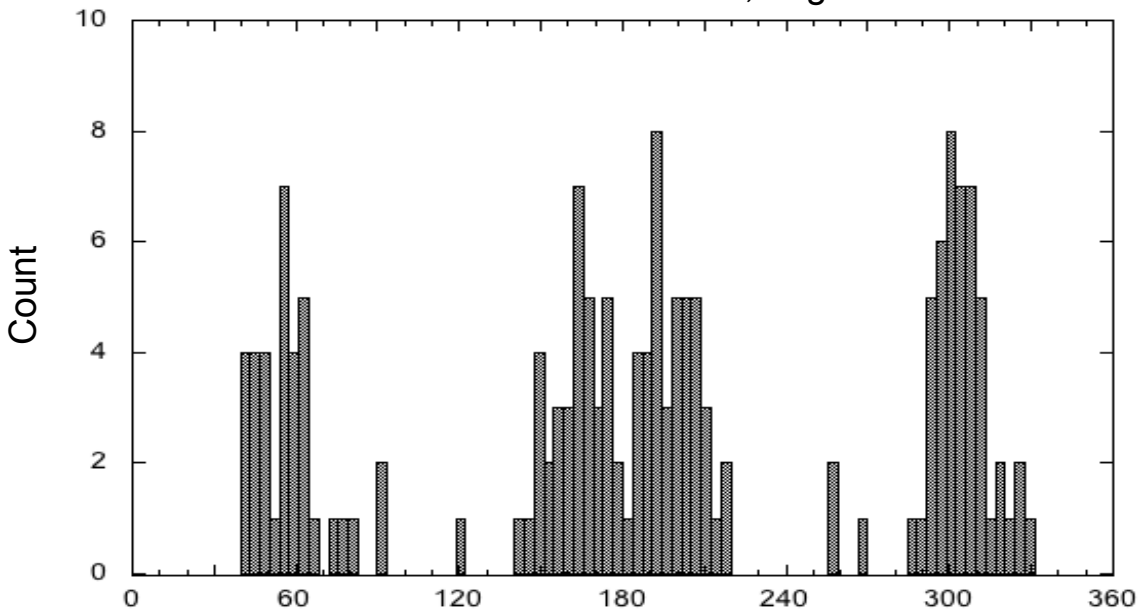


CSD comparison

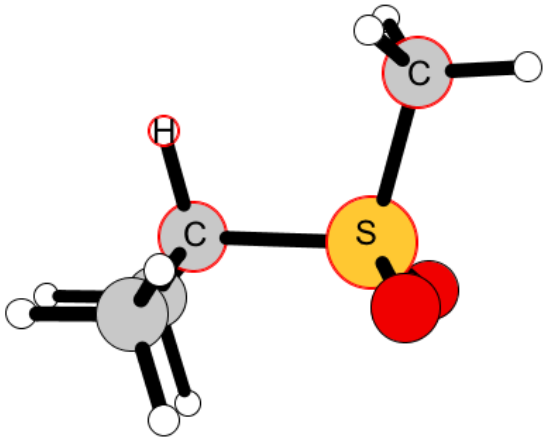


140 hits

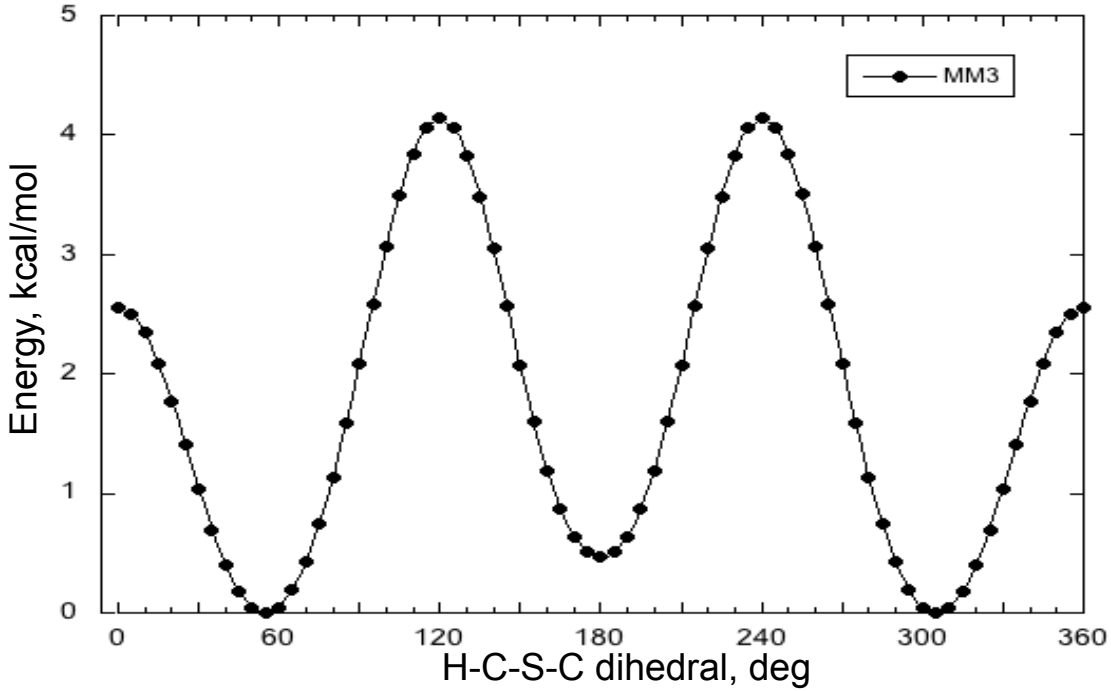
C–S = 1.83 ± 0.02 Å



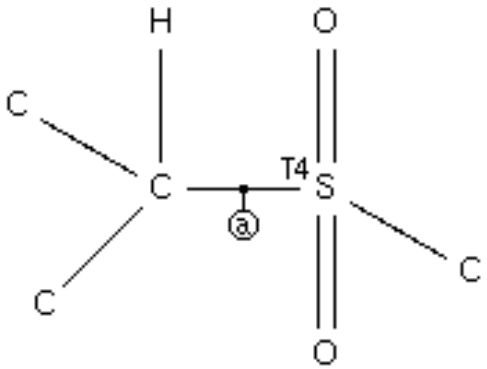
TYPE 1-2:8-4



Φ	E
55.0	0.00
180.0	0.46
305.0	0.00

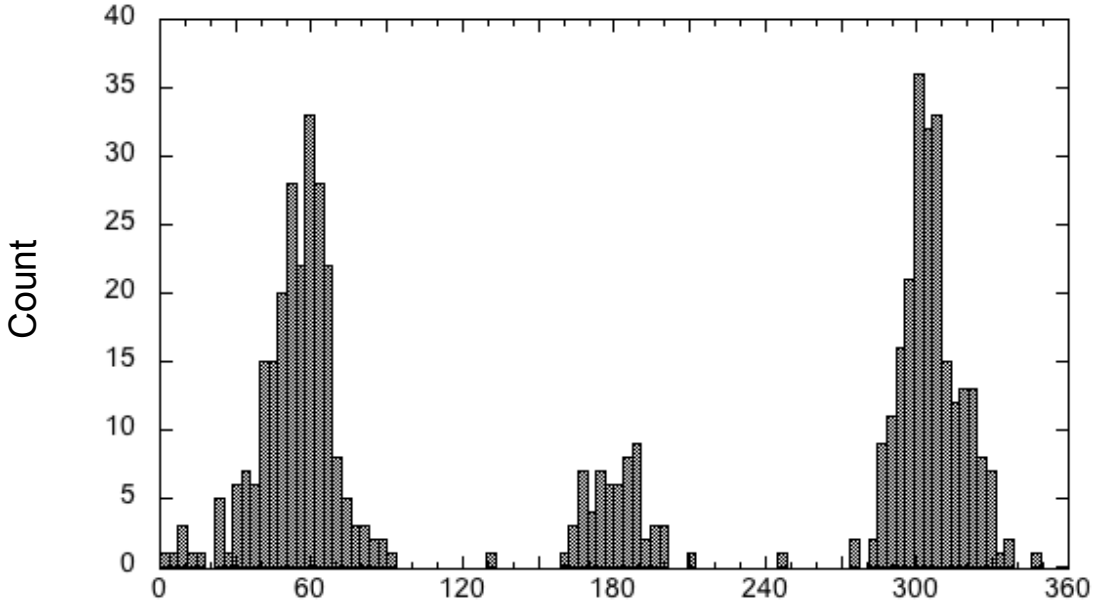


CSD comparison

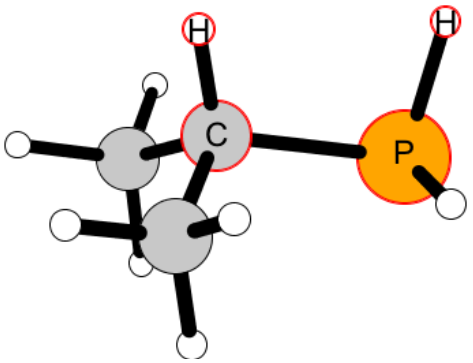


450 hits

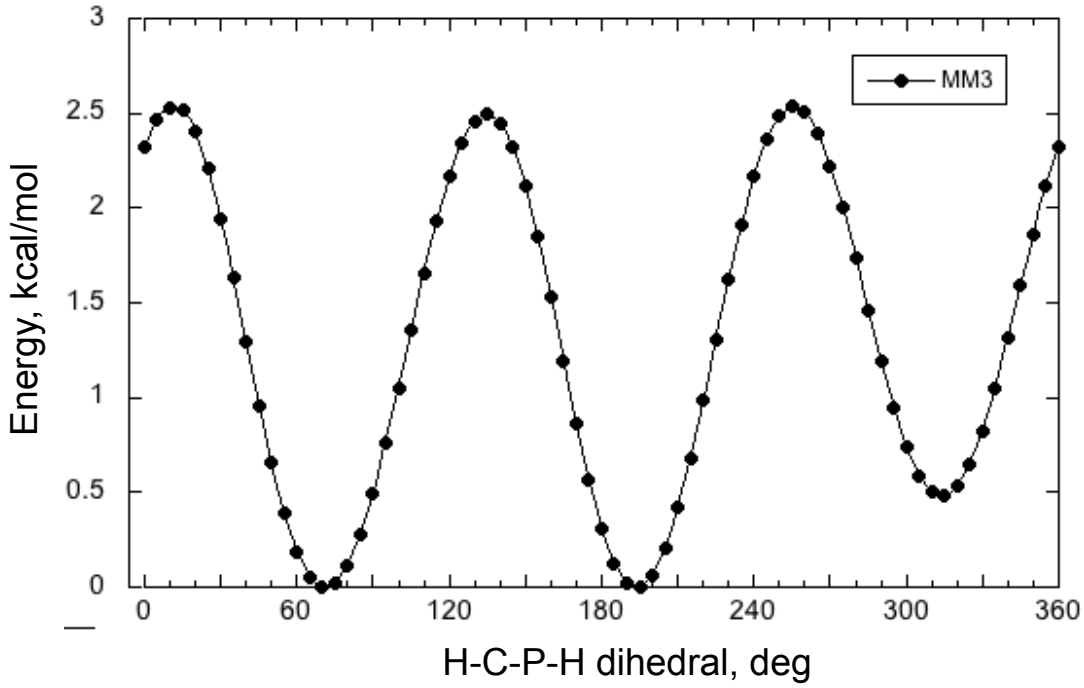
C–S = 1.80 ± 0.02 Å



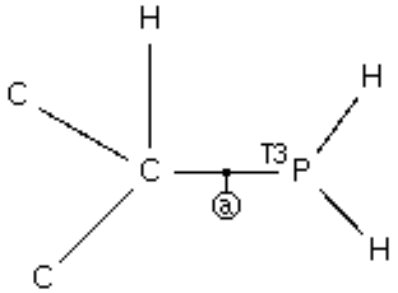
TYPE 1-2:9-1



Φ	E
70.0	0.00
195.0	0.00
315.0	0.48



CSD comparison

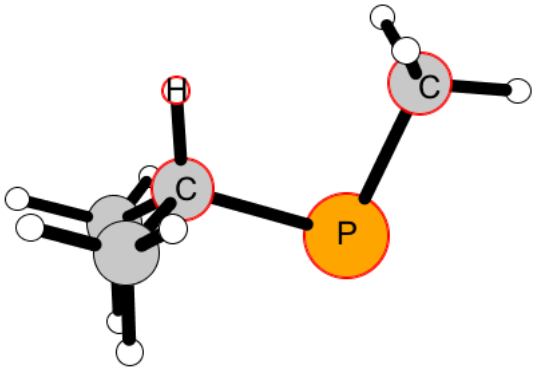


0 hits

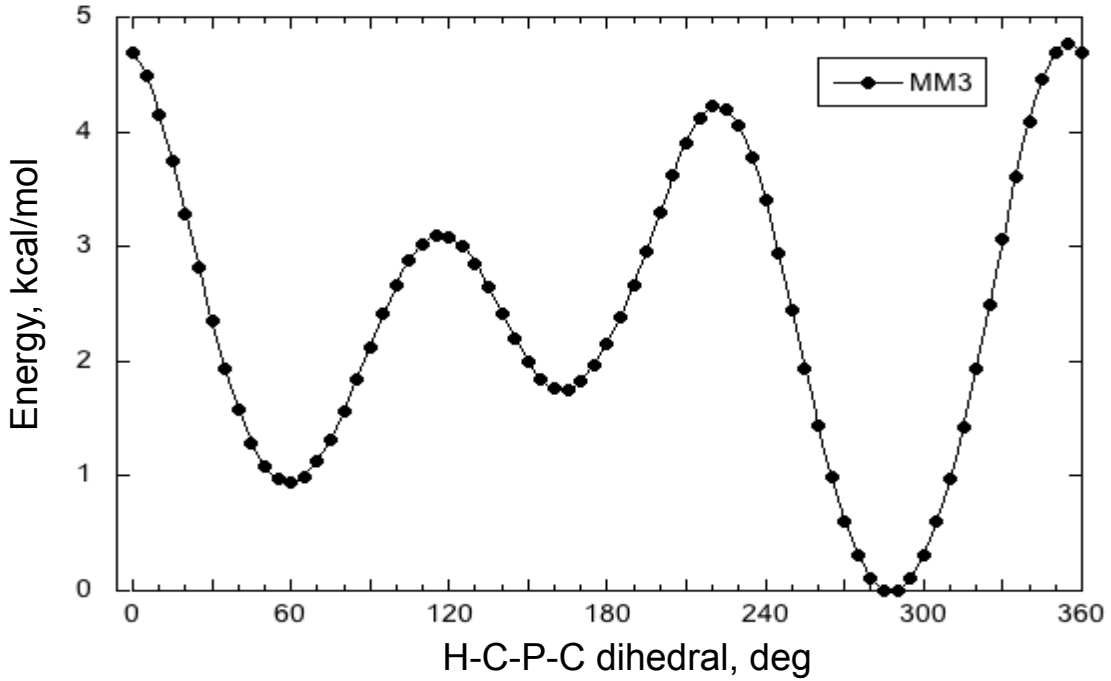
Count

NO CDB HITS

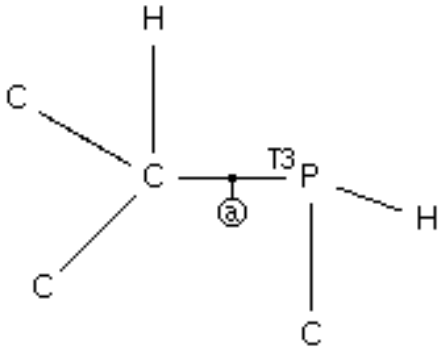
TYPE 1-2:9-2



Φ	E
60.0	0.95
165.0	1.76
285.0	0.00

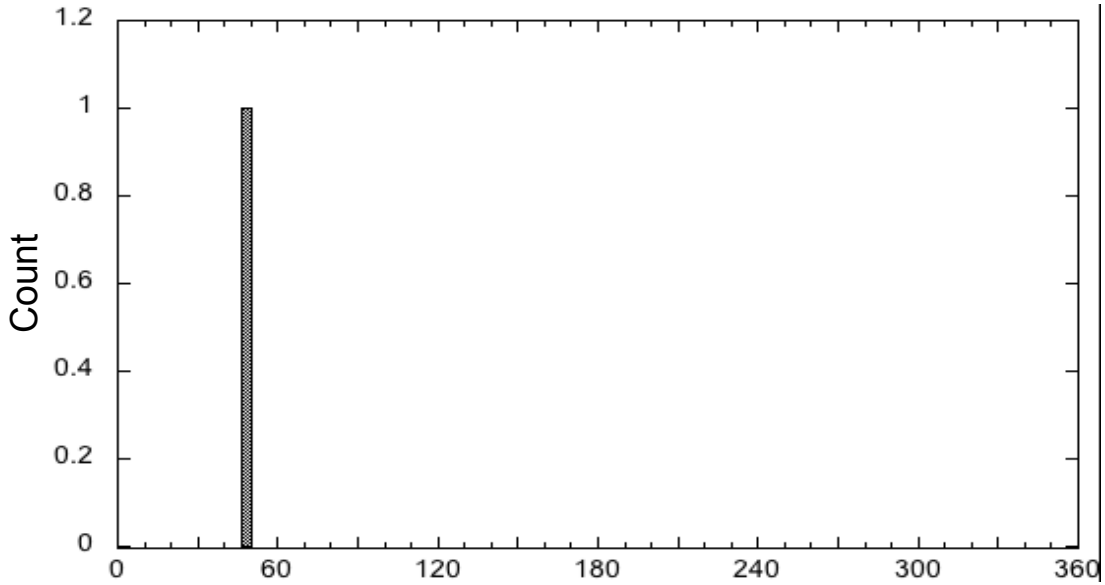


CSD comparison

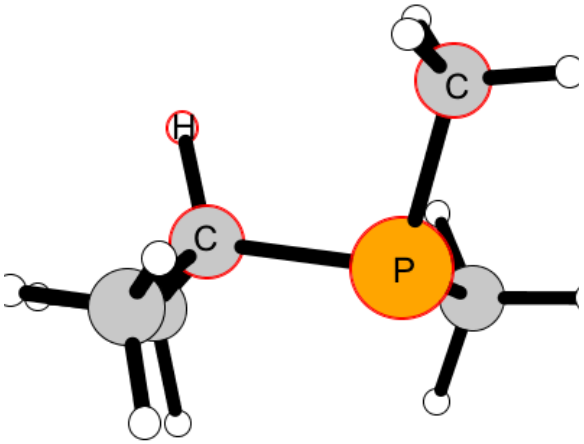


1 hit

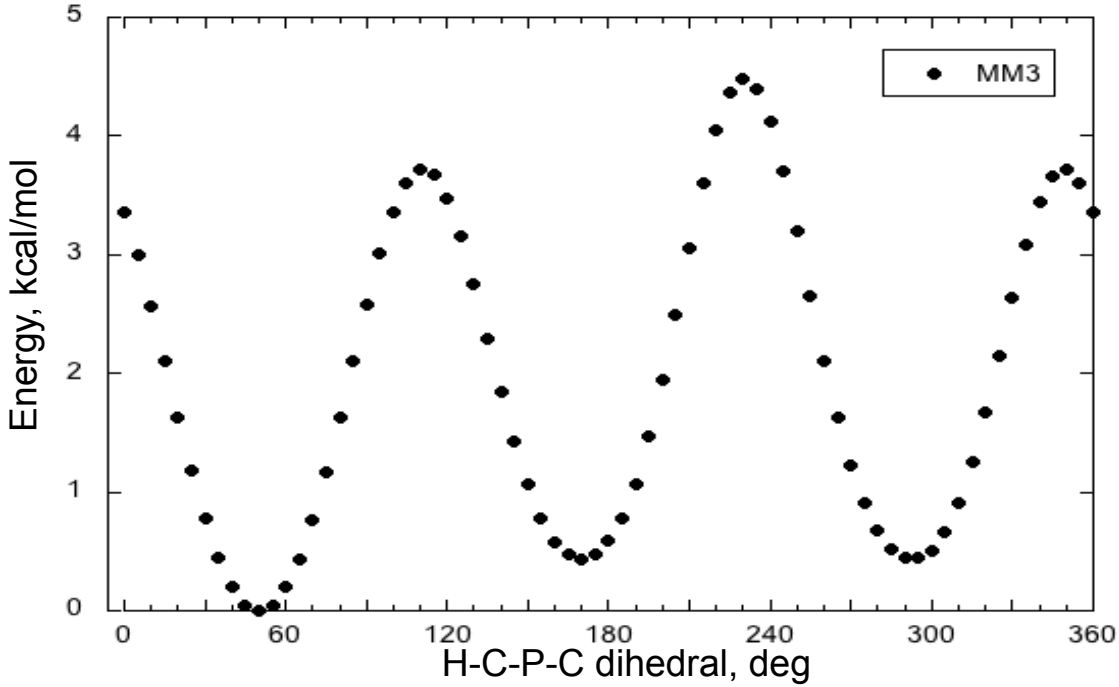
C–P = 1.89 Å



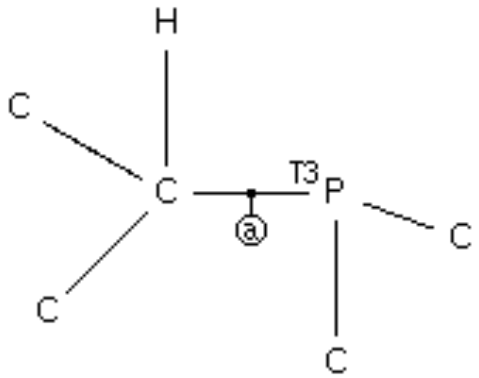
TYPE 1-2:9-4



Φ	E
50.0	0.00
170.0	0.44
295.0	0.44

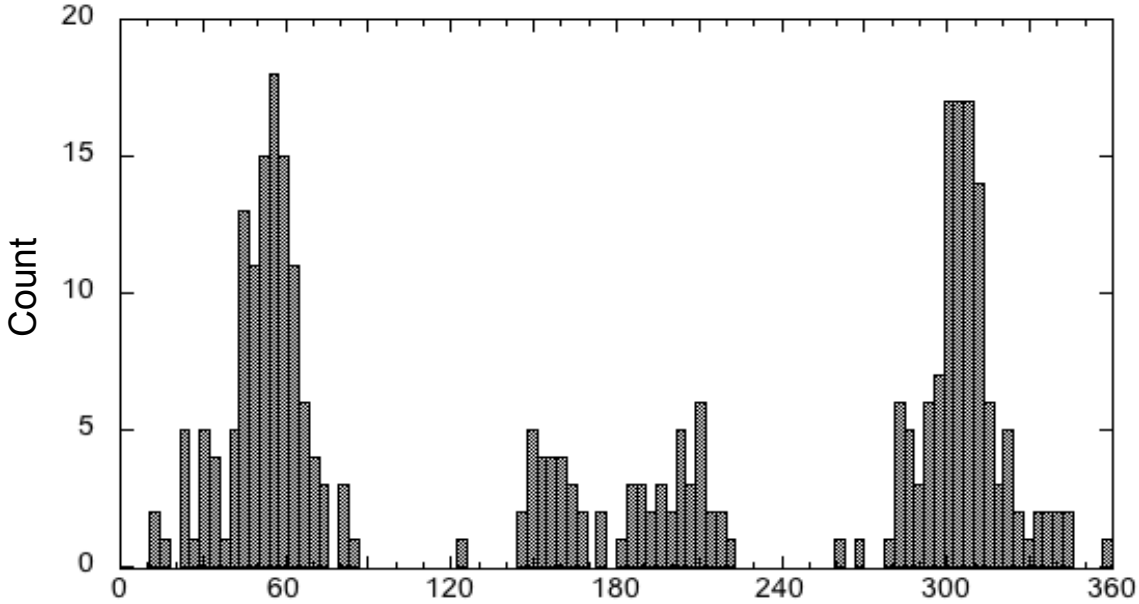


CSD comparison

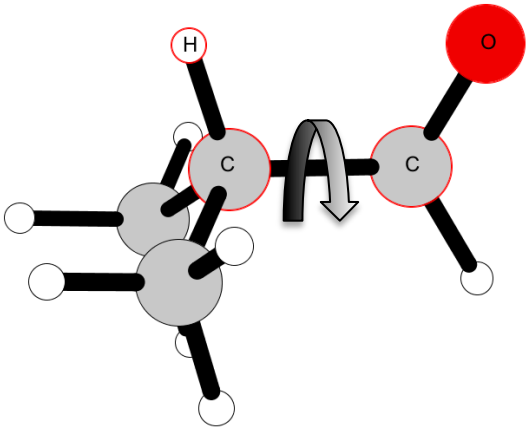


124 hits

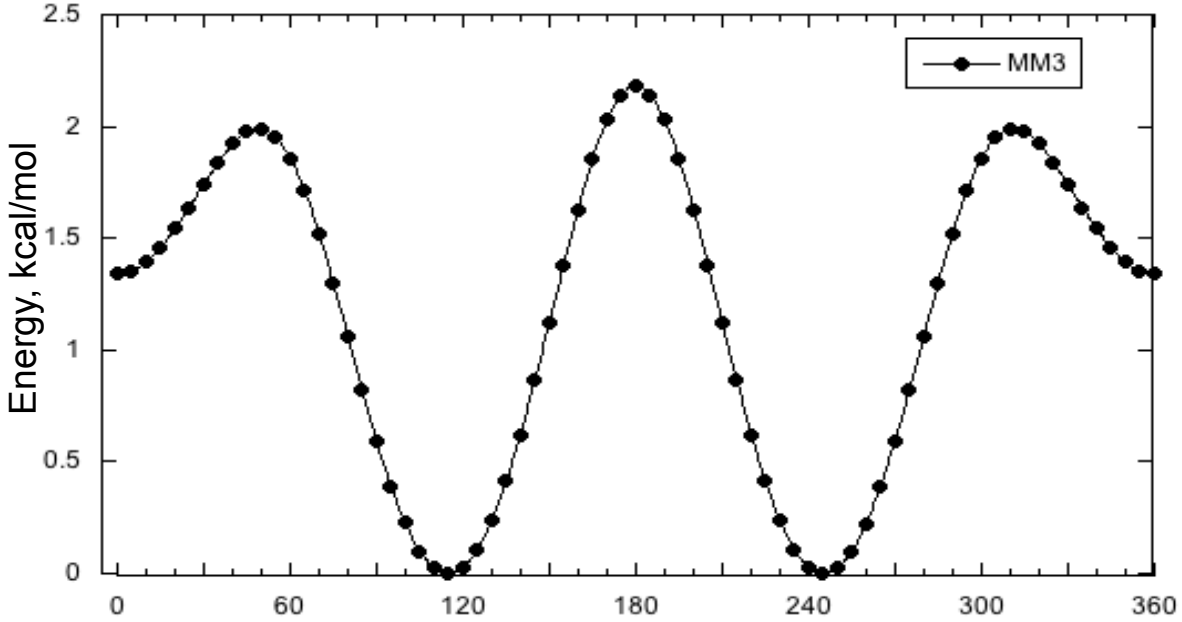
C–P = 1.87 ± 0.01 Å



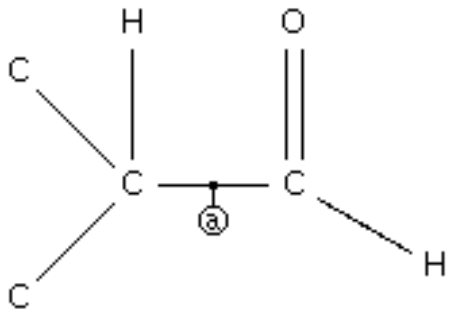
TYPE 1-2:10-1



Φ	E
0.00	1.34
115.00	0.00
245.00	0.00

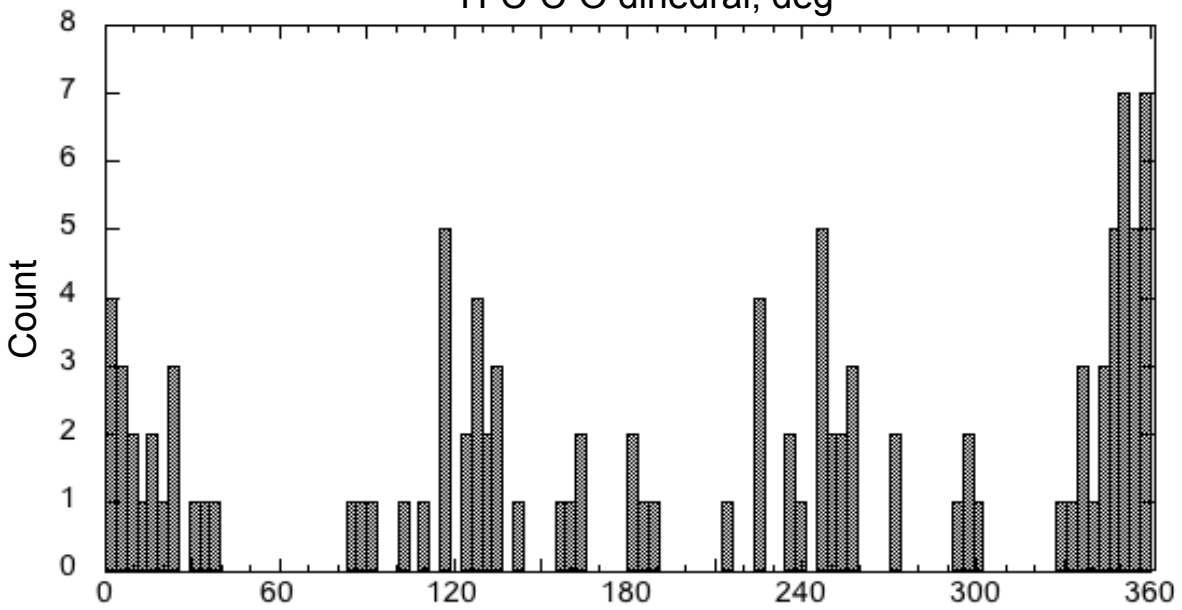


CSD comparison

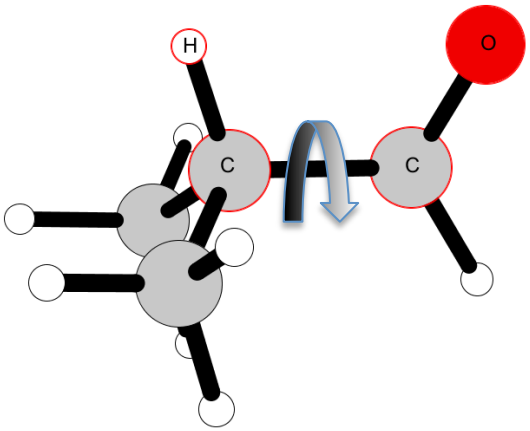


94 hits

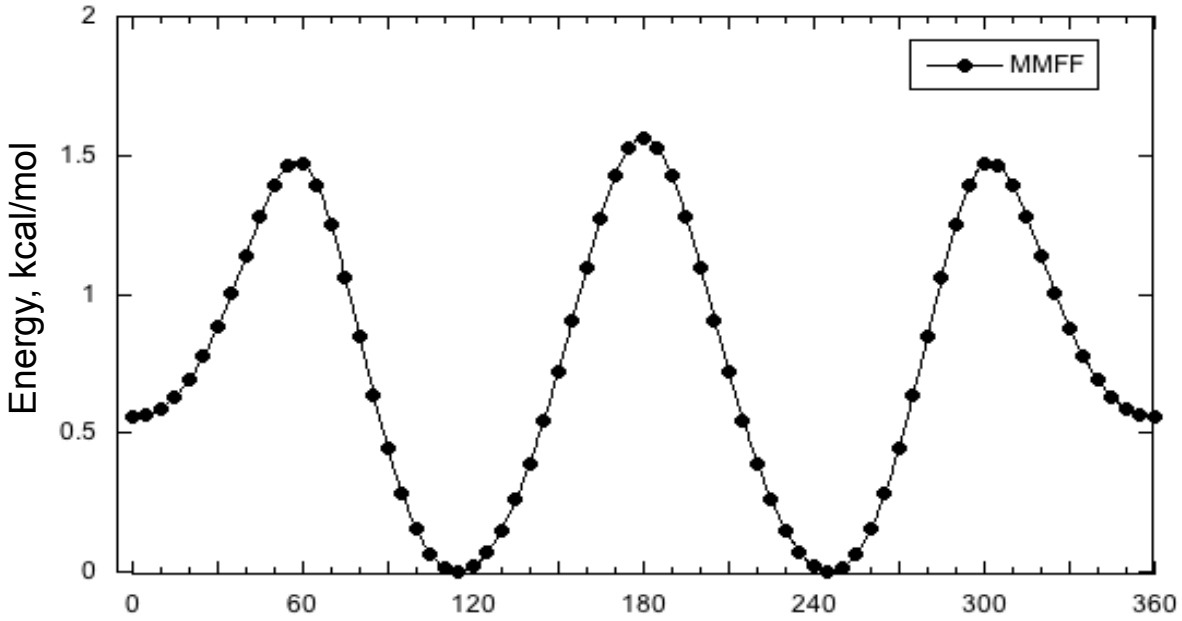
C–C = 1.50 ± 0.02 Å



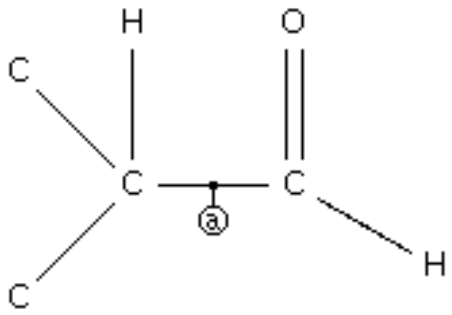
TYPE 1-2:10-1



Φ	E
0.00	0.56
115.00	0.00
245.00	0.00

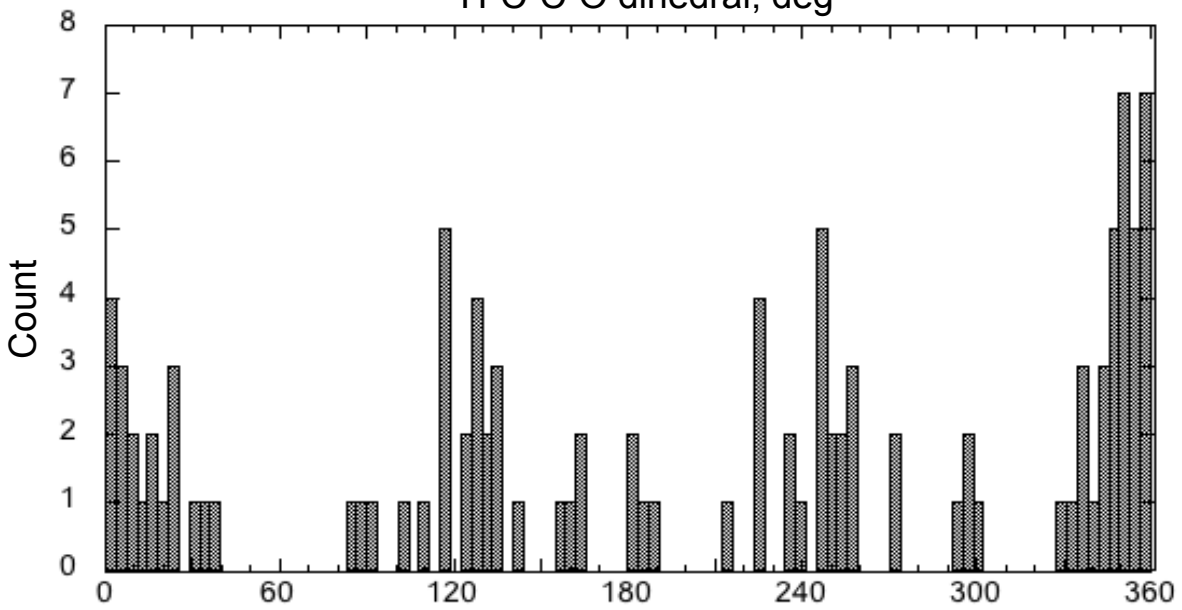


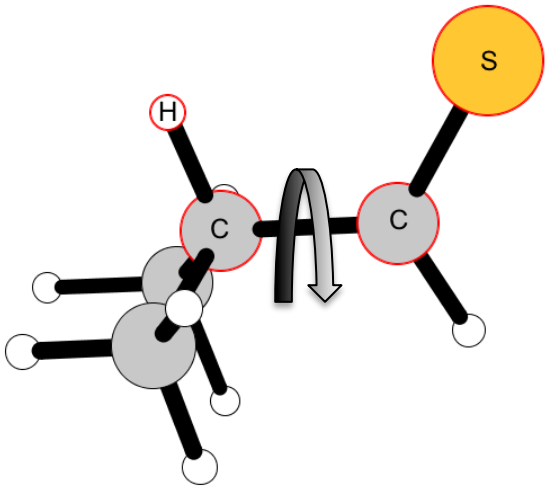
CSD comparison



94 hits

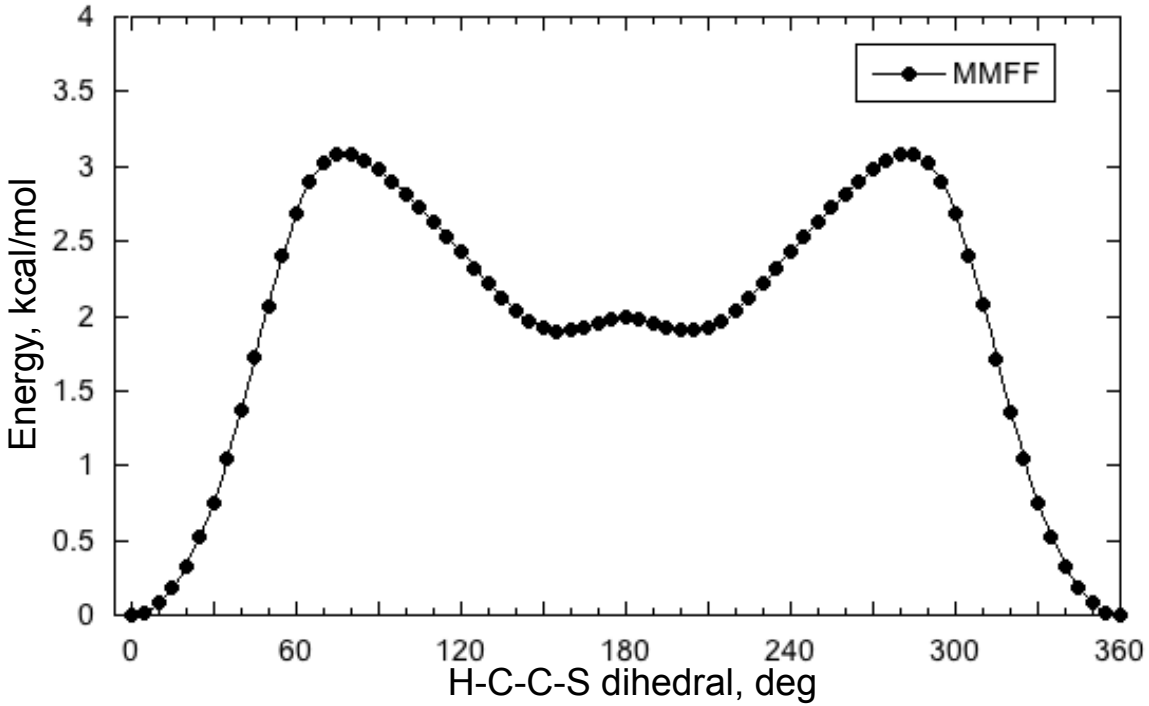
C–C = 1.50 ± 0.02 Å



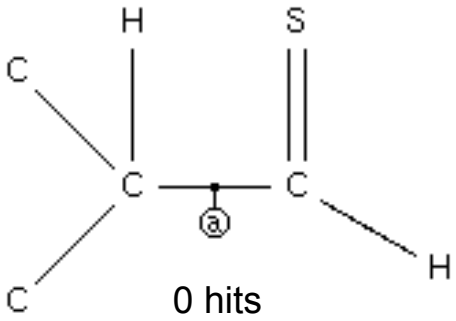


Φ	E
0.0	0.00
155.0	1.90
205.0	1.90

TYPE 1-2:10-2



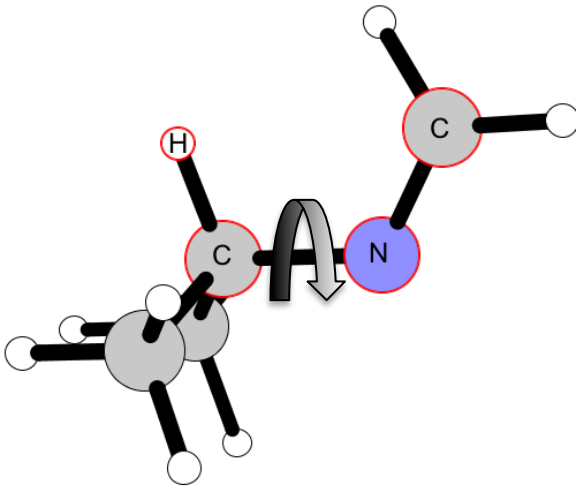
CSD comparison



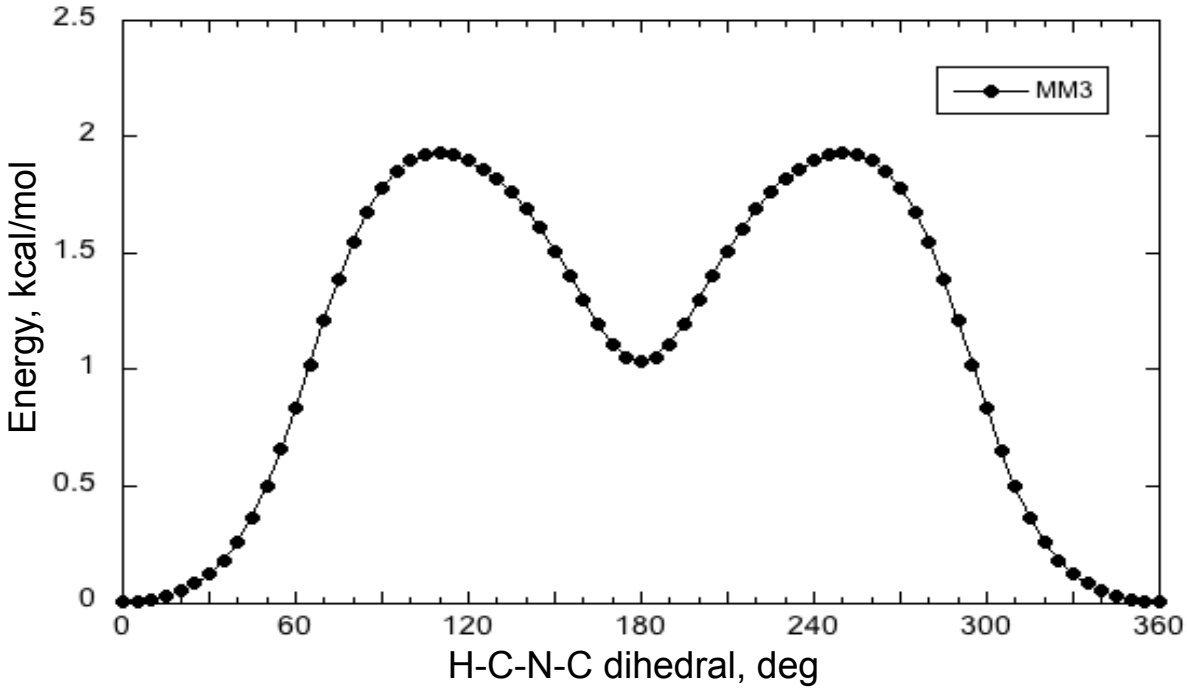
0 hits

NO HITS

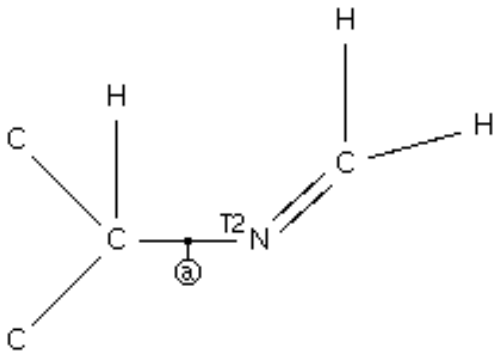
TYPE 1-2:10-3



Φ	E
0.00	0.00
180.00	1.03



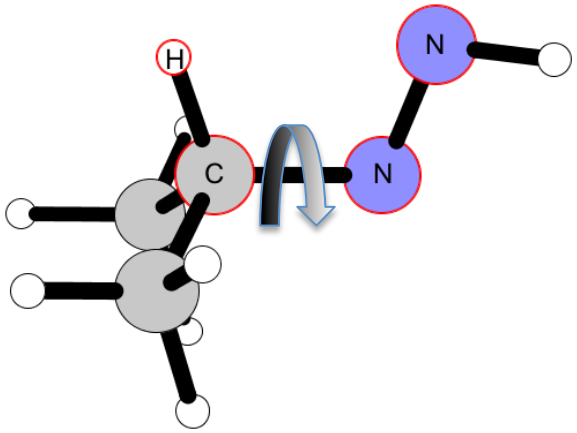
CSD comparison



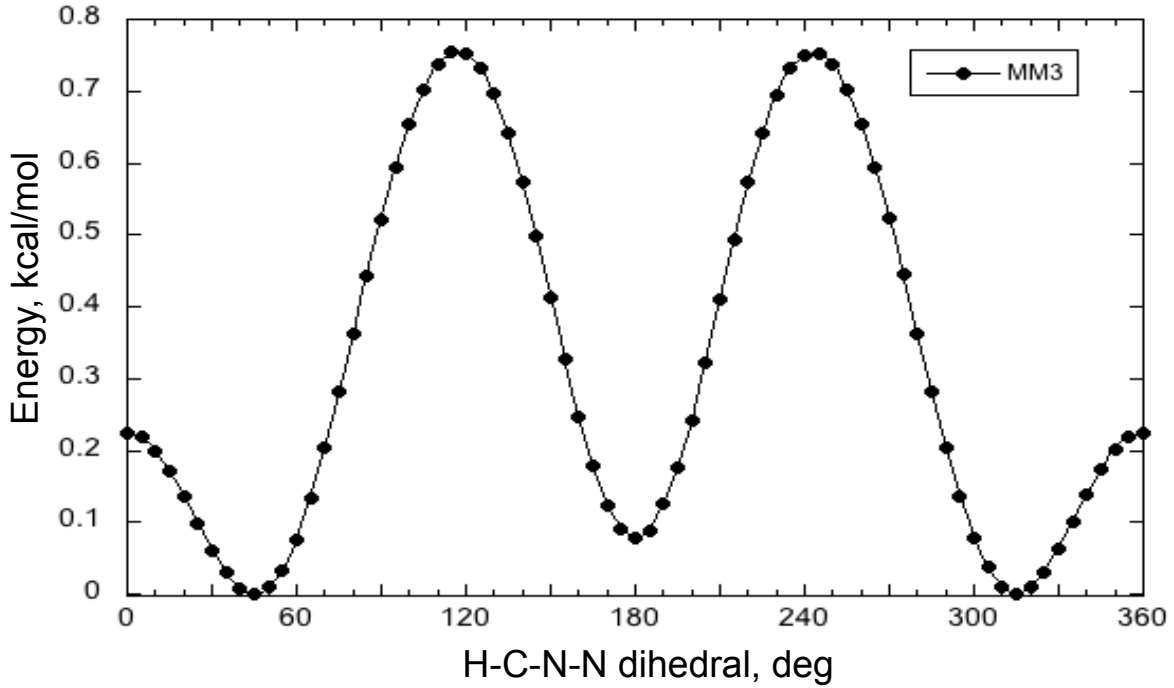
0 hits

NO HITS

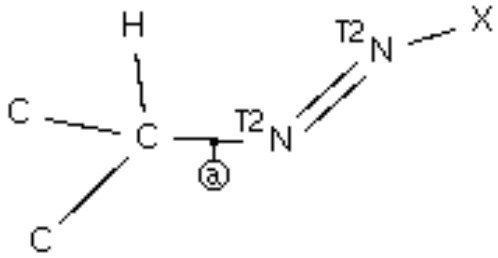
TYPE 1-2:10-4



Φ	E
45.00	0.00
180.00	0.09
315.00	0.00

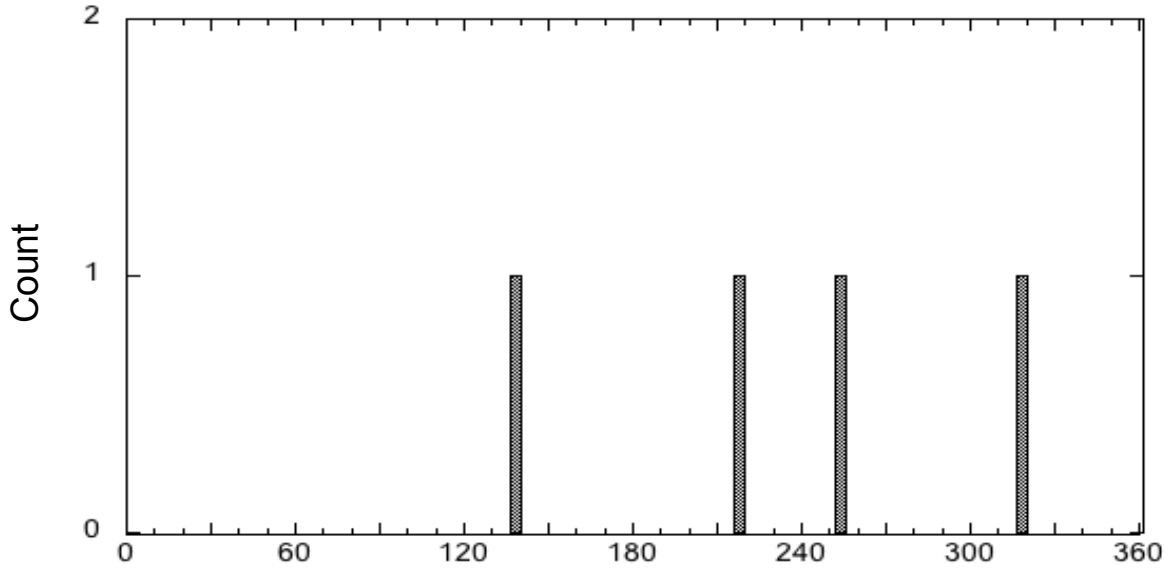


CSD comparison

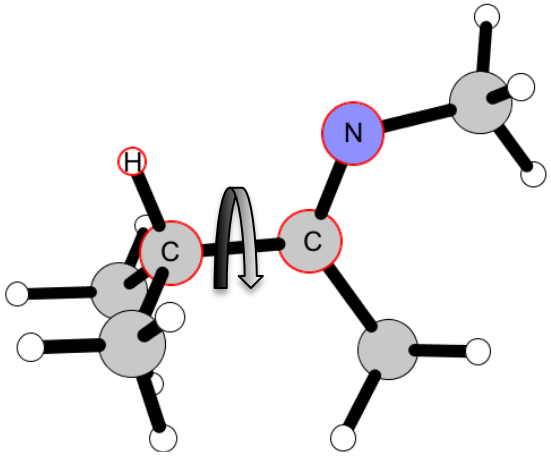


4 hits

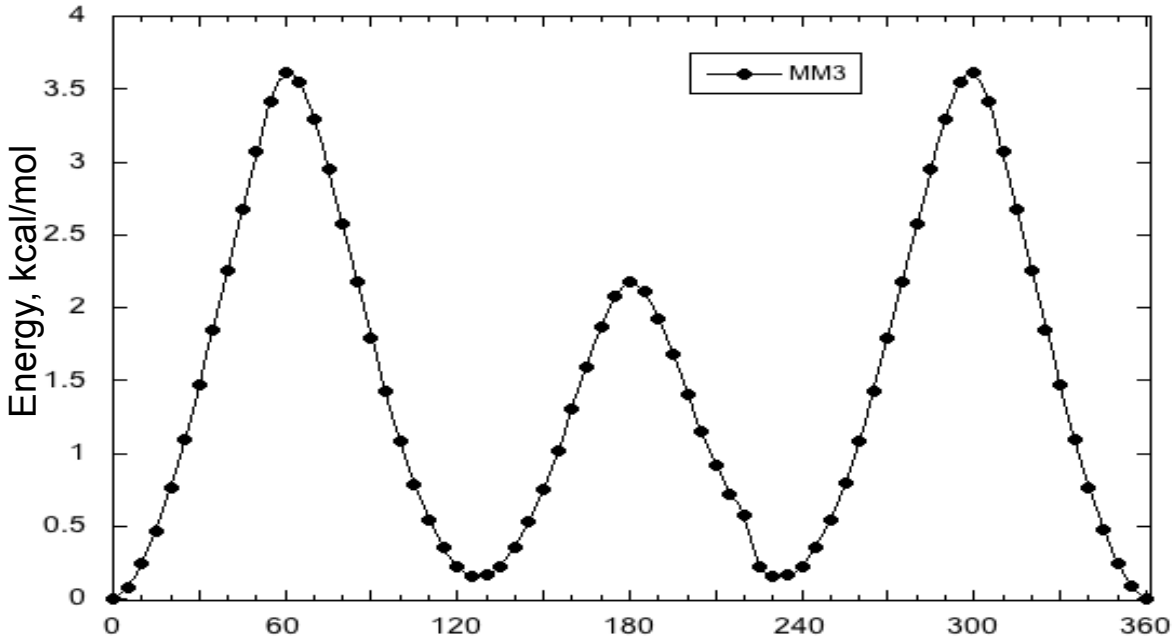
C–N = 1.48 ± 0.01 Å



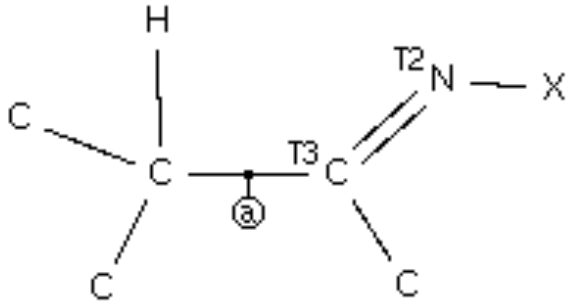
TYPE 1-2:11-1



Φ	E
0.00	0.00
125.00	0.16
230.00	0.16

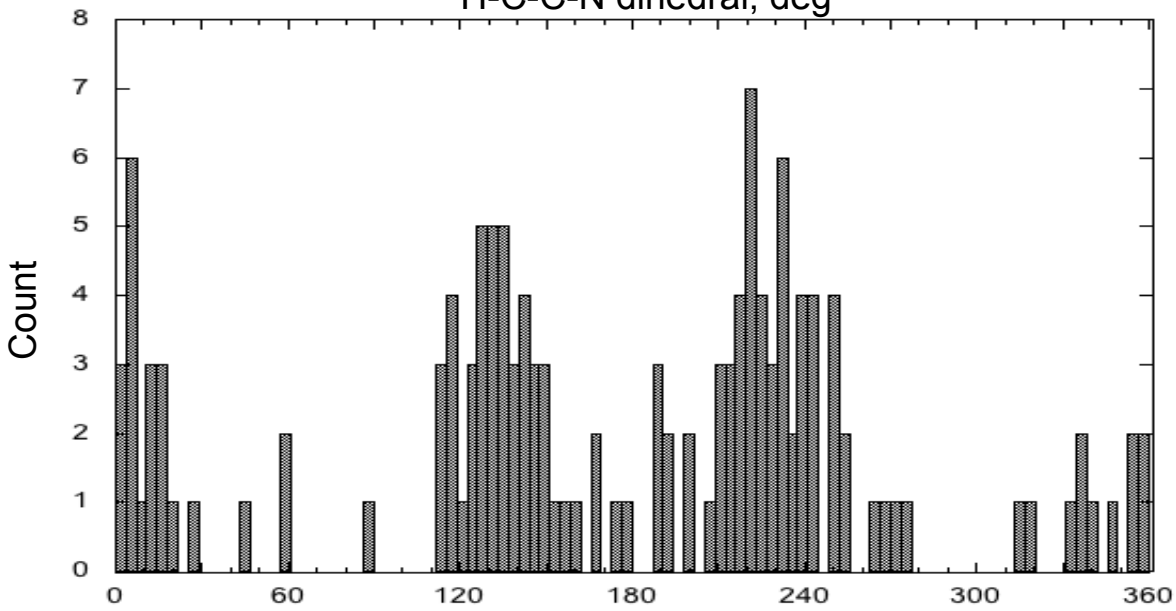


CSD comparison

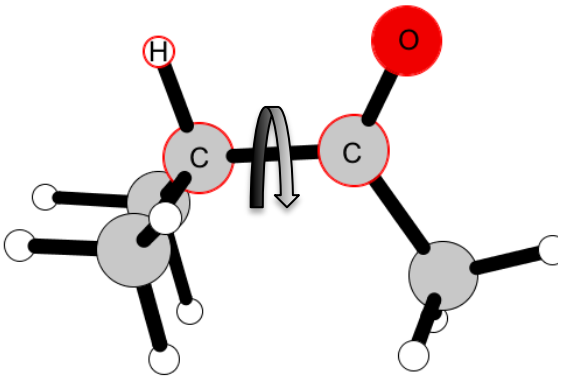


98 hits

C–C = 1.51 ± 0.02 Å

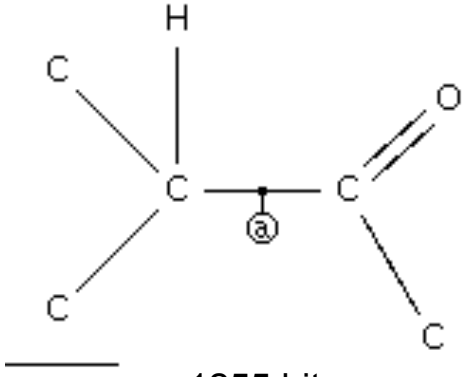


TYPE 1-2:11-2



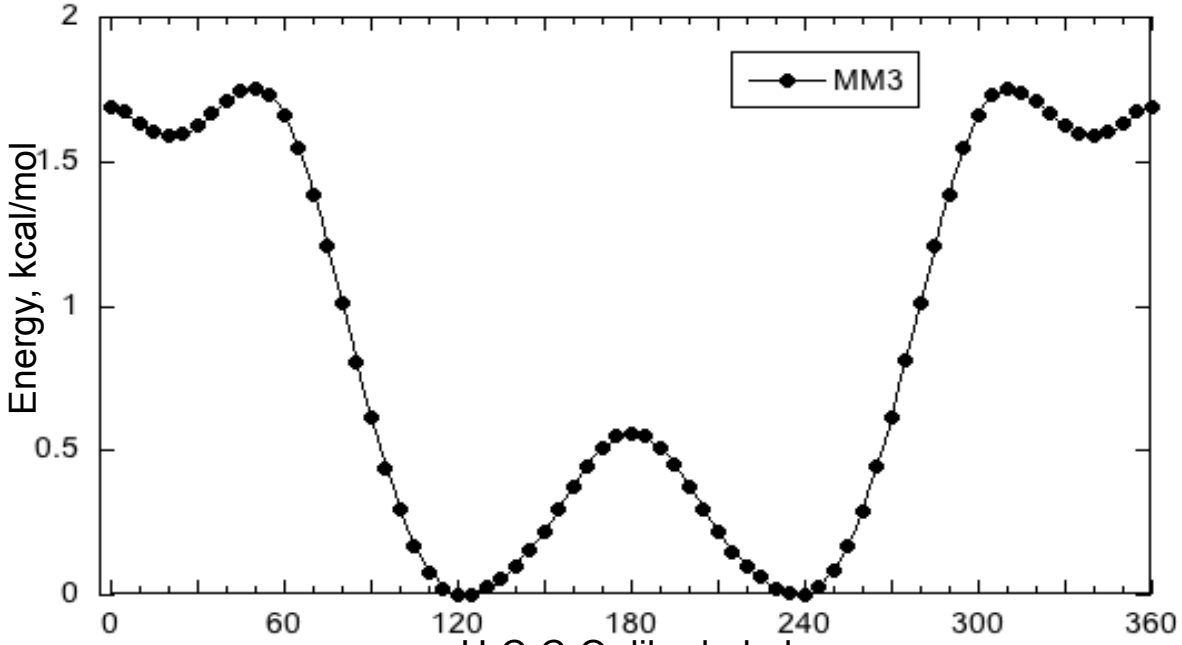
Φ	E
20.0	1.59
120.0	0.00
125.0	0.00
235.0	0.00
240.0	0.00
340.0	1.59

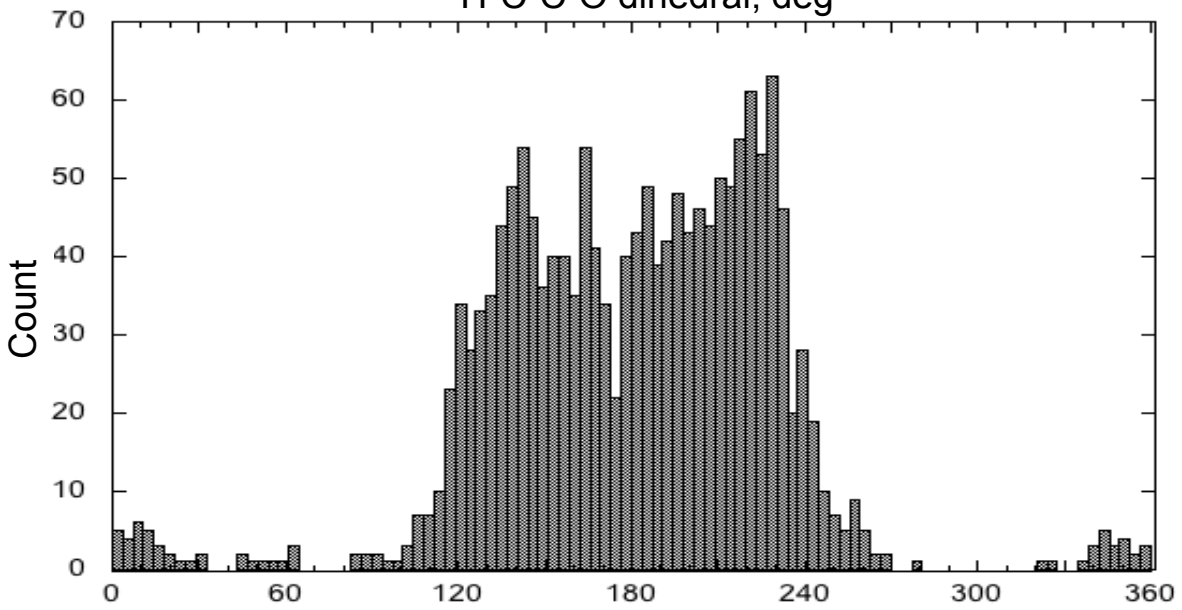
CSD comparison



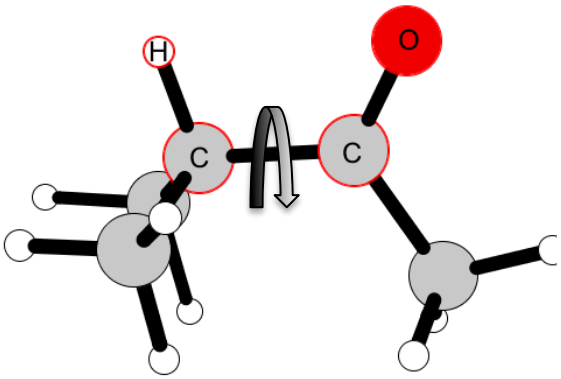
1255 hits

C–C = 1.52 ± 0.02 Å



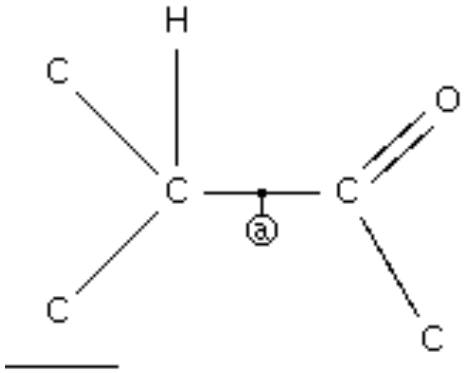


TYPE 1-2:11-2



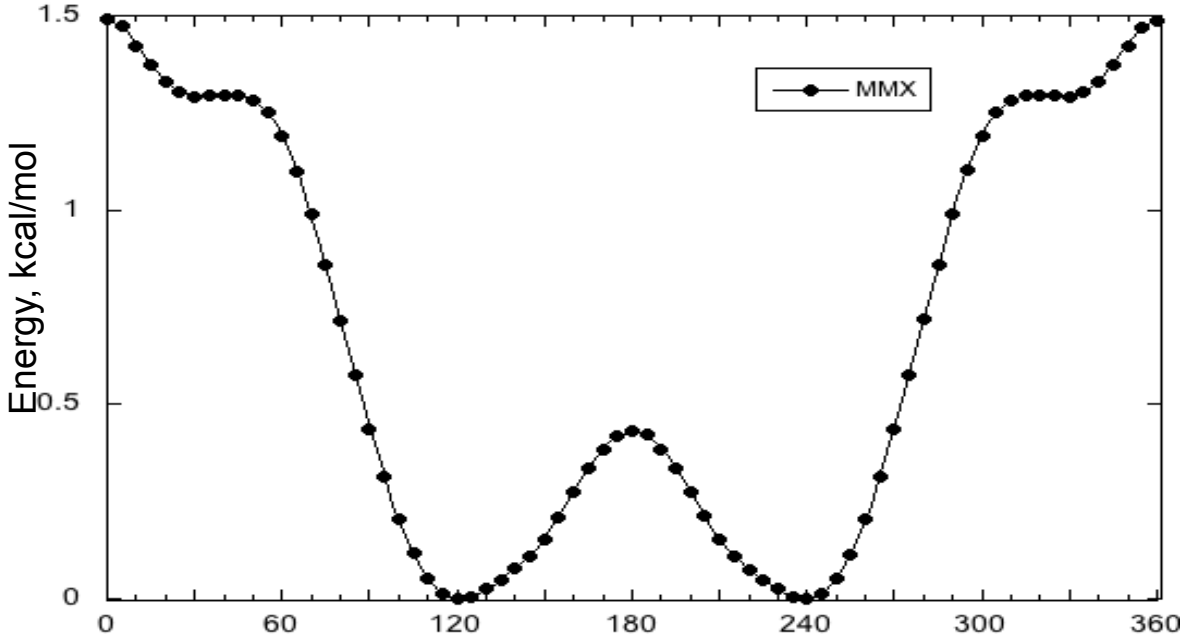
Φ	E
30.00	1.29
120.00	0.00
240.00	0.00
330.00	1.29

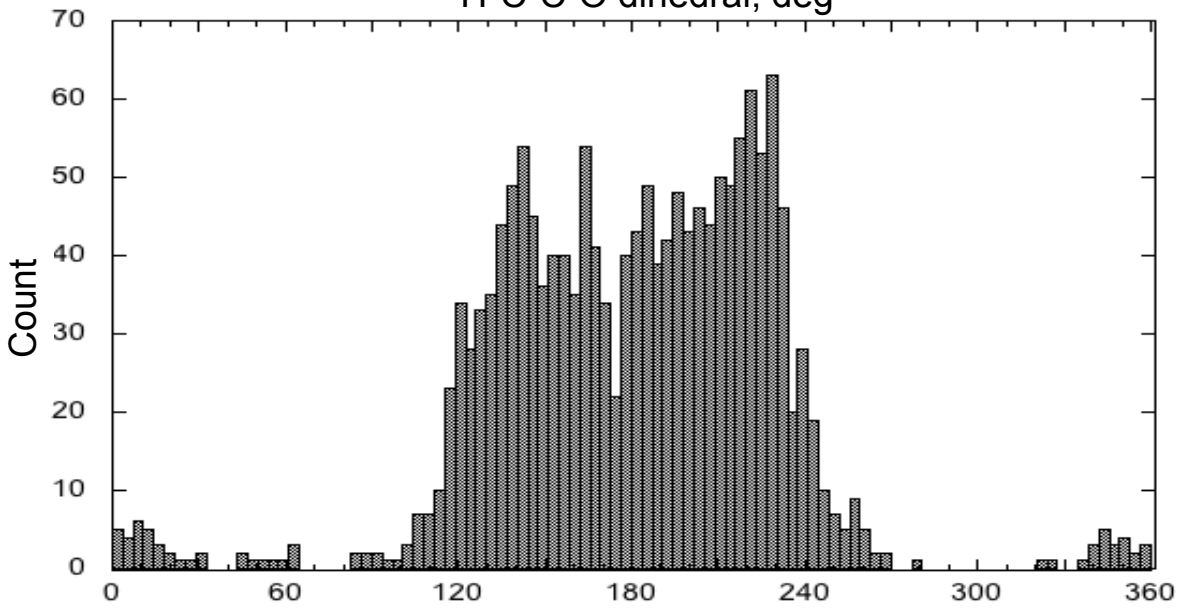
CSD comparison



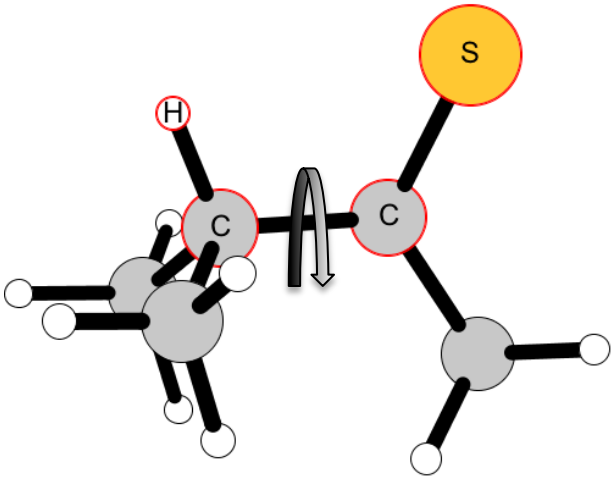
1255 hits

C–C = 1.52 ± 0.02 Å

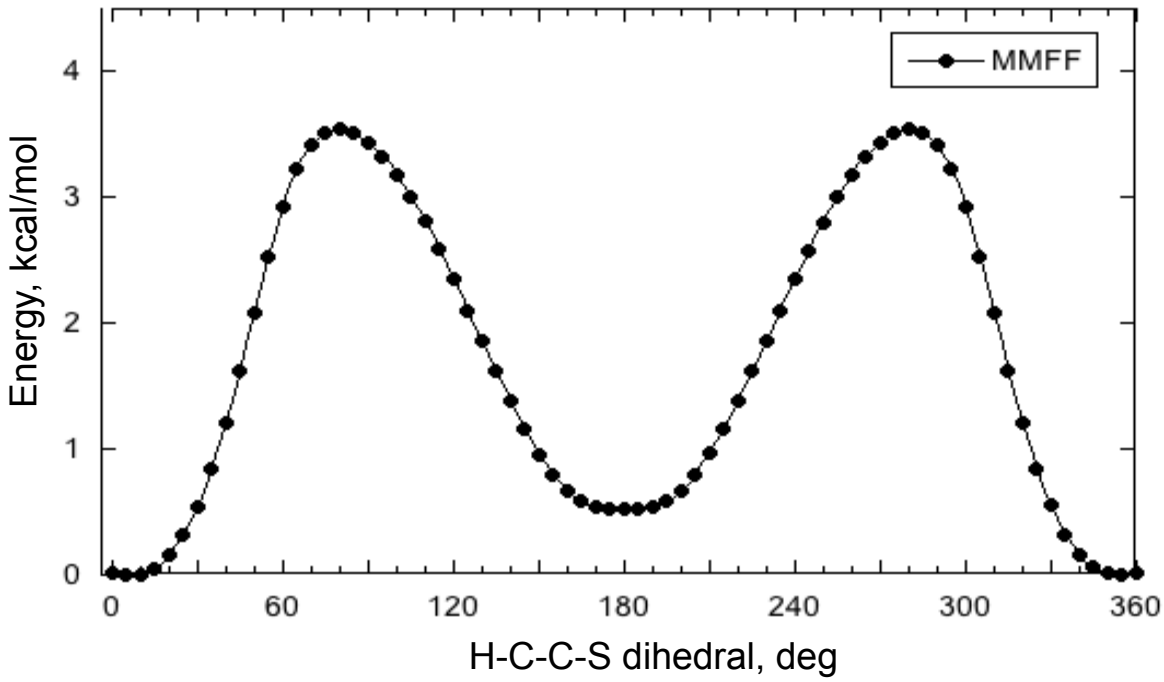




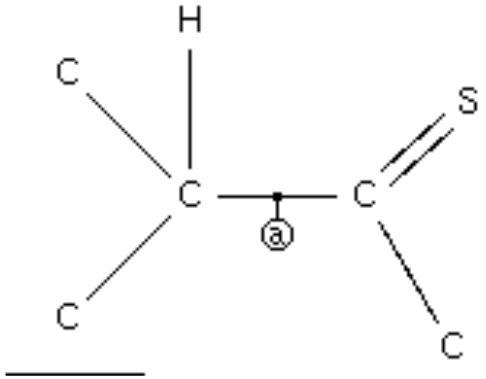
TYPE 1-2:11-3



Φ	E
5.0	0.00
175.0	0.53
180.0	0.53
185.0	0.53
355.0	0.00



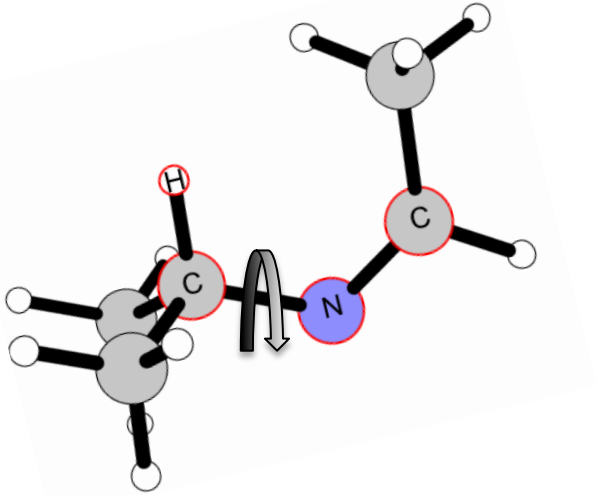
CSD comparison



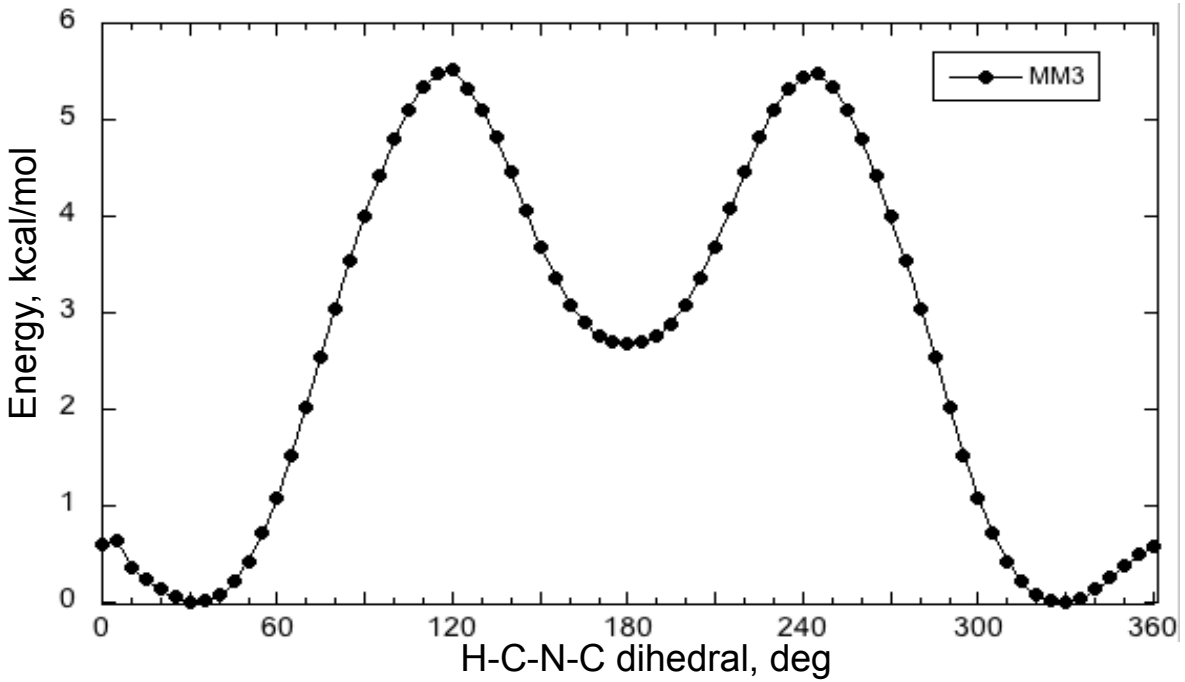
0 hits

NO HITS

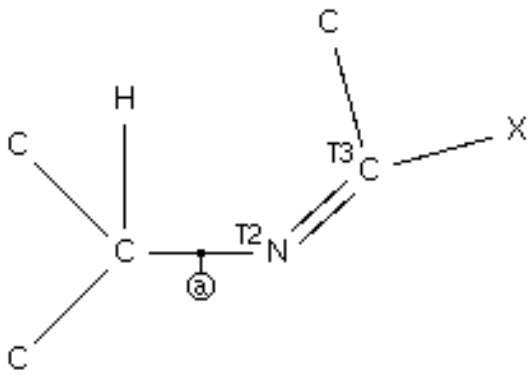
TYPE 1-2:12-1



Φ	E
30.00	0.00
185.00	2.71
330.00	0.00

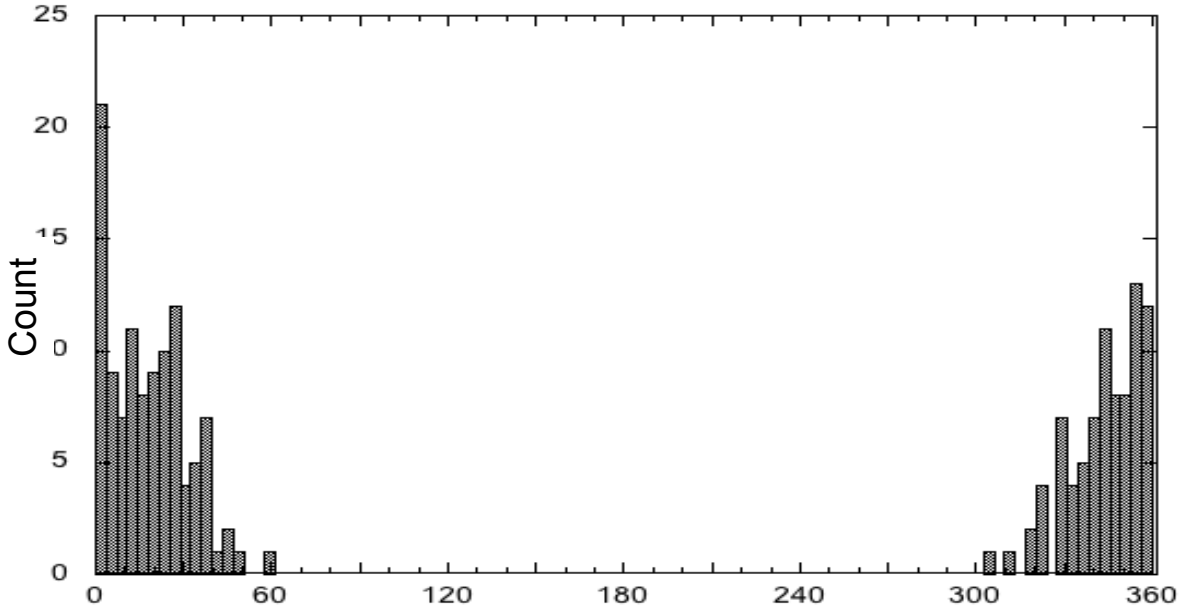


CSD comparison

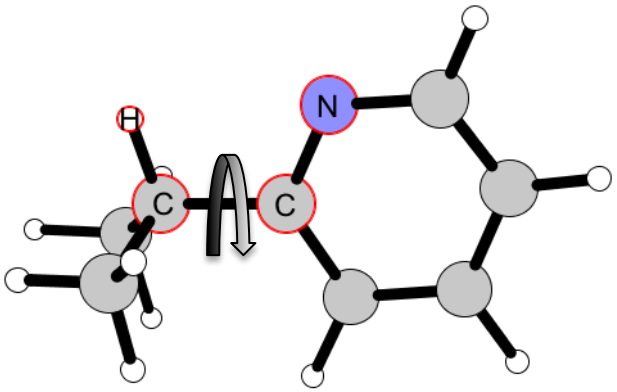


152 hits

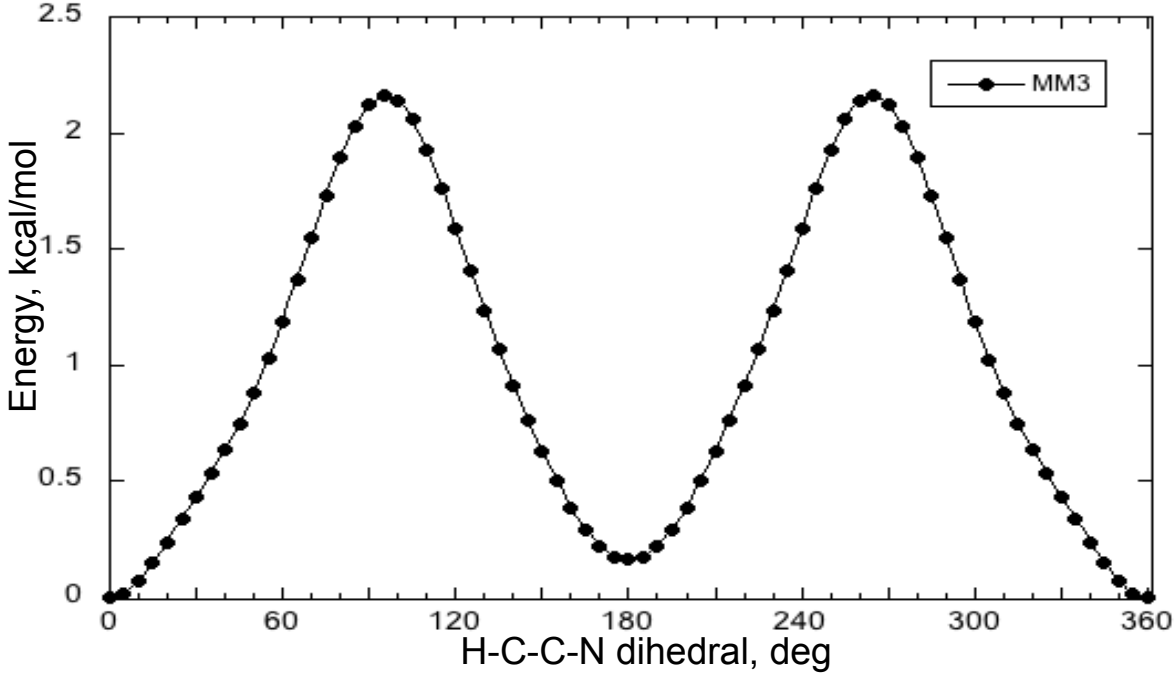
C–N = 1.46 ± 0.01 Å



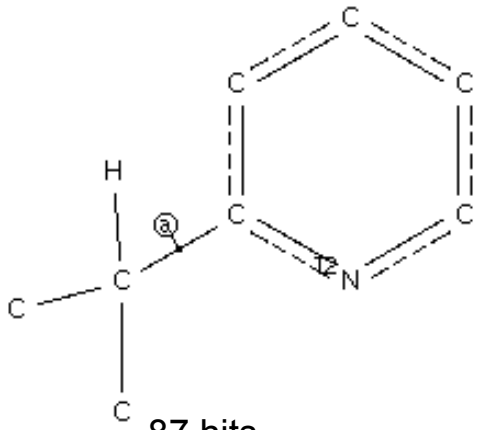
TYPE 1-2:13-1



Φ	E
0.00	0.00
180.00	0.16

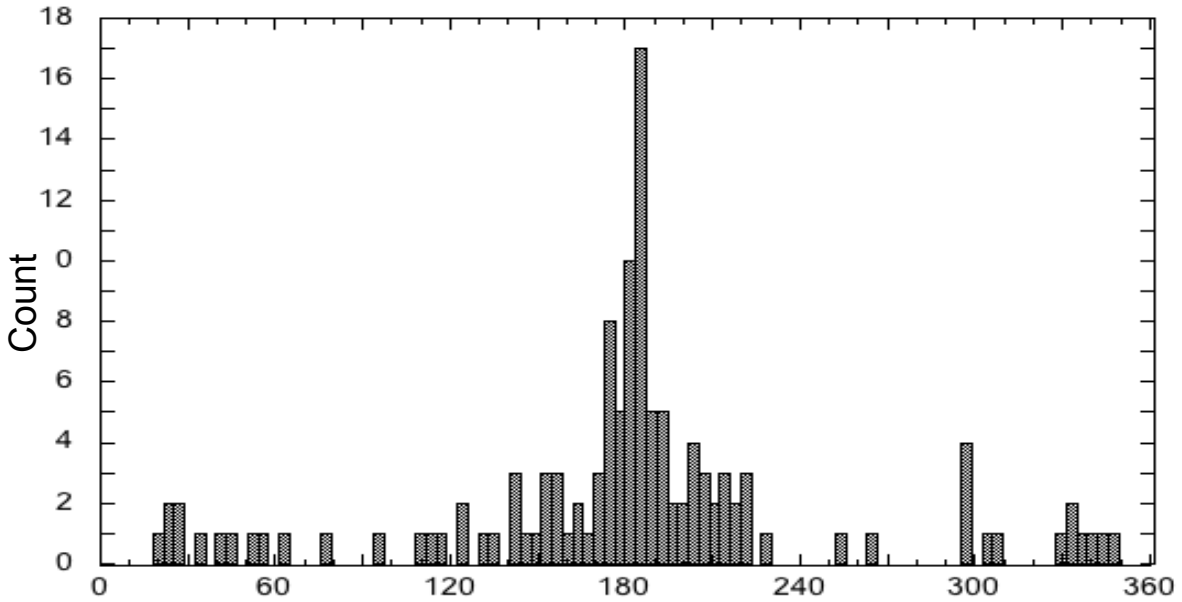


CSD comparison

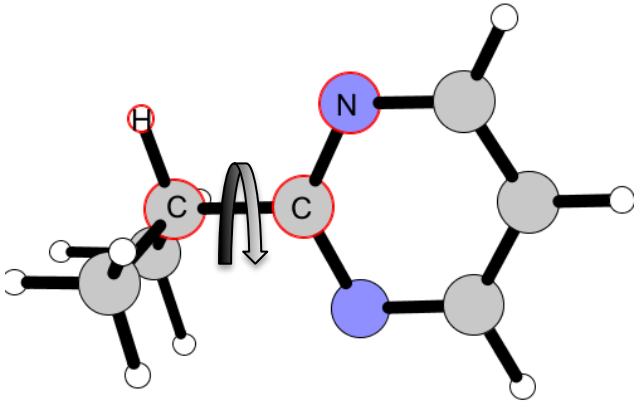


87 hits

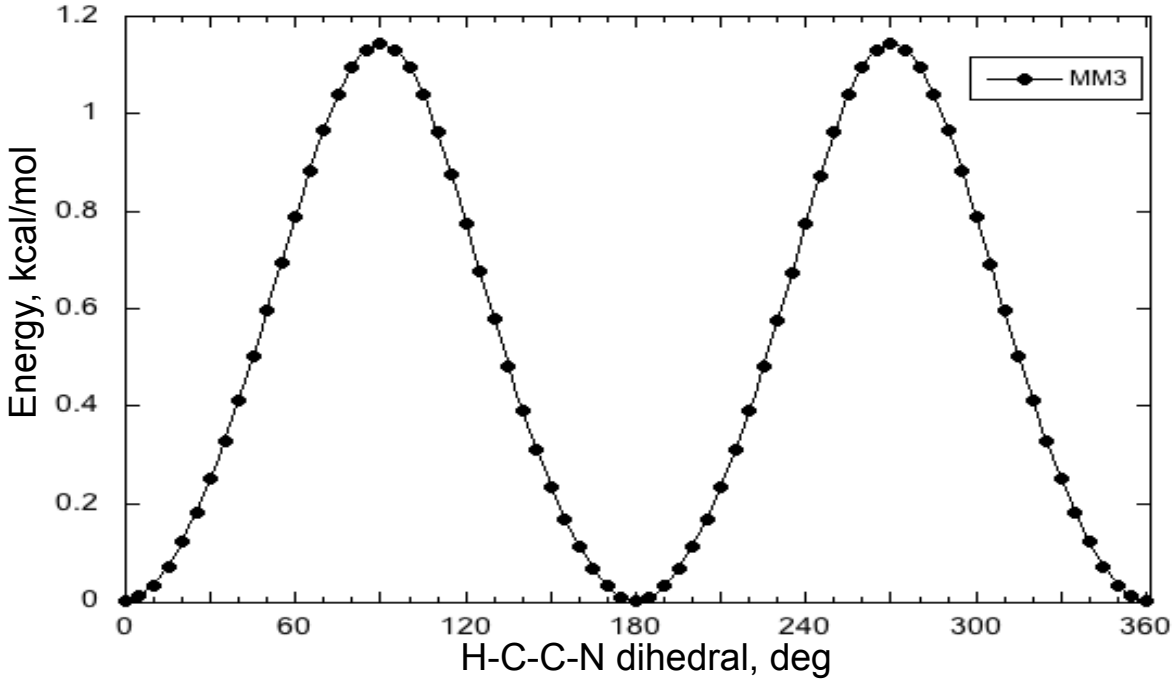
C–C = 1.51 ± 0.02 Å



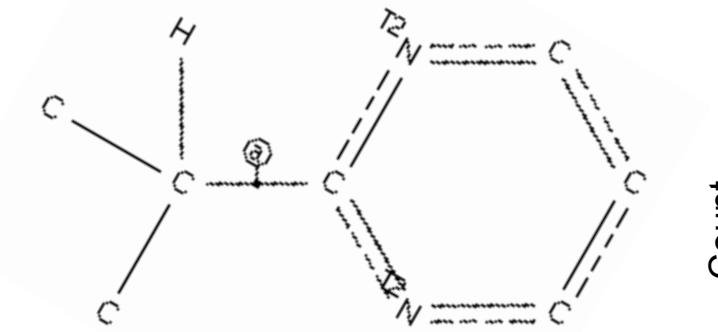
TYPE 1-2:13-2



Φ	E
0.00	0.00
180.00	0.00

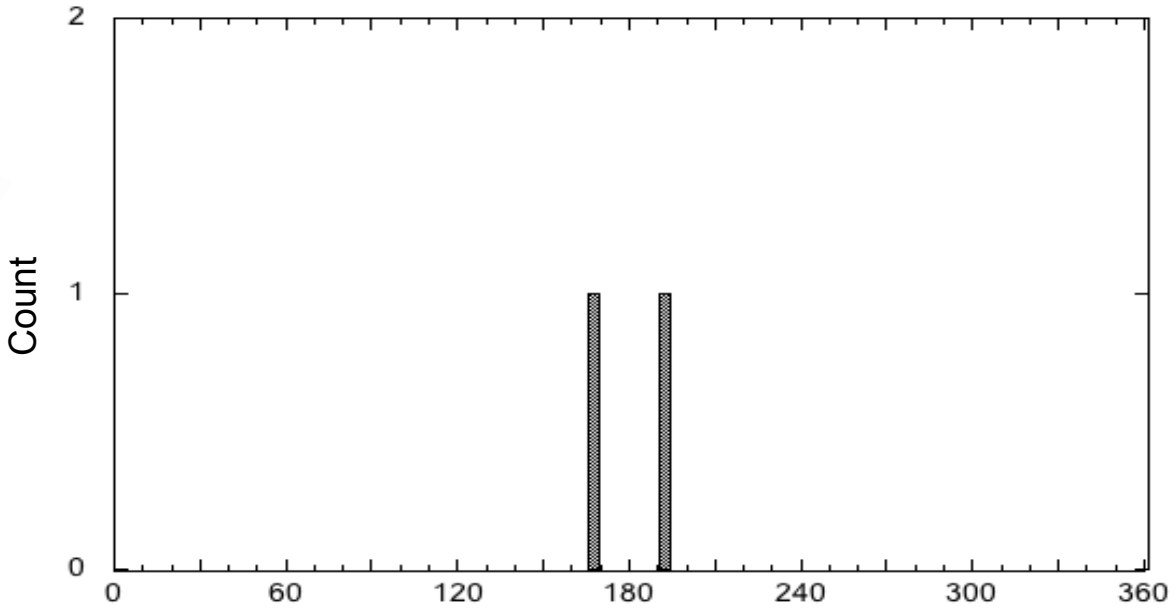


CSD comparison

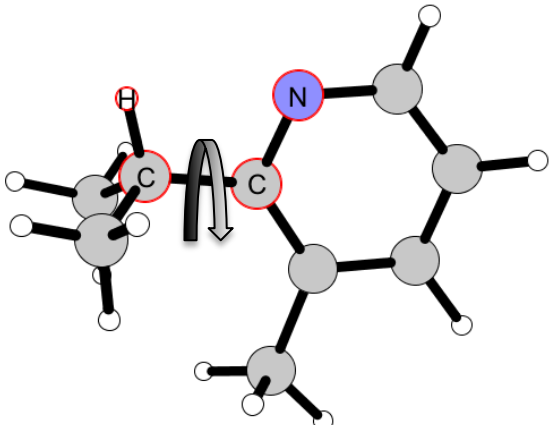


2 hits

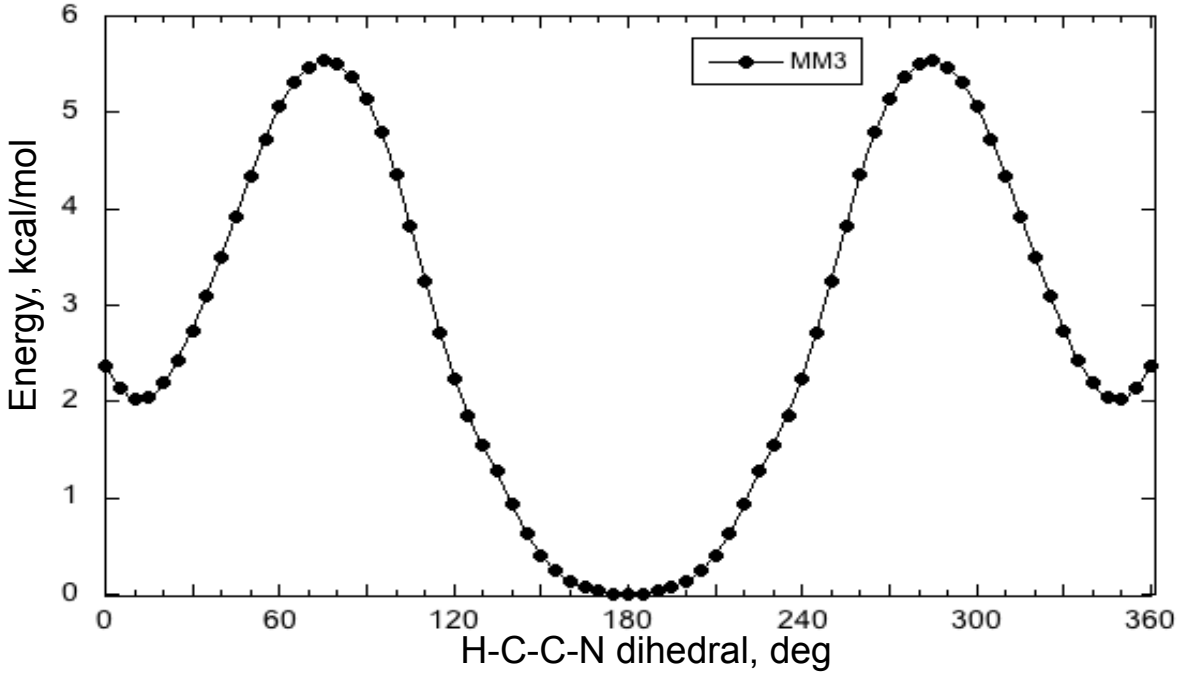
C–C = 1.50 ± 0.02 Å



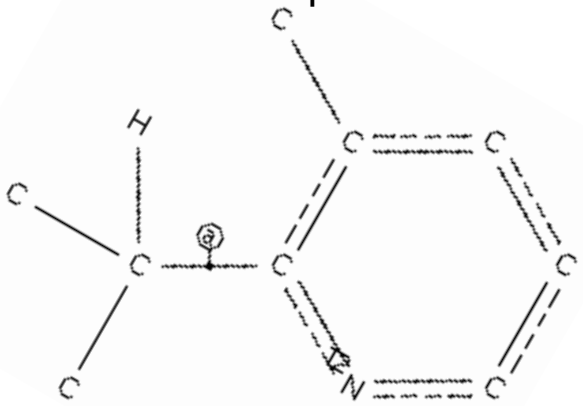
TYPE 1-2:13-3



Φ	E
10.00	2.03
180.00	0.00
350.00	2.03

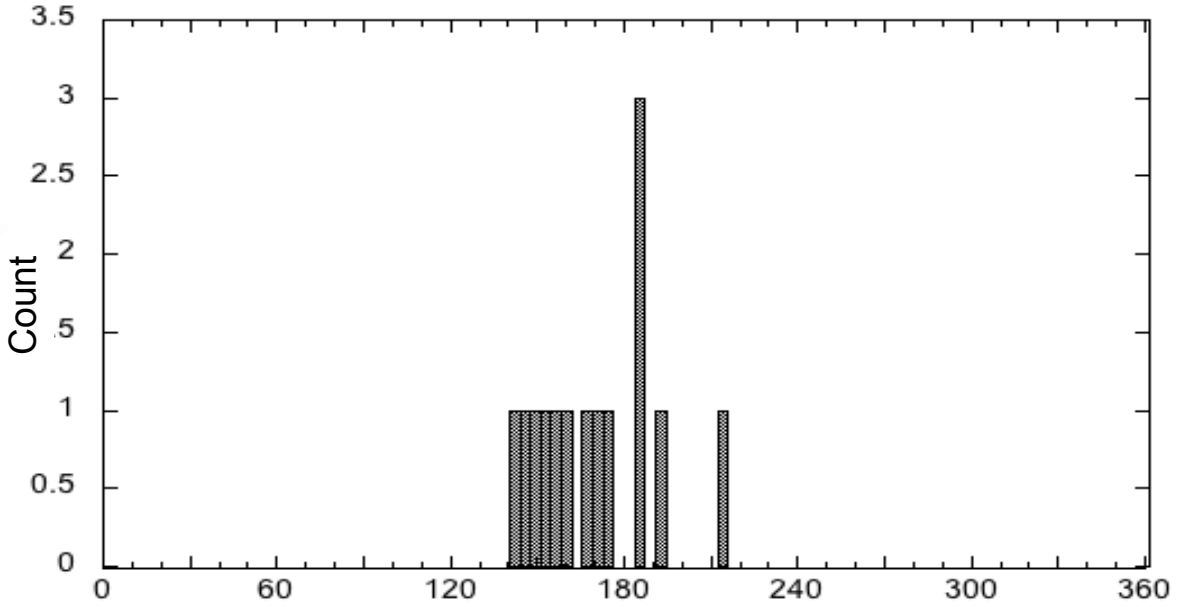


CSD comparison

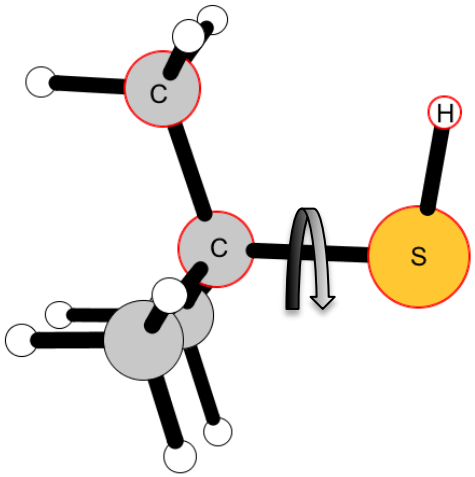


10 hits

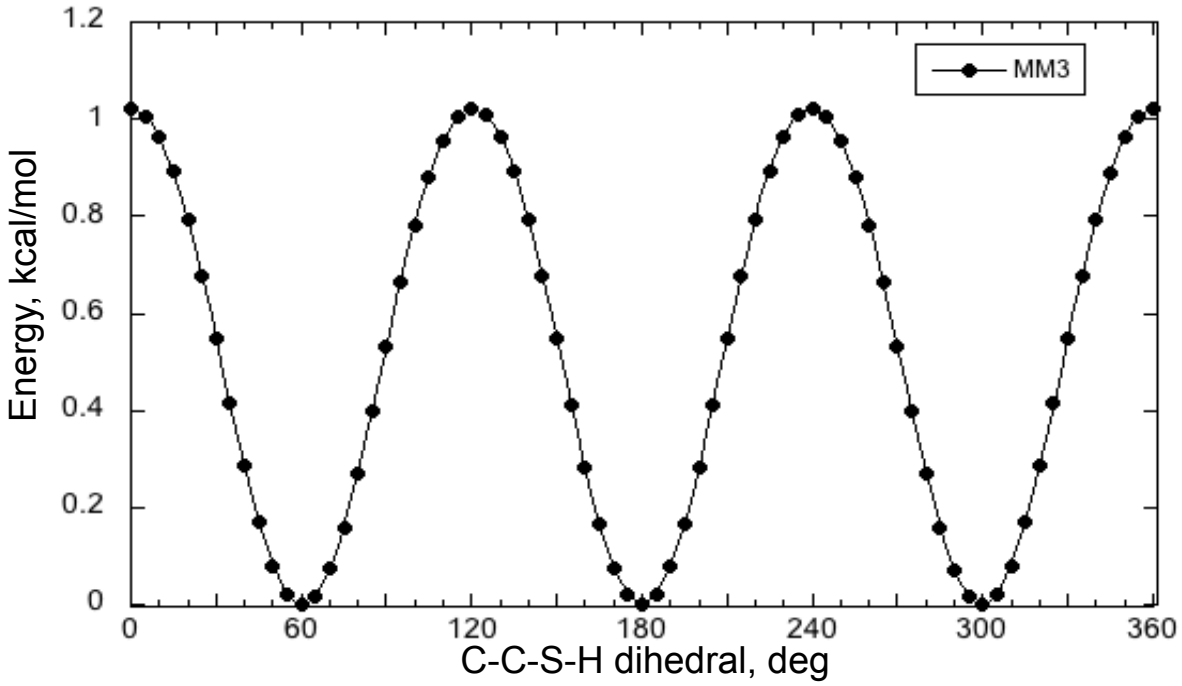
C–C = 1.50 ± 0.01 Å



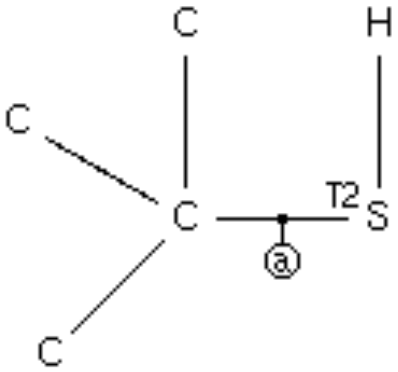
TYPE 1-3:5-4



Φ	E
60.00	0.00
180.00	0.00
300.00	0.00

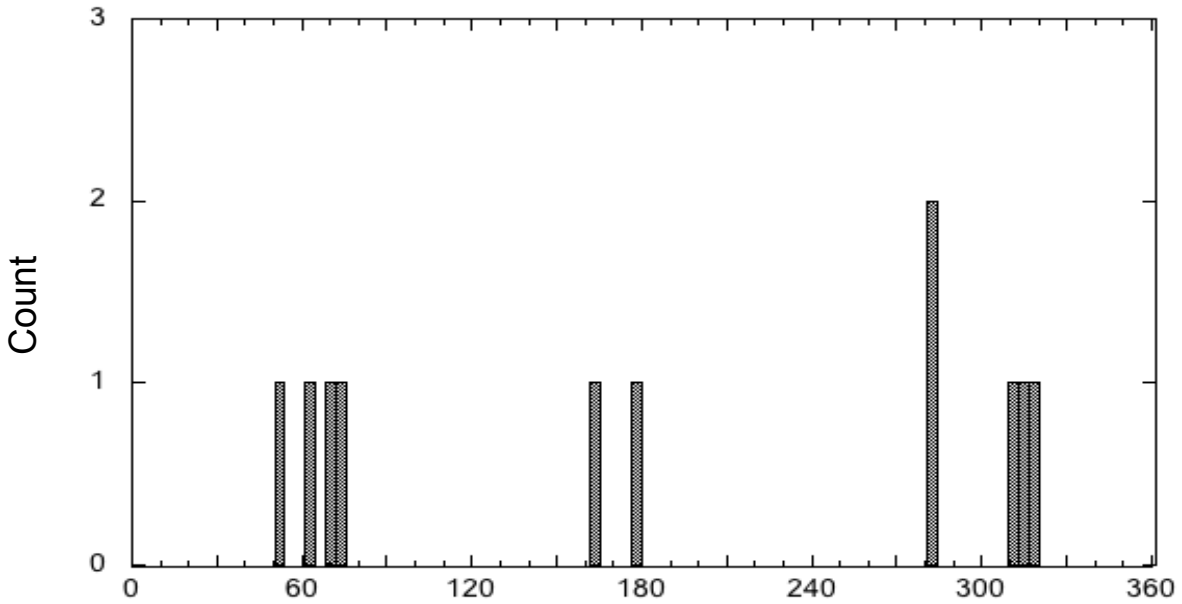


CSD comparison

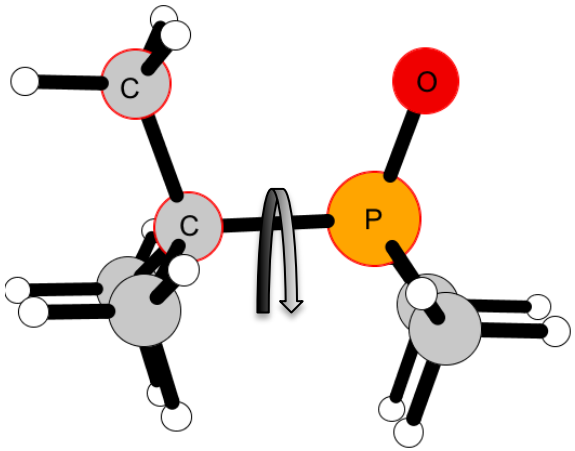


8 hits

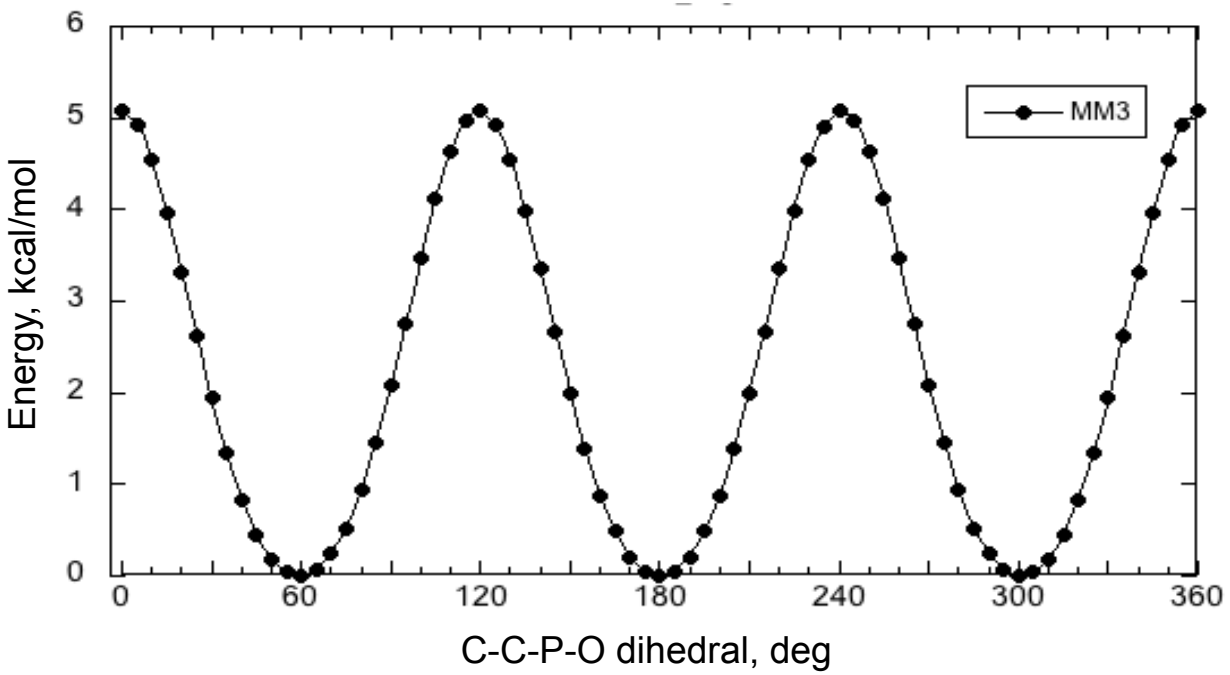
C–S = 1.85 ± 0.01 Å



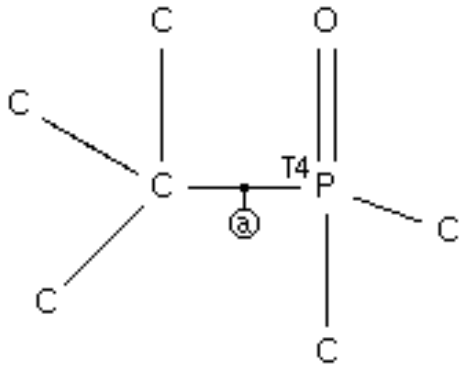
TYPE 1-3:8-1



Φ	E
60.0	0.00
180.0	0.00
300.0	0.00

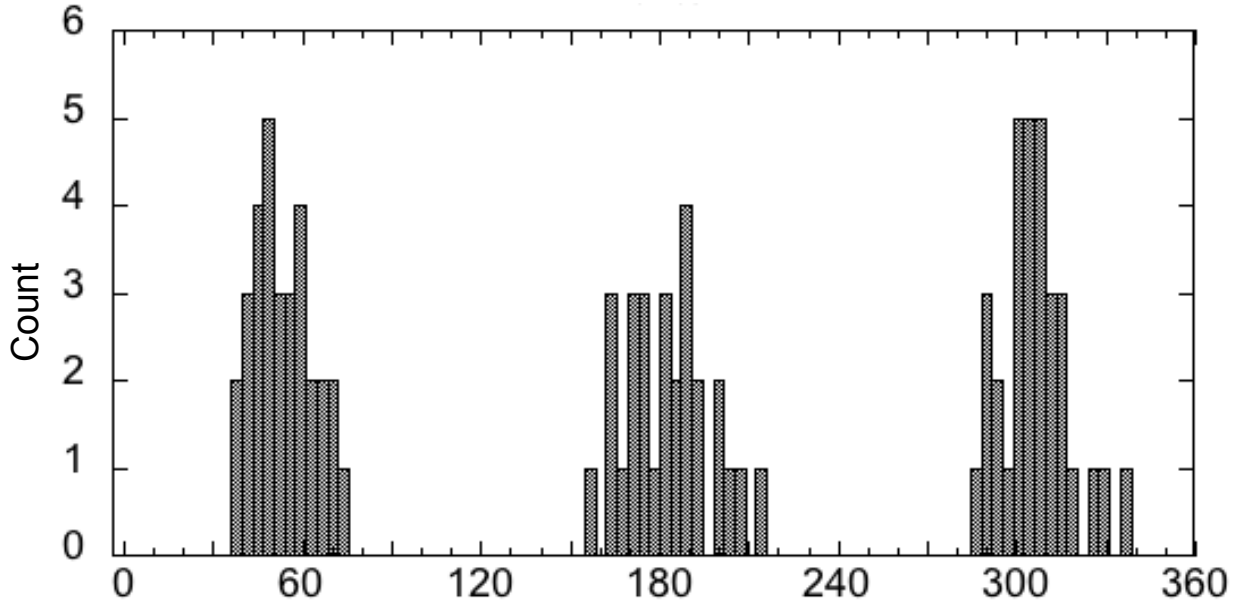


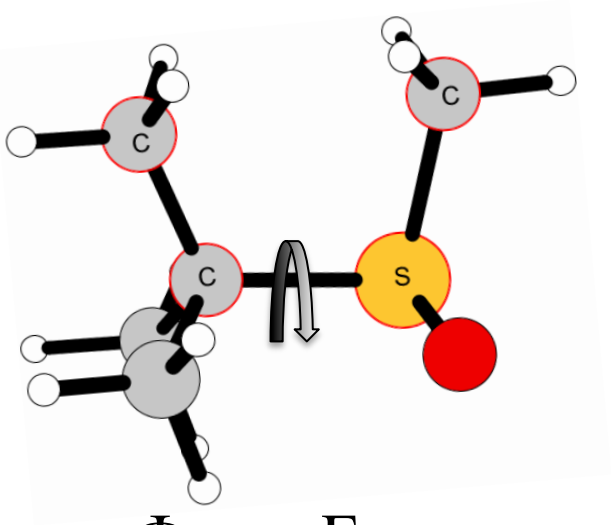
CSD comparison



52 hits

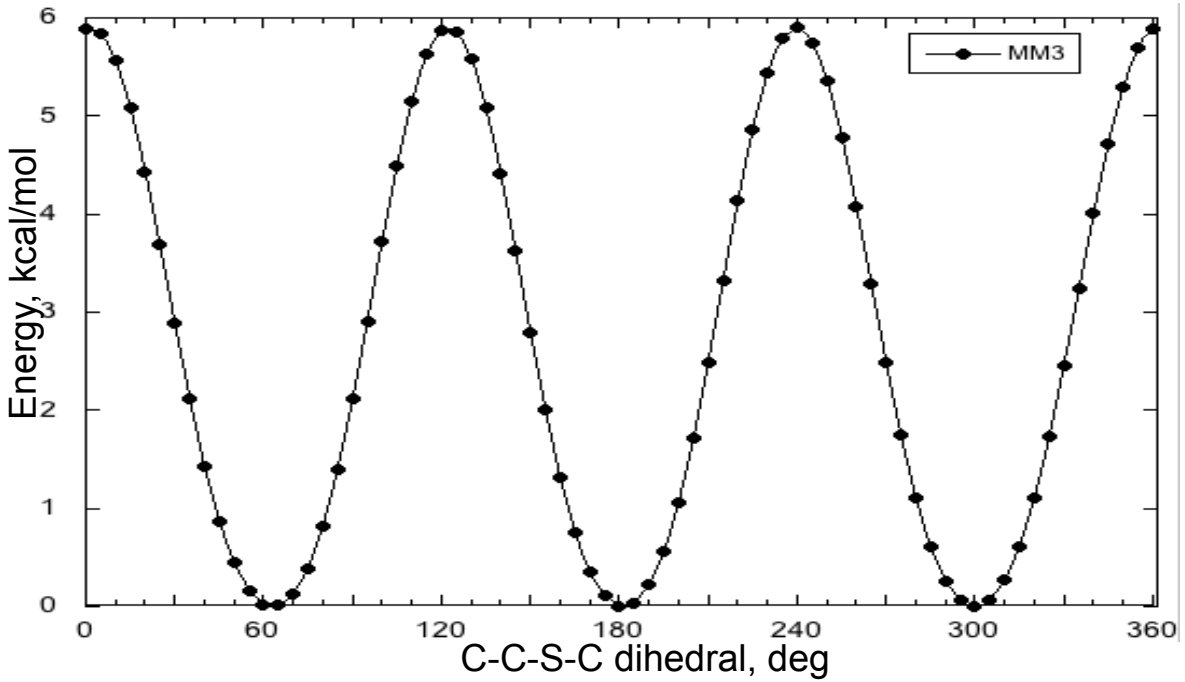
C–P = 1.85 ± 0.02 Å



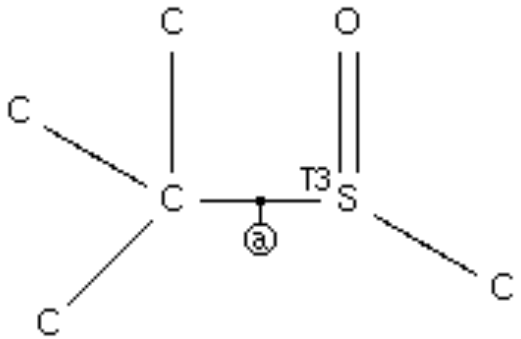


TYPE 1-3:8-2

Φ	E
65.00	0.01
180.00	0.00
300.00	0.00

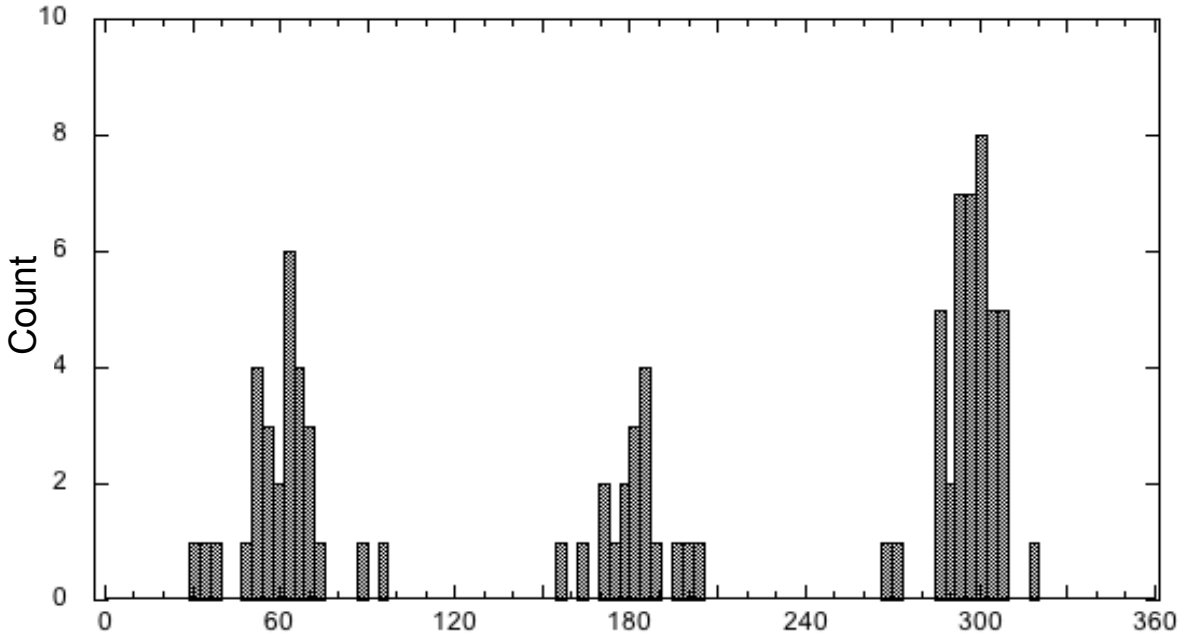


CSD comparison

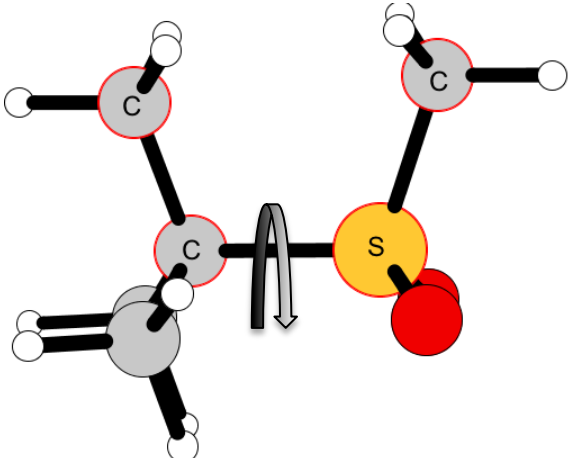


79 hits

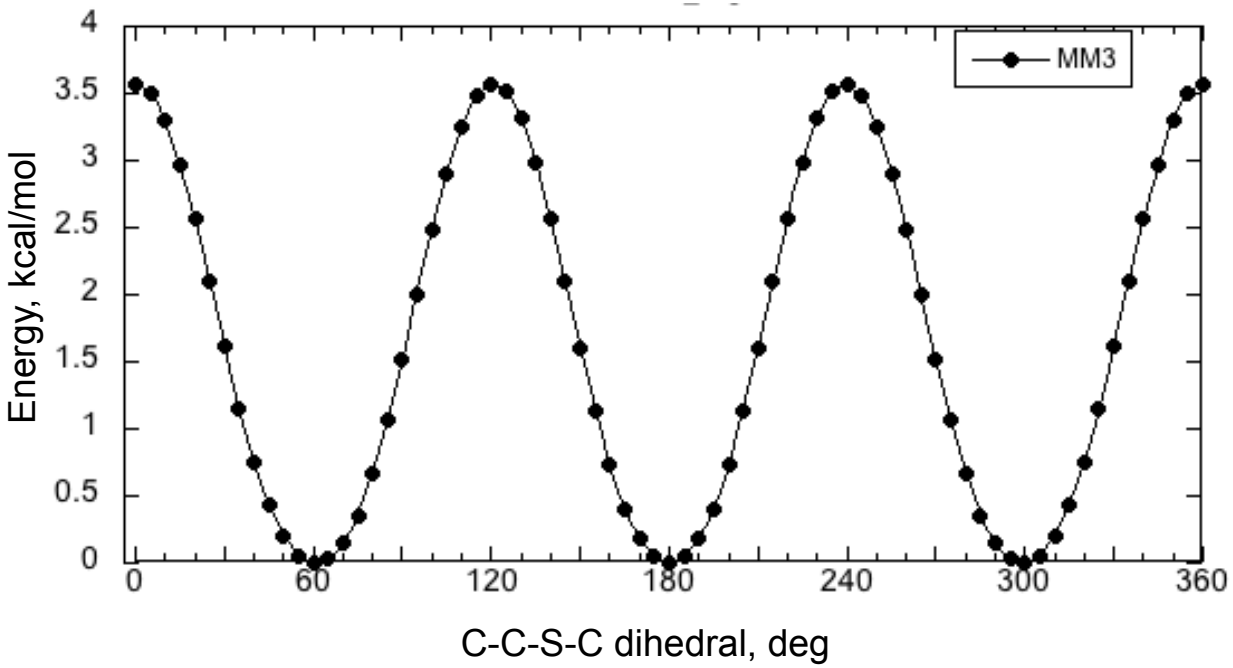
C–S = 1.86 ± 0.02 Å



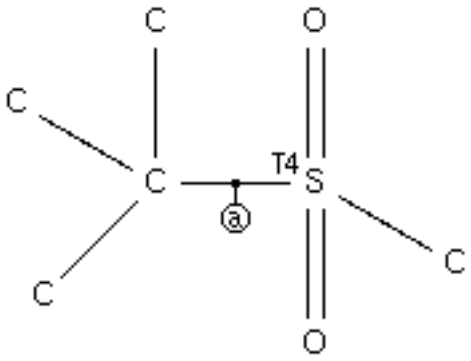
TYPE 1-3:8-4

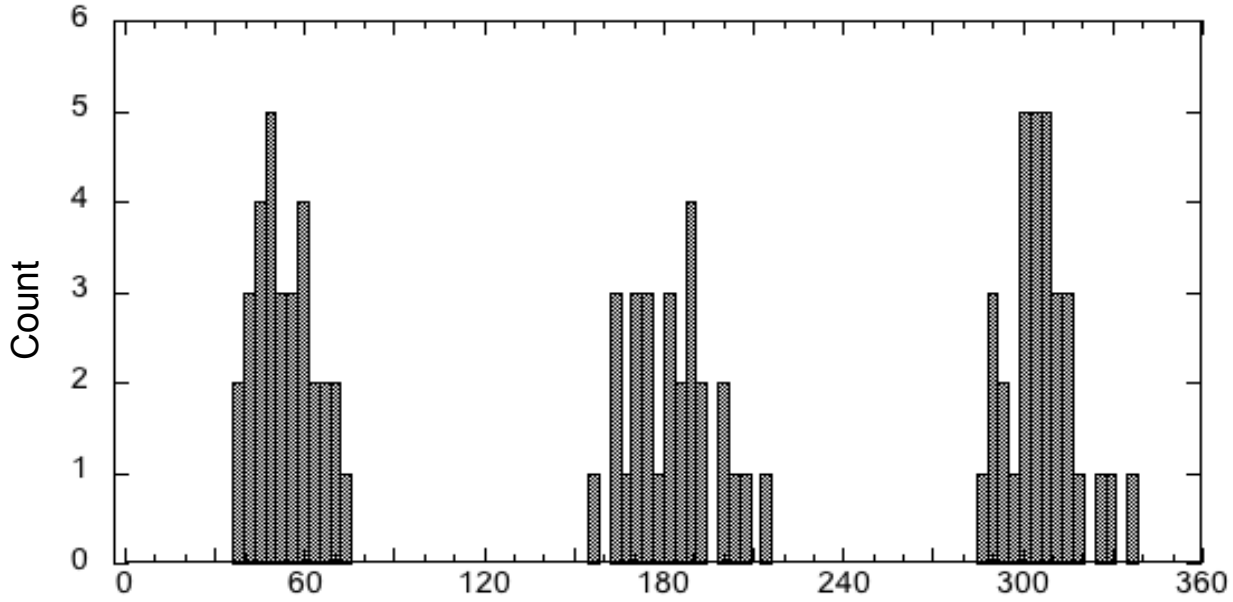


Φ	E
60.0	0.00
180.0	0.00
300.0	0.00



CSD comparison

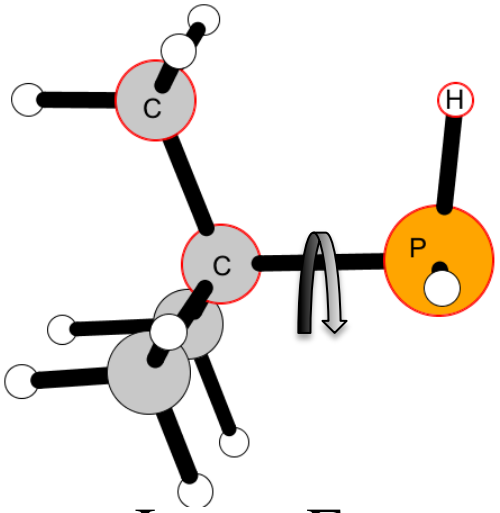




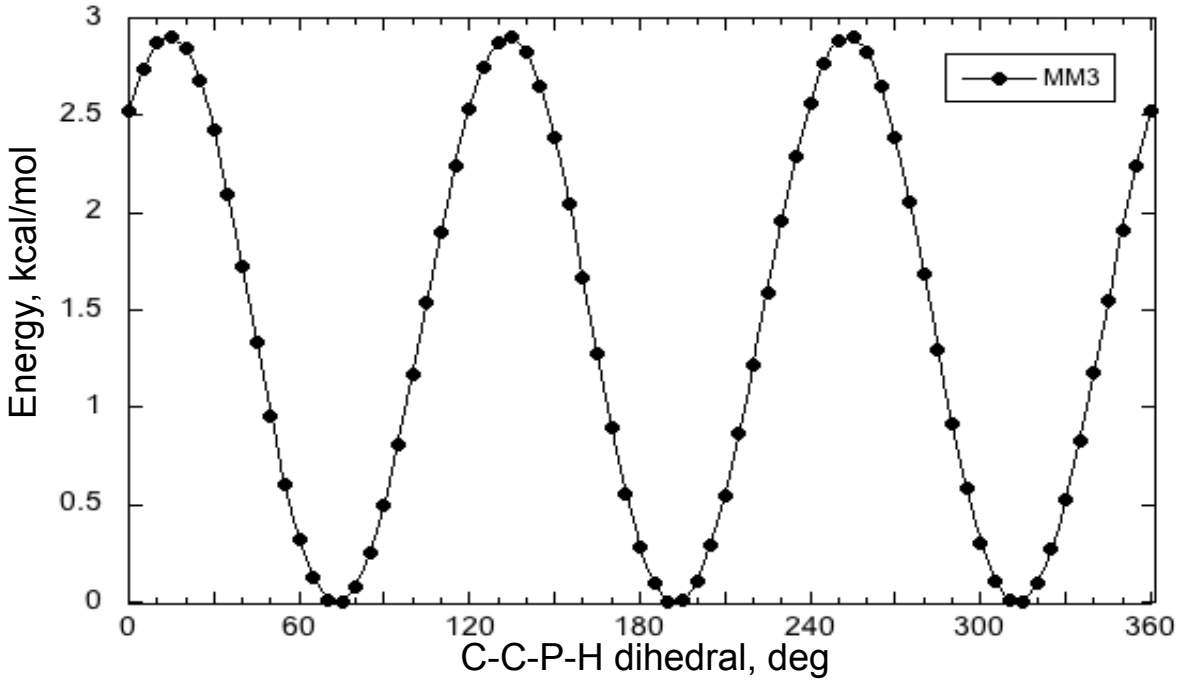
201 hits

C–S = 1.84 ± 0.02 Å

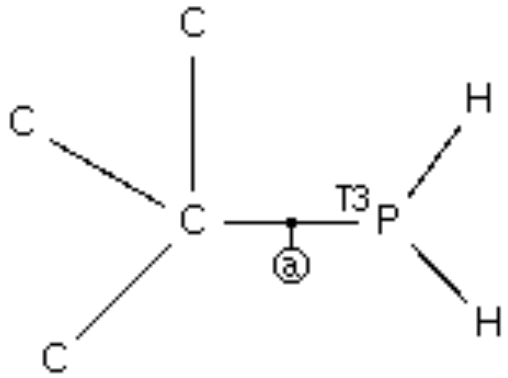
TYPE 1-3:9-1



Φ	E
75.00	0.00
190.00	0.00
315.00	0.00

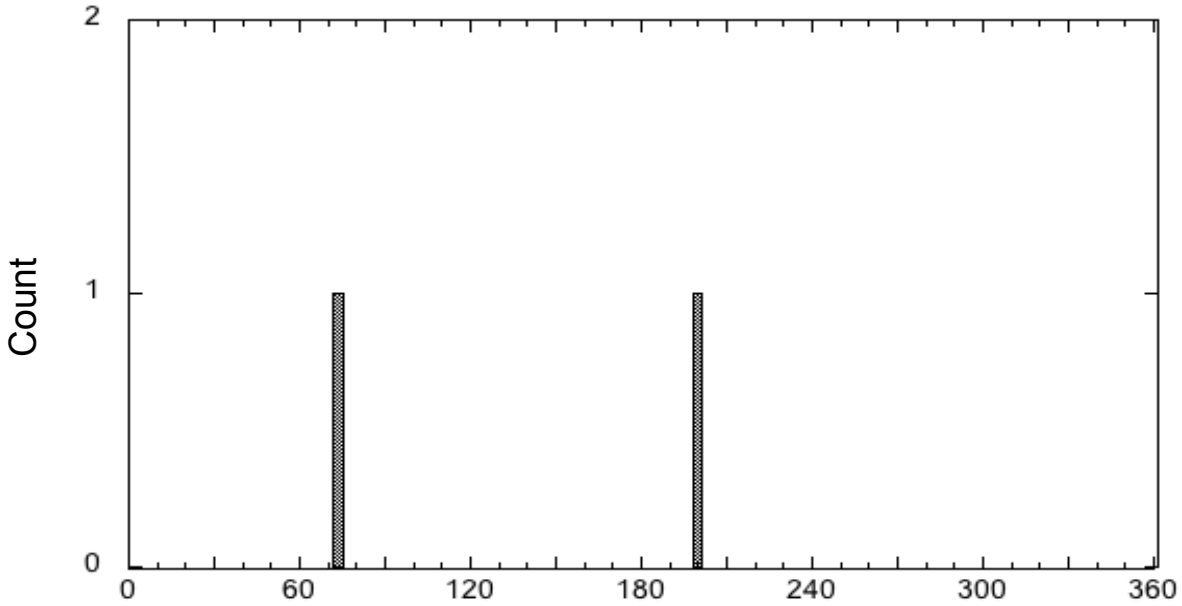


CSD comparison

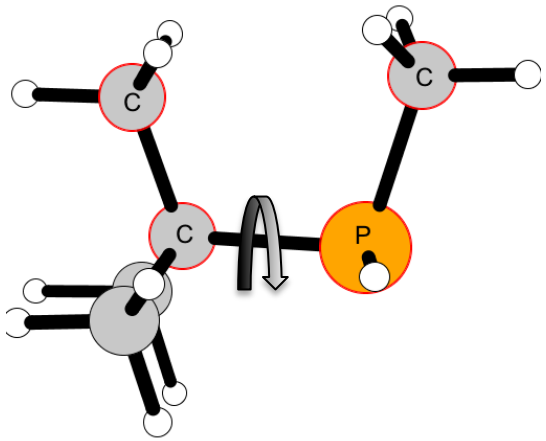


2 hits

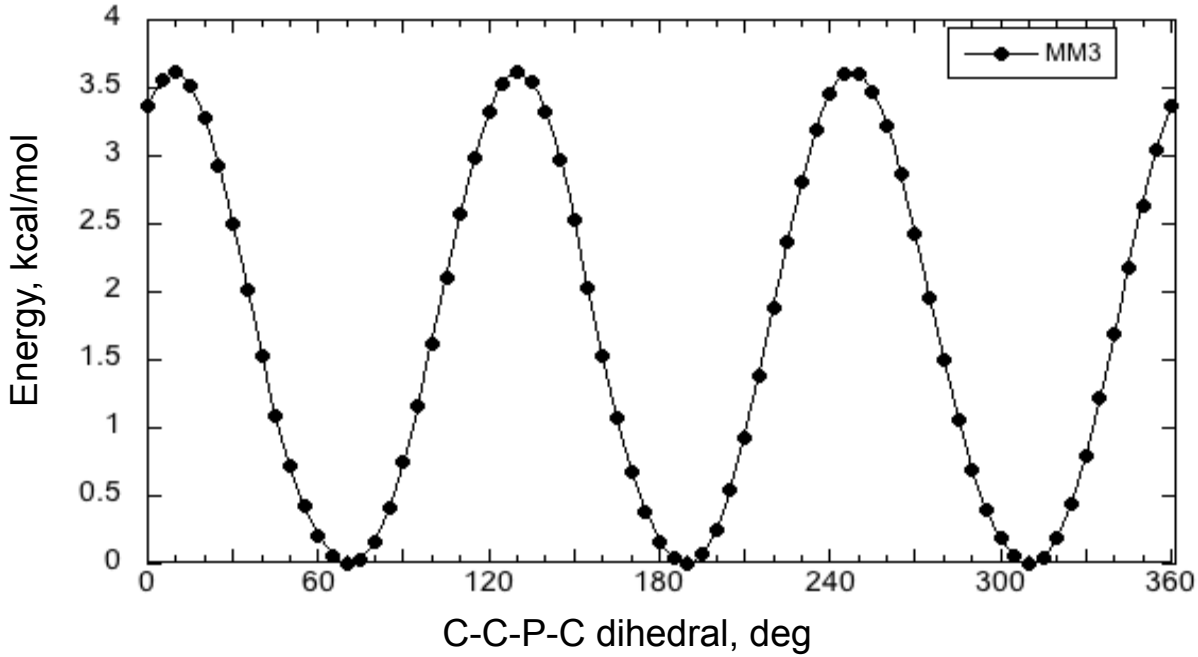
C–P = 1.86 ± 0.00 Å



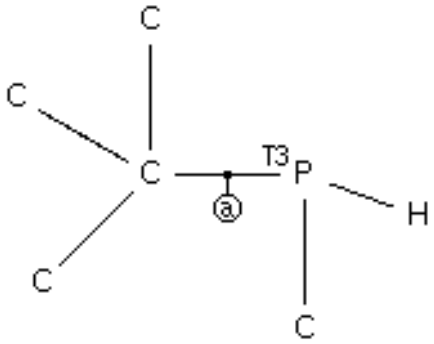
TYPE 1-3:9-2



Φ	E
70.0	0.00
190.0	0.00
310.0	0.00



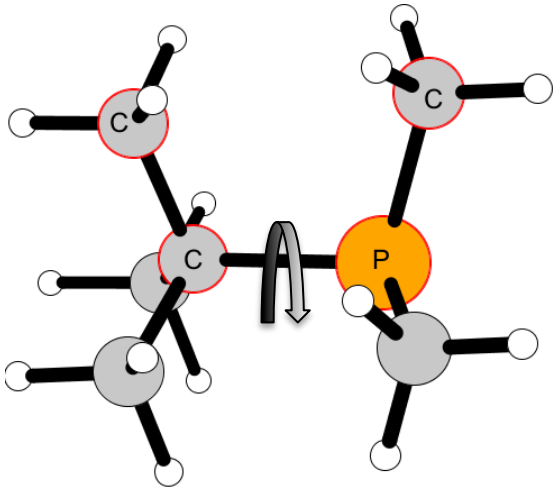
CSD comparison



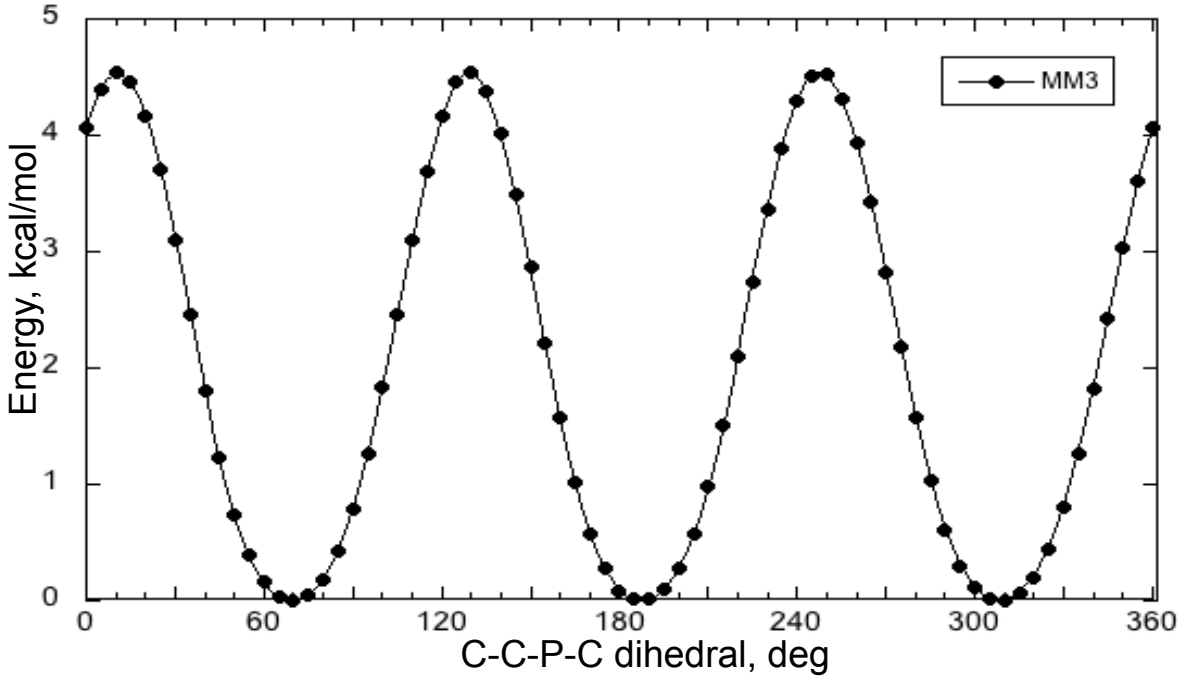
0 hits

NO CDB HITS

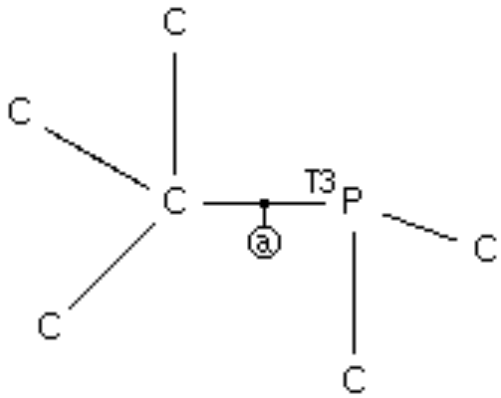
TYPE 1-3:9-4



Φ	E
70.00	0.00
185.00	0.01
310.00	0.00

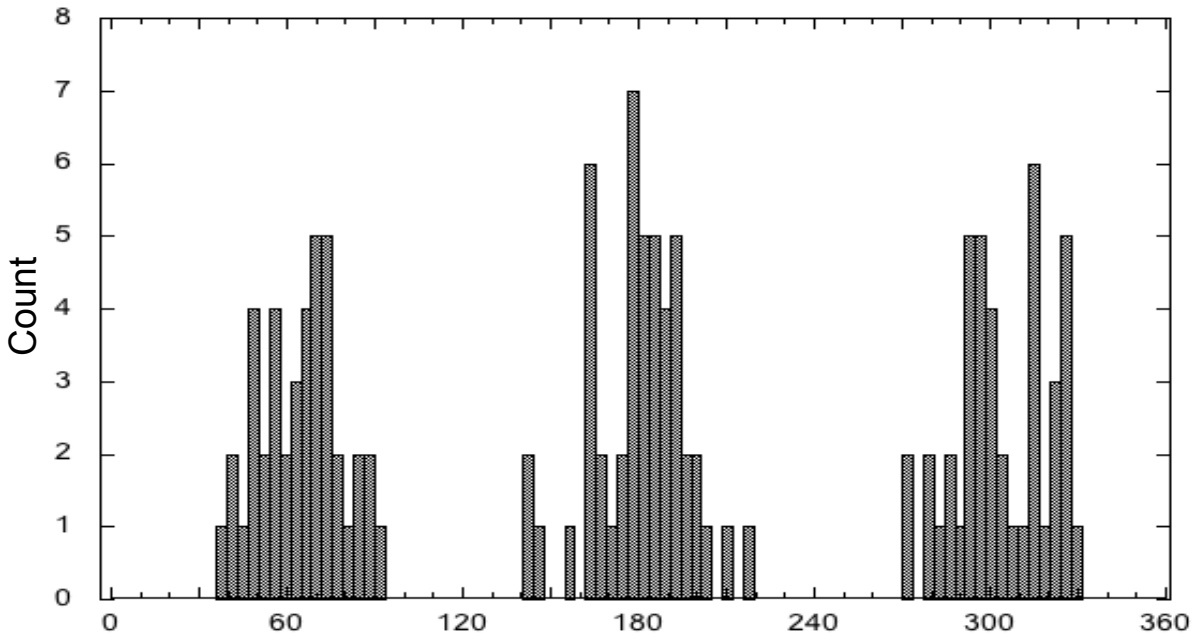


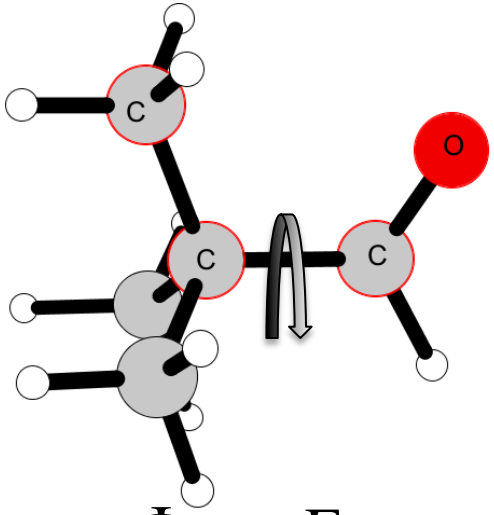
CSD comparison



63 hits

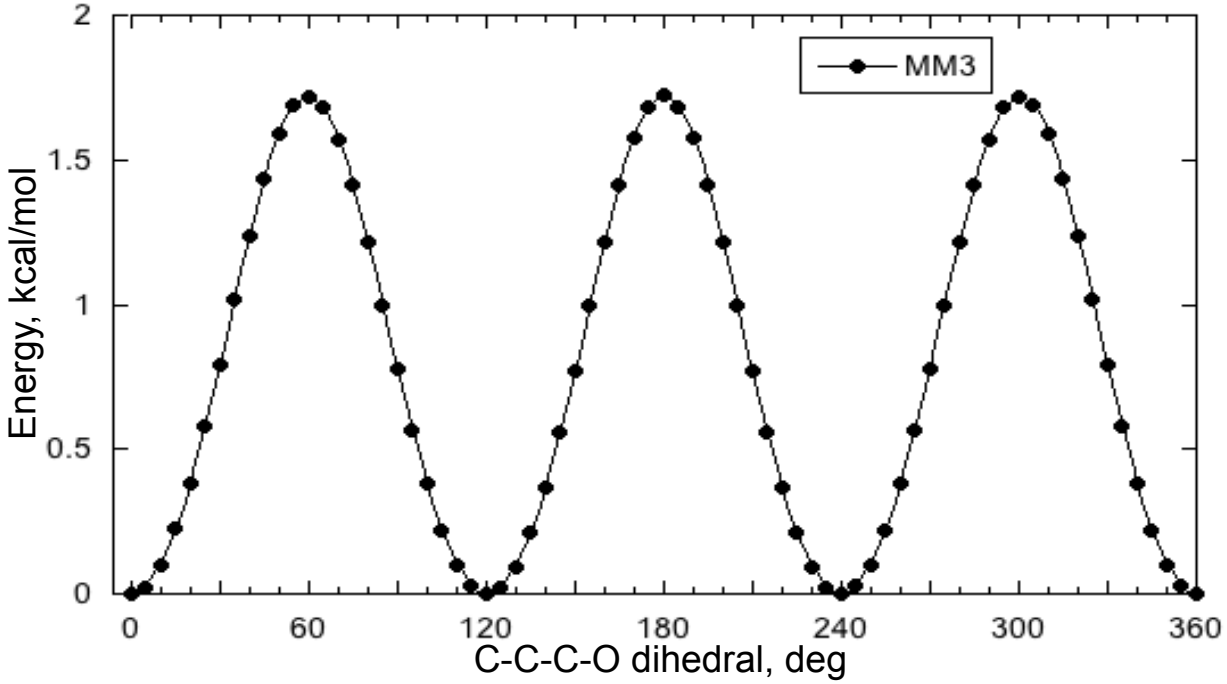
C–P = 1.89 ± 0.02 Å



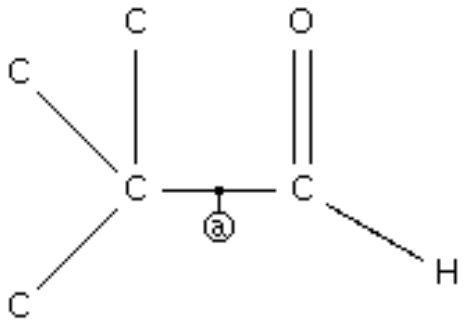


Φ	E
0.0	0.00
120.0	0.00
240.0	0.00

TYPE 1-3:10-1

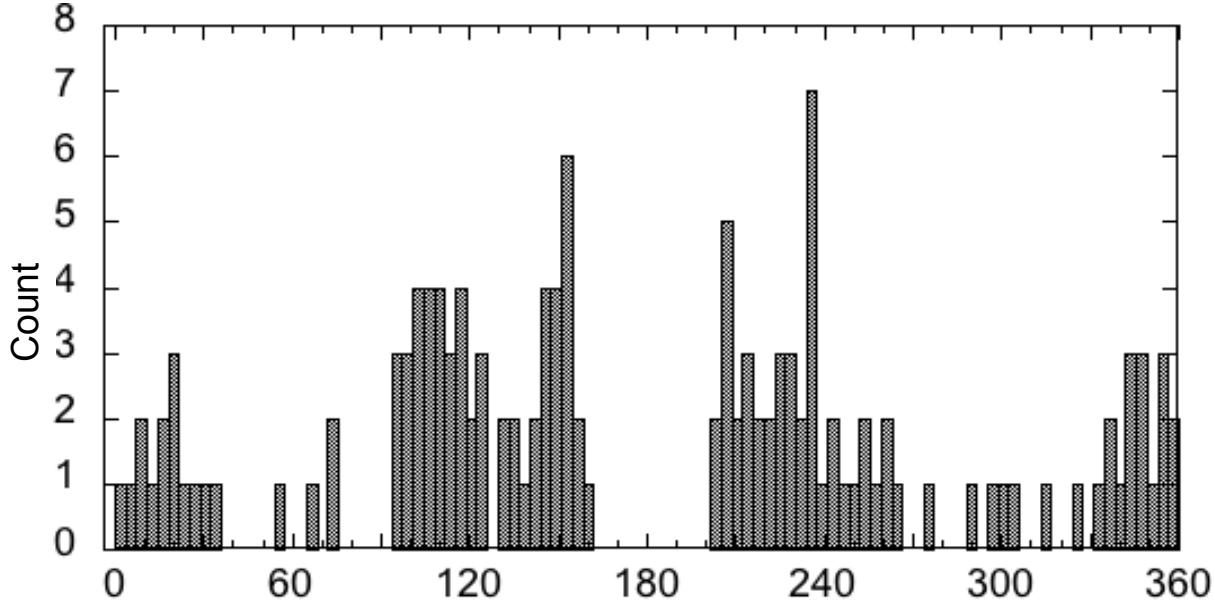


CSD comparison

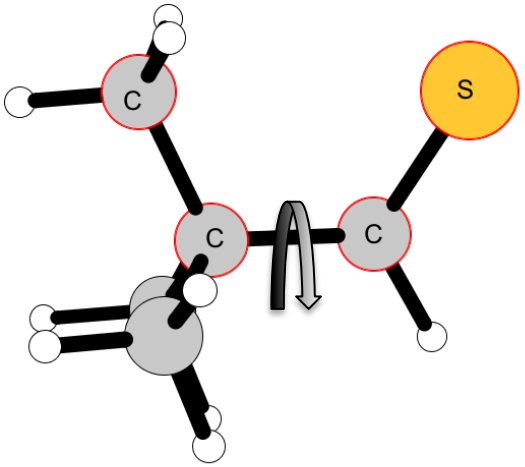


114 hits

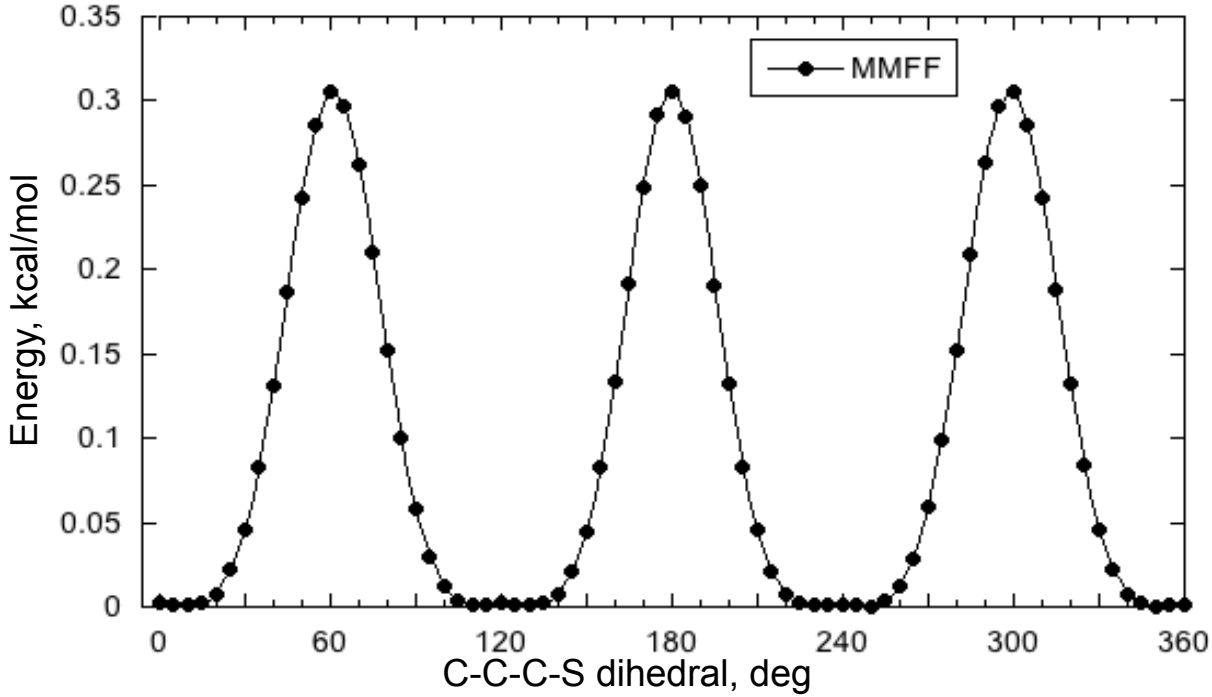
C–C = 1.52 ± 0.02 Å



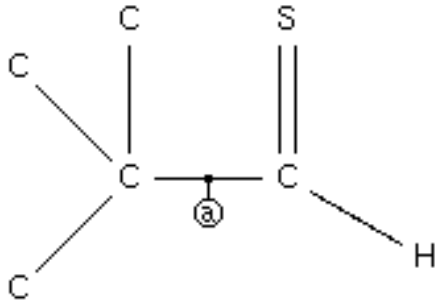
TYPE 1-3:10-2



Φ	E
0.0 : 15.0	0.00
105.0 : 135.0	0.00
345.0 : 355.0	0.00



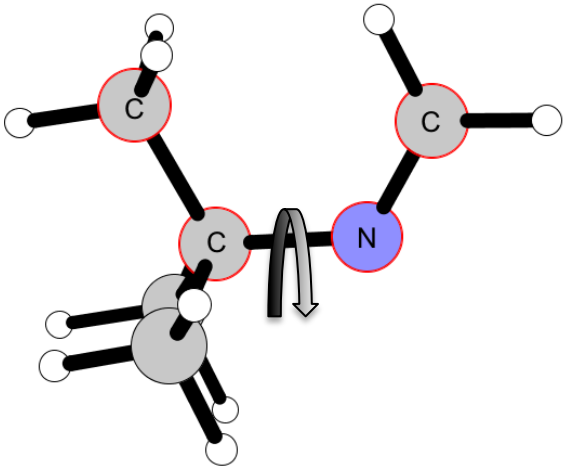
CSD comparison



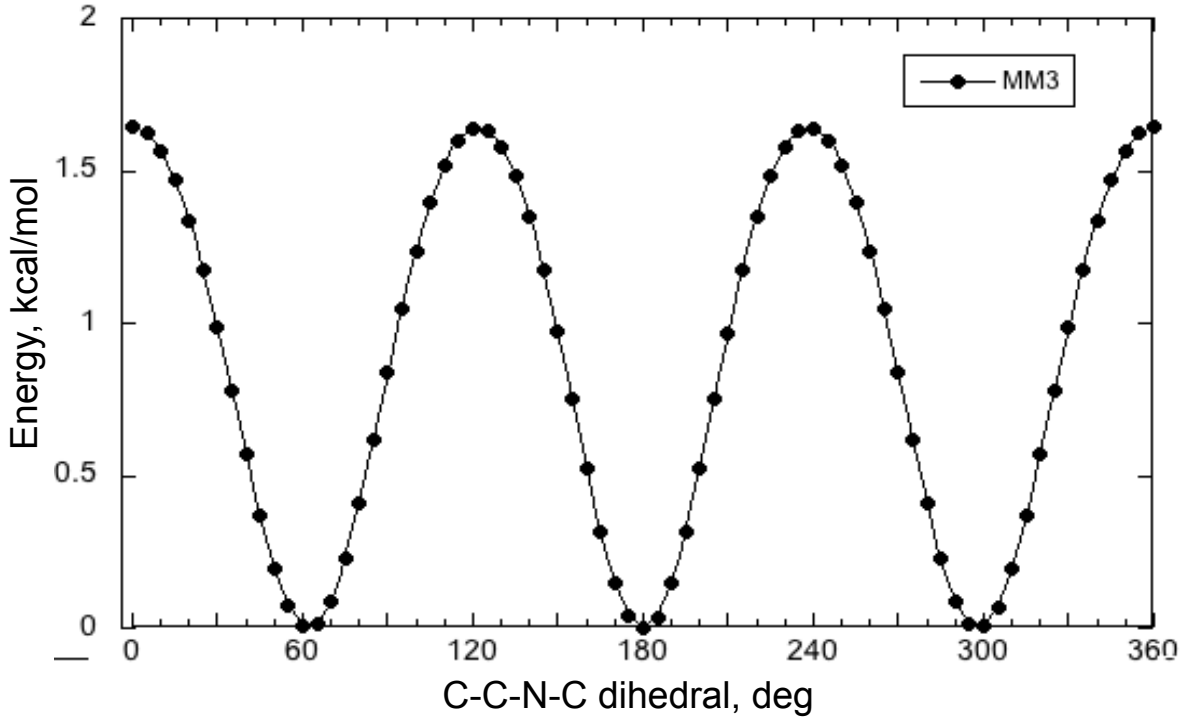
0 hits

NO CDB HITS

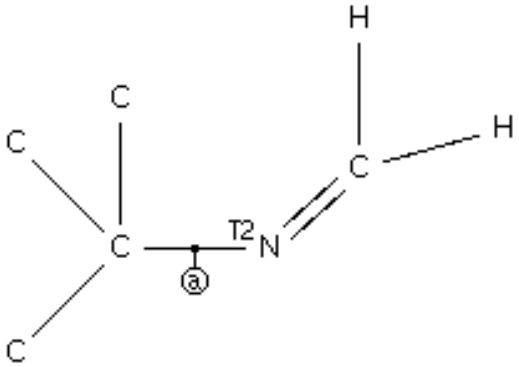
TYPE 1-3:10-3



Φ	E
60.0	0.01
180.0	0.00
300.0	0.01



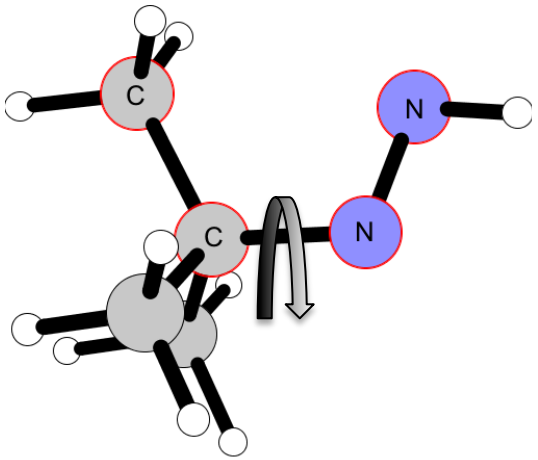
CSD comparison



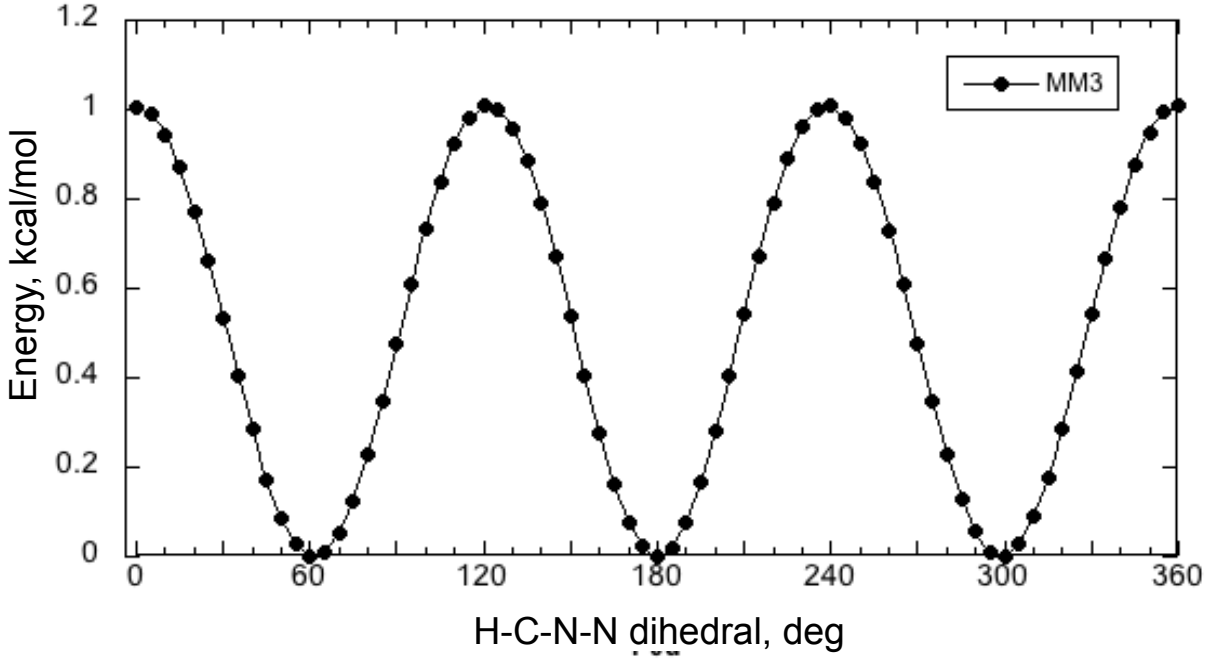
0 hits

NO CDB HITS

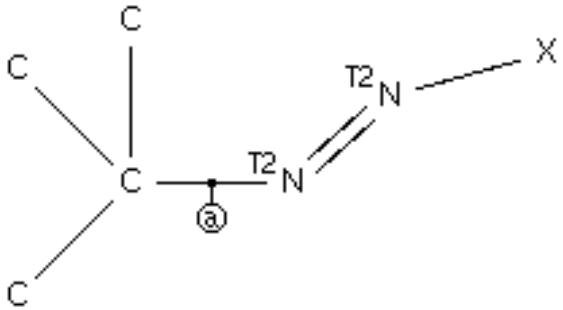
TYPE 1-3:10-4



Φ	E
60.0	0.00
180.0	0.00
300.0	0.00

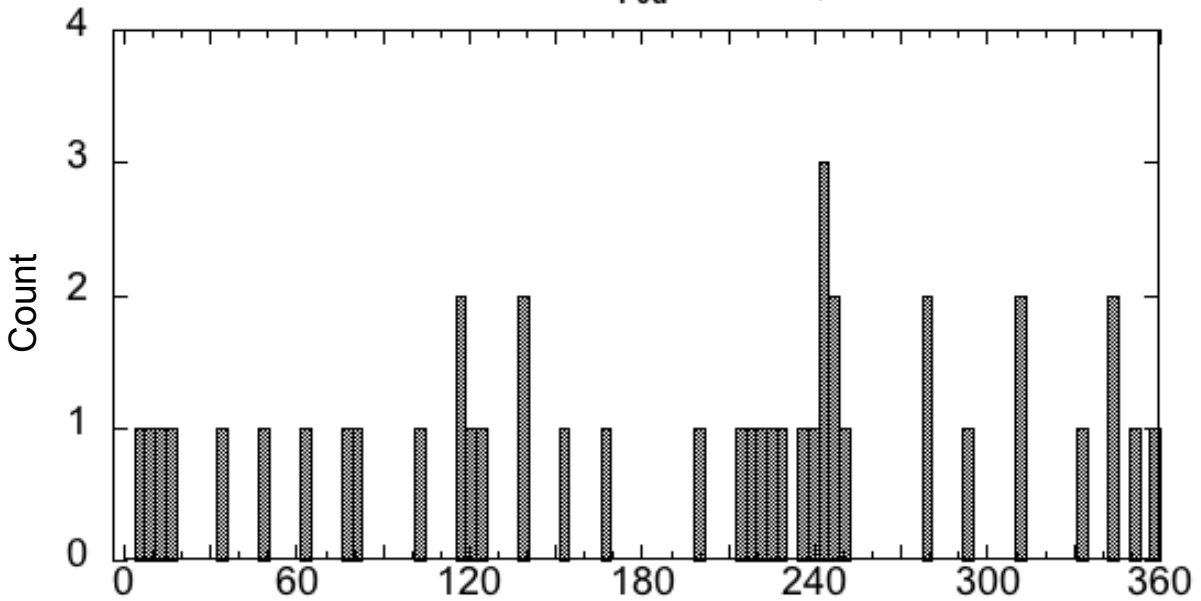


CSD comparison

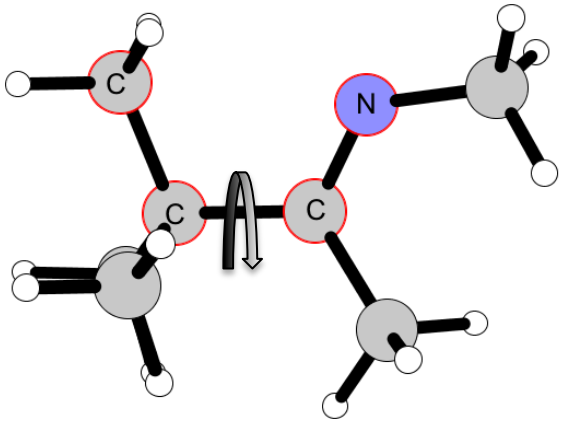


32 hits

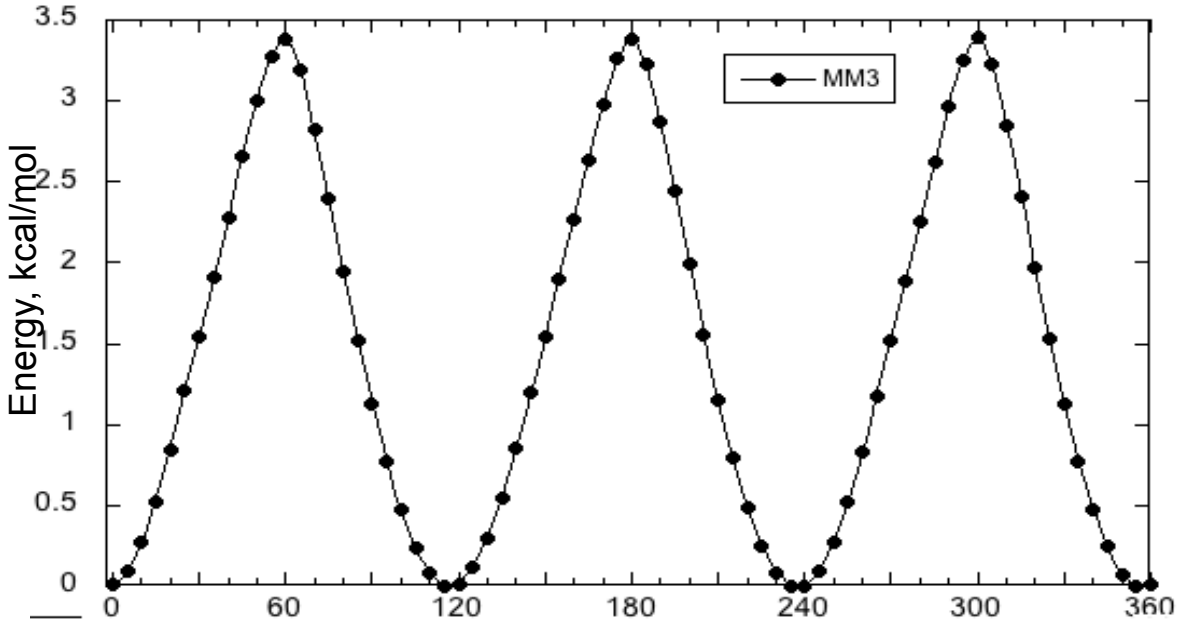
C–N = 1.50 ± 0.02 Å



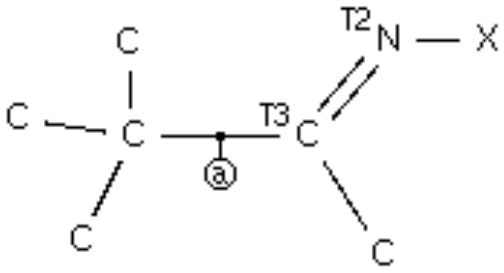
TYPE 1-3:11-1

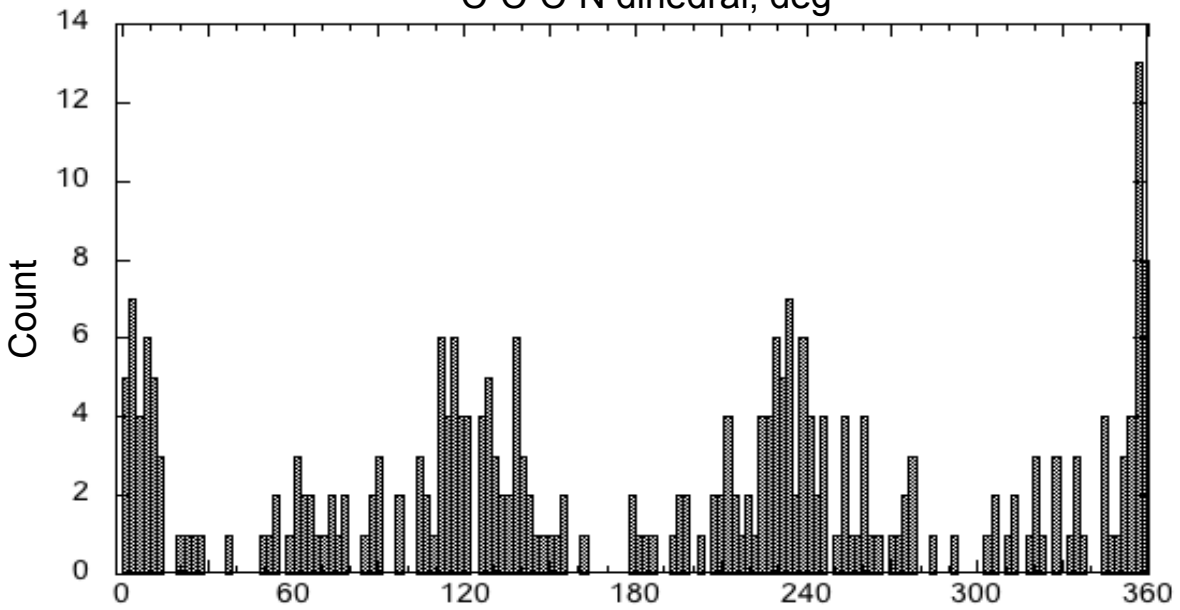


Φ	E
115.0	0.00
240.0	0.00
355.0	0.00



CSD comparison

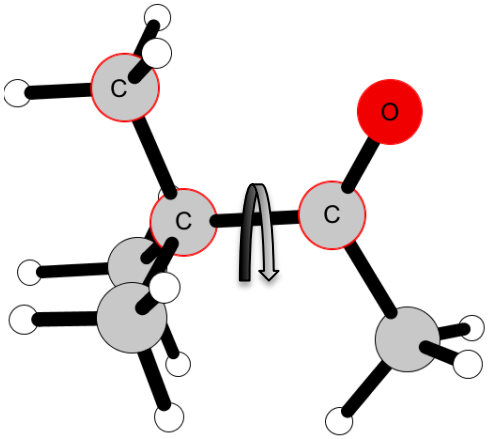




194 hits

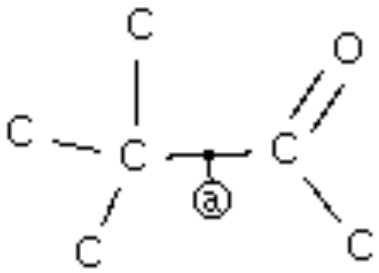
C–C = 1.52 ± 0.01 Å

TYPE 1-3:11-2



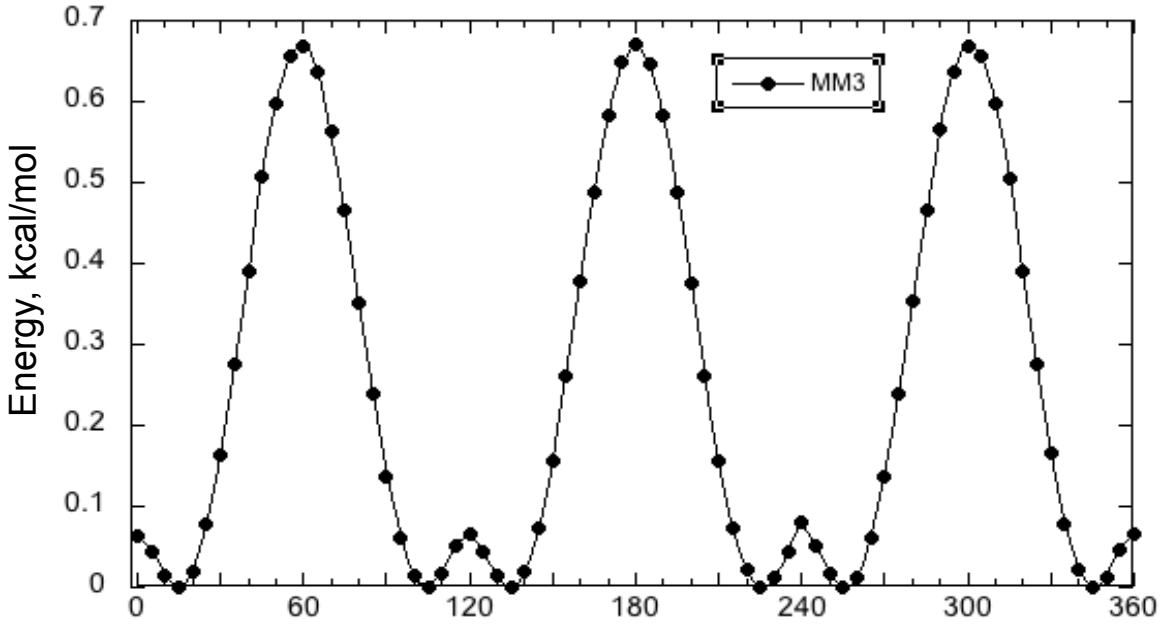
Φ	E
15.0	0.00
105.0	0.00
135.0	0.00
225.0	0.00
255.0	0.00
345.0	0.00

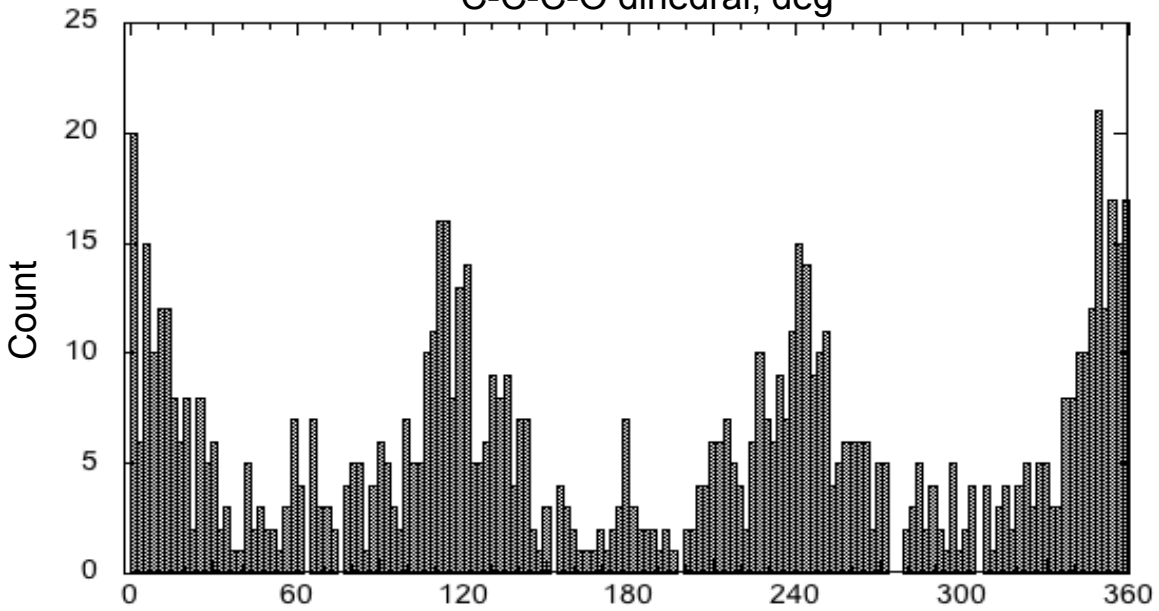
CSD comparison



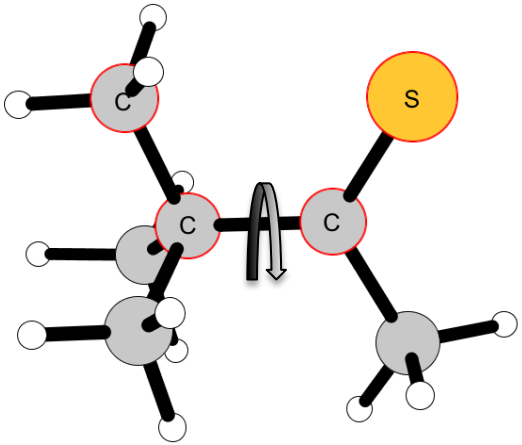
611 hits

C–C = 1.53 ± 0.02 Å



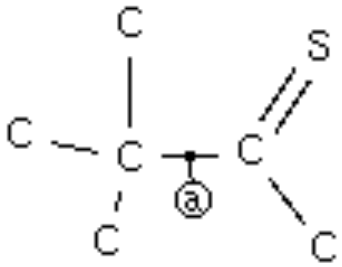


TYPE 1-3:11-3



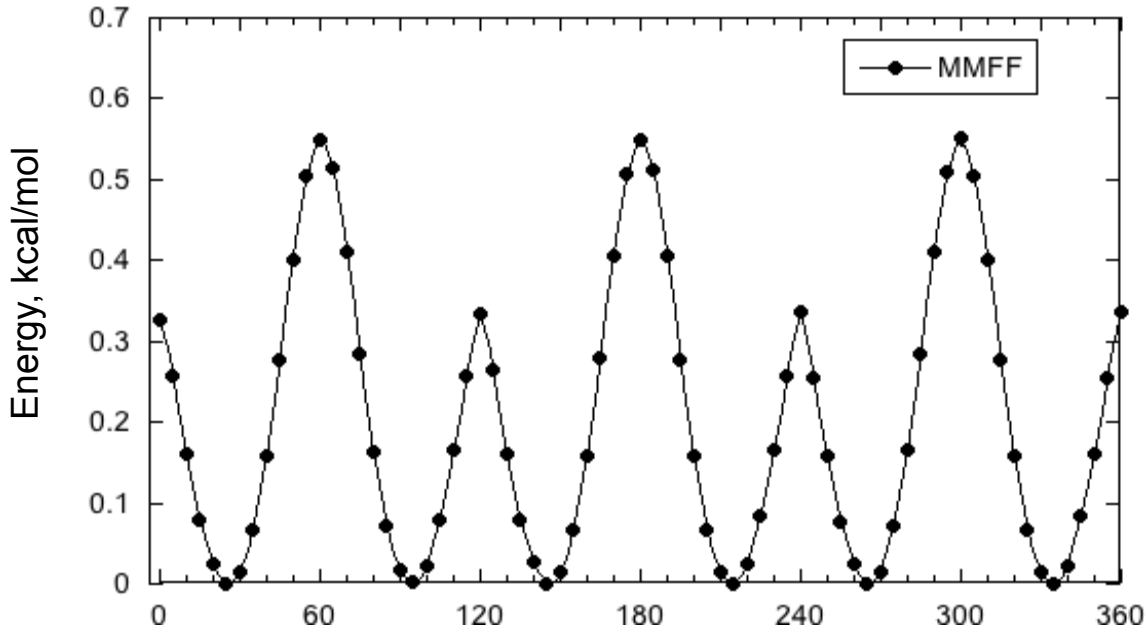
Φ	E
25.0	0.00
95.0	0.00
145.0	0.00
215.0	0.00
265.0	0.00
335.0	0.00

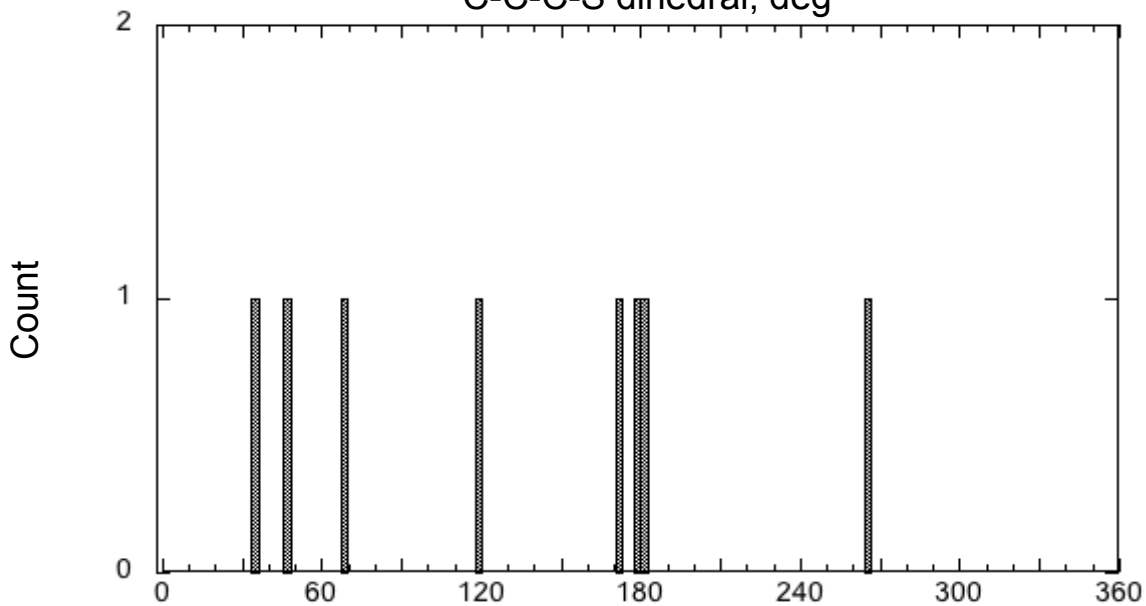
CSD comparison



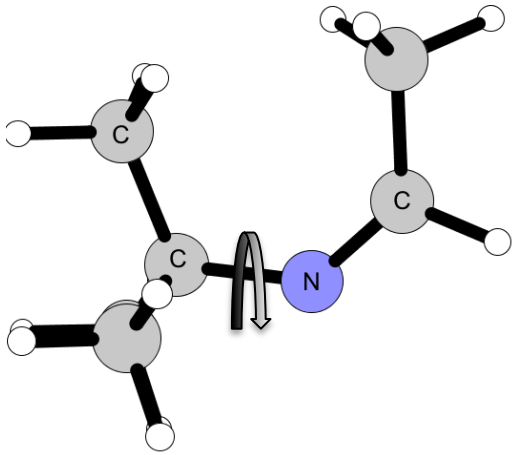
7 hits

C–C = 1.53 ± 0.01 Å

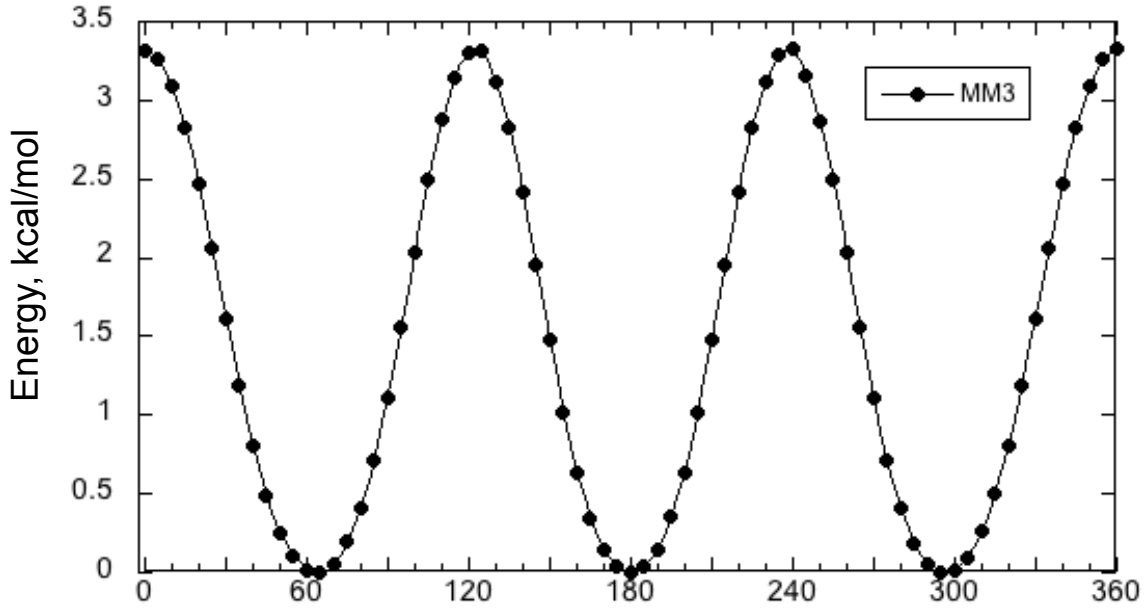




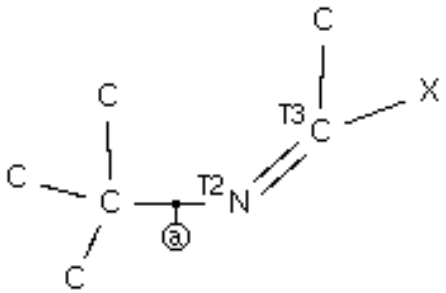
TYPE 1-3:12-1



Φ	E
65.0	0.00
180.0	0.00
295.0	0.00

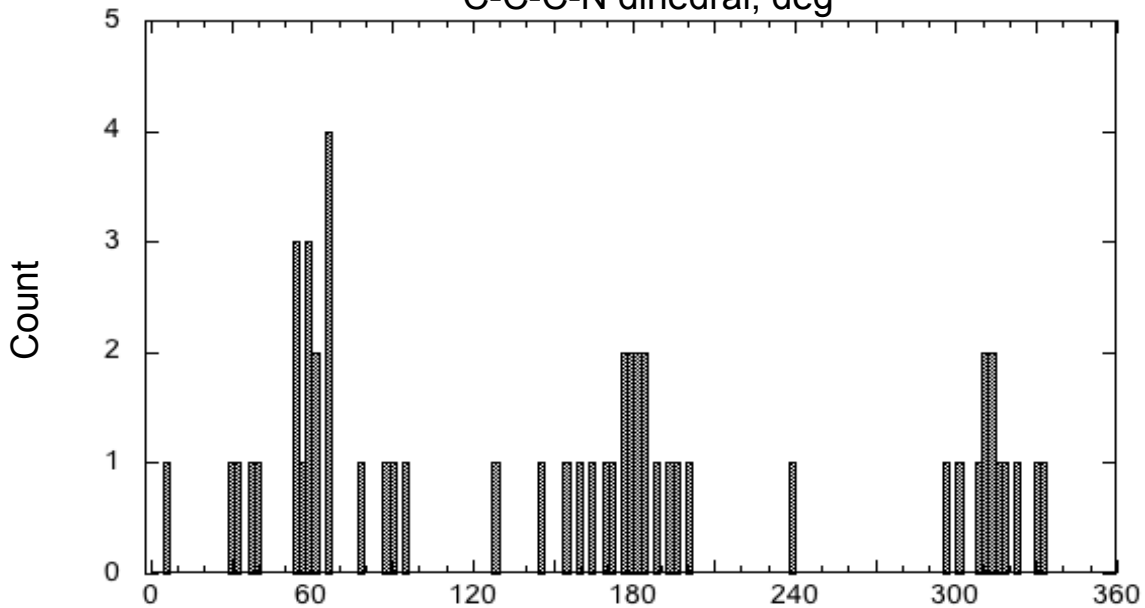


CSD comparison

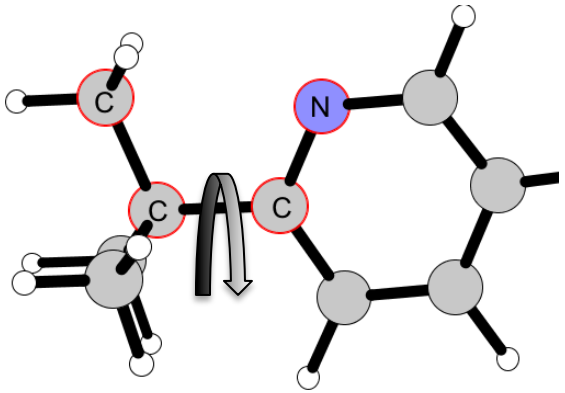


44 hits

C–N = 1.48 ± 0.01 Å

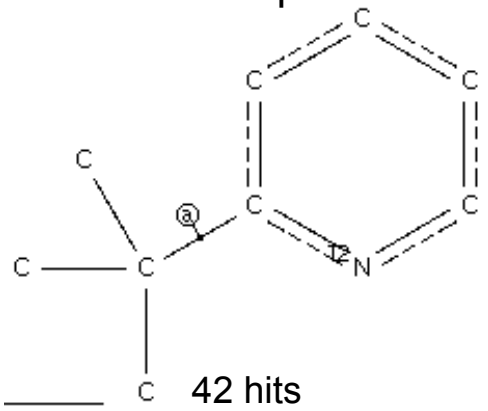


TYPE 1-3:13-1



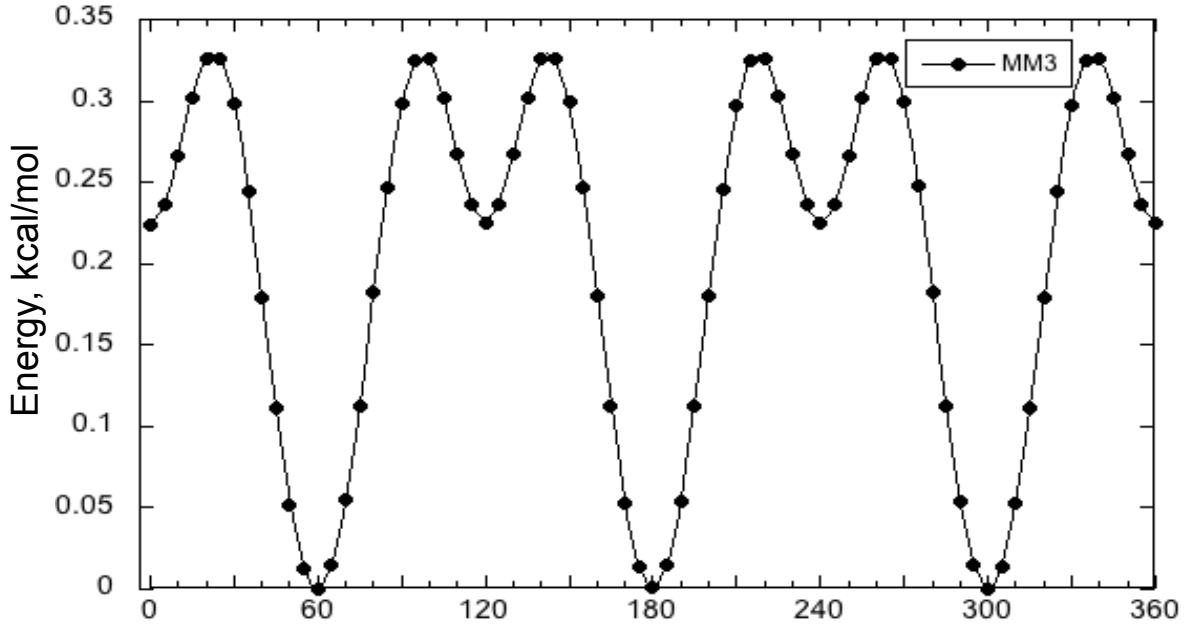
Φ	E
0.0	0.22
60.0	0.00
120.0	0.22
180.0	0.00
240.0	0.22
300.0	0.00

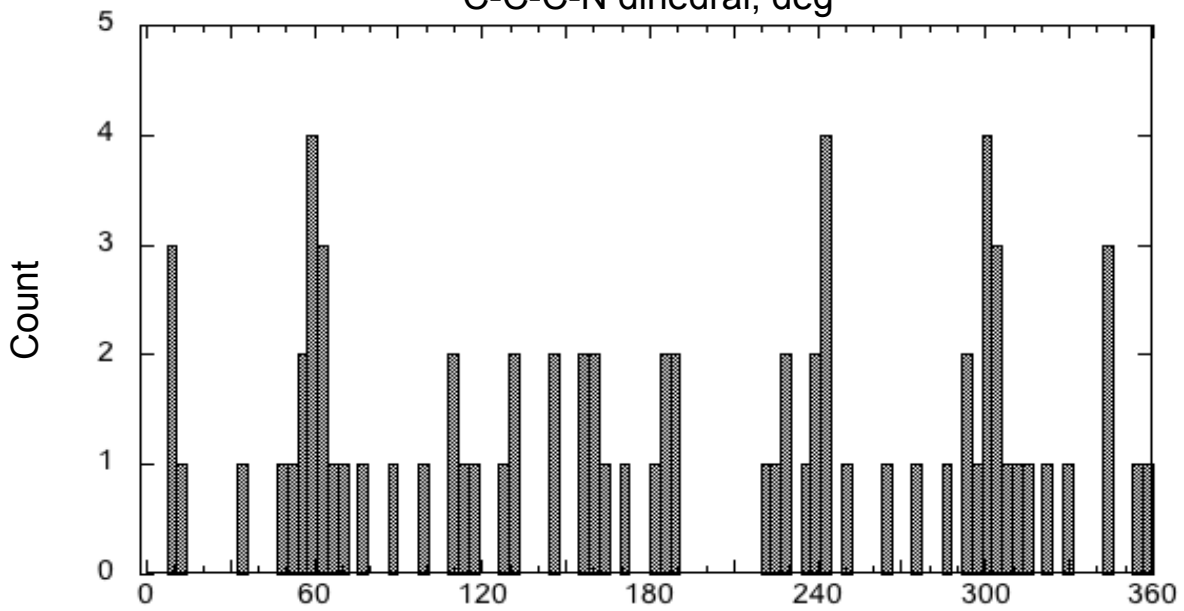
CSD comparison



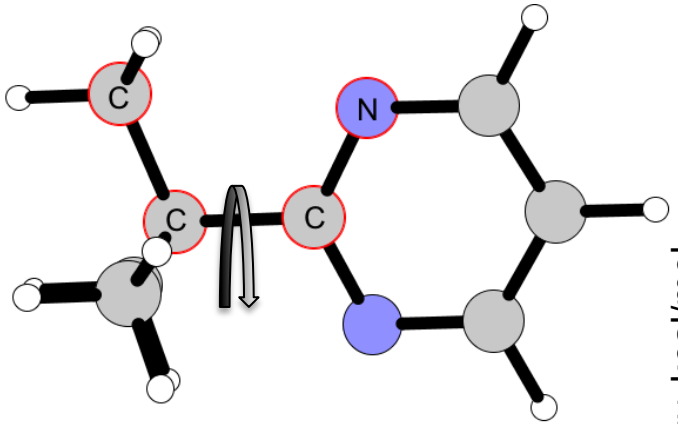
42 hits

C–C = 1.53 ± 0.01 Å



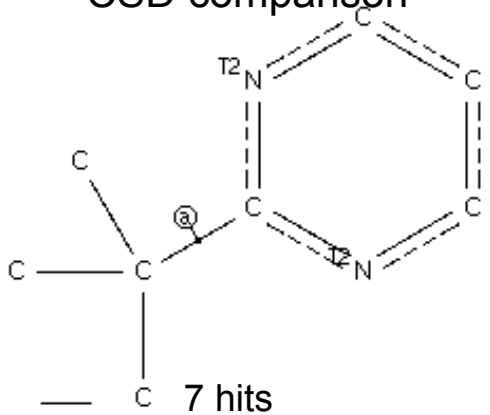


TYPE 1-3:13-2



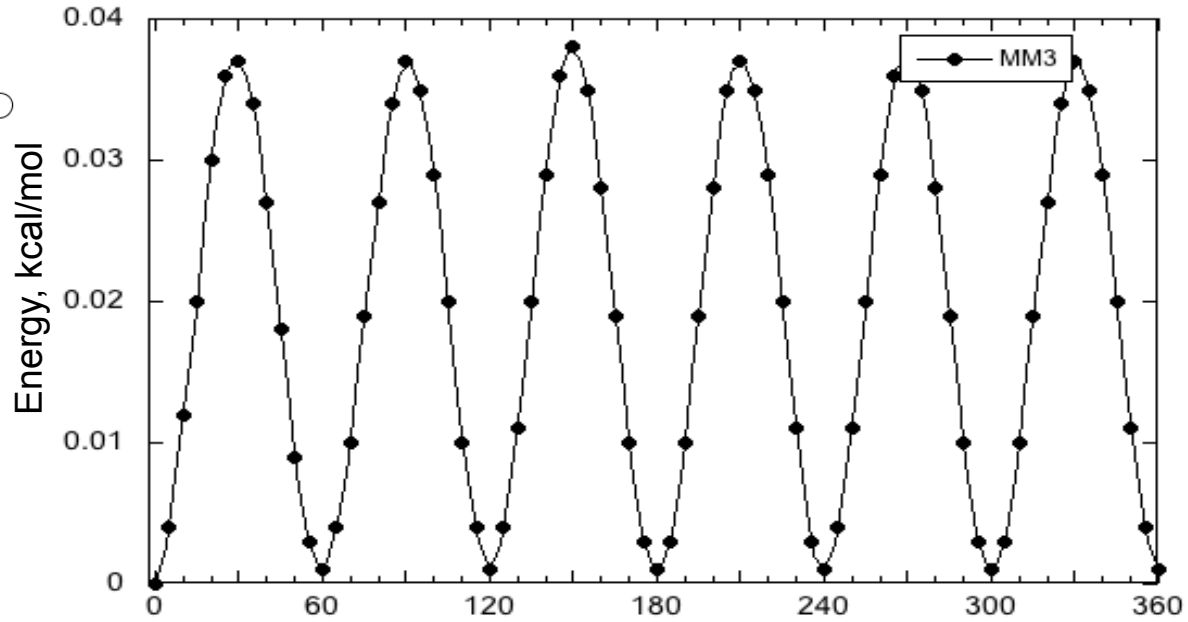
Φ	E
0.0	0.00
60.0	0.00
120.0	0.00
180.0	0.00
240.0	0.00
300.0	0.00

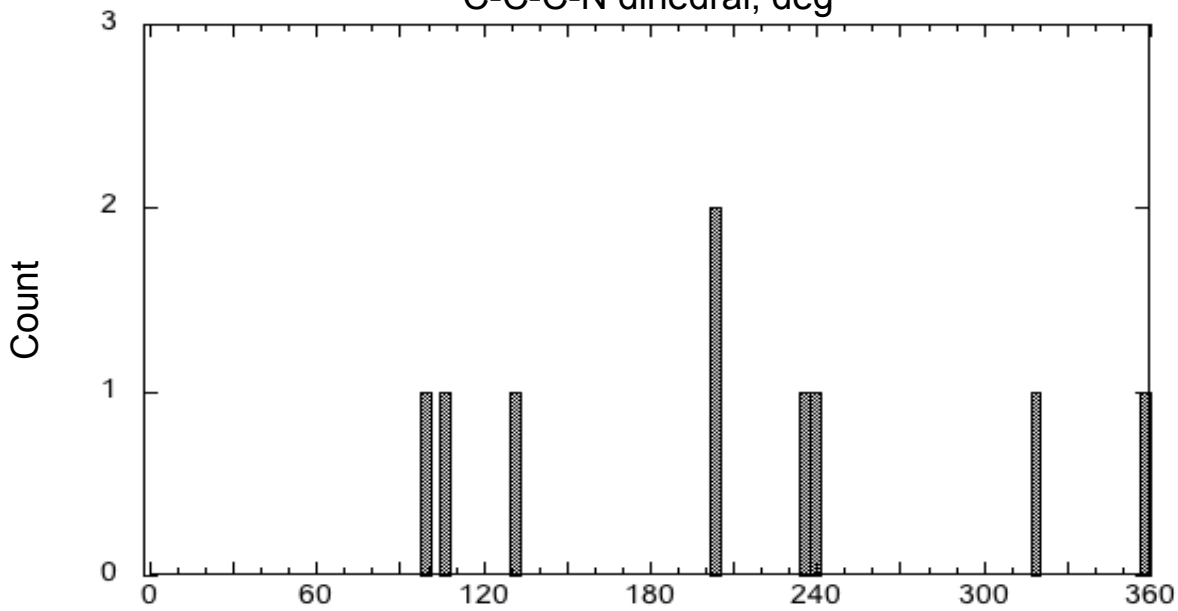
CSD comparison



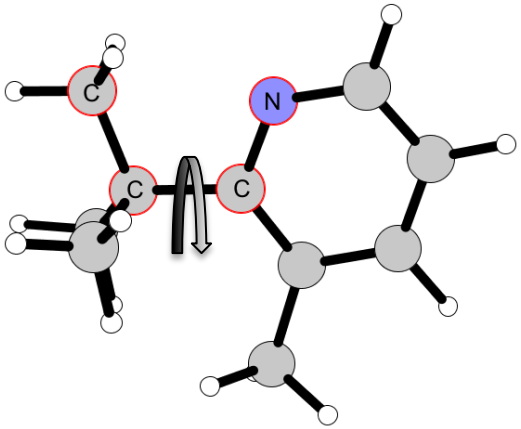
7 hits

C–C = 1.53 ± 0.01 Å



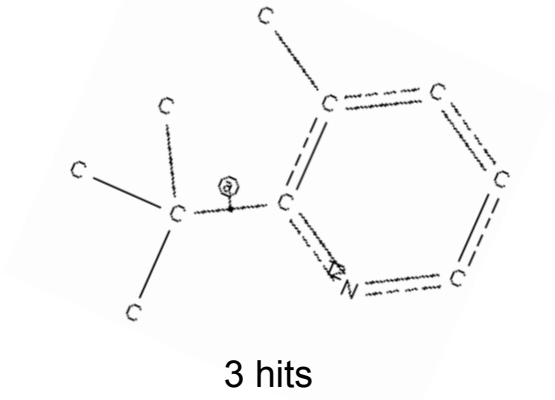


TYPE 1-3:13-3



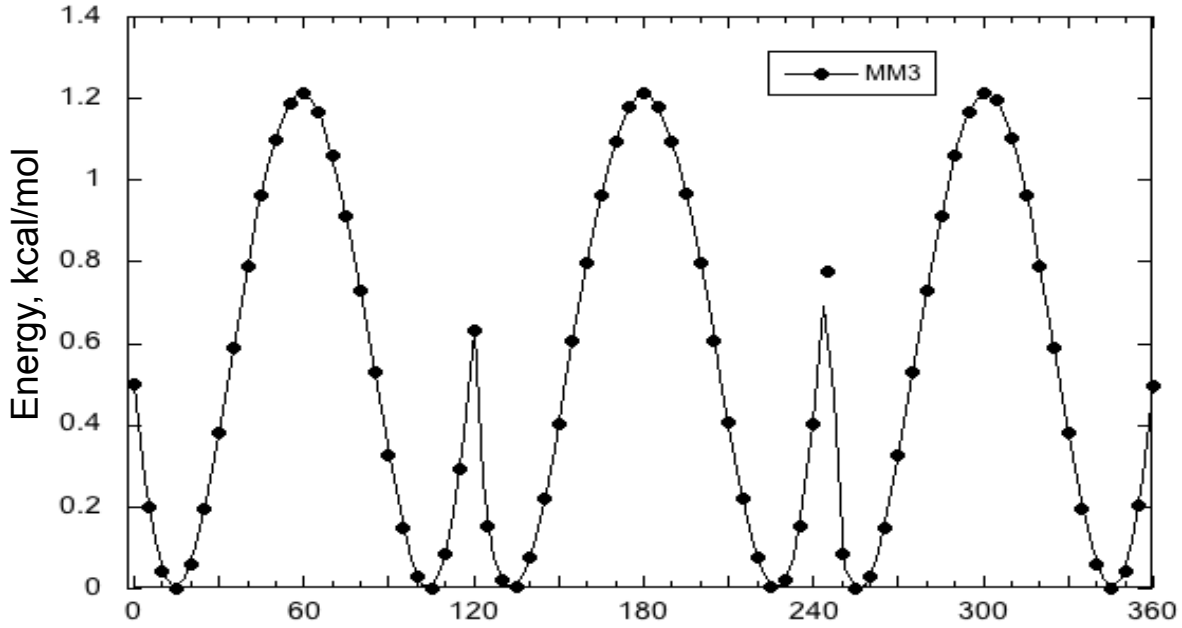
Φ	E
15.0	0.00
105.0	0.00
135.0	0.00
225.0	0.00
255.0	0.00
345.00	0.00

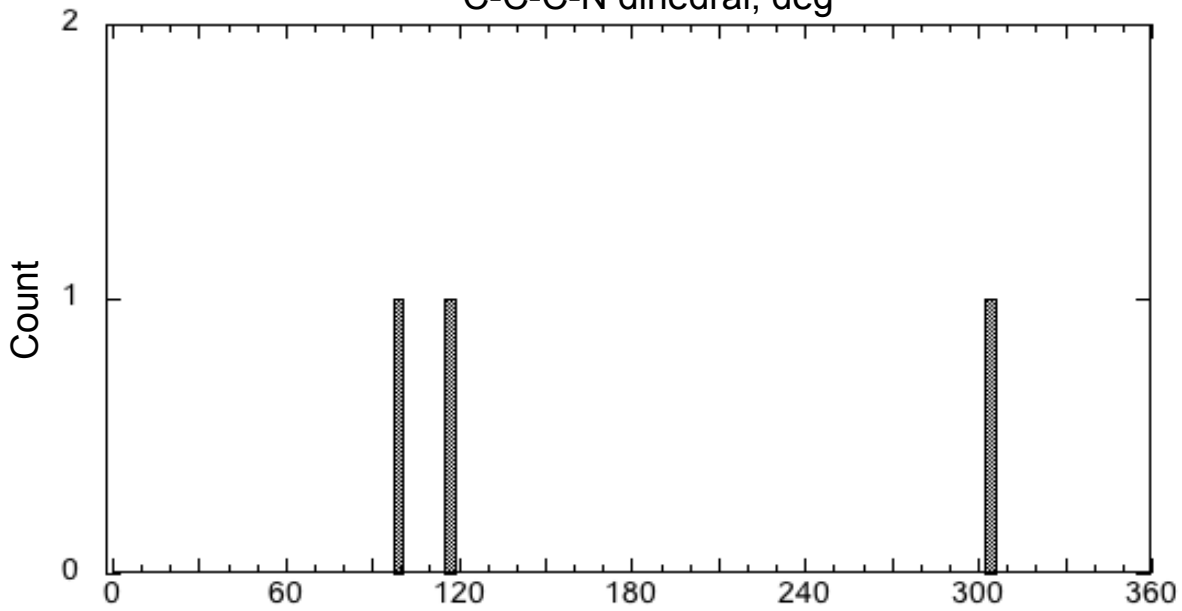
CSD comparison



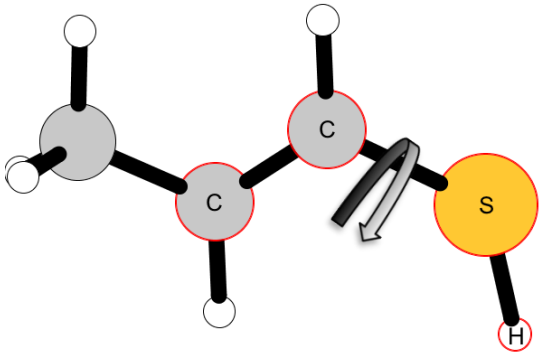
3 hits

C–C = 1.54 ± 0.01 Å

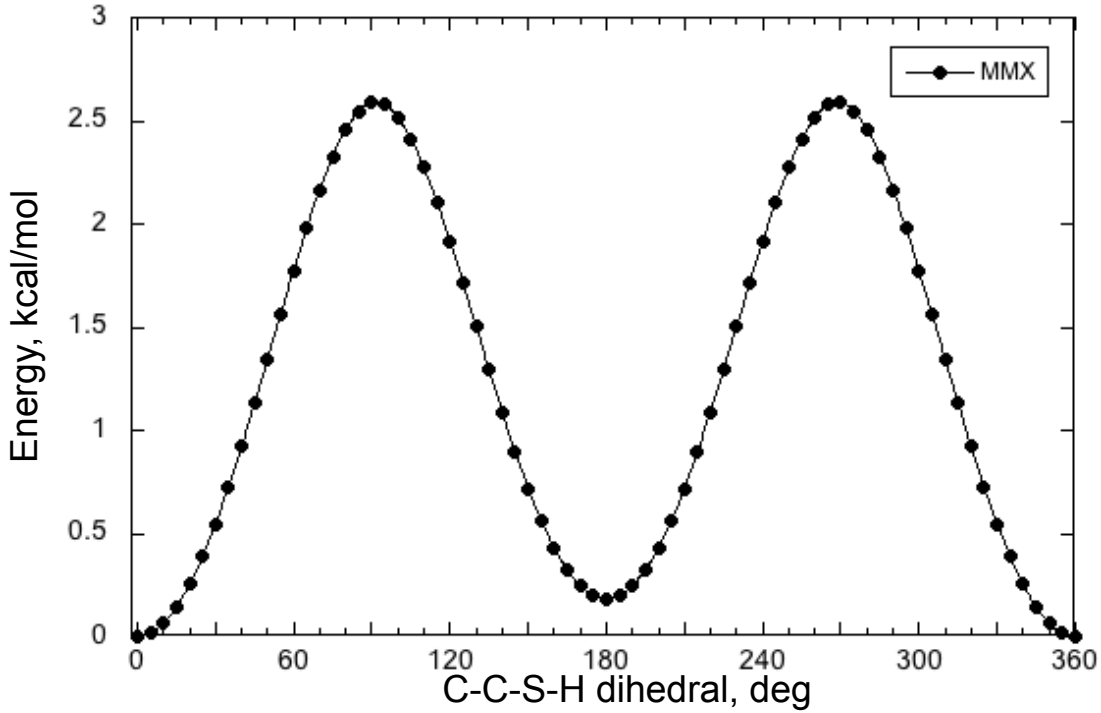




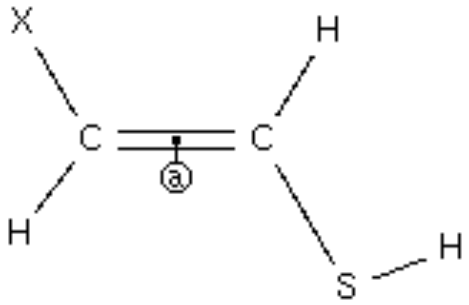
TYPE 2-1:5-4



Φ	E
0.0	0.00
180.0	0.19

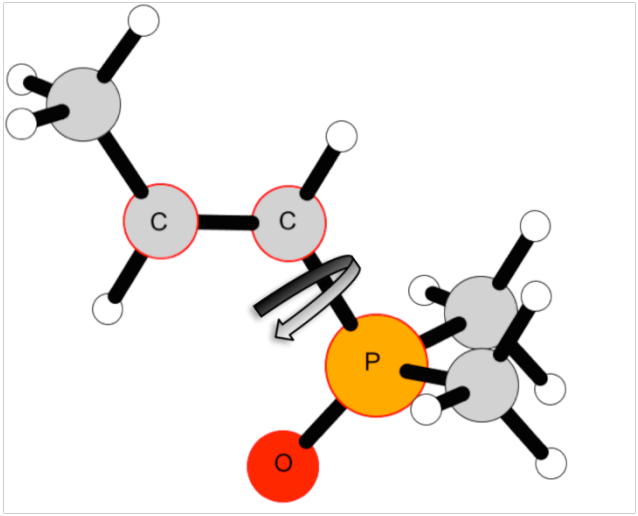


CSD comparison

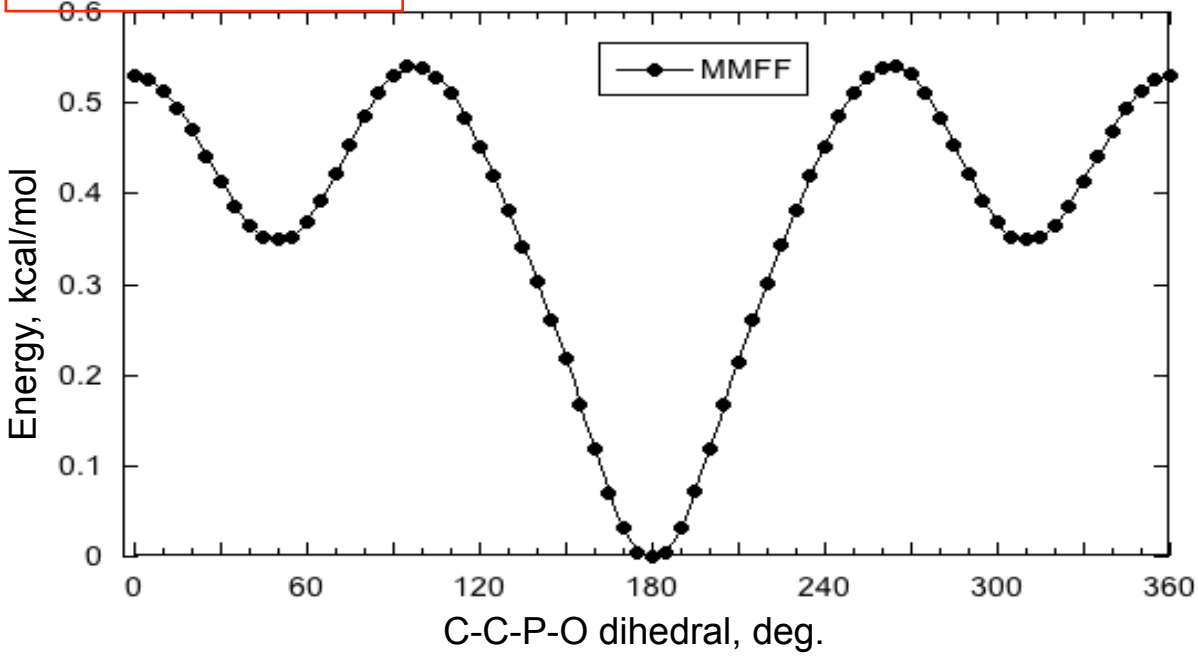


0 hits

NO CDB HITS

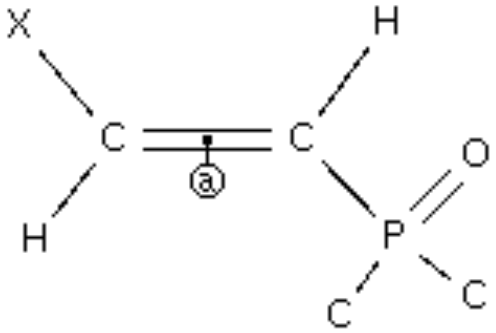


TYPE 2-1:8-1



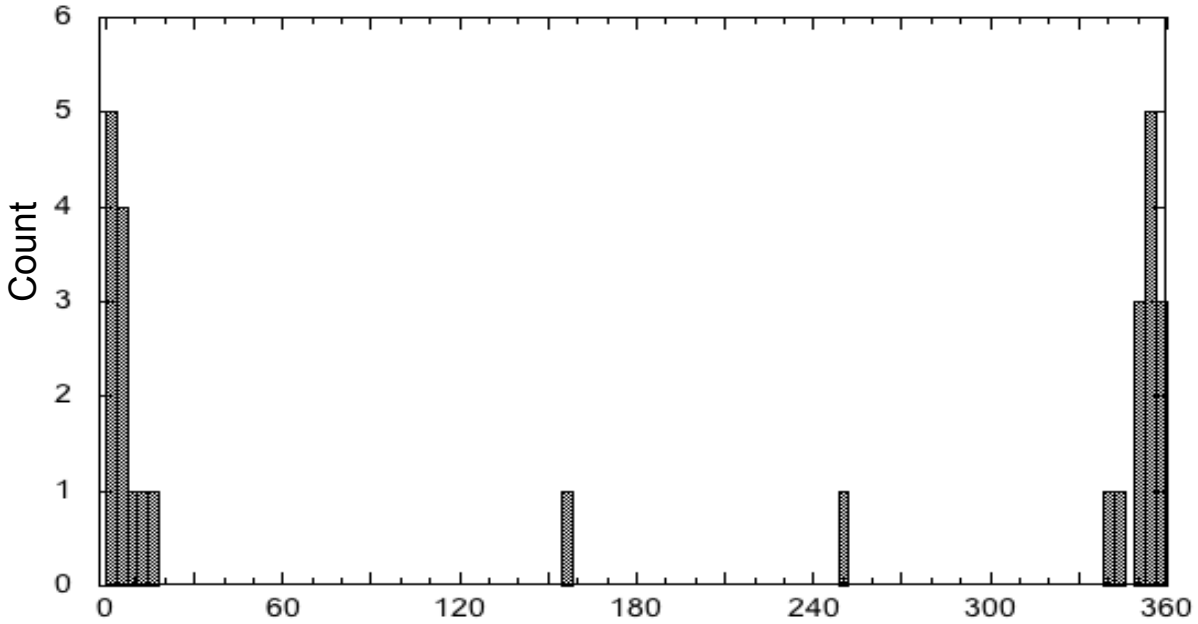
Φ	E
50.0	0.35
180.0	0.00
310.0	0.35

CSD comparison

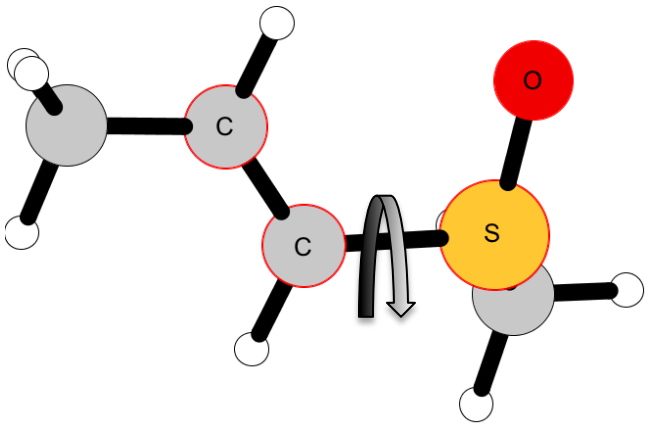


18 Hits

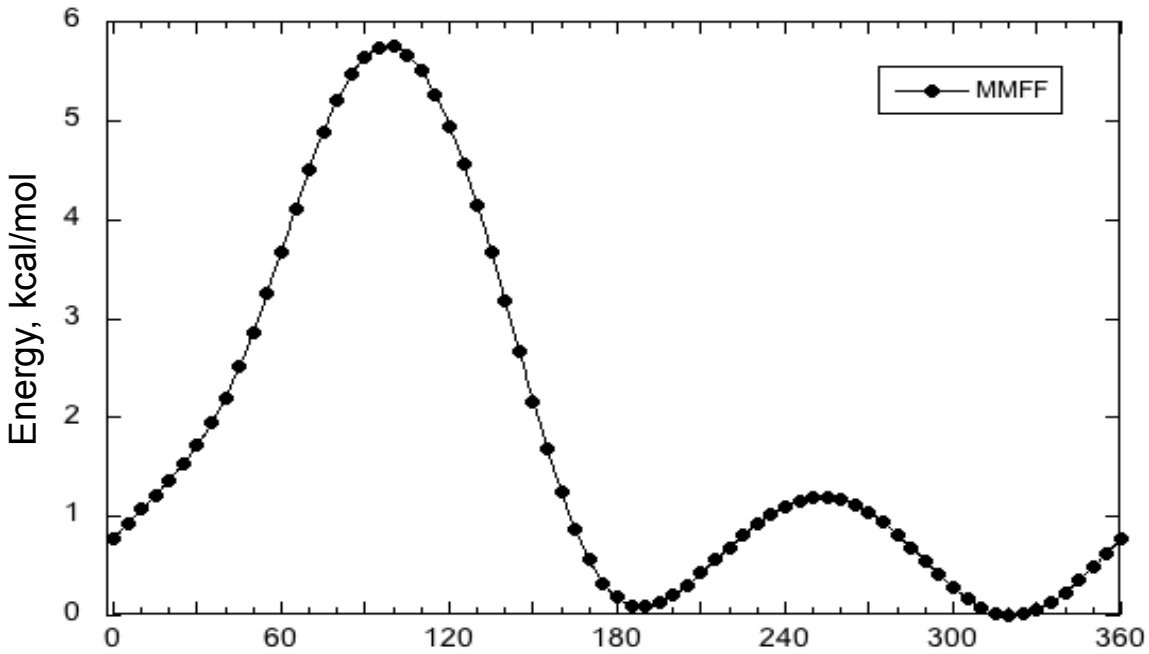
C-P = 1.78 ± 0.01 Å



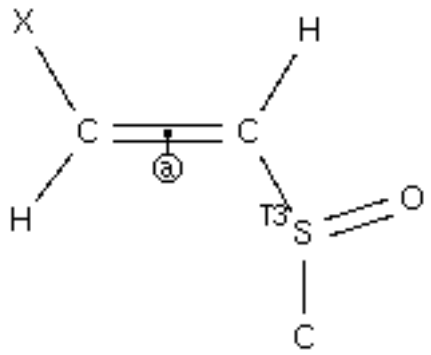
TYPE 2-1:8-2



Φ	E
190.0	0.09
320.0	0.00

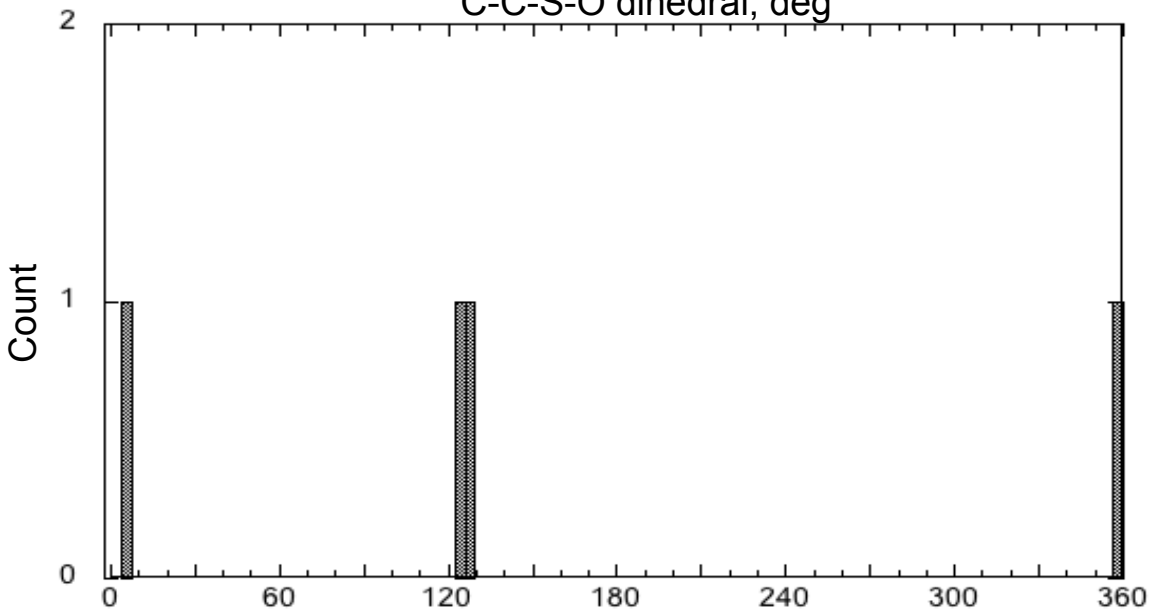


CSD comparison

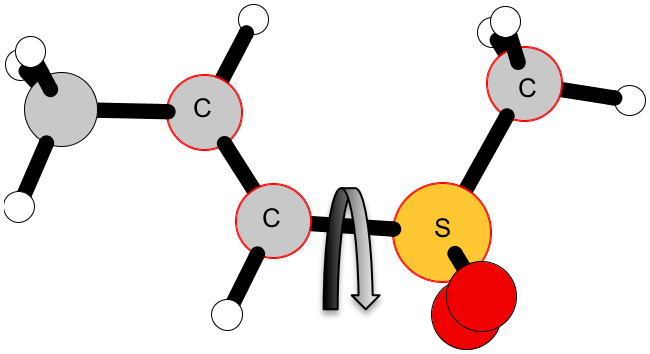


4 hits

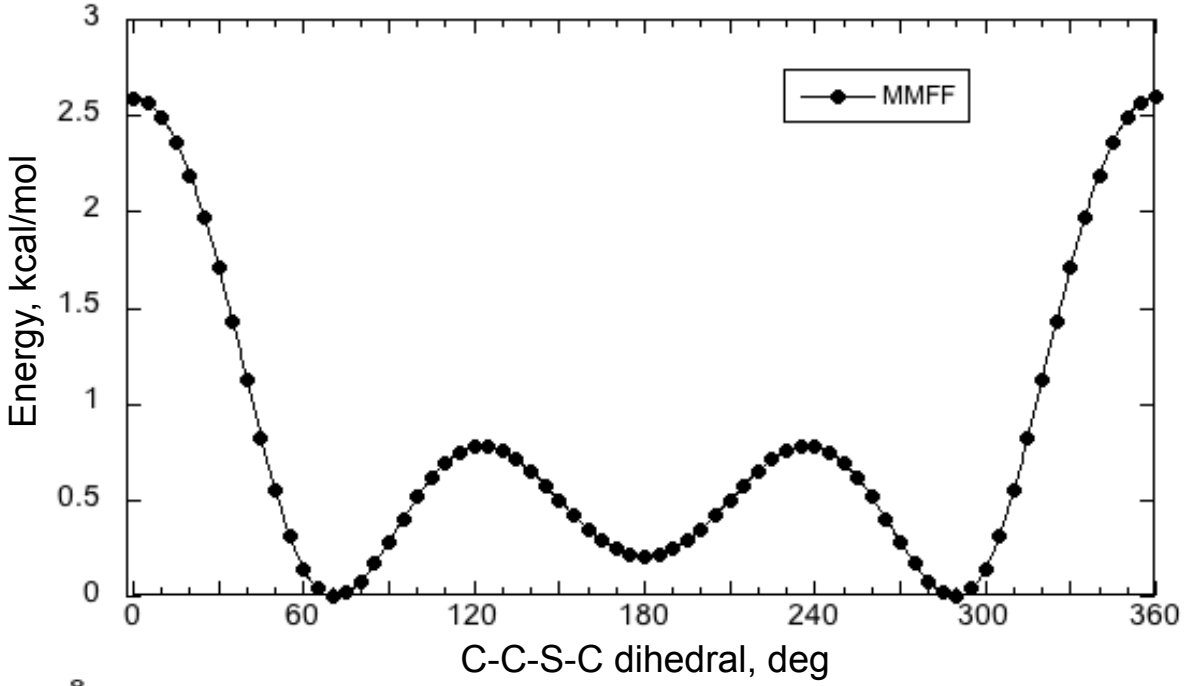
C–S = 1.77 ± 0.00 Å



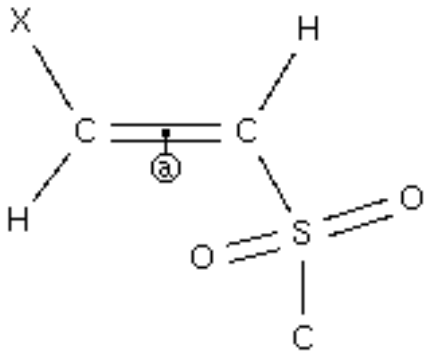
TYPE 2-1:8-4



Φ	E
70.0	0.00
180.0	0.21
290.0	0.00

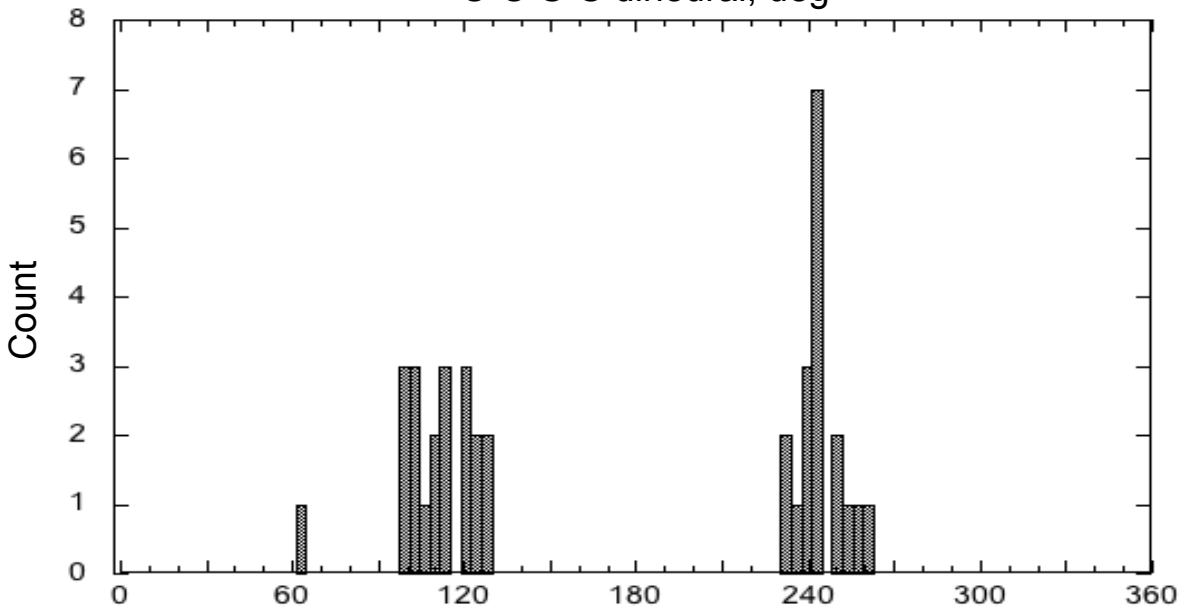


CSD comparison

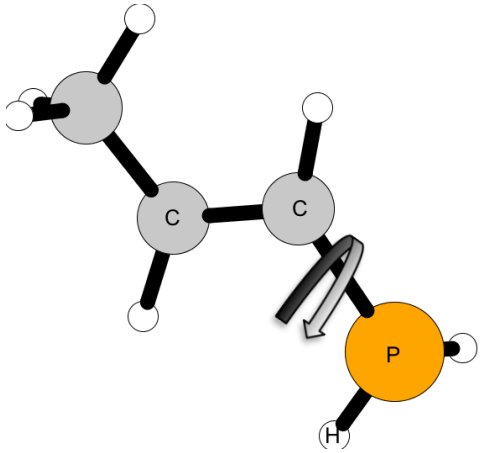


32 hits

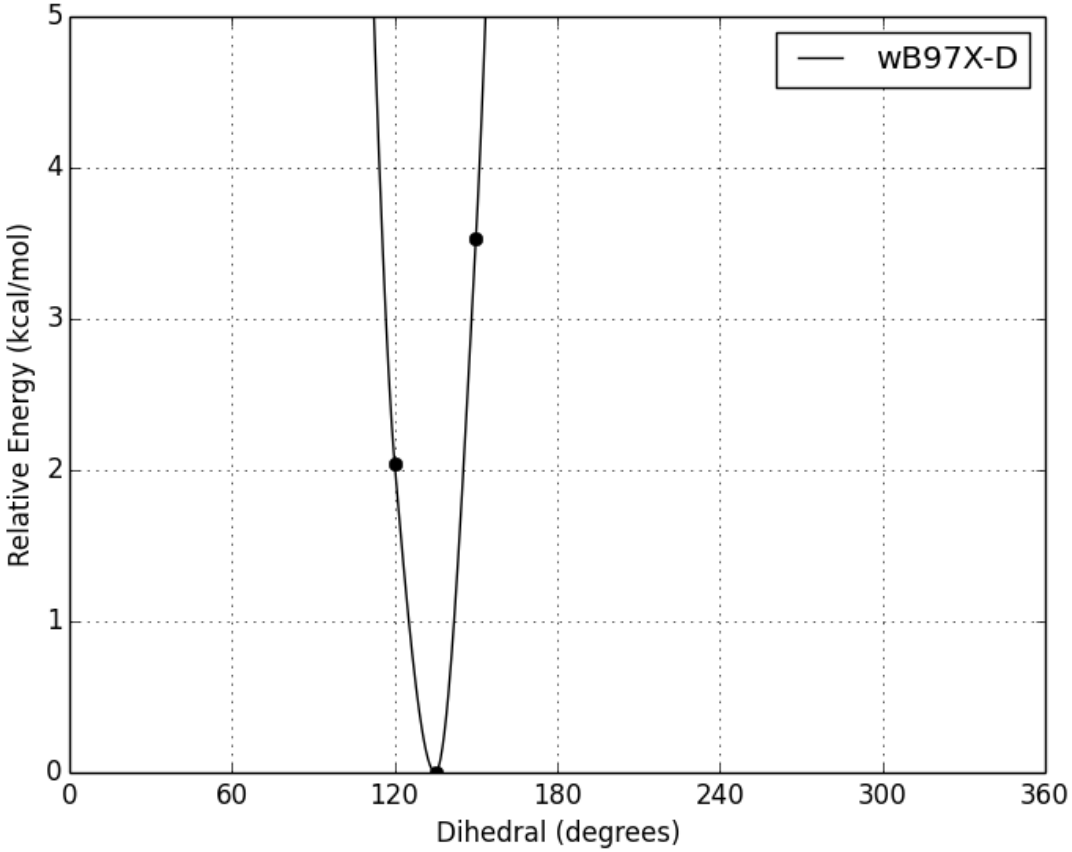
C–S = 1.74 ± 0.01 Å



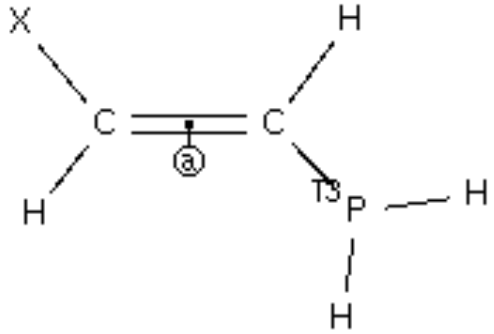
TYPE 2-1:9-1



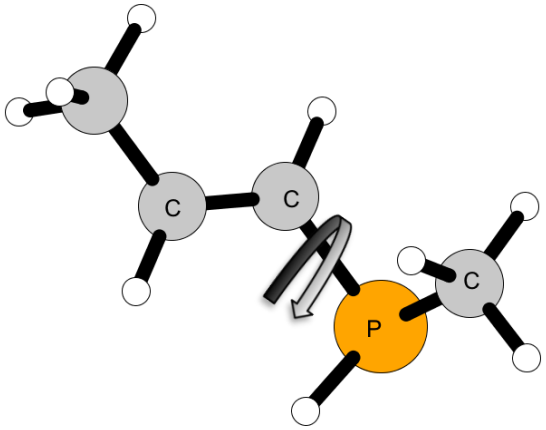
Φ	E
135.0	0.00



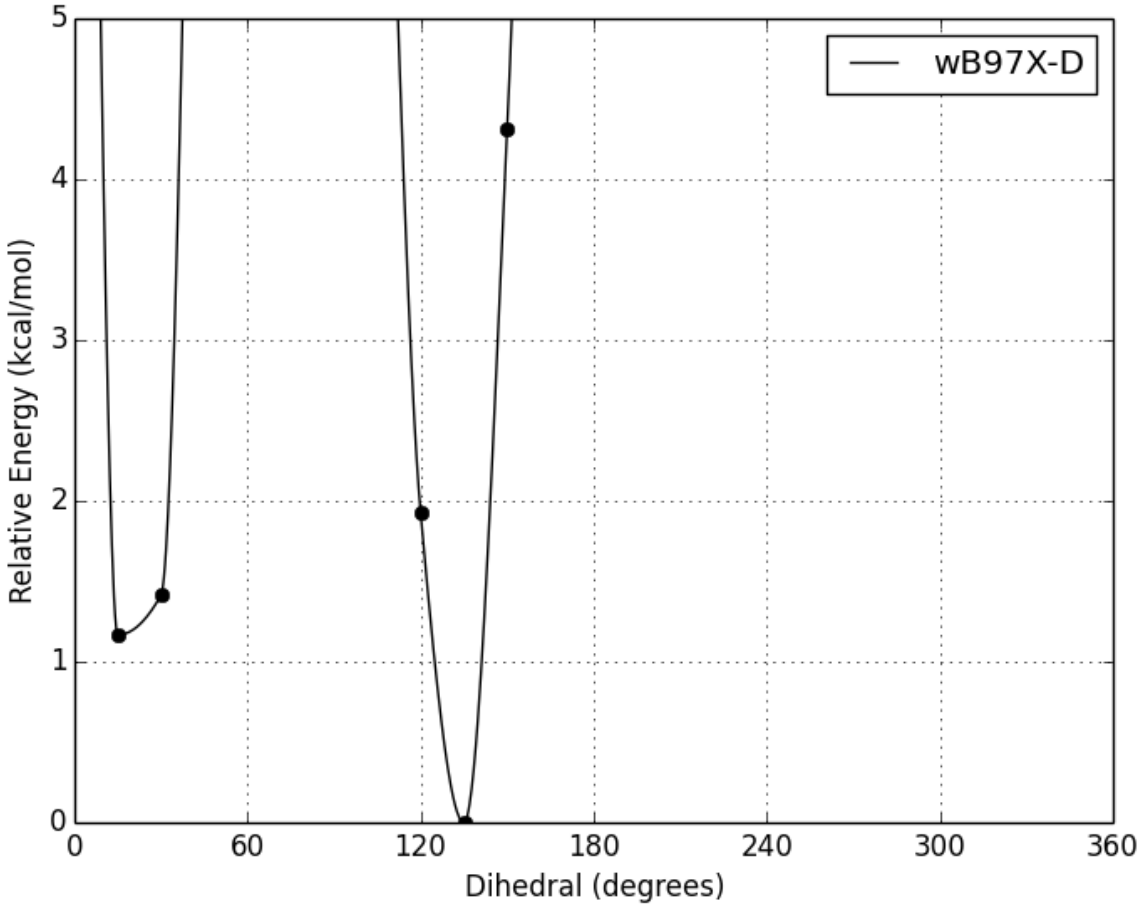
CSD comparison



NO CDB HITS

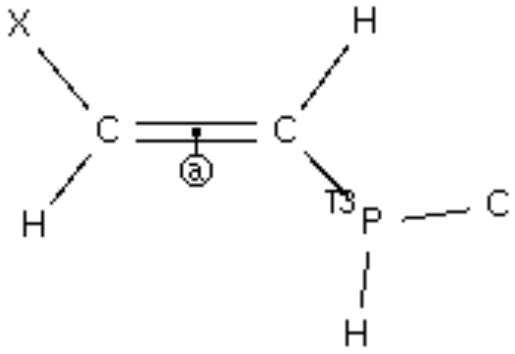


TYPE 2-1:9-2



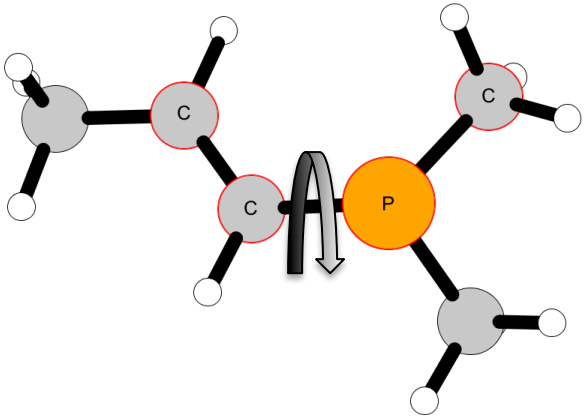
Φ	E
15.0	1.17
135.0	0.00

CSD comparison

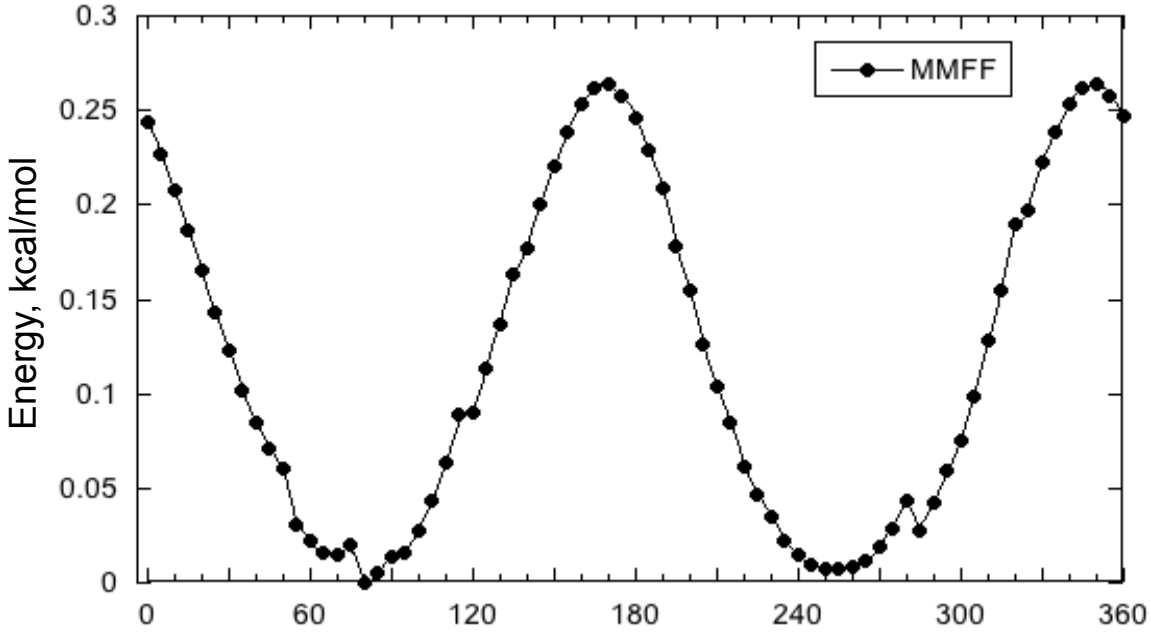


NO CDB HITS

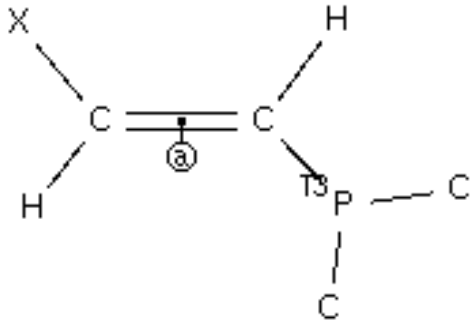
TYPE 2-1:9-4



Φ	E
80.0	0.00
270.0	0.01

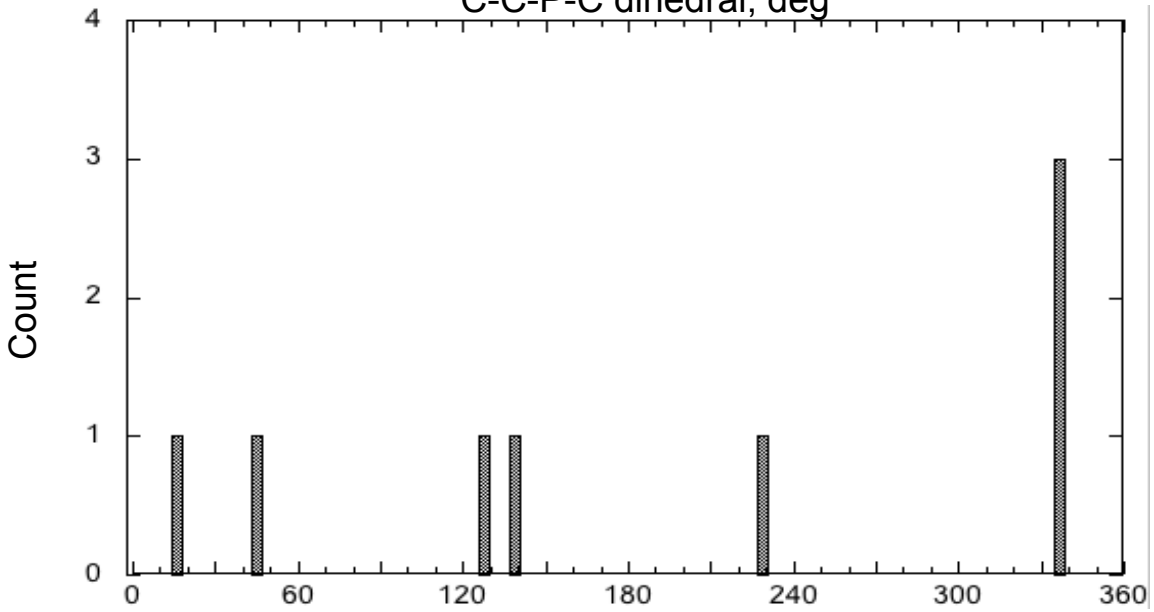


CSD comparison

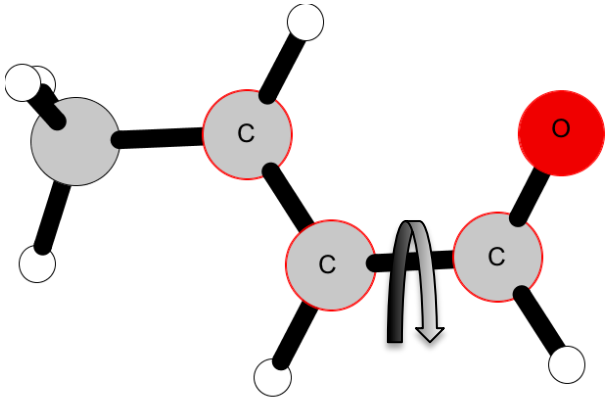


6 hits

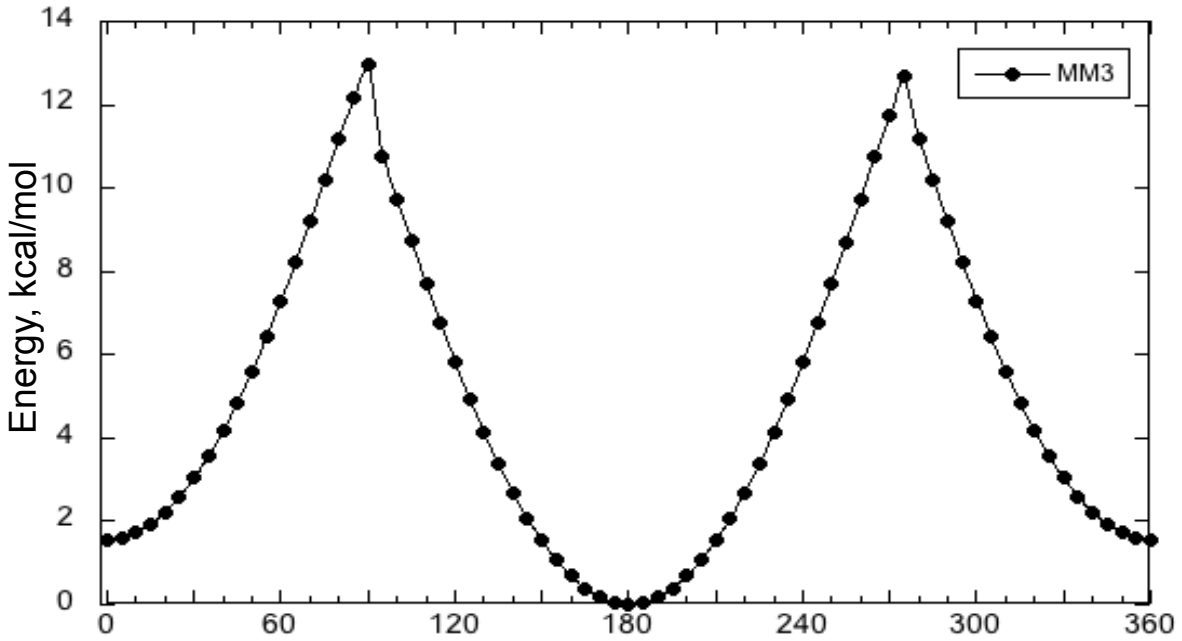
C–P = 1.81 ± 0.02 Å



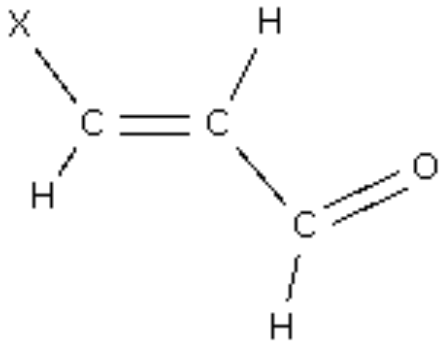
TYPE 2-1:10-1



Φ	E
0.0	1.57
180.0	0.00

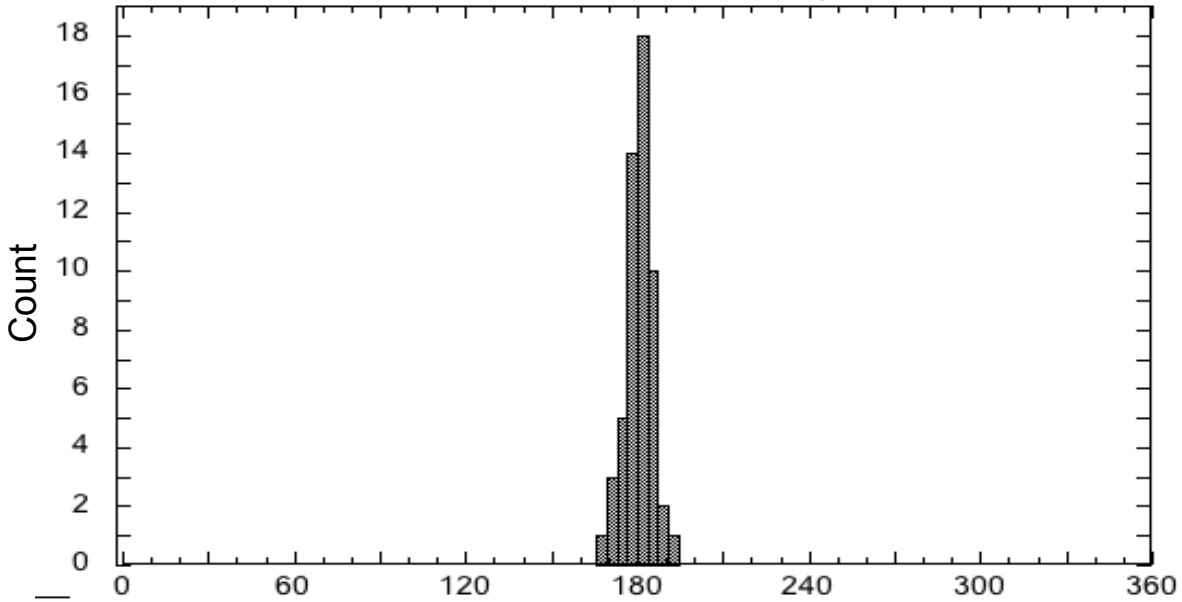


CSD comparison

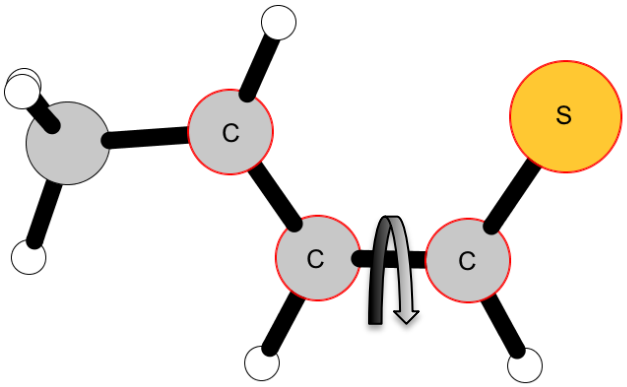


43 hits

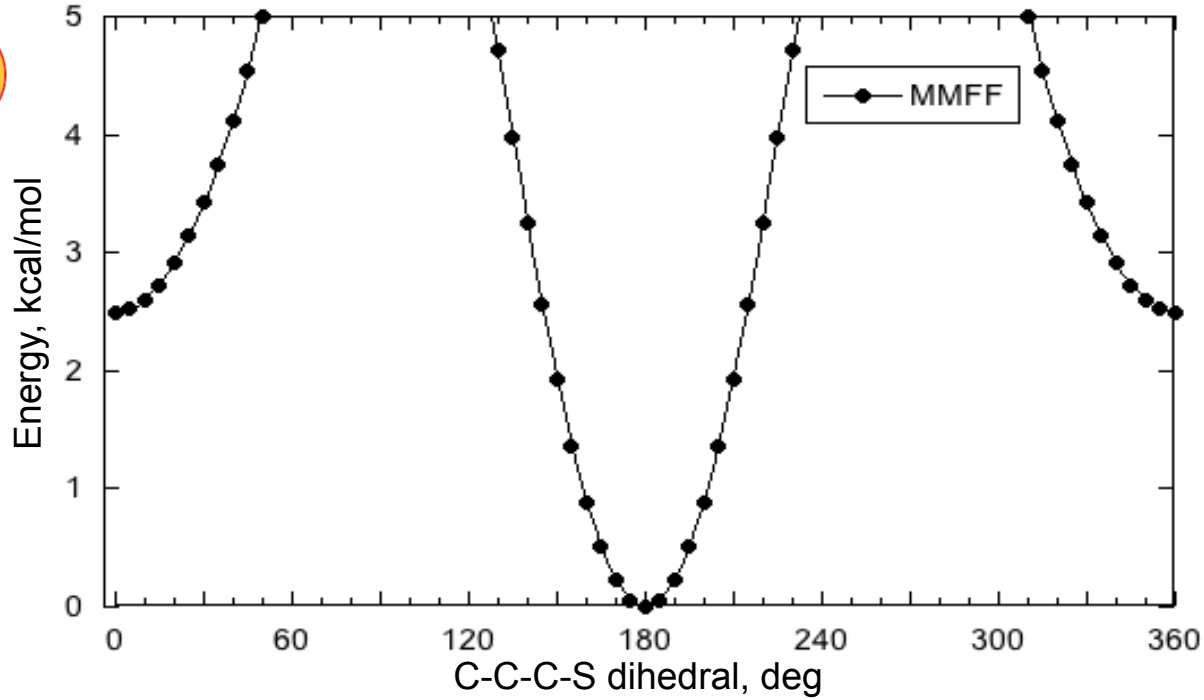
C–C = 1.43 ± 0.02 Å



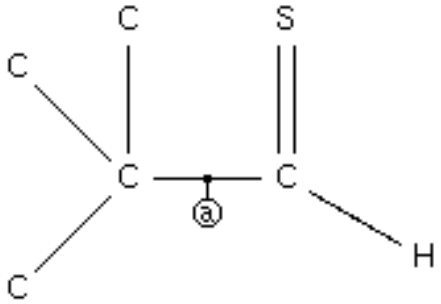
TYPE 2-1:10-2



Φ	E
0.0	2.50
180.0	0.00



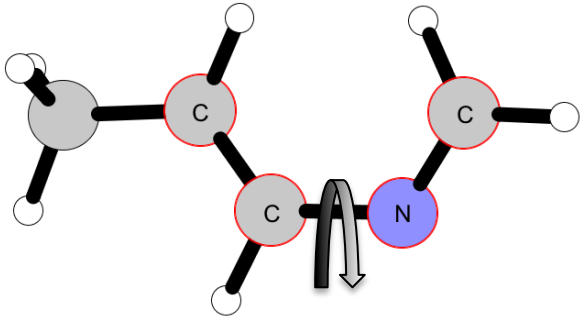
CSD comparison



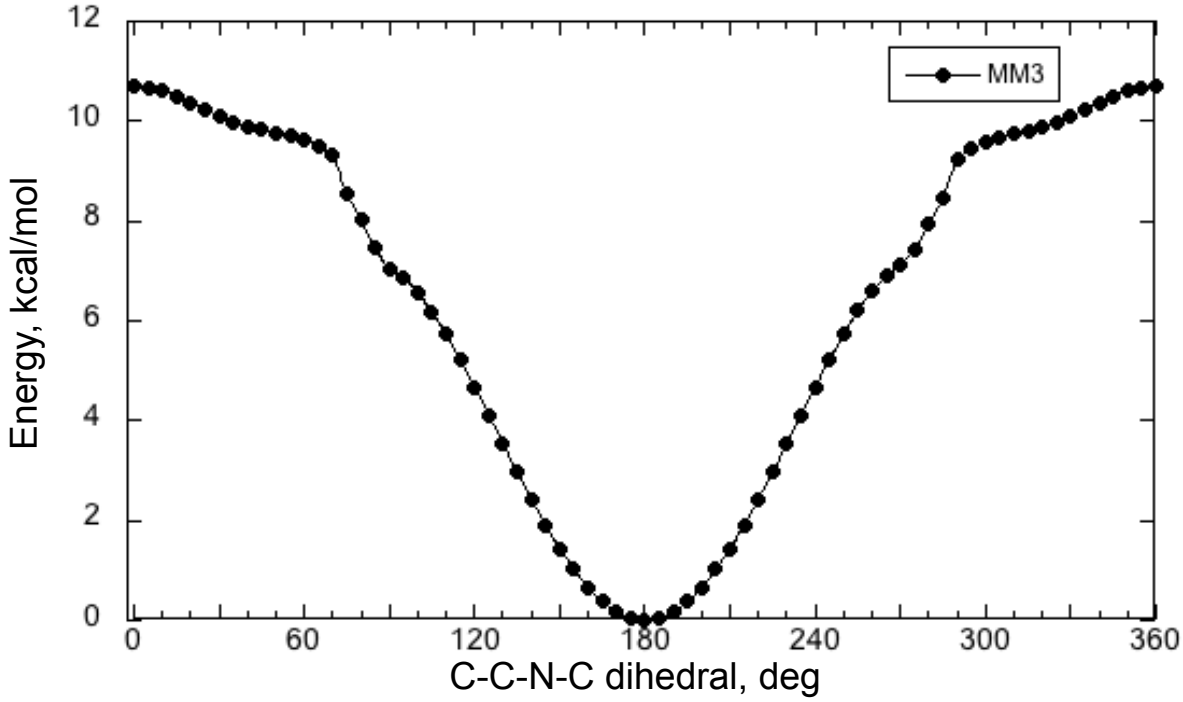
0 hits

NO CSD HITS

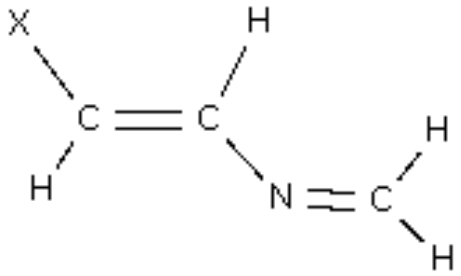
TYPE 2-1:10-3



Φ	E
180.0	0.00



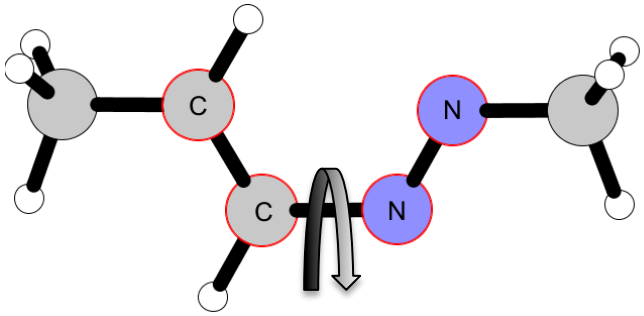
CSD comparison



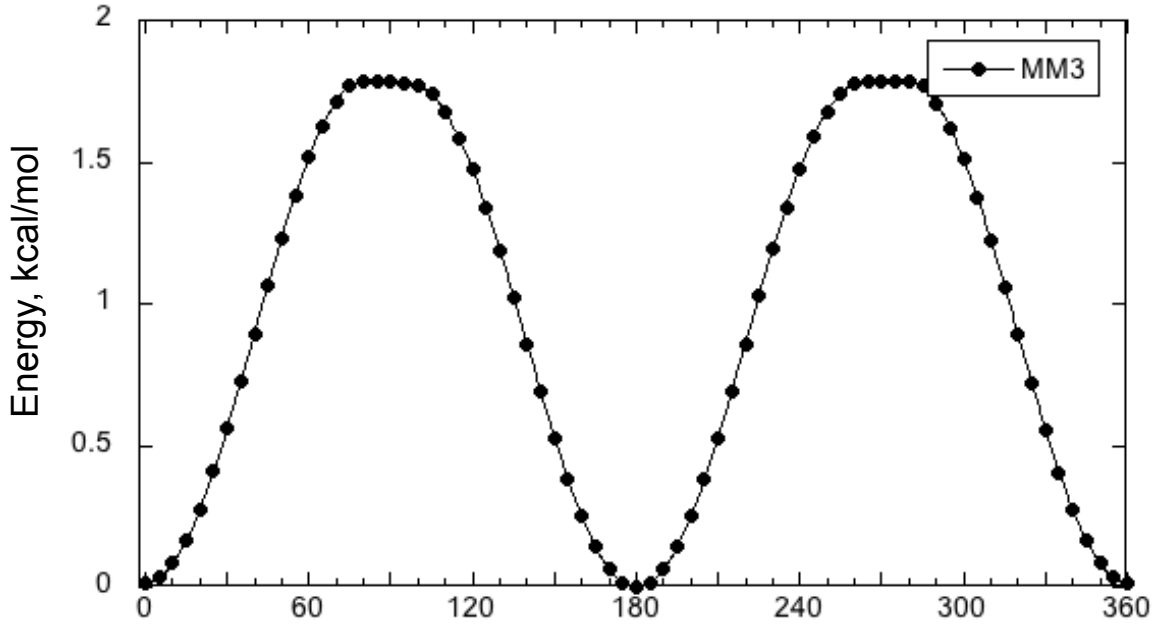
0 hits

NO CSD HITS

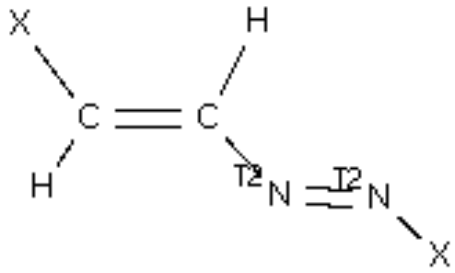
TYPE 2-1:10-4



Φ	E
0.00	0.02
180.00	0.00

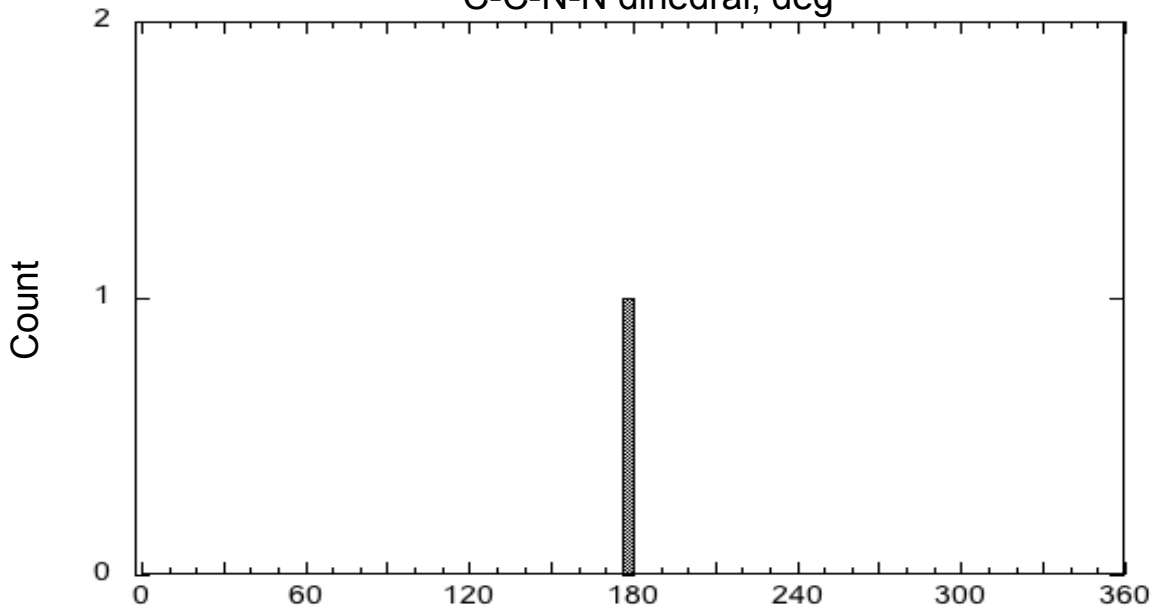


CSD comparison

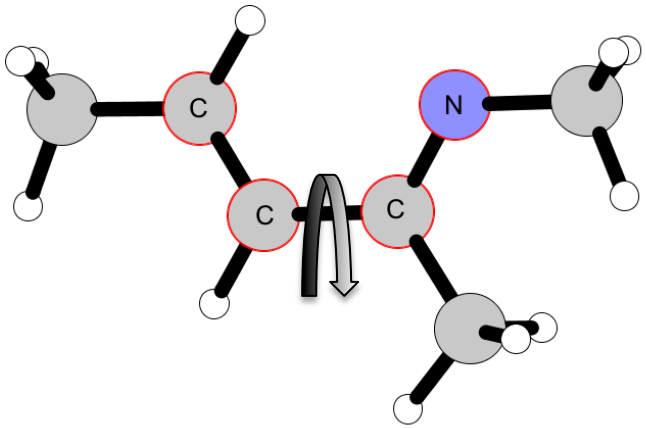


1 hit

C–N = 1.40 Å

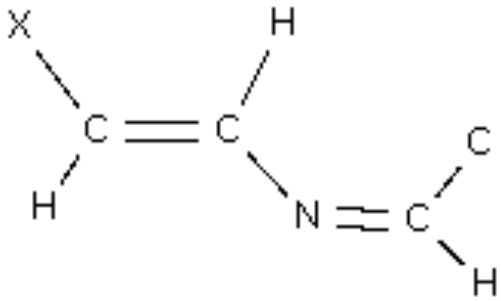


TYPE 2-1:11-1



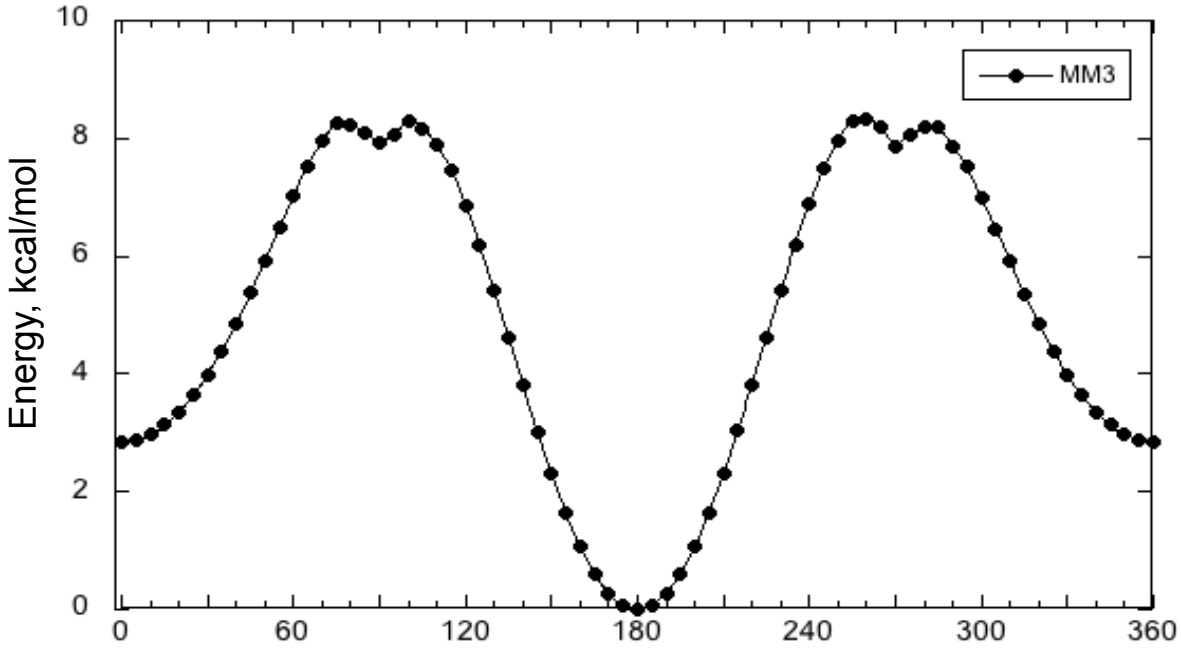
Φ	E
0.0	2.85
90.0	7.91
180.0	0.00
270.0	7.87

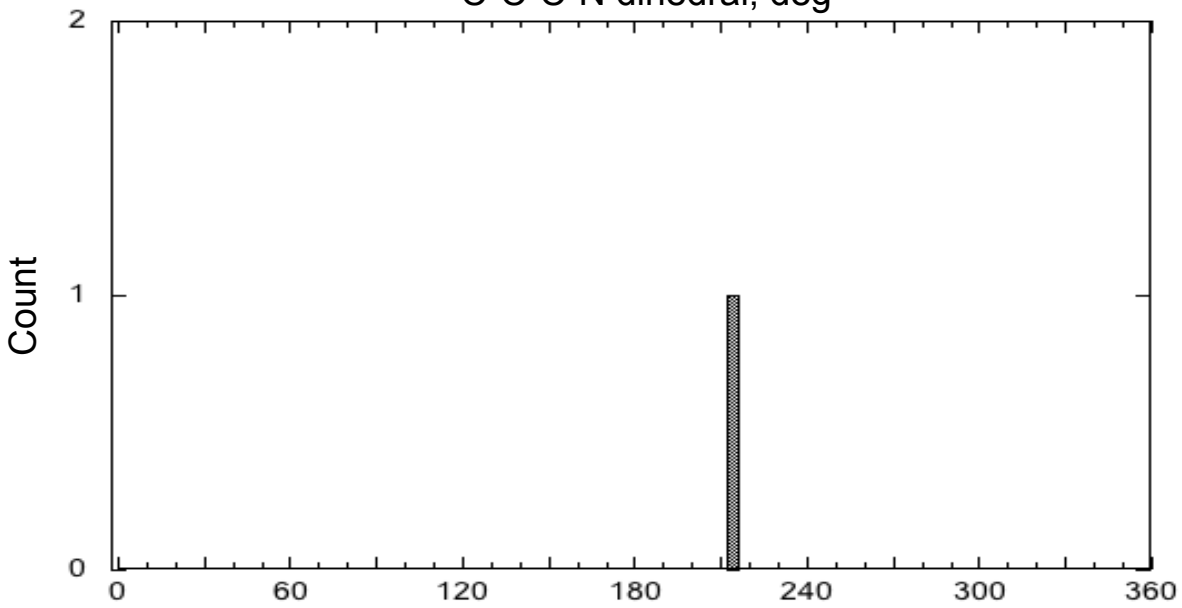
CSD comparison



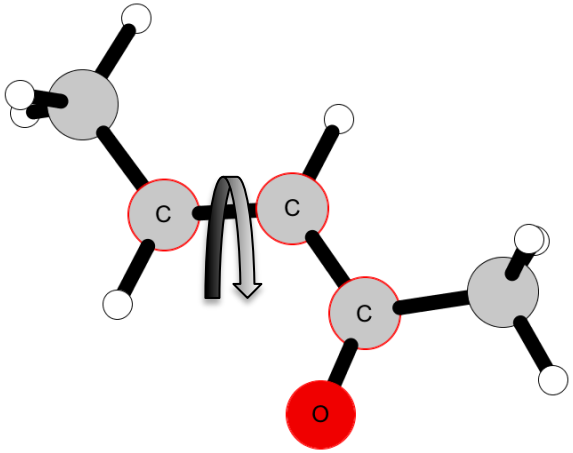
1 hits

C–C = 1.46 Å

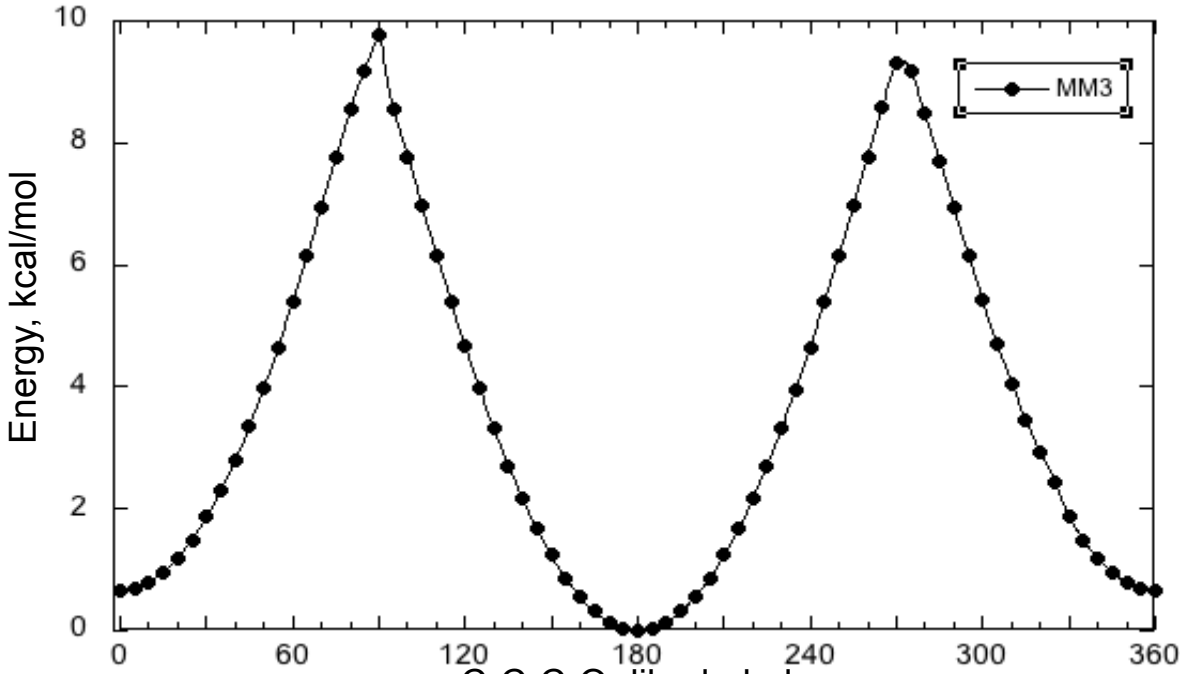




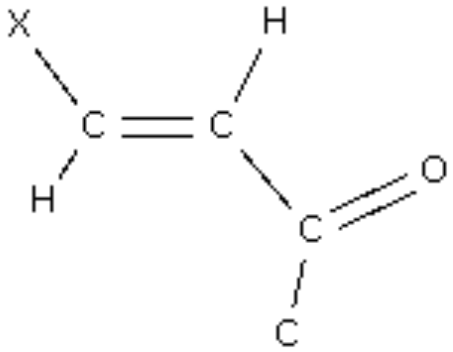
TYPE 2-1:11-2



Φ	E
0.0	0.66
180.0	0.00

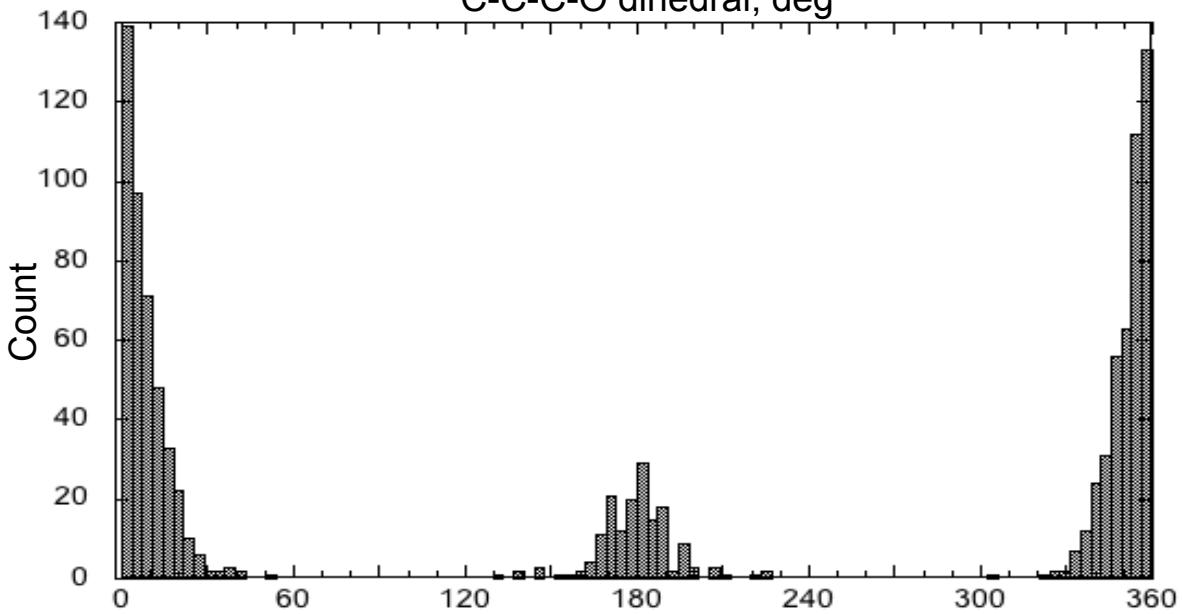


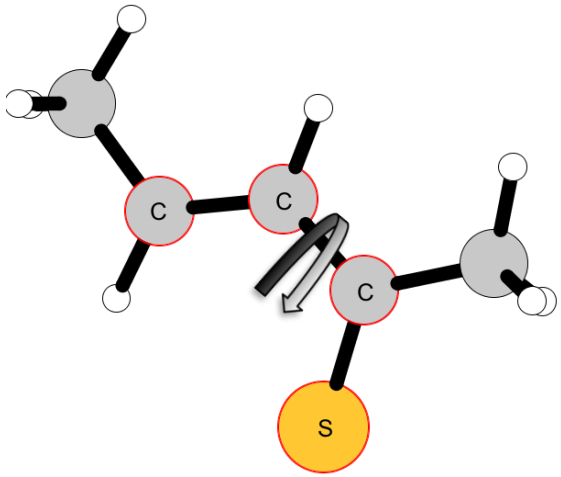
CSD comparison



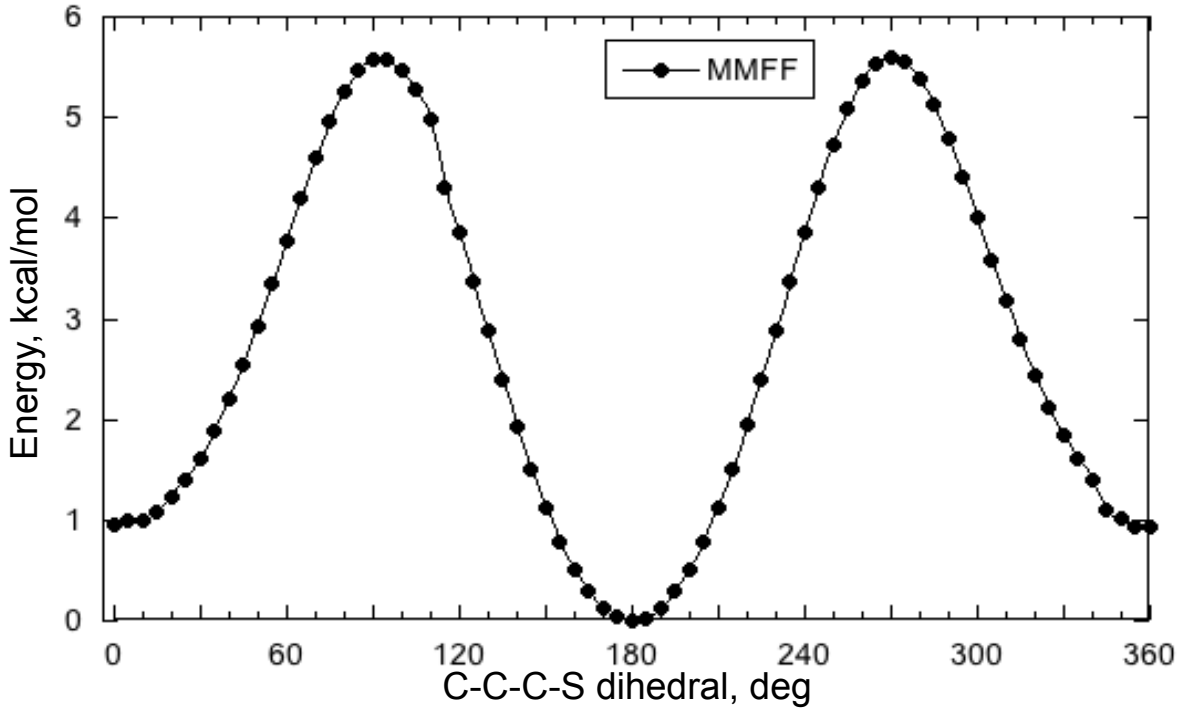
860 hits

C–C = 1.47 ± 0.02 Å



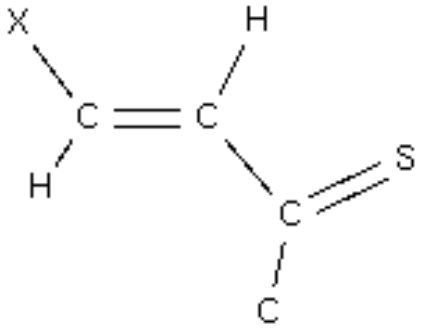


TYPE 2-1:11-3



Φ	E
0.0	0.95
180.0	0.00

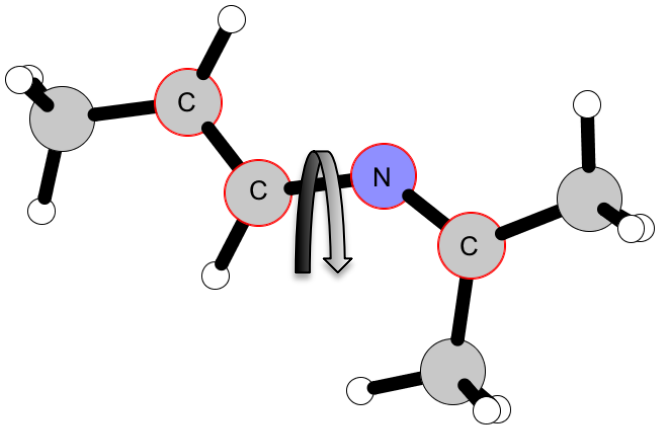
CSD comparison



0 hits

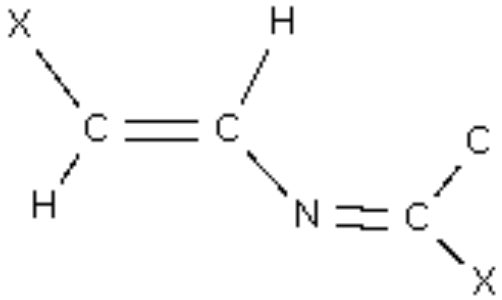
NO CSD HITS

TYPE 2-1:12-1



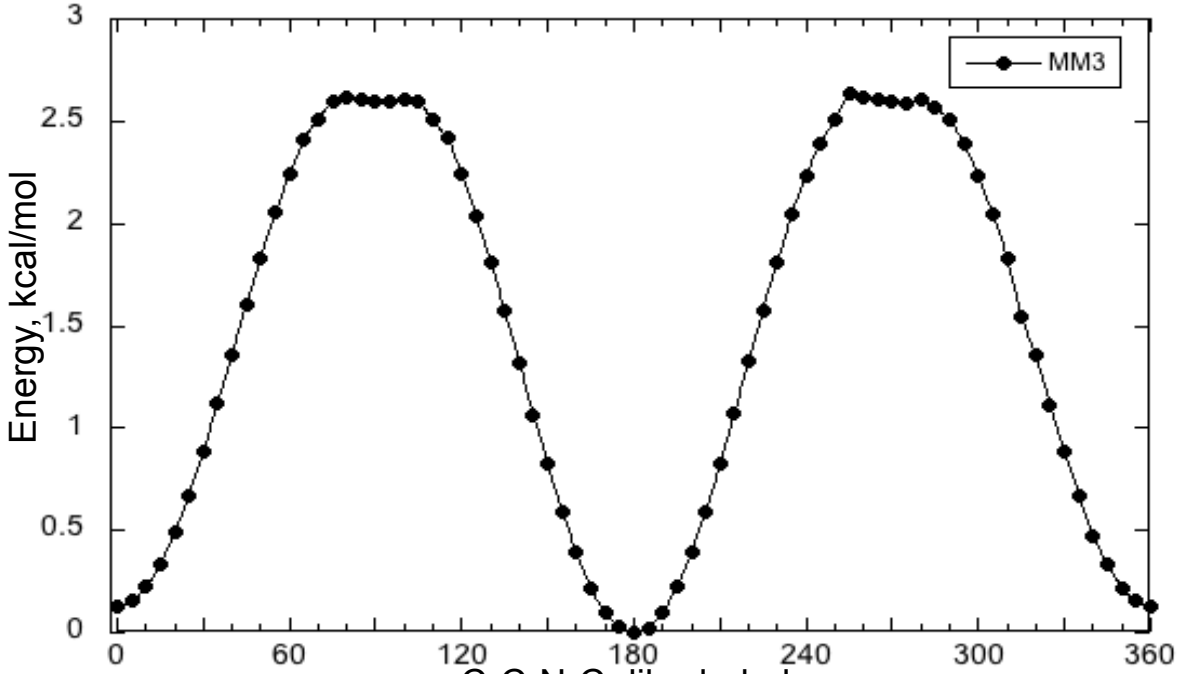
Φ	E
0.0	0.13
90.0	2.60
180.0	0.00
270.0	2.60

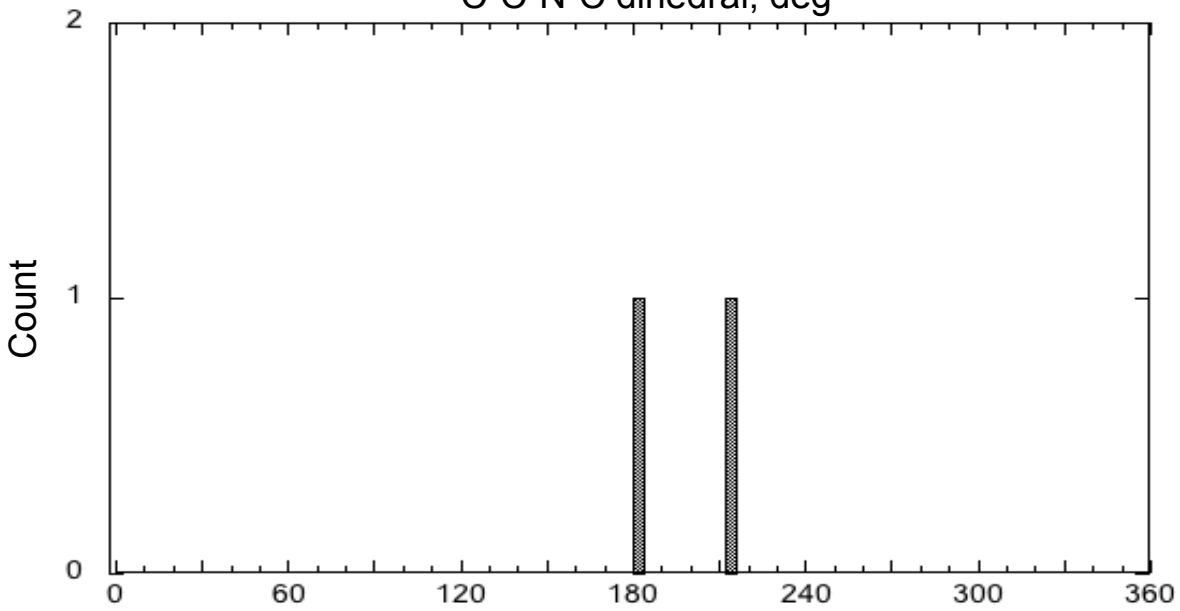
CSD comparison



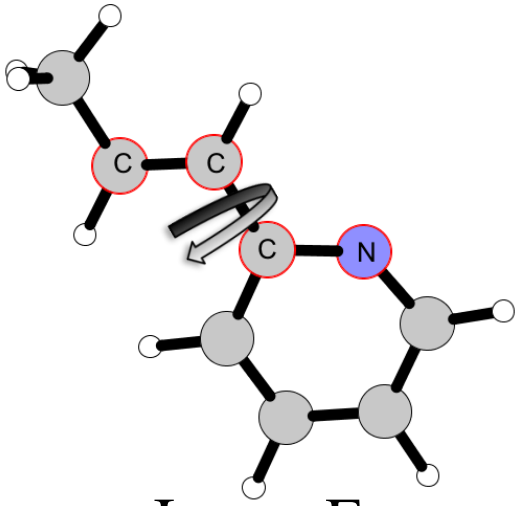
2 hits

C–N = 1.41 ± 0.06 Å

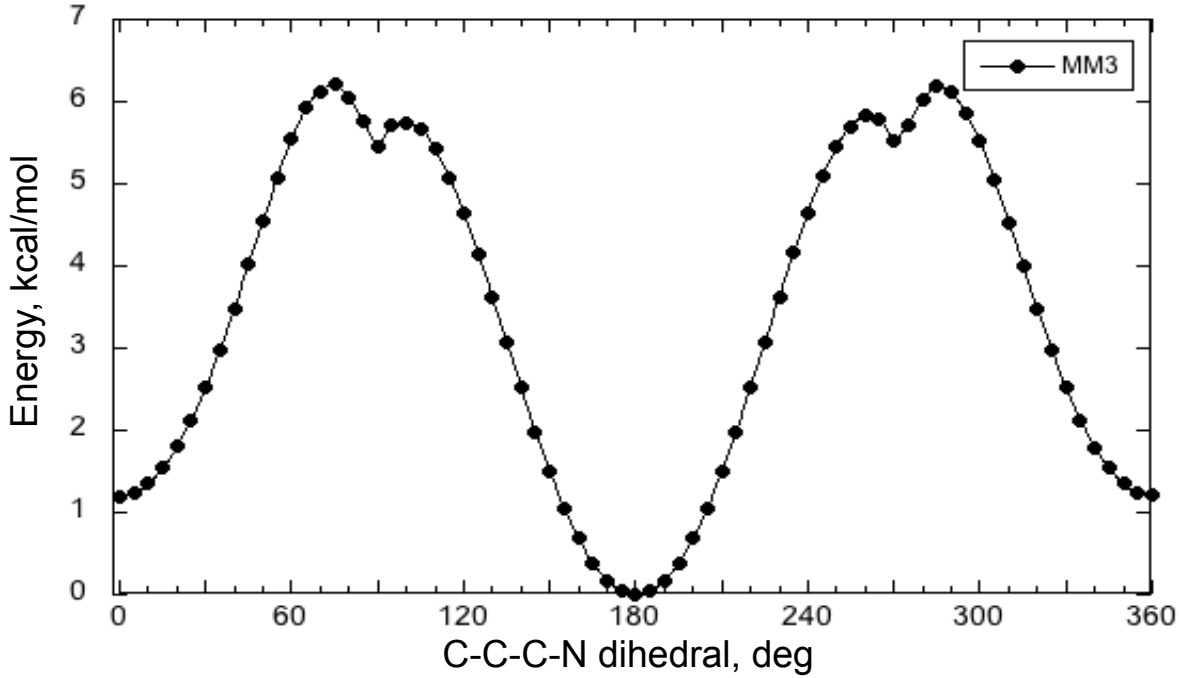




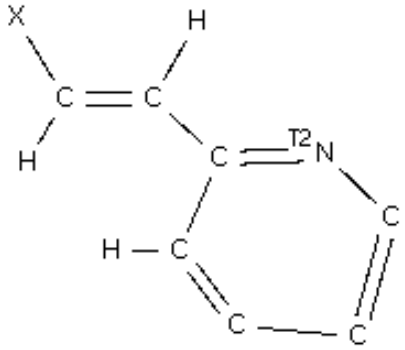
TYPE 2-1:13-1



Φ	E
0.0	1.20
90.0	5.44
180.0	0.00
270.0	5.52

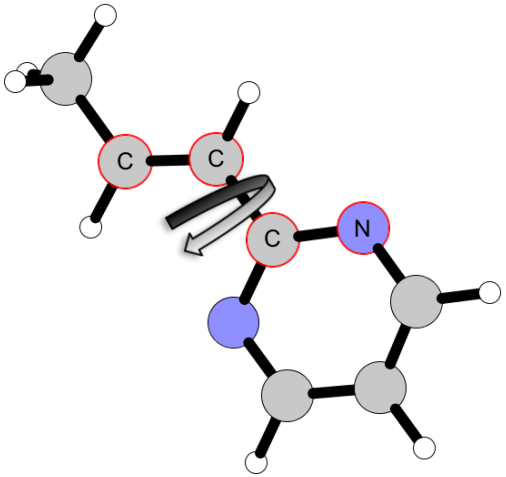


CSD comparison

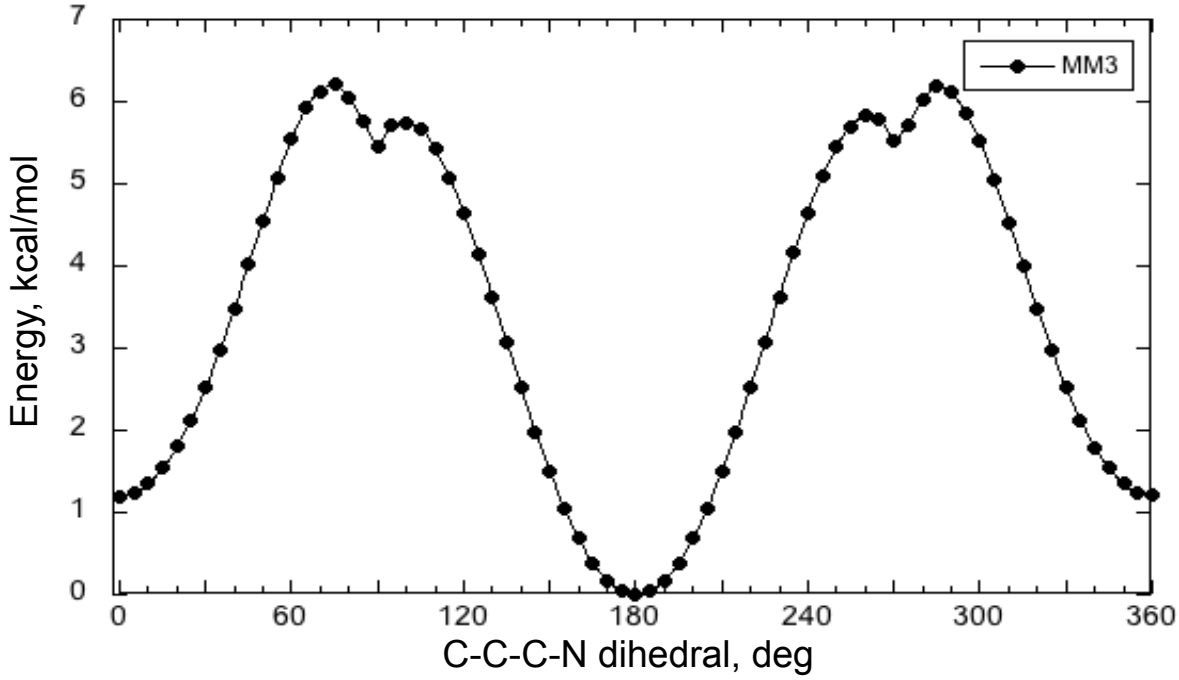


0 hits

NO CSD HITS

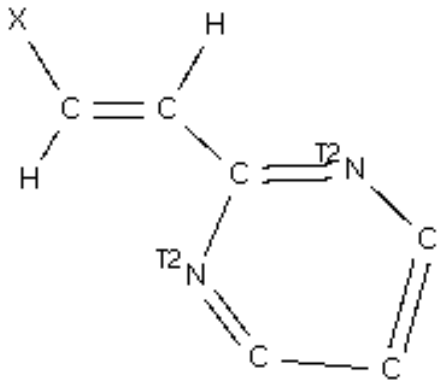


TYPE 2-1:13-2



Φ	E
0.0	1.20
90.0	5.44
180.0	0.00
270.0	5.52

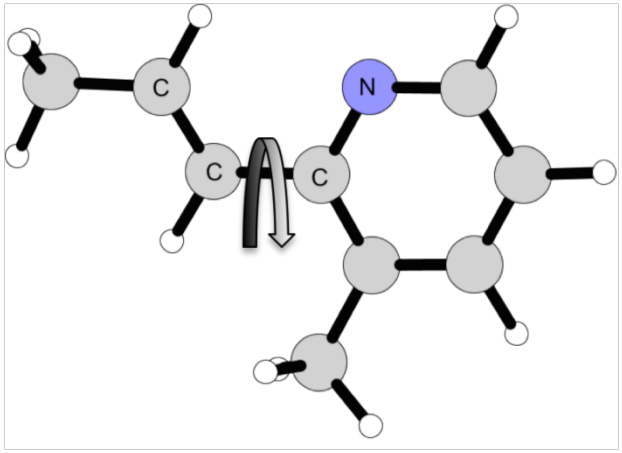
CSD comparison



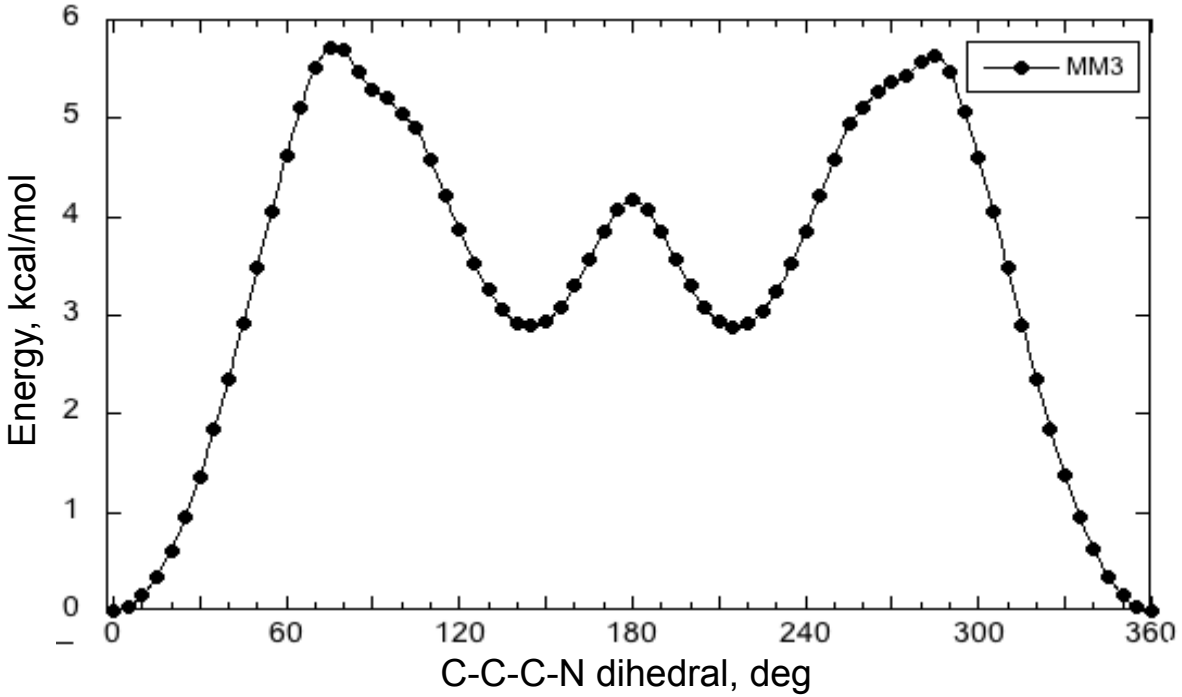
0 hits

NO CSD HITS

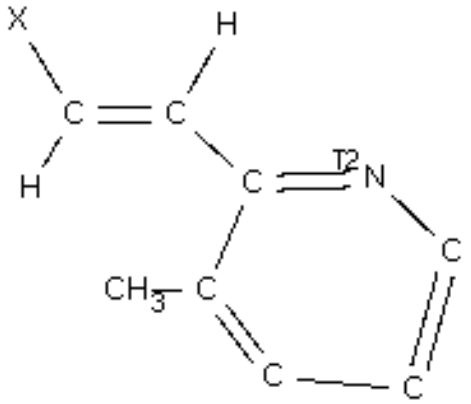
TYPE 2-1:13-3



Φ	E
0.0	0.00
145.0	2.89
215.0	2.88



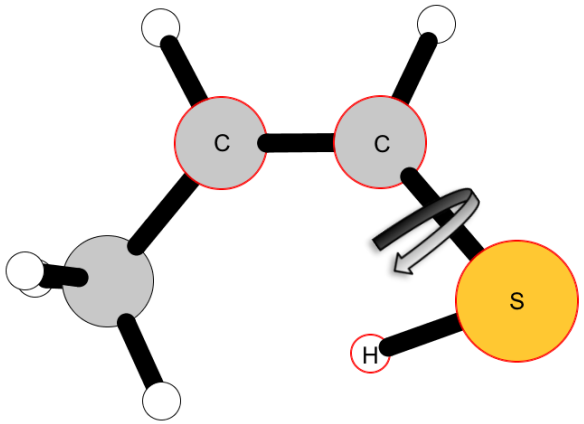
CSD comparison



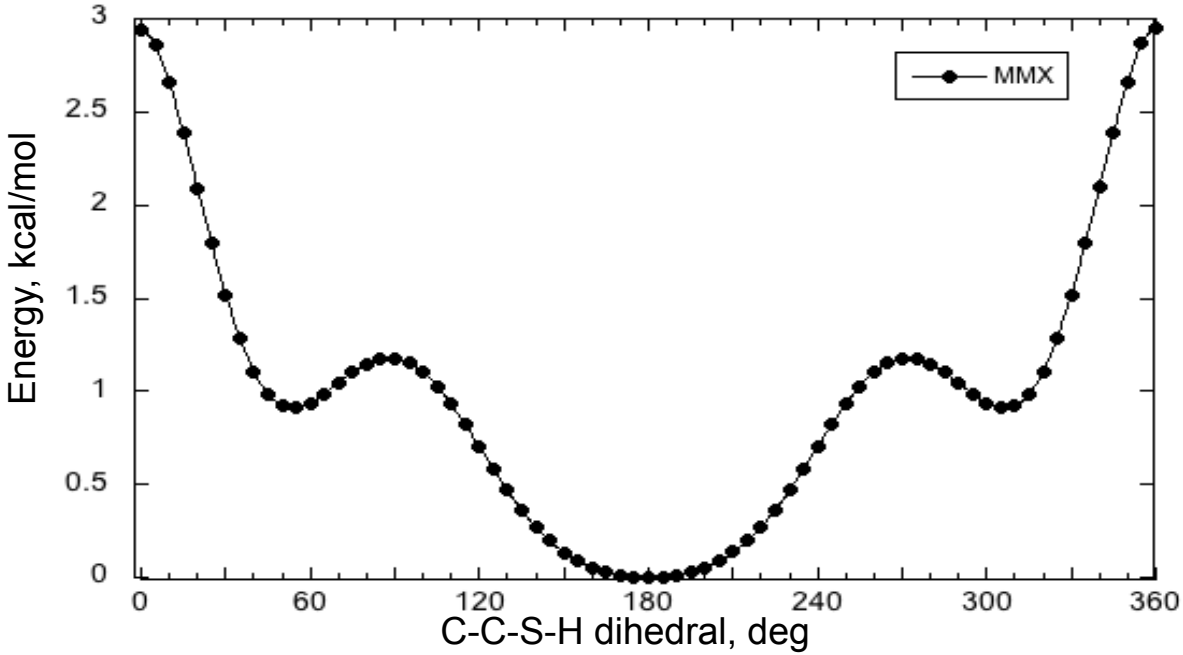
0 hits

NO CSD HITS

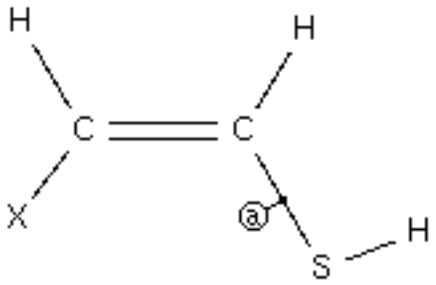
TYPE 2-2:5-4



Φ	E
55.0	0.91
180.0	0.00
305.0	0.91



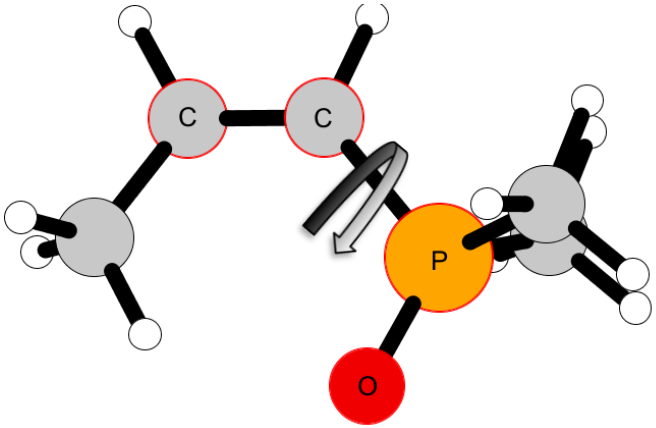
CSD comparison



0 hits

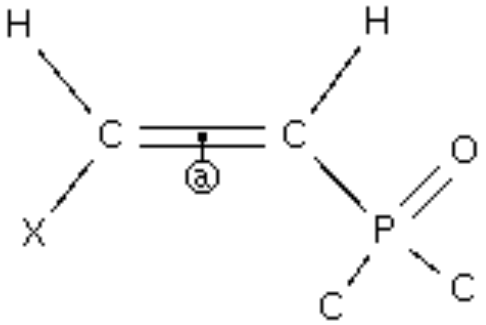
NO CSD HITS

TYPE 2-2:8-1



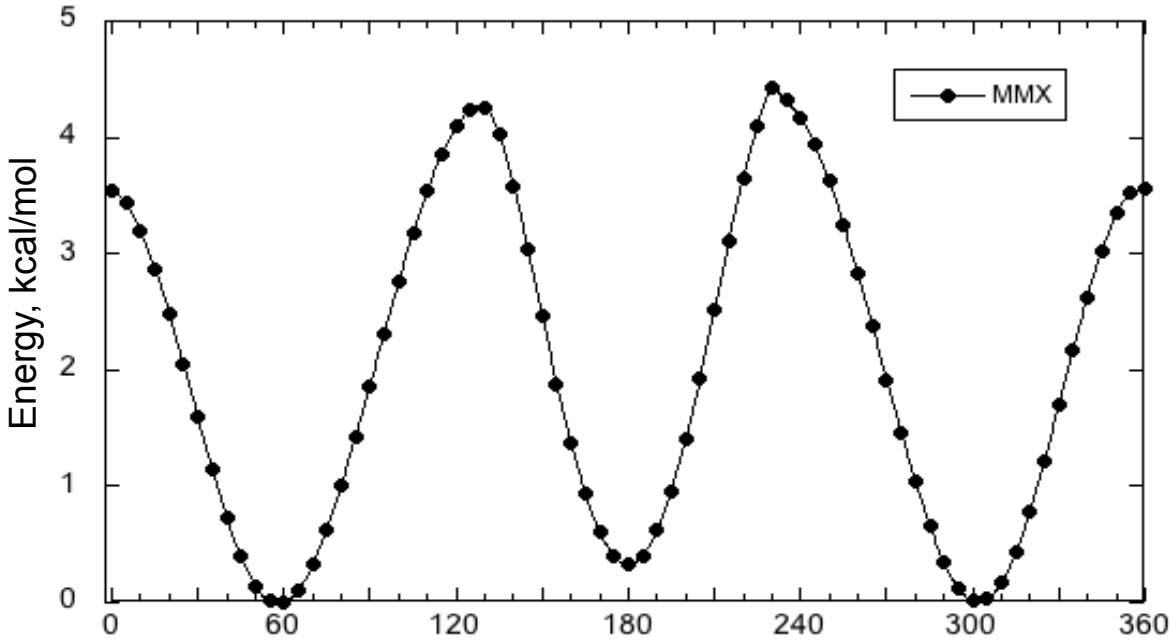
Φ	E
60.0	0.00
180.0	0.33
305.0	0.03

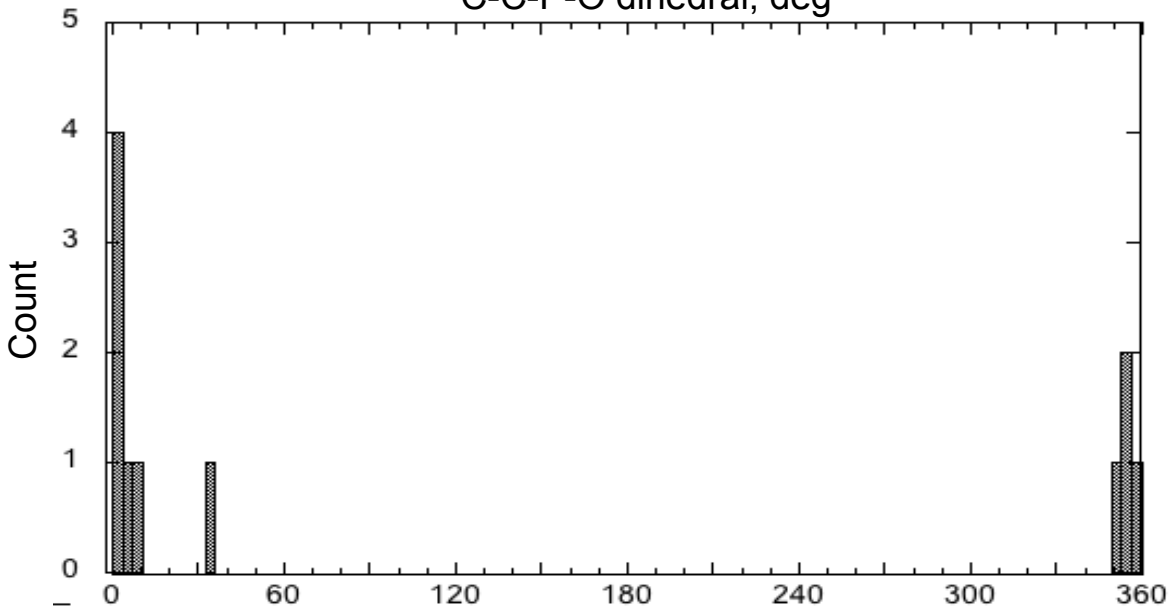
CSD comparison



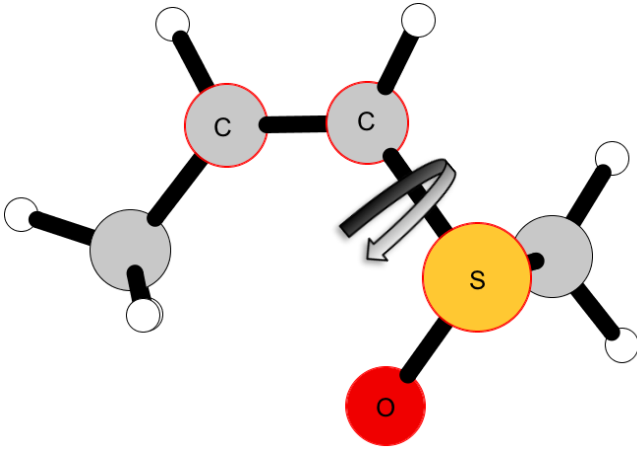
10 hits

C–P = 1.78 ± 0.02 Å

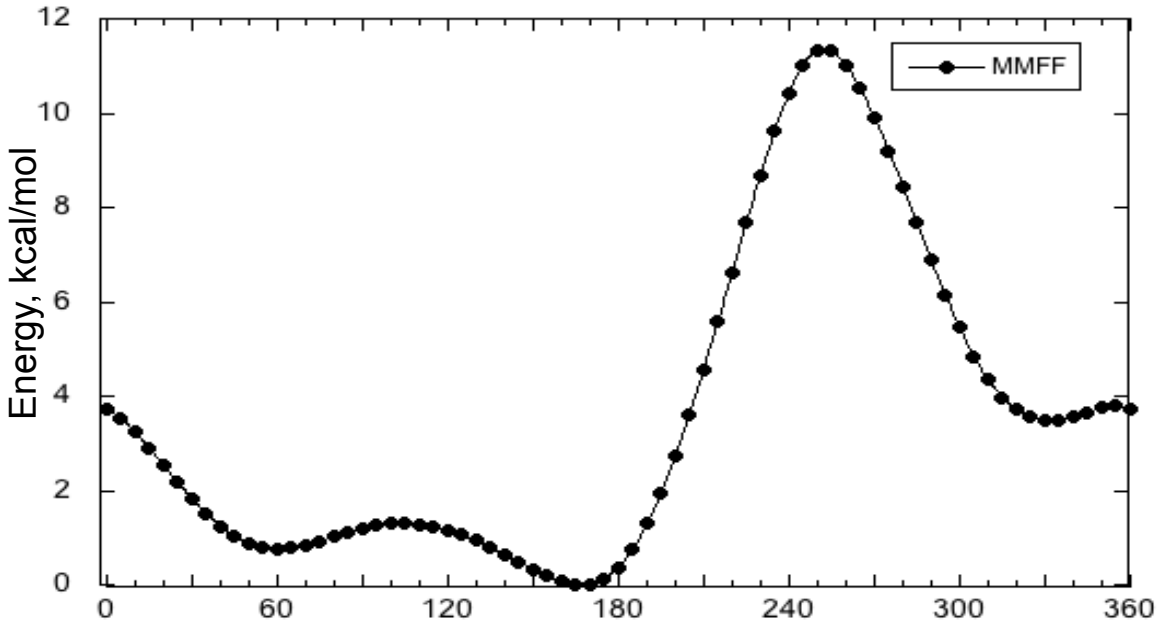




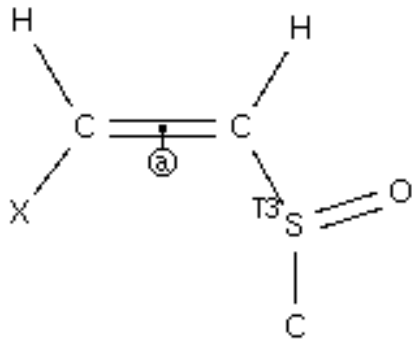
TYPE 2-2:8-2



Φ	E
60.0	0.77
165.00	0.00
330.00	3.49

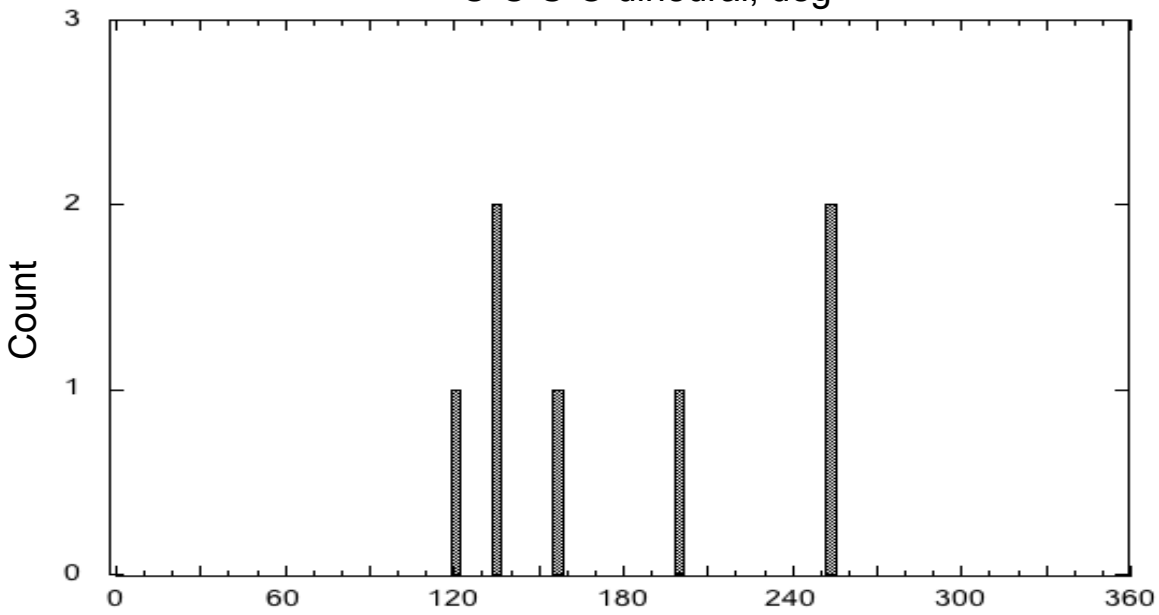


CSD comparison

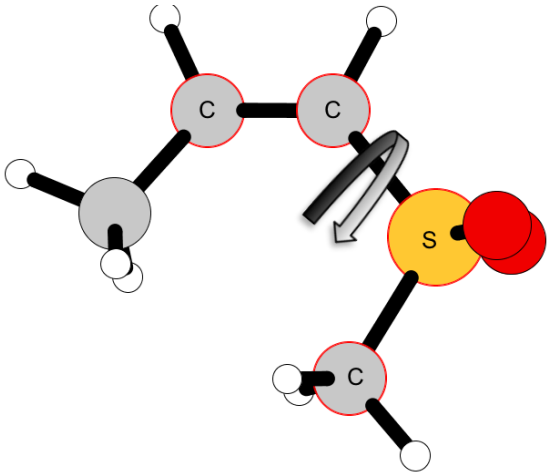


6 hits

C–S = 1.78 ± 0.02 Å

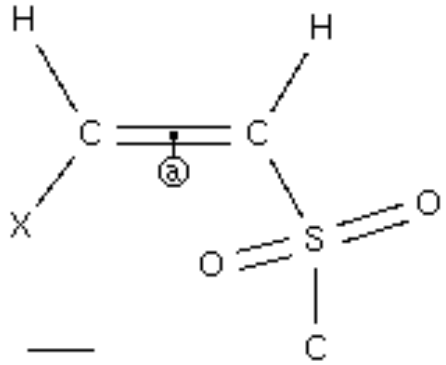


TYPE 2-2:8-4



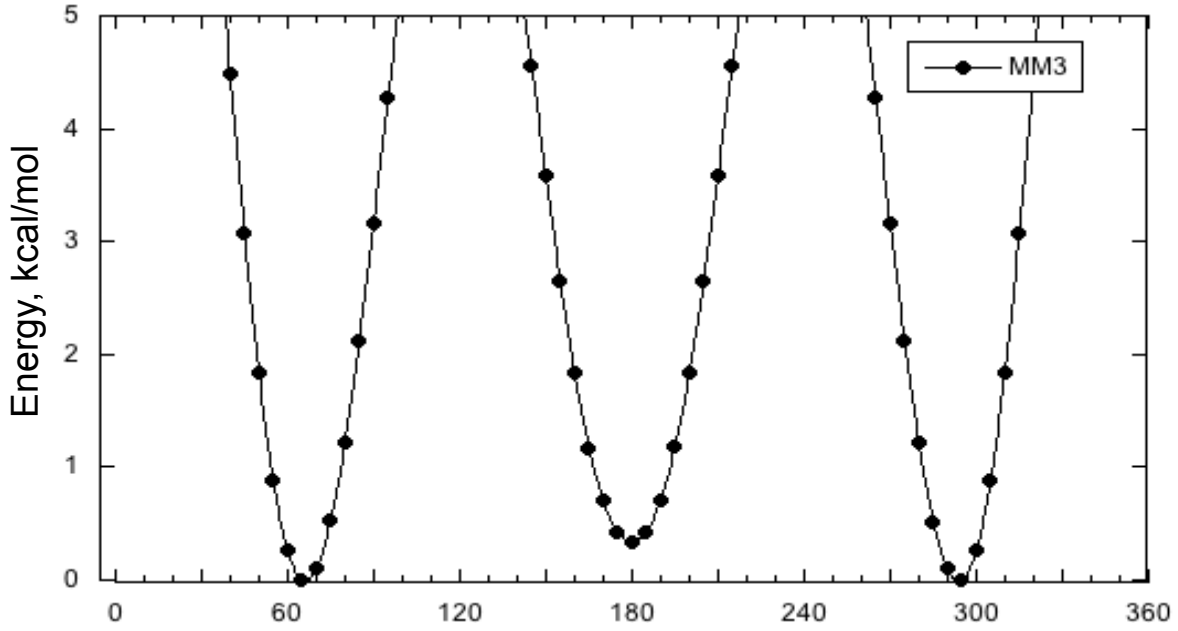
Φ	E
65.0	0.00
180.0	0.34
295.0	0.00

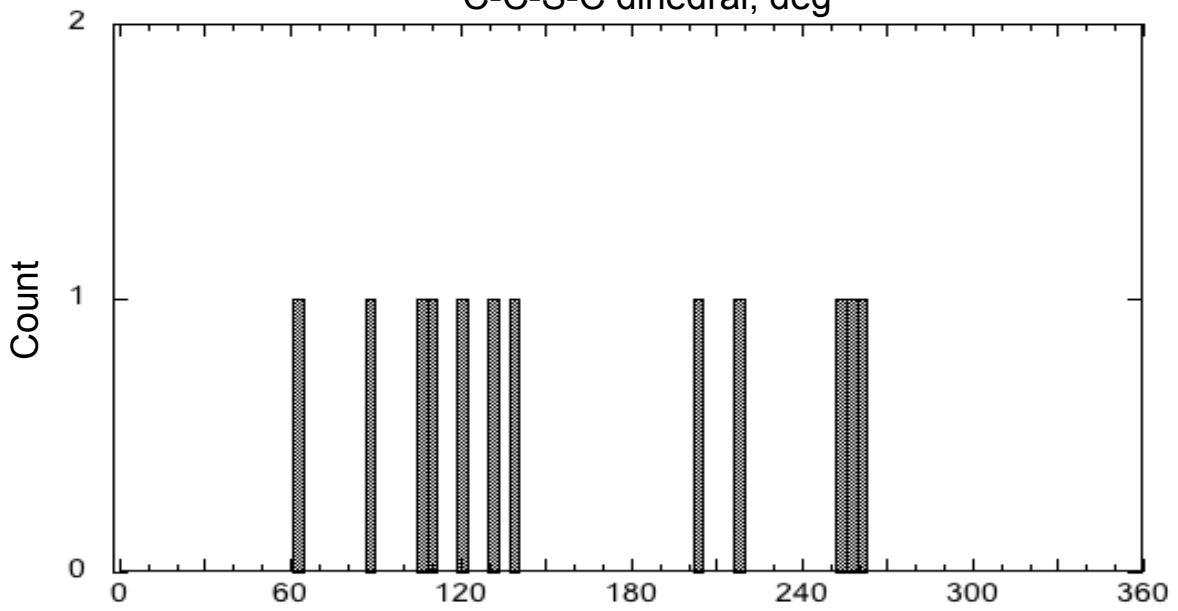
CSD comparison



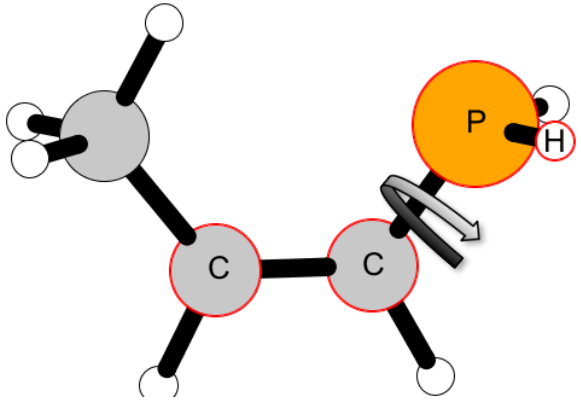
10 hits

C–S = 1.72 ± 0.11 Å

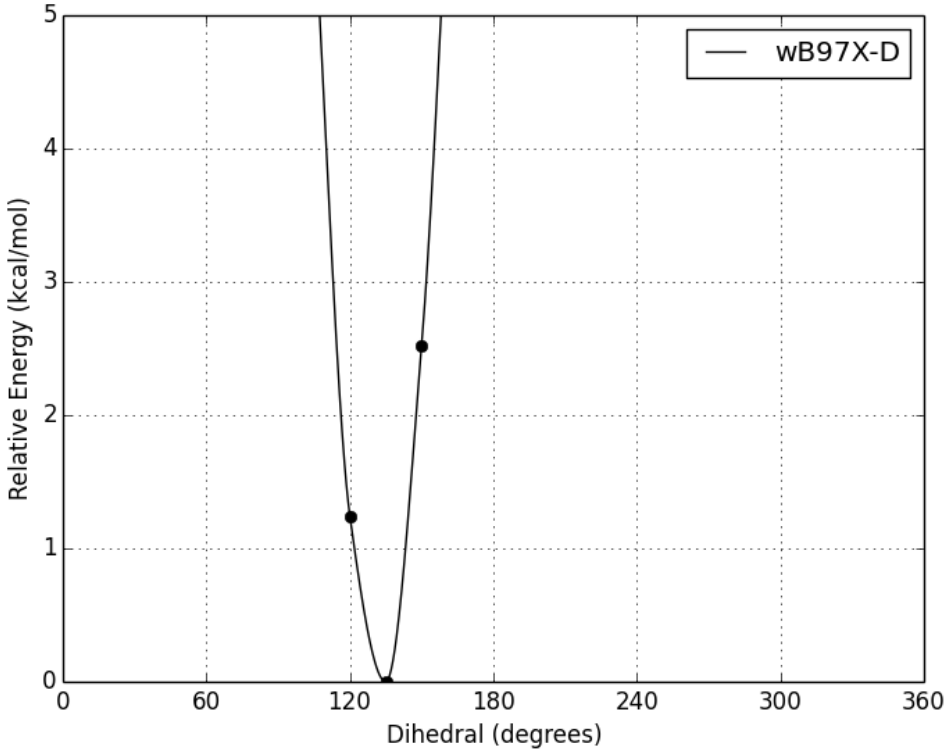




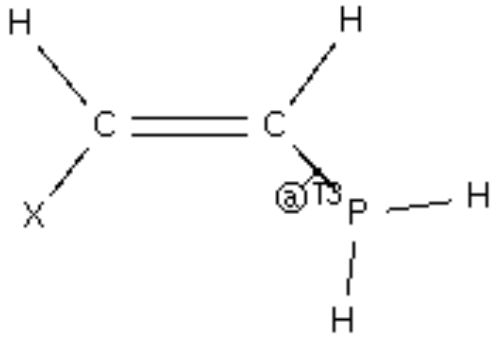
TYPE 2-2:9-1



Φ	E
135.0	0.00

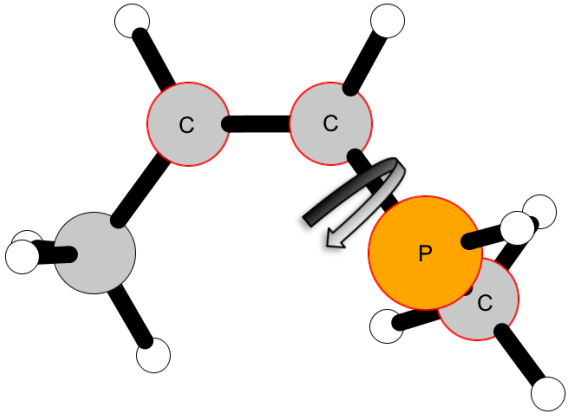


CSD comparison

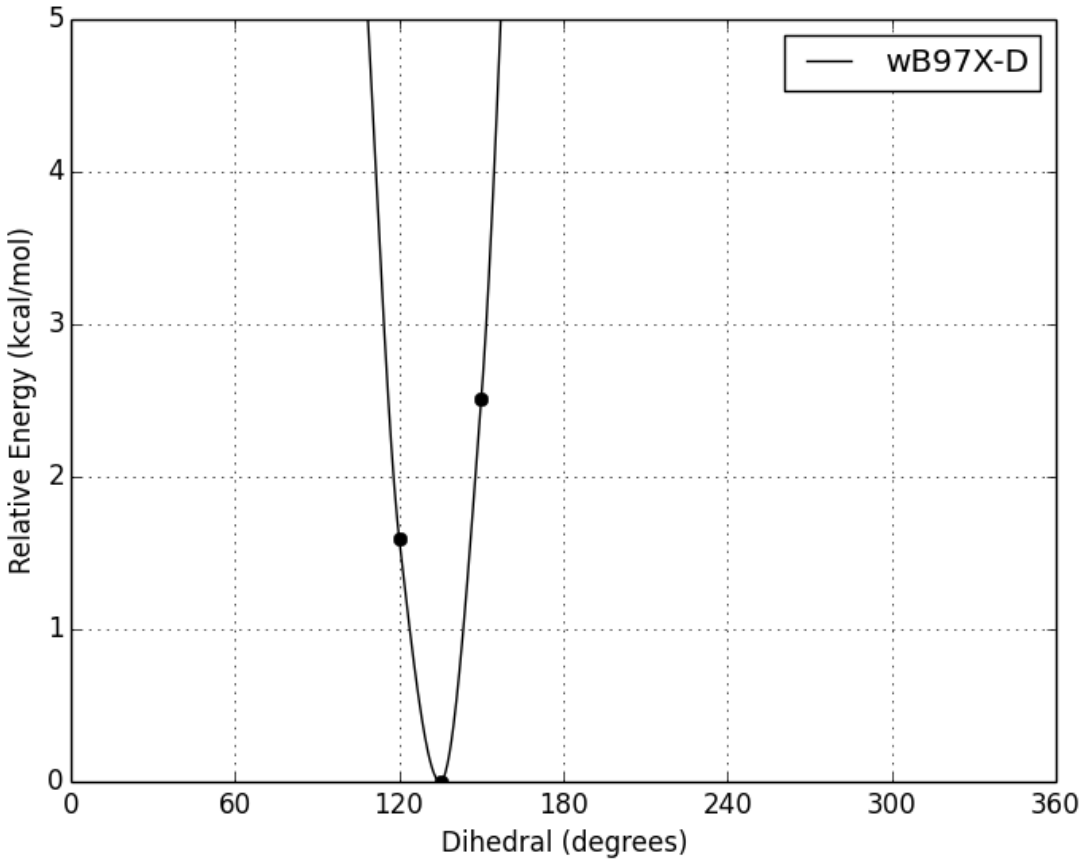


NO CDB HITS

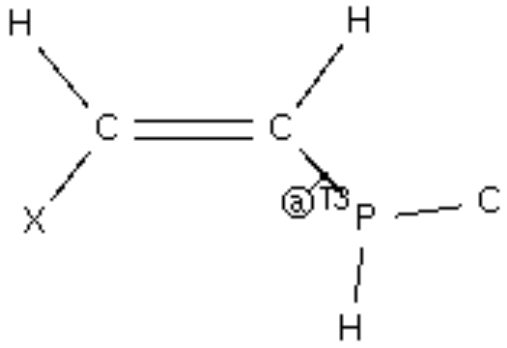
TYPE 2-2:9-2



Φ	E
135.0	0.00

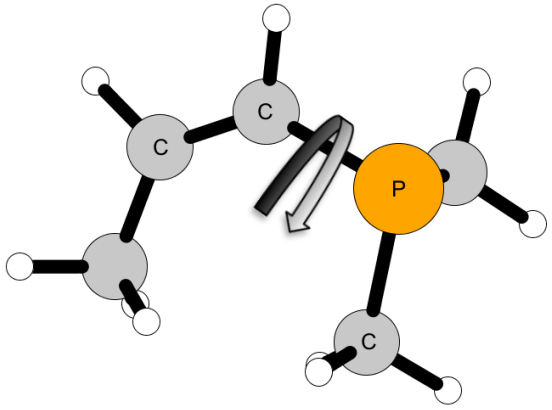


CSD comparison

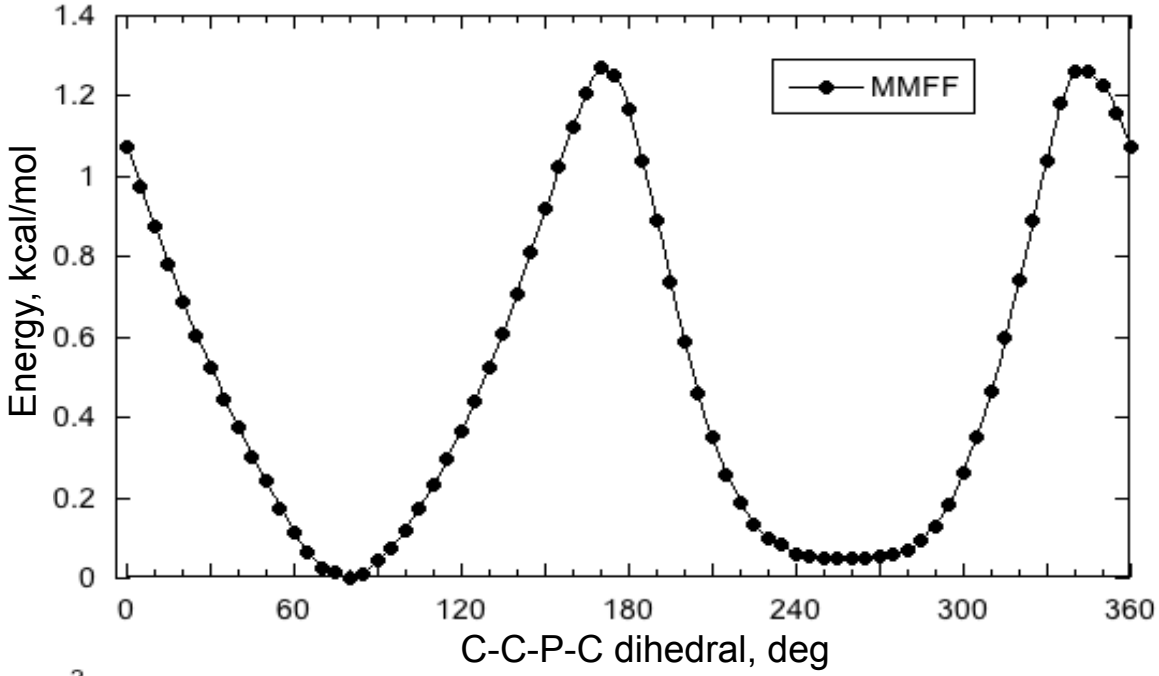


NO CDB HITS

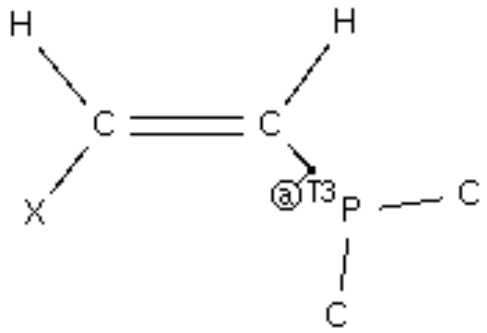
TYPE 2-2:9-4



Φ	E
80.0	0.00
260.0	0.05

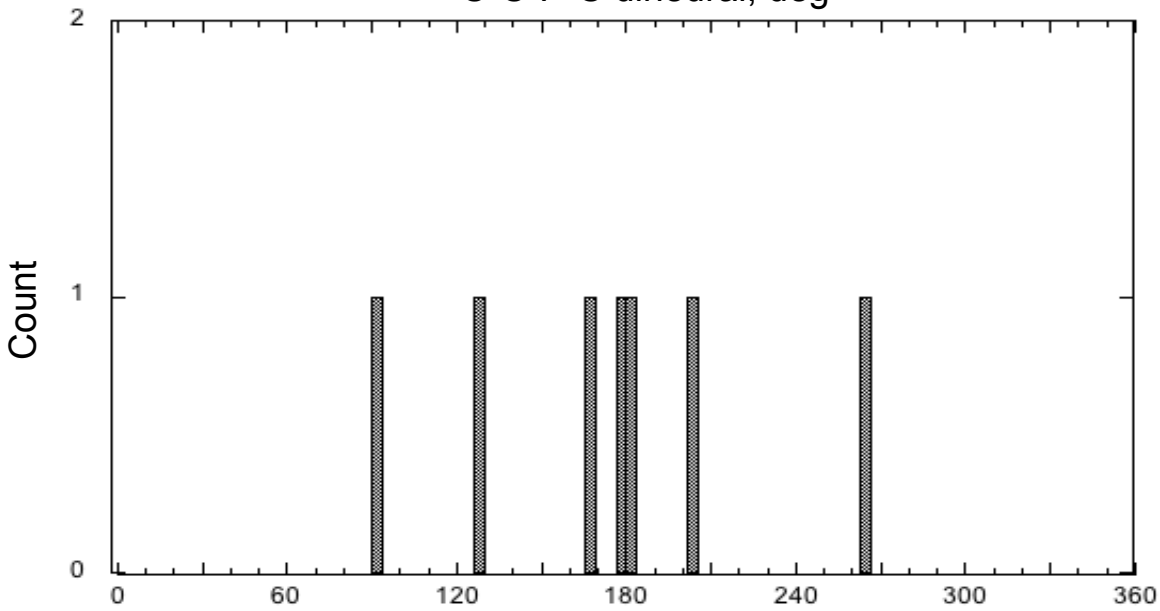


CSD comparison

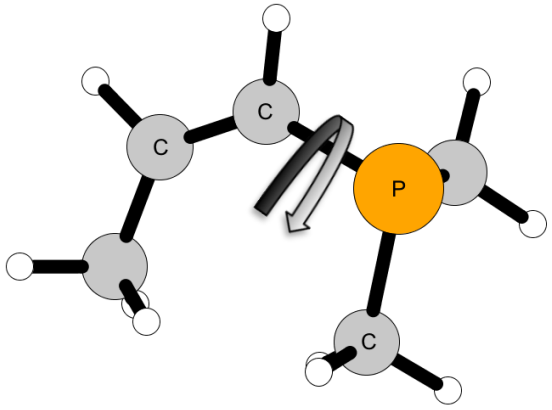


5 hits

C–P = 1.82 ± 0.00 Å

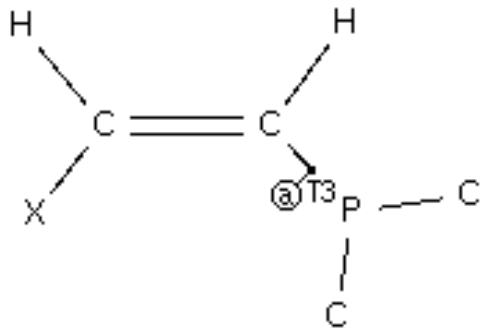


TYPE 2-2:9-4



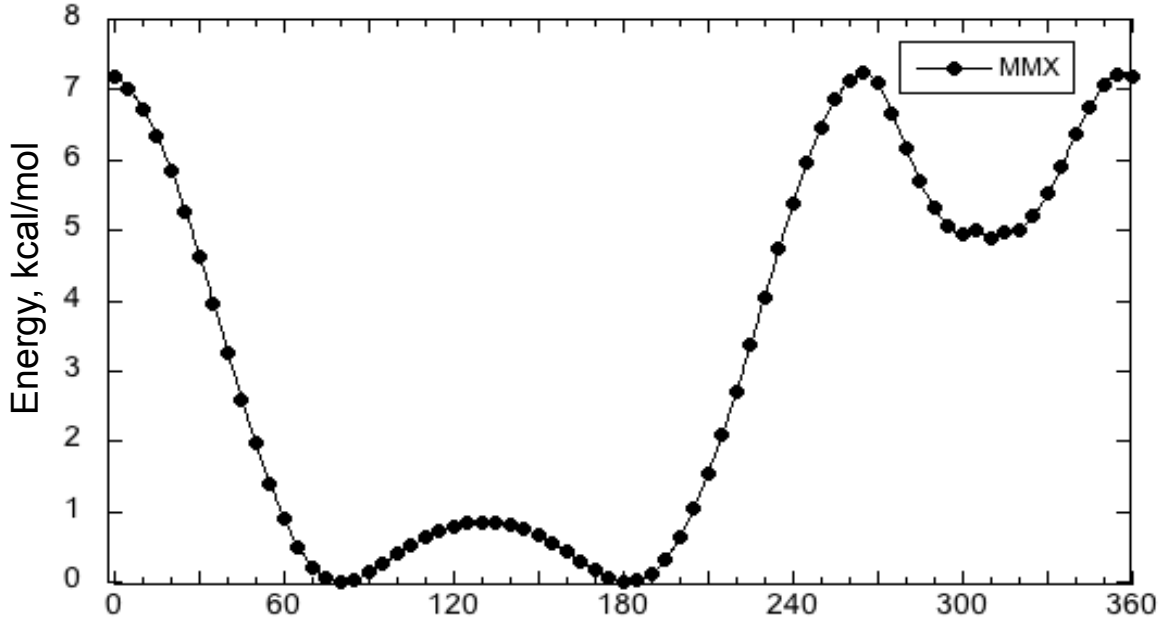
Φ	E
80.0	0.00
180.0	0.01
310.0	4.89

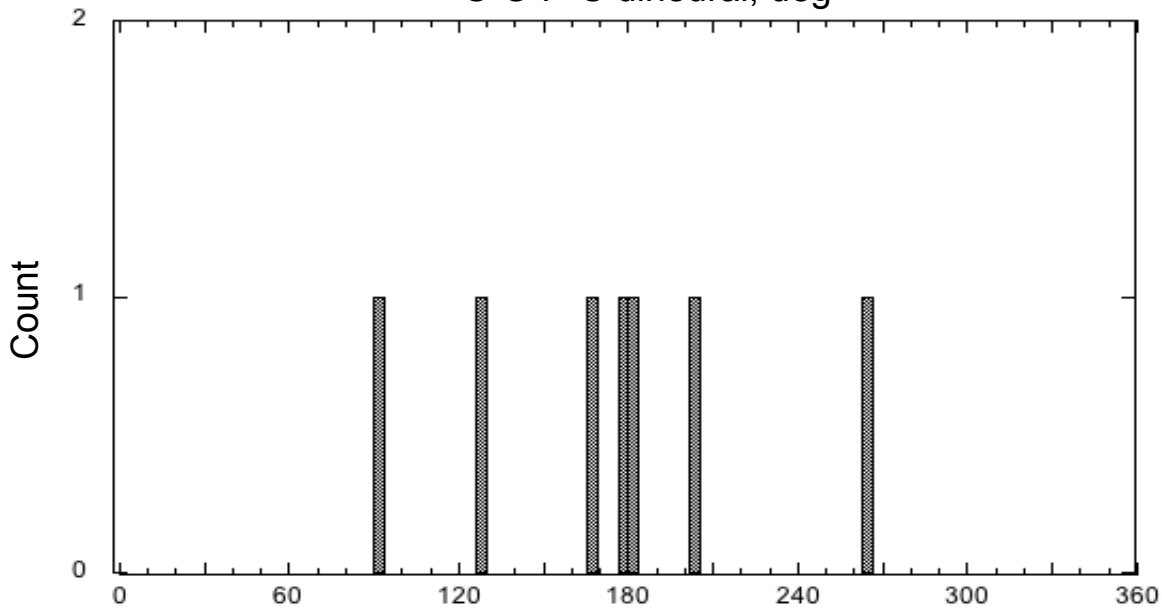
CSD comparison

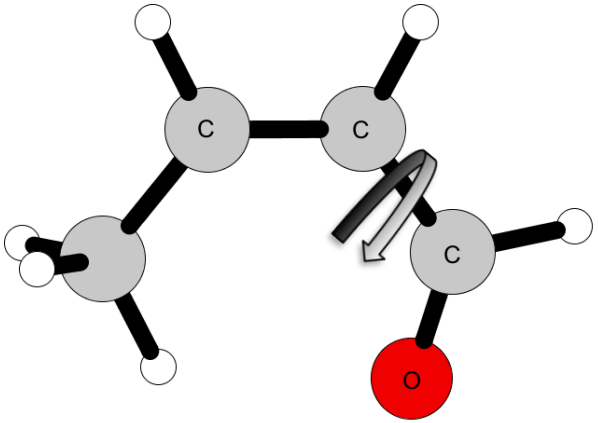


5 hits

C–P = 1.82± 0.00 Å

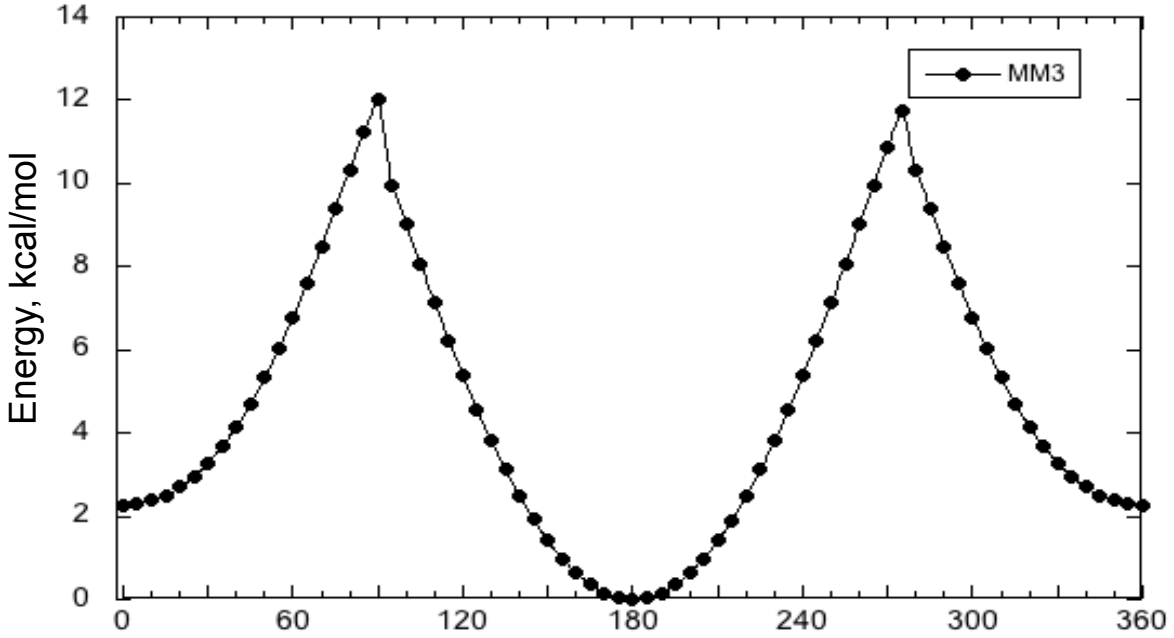




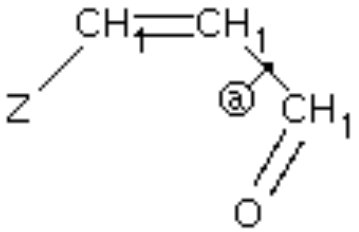


TYPE 2-2:10-1

Φ	E
0.0	2.28
185.0	0.00

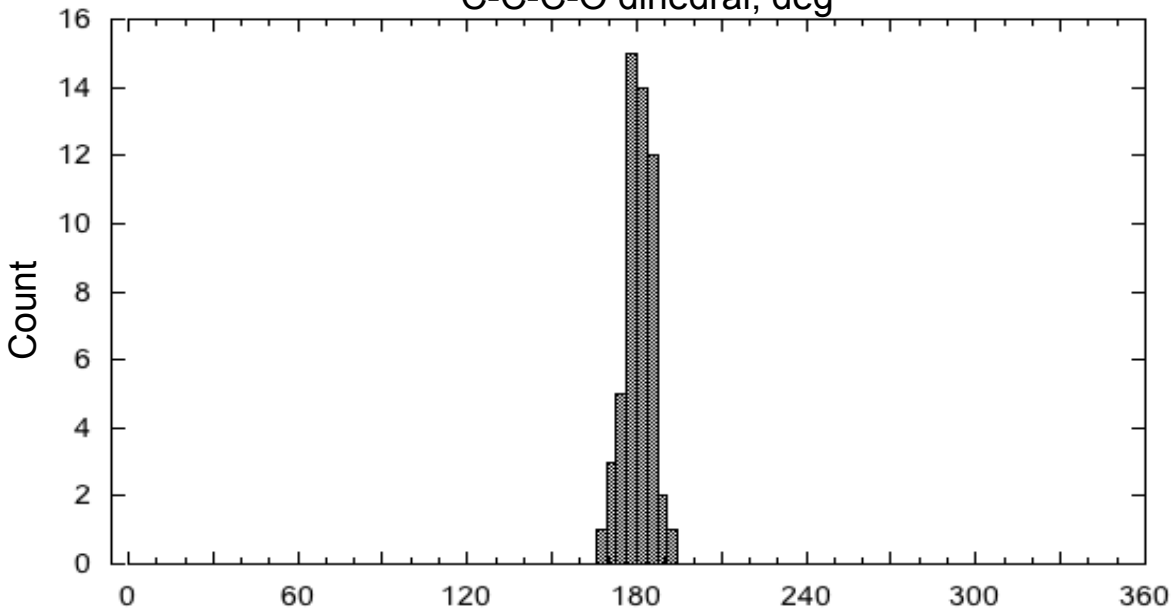


CSD comparison

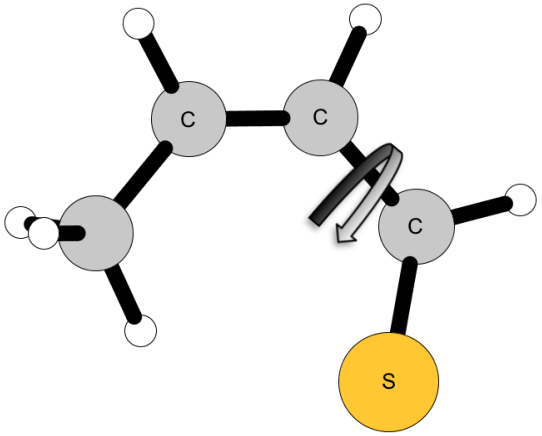


43 hits

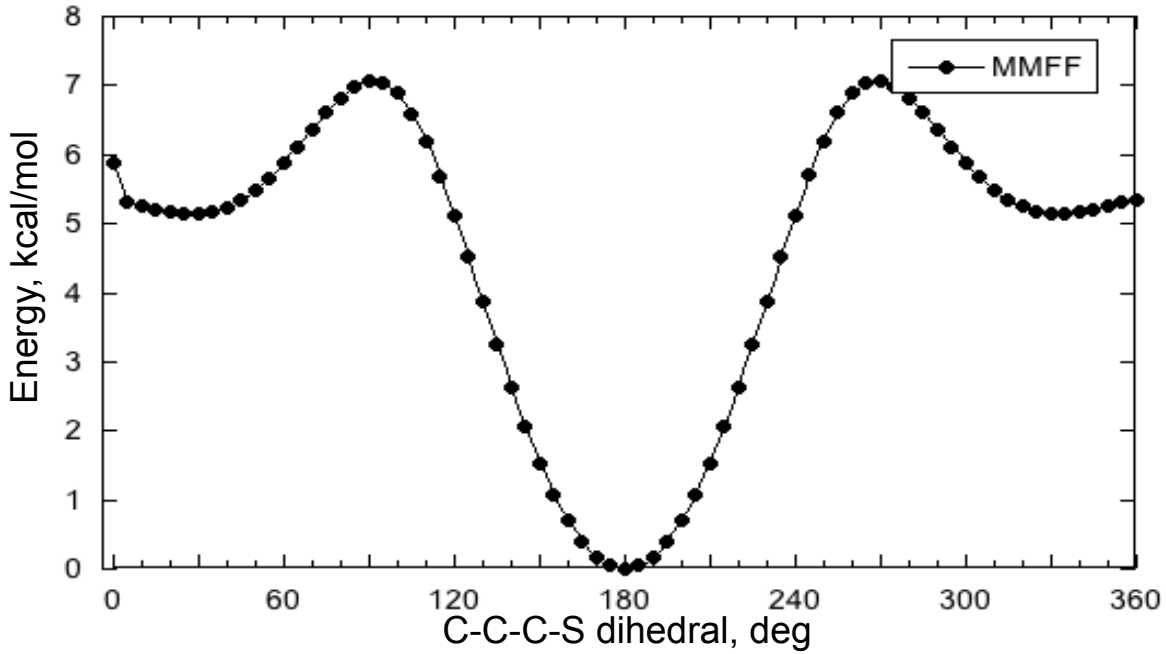
C–C = 1.44 ± 0.02 Å



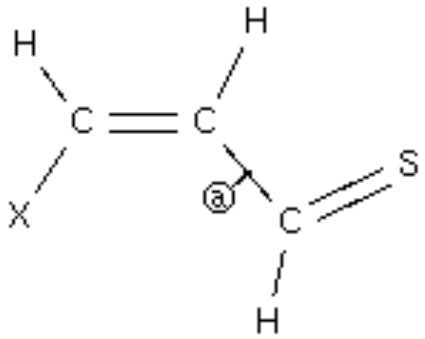
TYPE 2-2:10-2



Φ	E
25.0	5.14
180.0	0.00
335.0	5.14



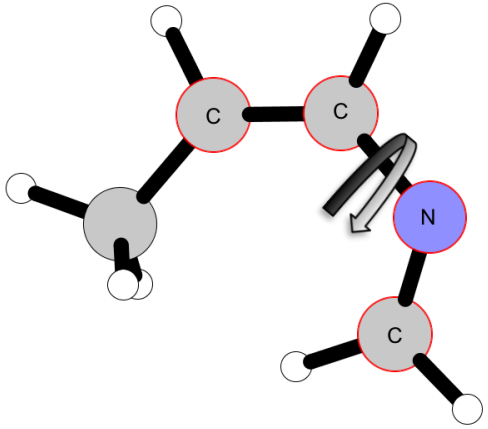
CSD comparison



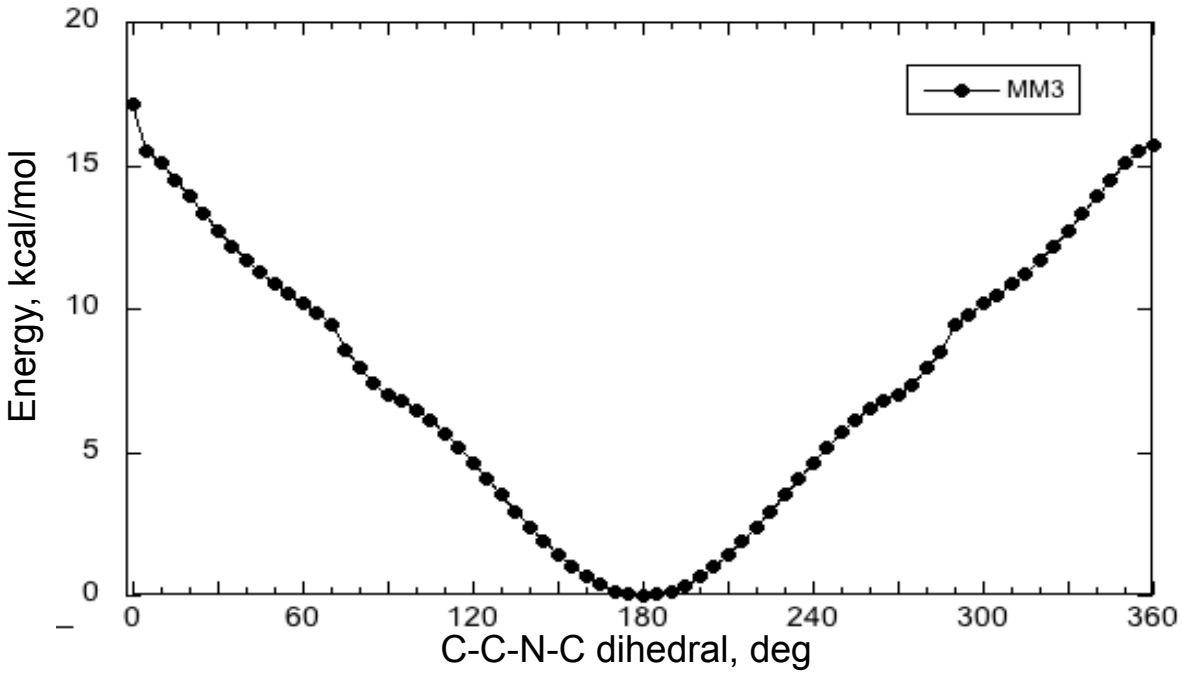
0 hits

NO CSD HITS

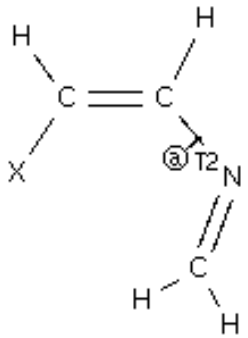
TYPE 2-2:10-3



Φ	E
180.0	0.00

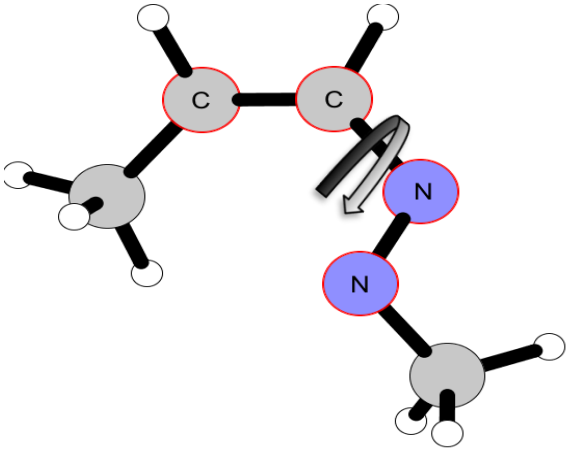


CSD comparison

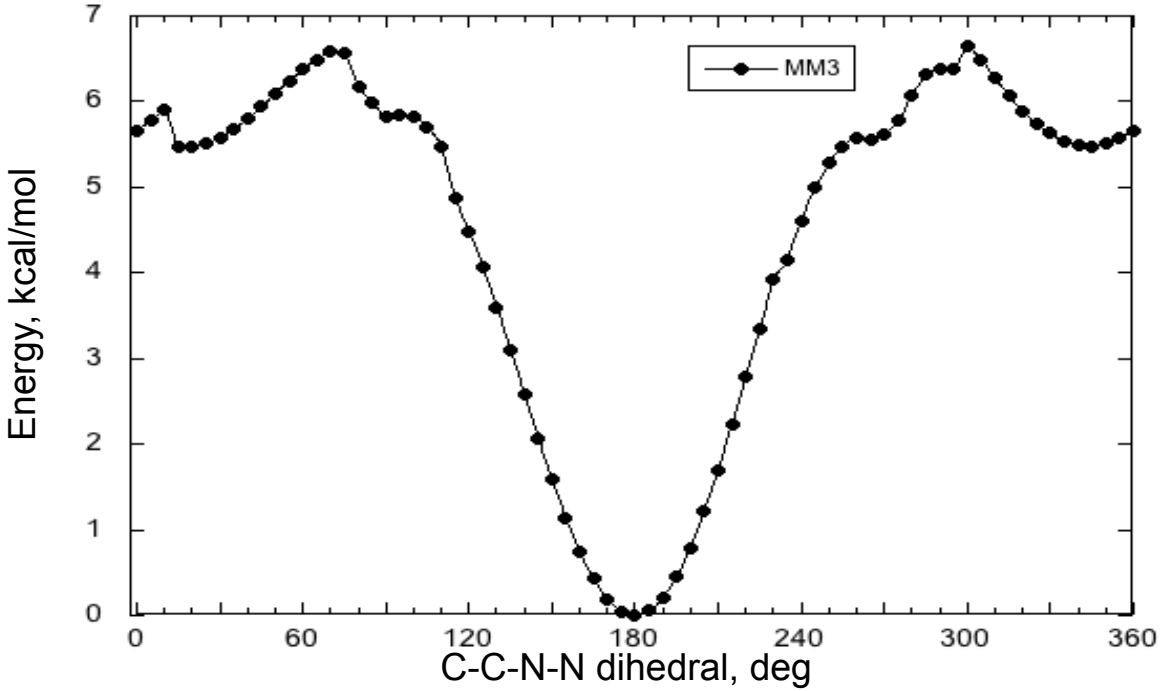


0 hits

NO CSD HITS

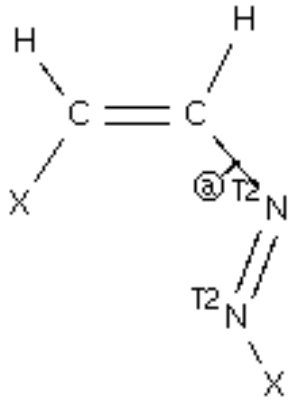


TYPE 2-2:10-4

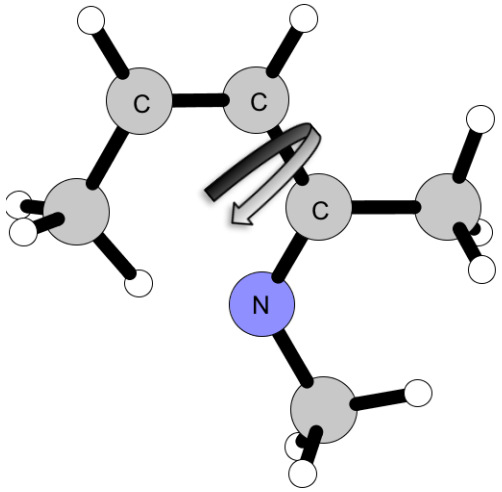


Φ	E
15.0	5.47
90.0	5.82
180.0	0.00
265.0	5.56
345.0	5.48

CSD comparison



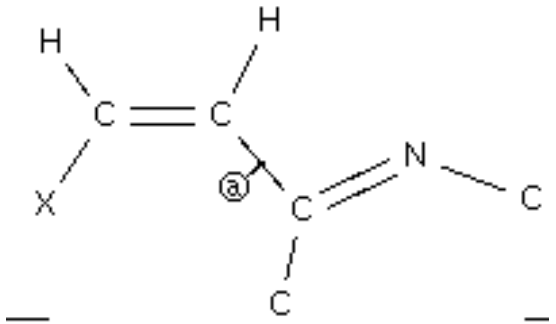
NO CSD HITS



TYPE 2-2:11-1

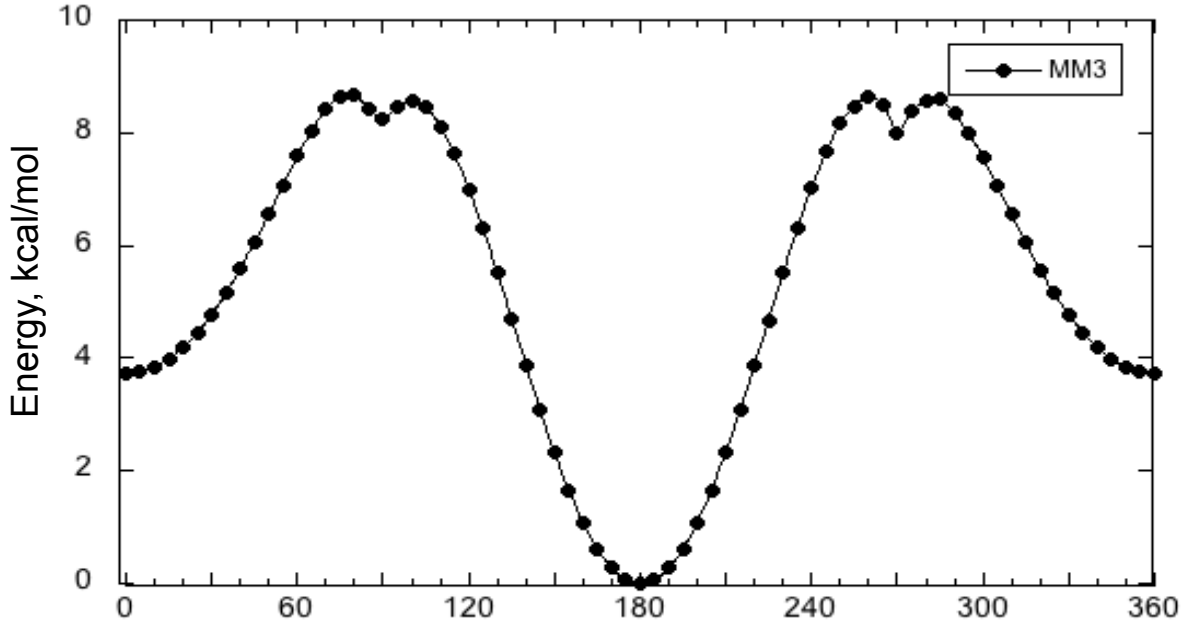
Φ	E
0.0	3.74
90.0	8.22
180.0	0.00
270.0	8.00

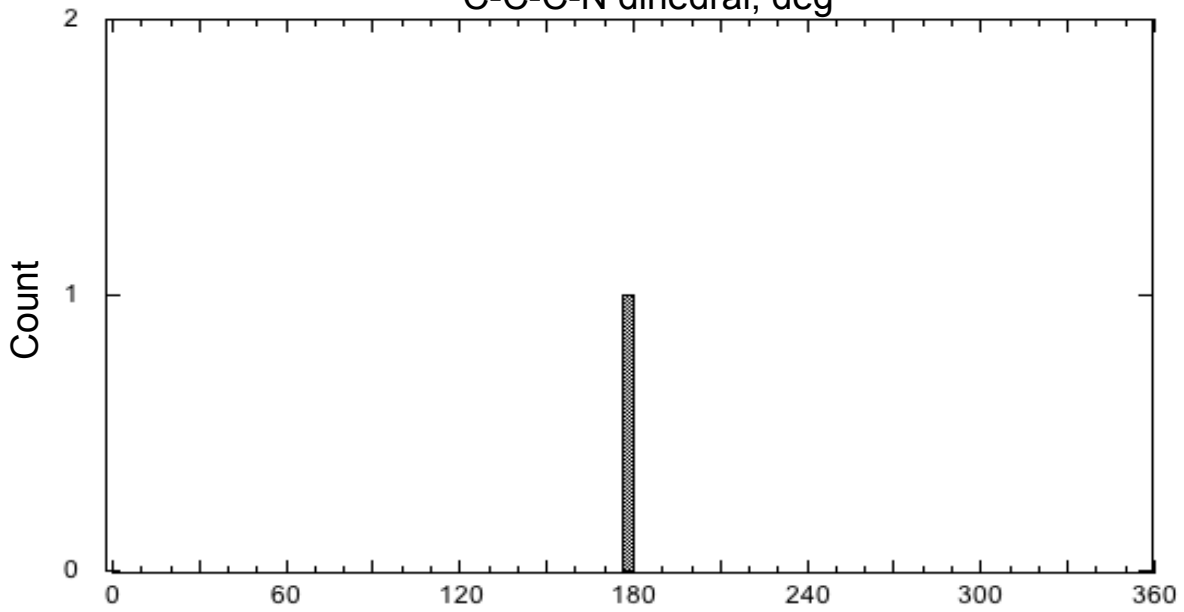
CSD comparison



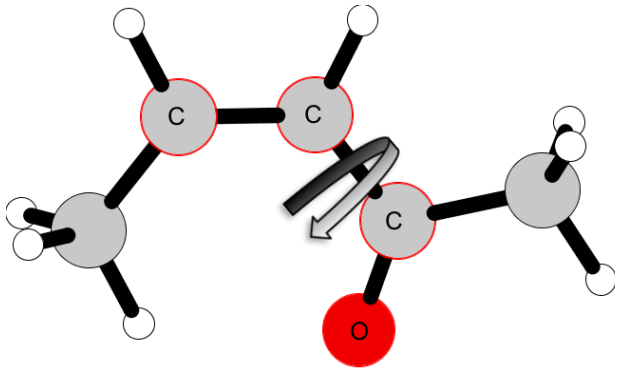
1 hits

C–C = 1.45 Å

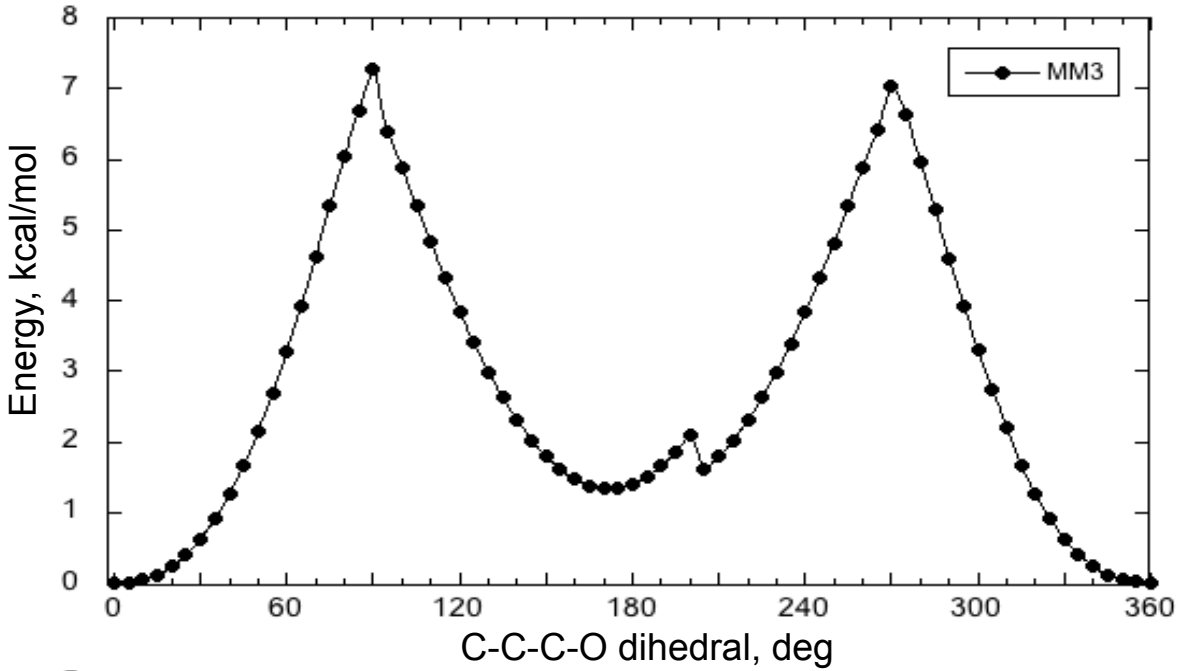




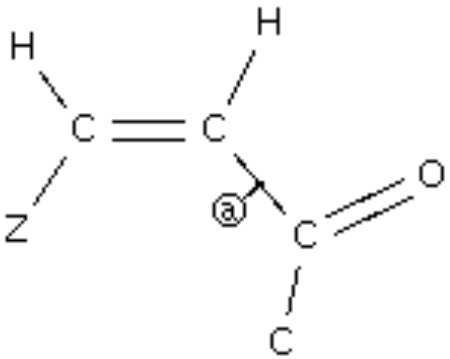
TYPE 2-2:11-2



Φ	E
0.0	0.00
170.0	1.34

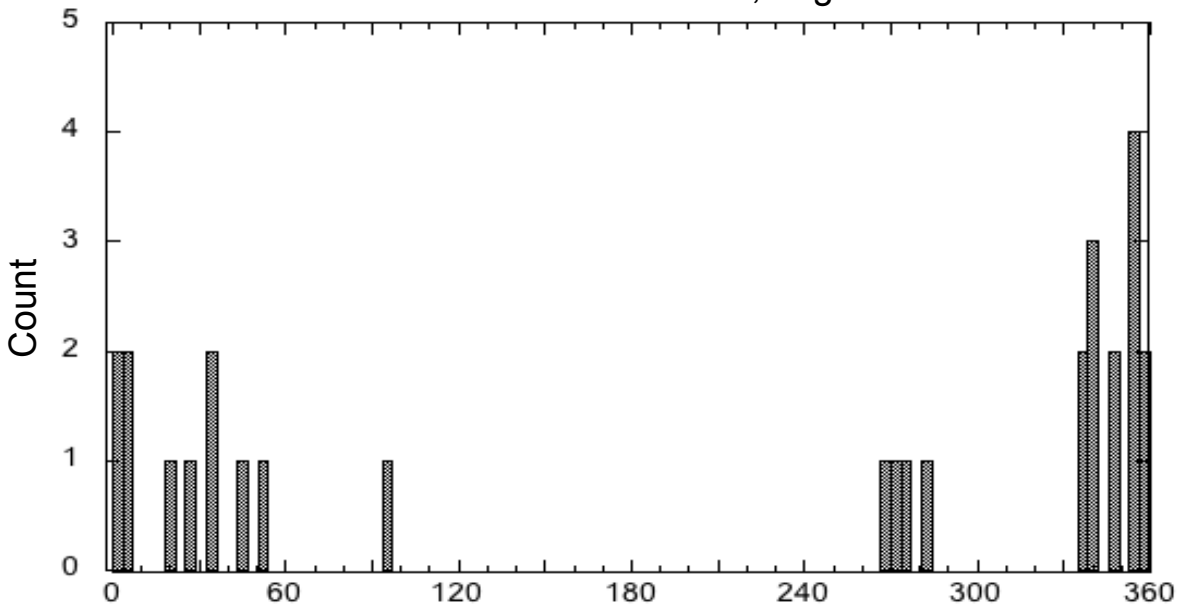


CSD comparison

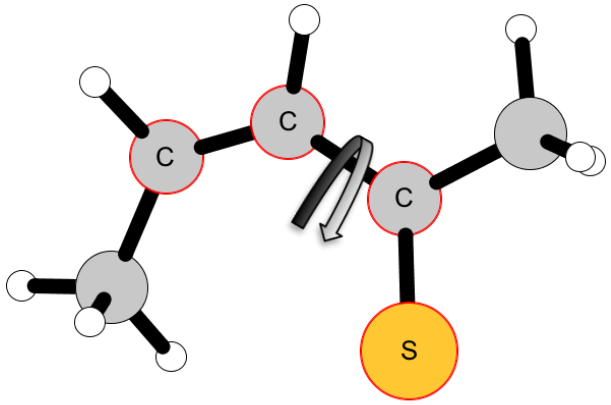


21 hits

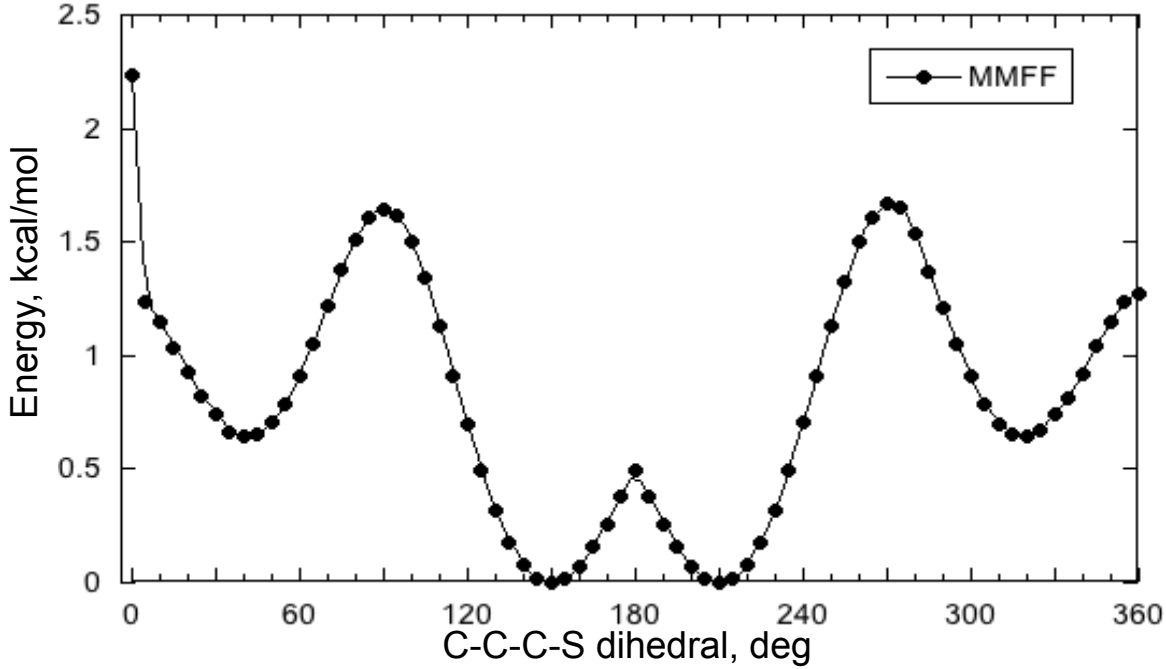
C–C = 1.47 ± 0.03 Å



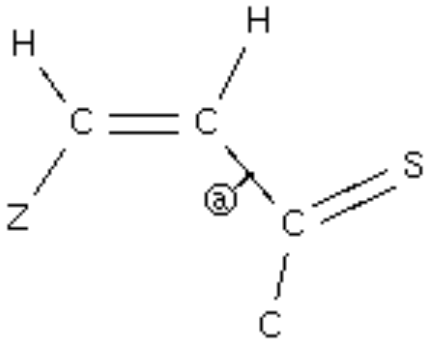
TYPE 2-2:11-3



Φ	E
40.0	0.65
150.0	0.00
210.0	0.00
320.0	0.65



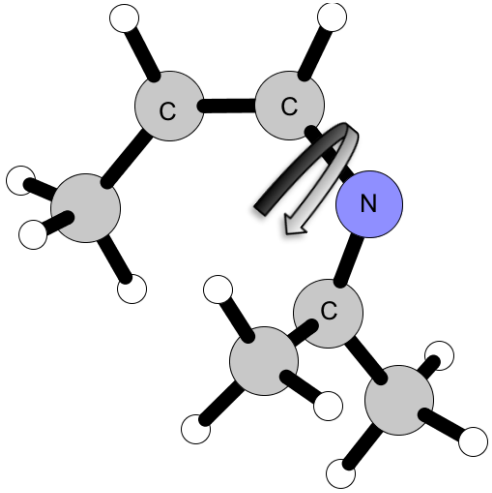
CSD comparison



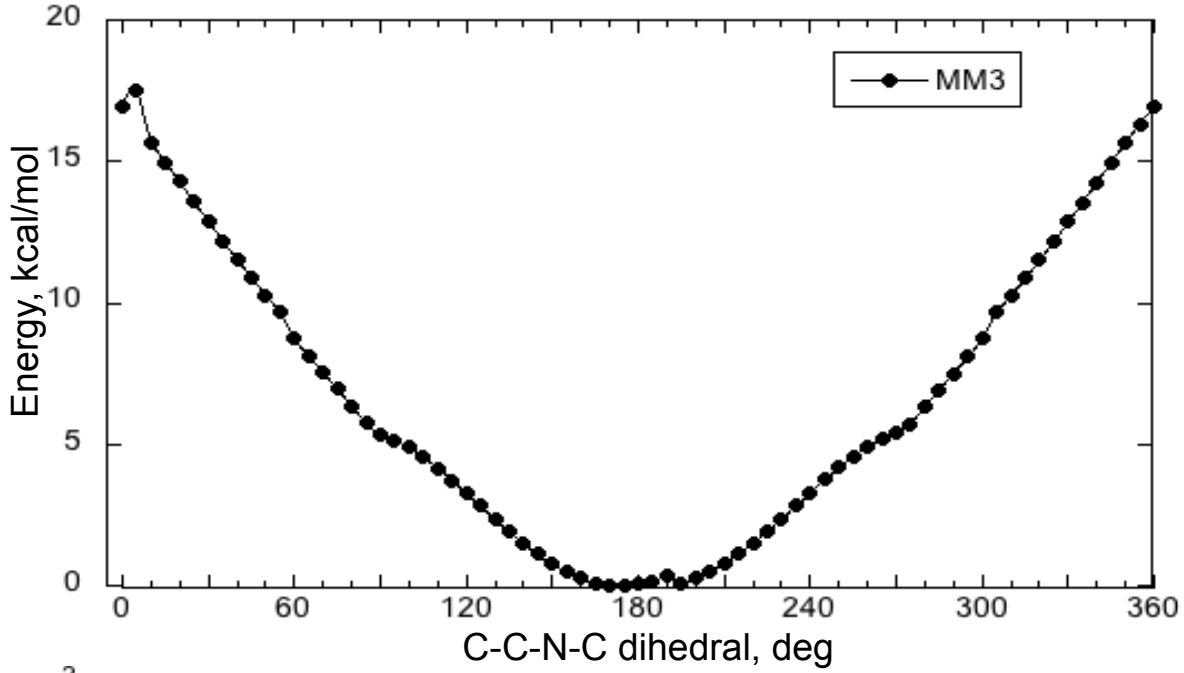
0 hits

NO CSD HITS

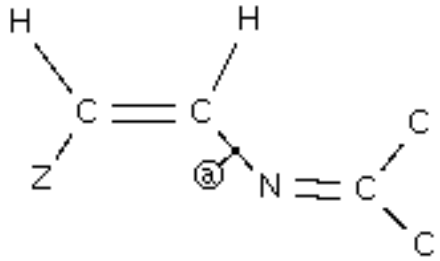
TYPE 2-2:12-1



Φ	E
175.0	0.00

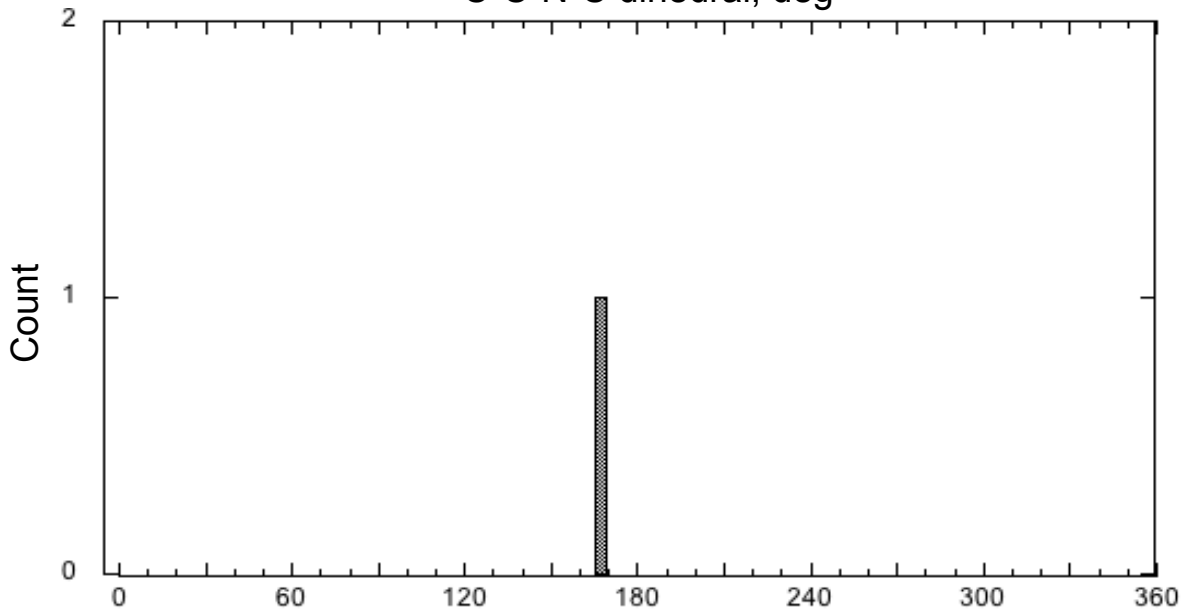


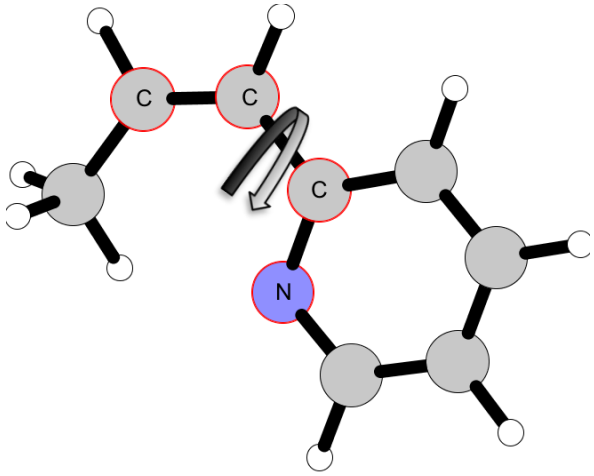
CSD comparison



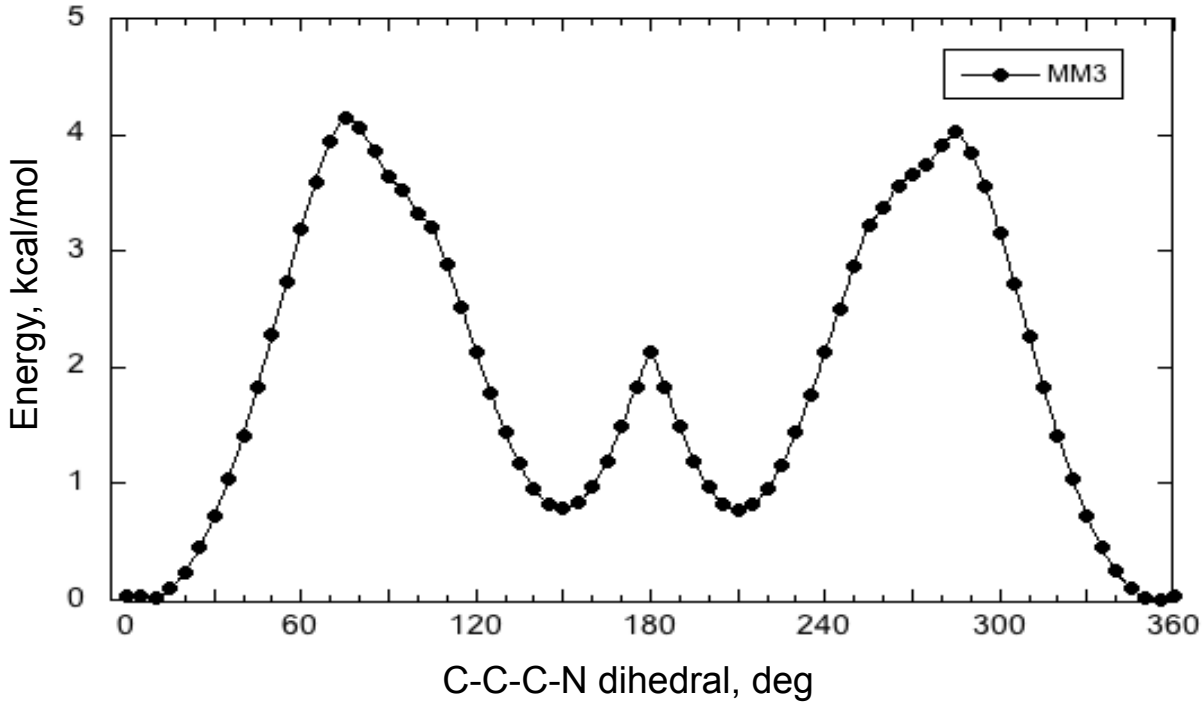
1 hits

C–N = 1.38 Å



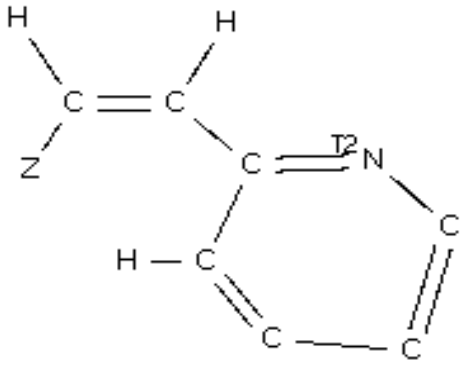


TYPE 2-2:13-1



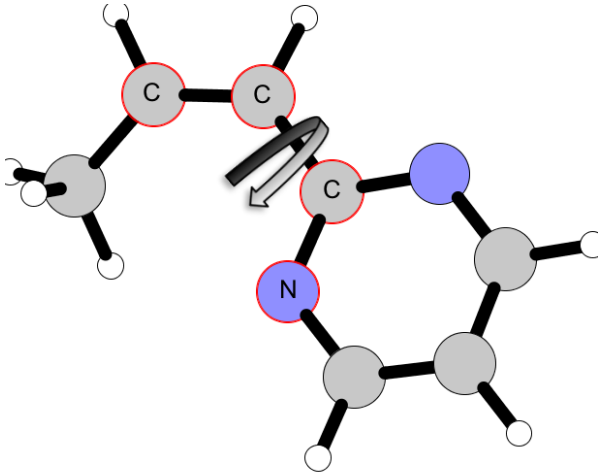
Φ	E
10.0	0.02
150.0	0.79
210.0	0.78
355.0	0.00

CSD comparison

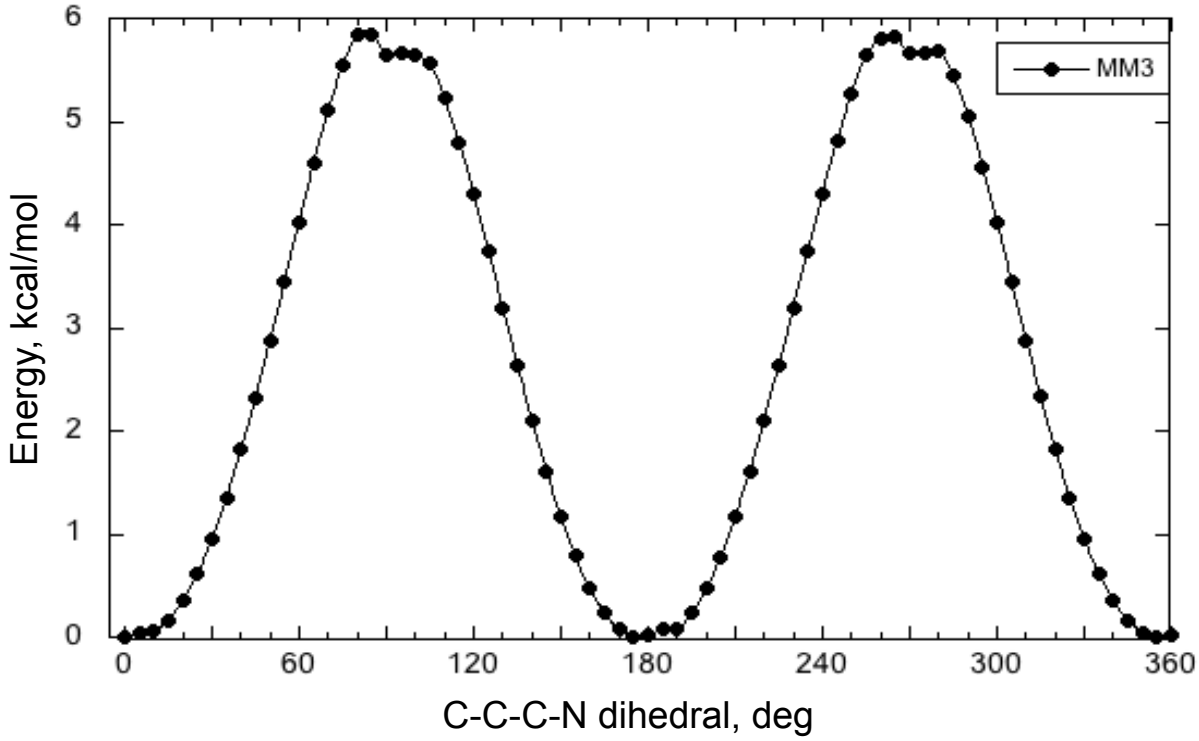


0 hits

NO CSD HITS

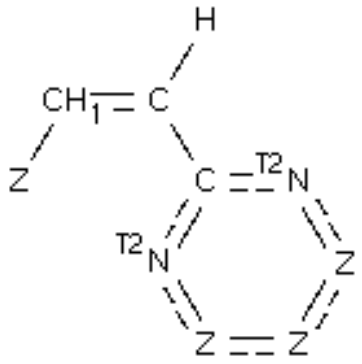


TYPE 2-2:13-2



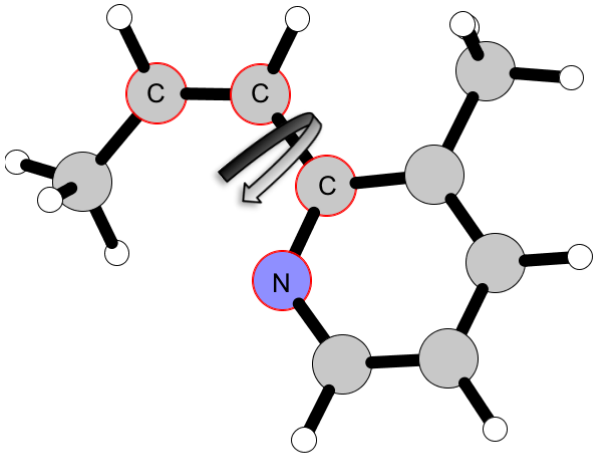
Φ	E
0.0	0.01
175.0	0.01
355.0	0.00

CSD comparison

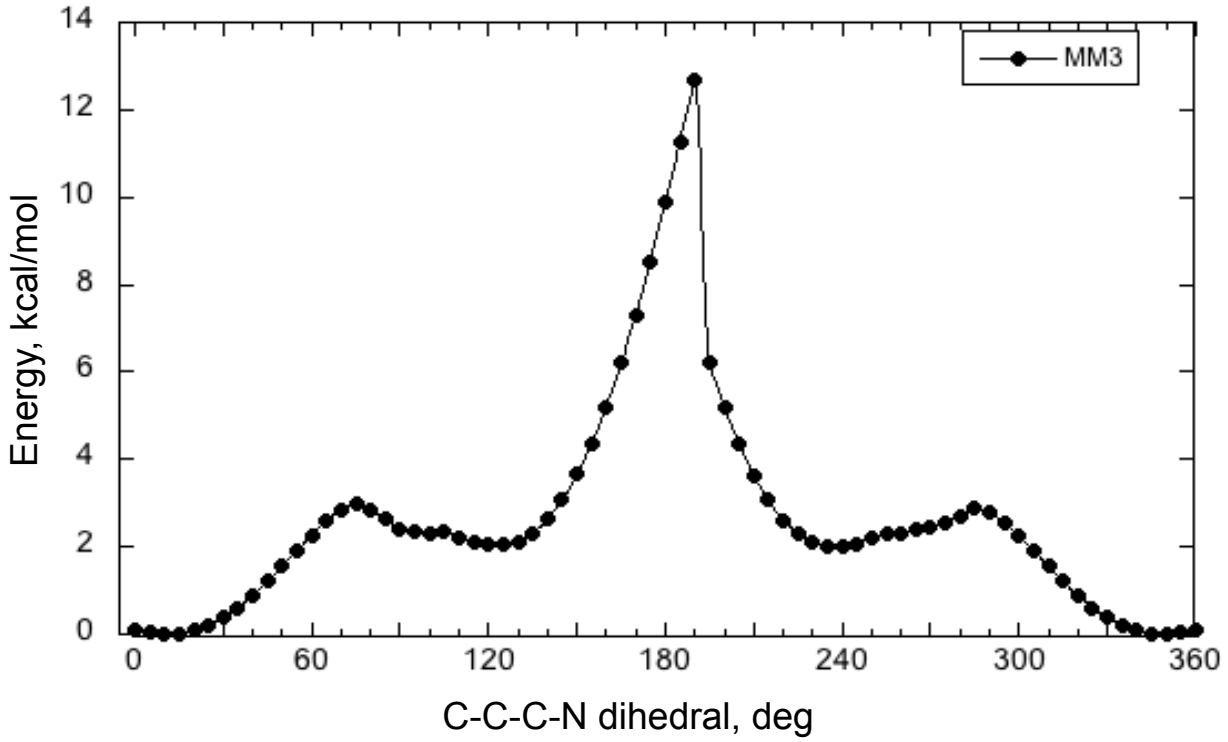


0 hits

NO CSD HITS

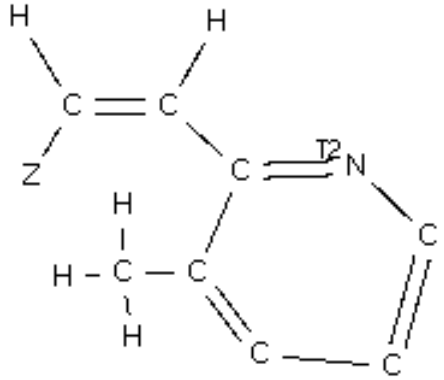


TYPE 2-2:13-3



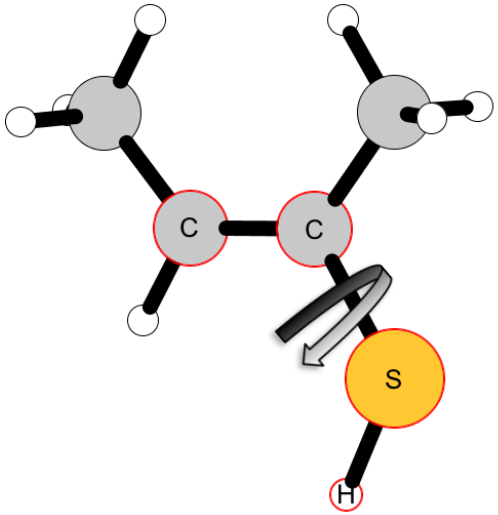
Φ	E
10.0	0.00
125.0	2.04
240.0	2.01
350.0	0.01

CSD comparison



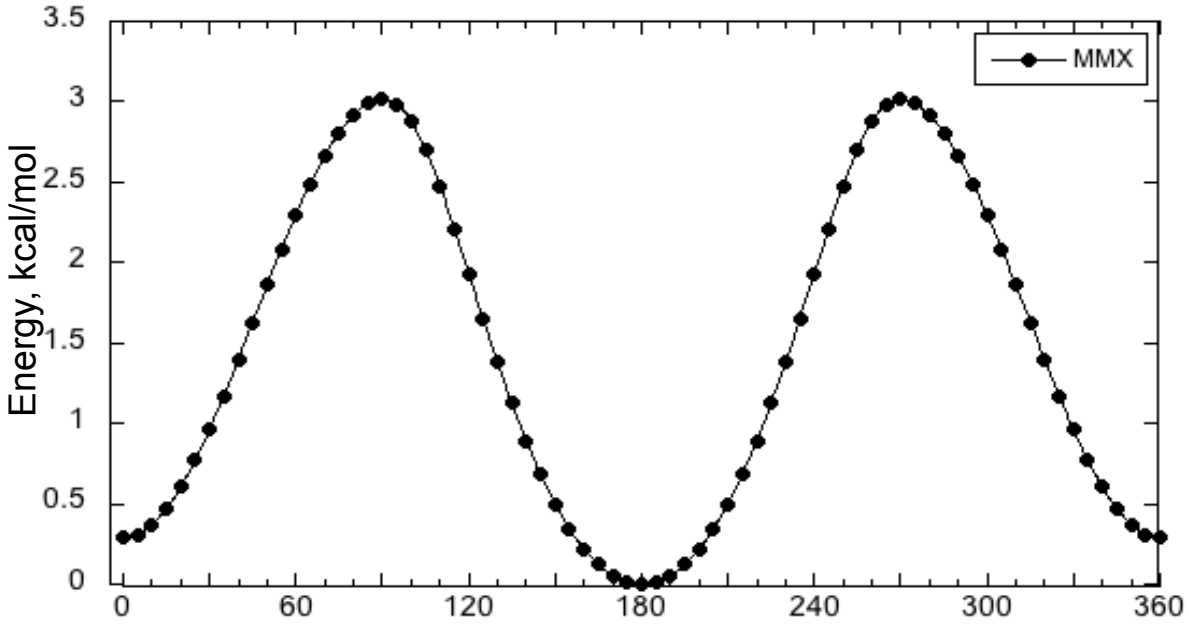
0 hits

NO CSD HITS

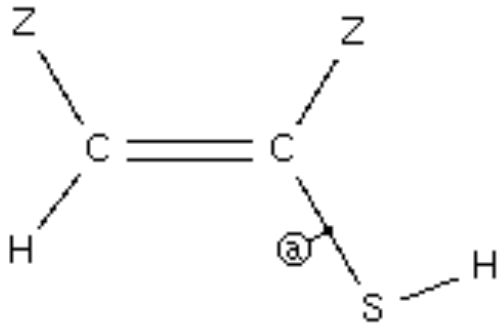


TYPE 2-3:5-4

Φ	E
0.00	0.29
180.0	0.00

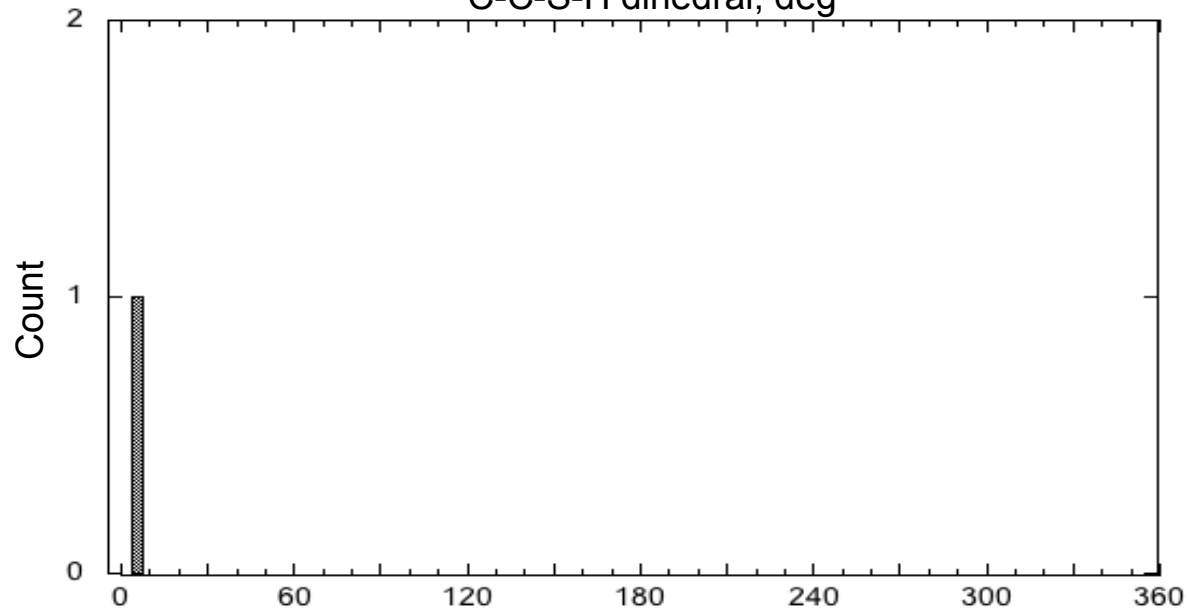


CSD comparison

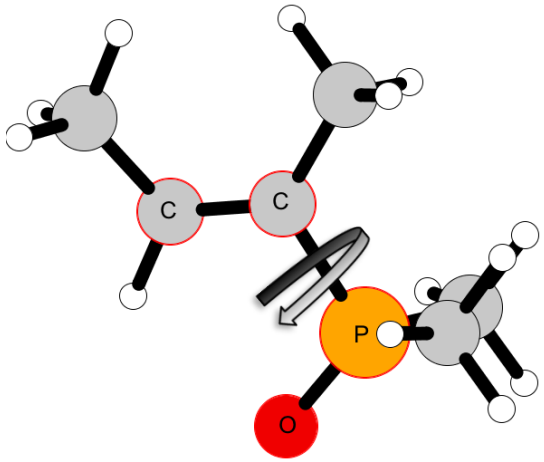


1 hits

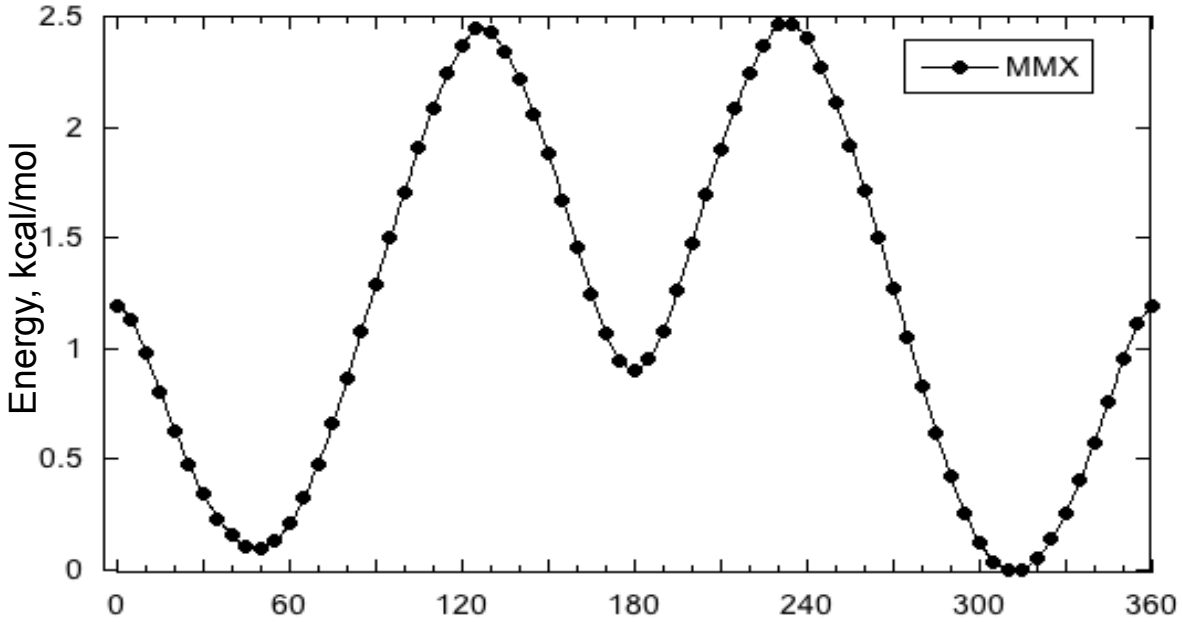
C–S = 1.75Å



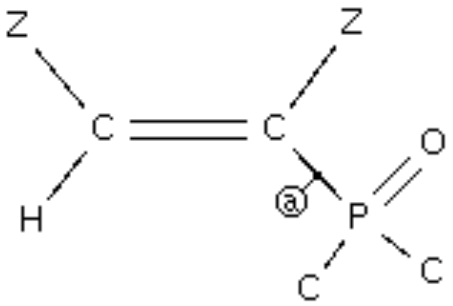
TYPE 2-3:8-1



Φ	E
55.0	0.10
180.0	0.90
310.0	0.00

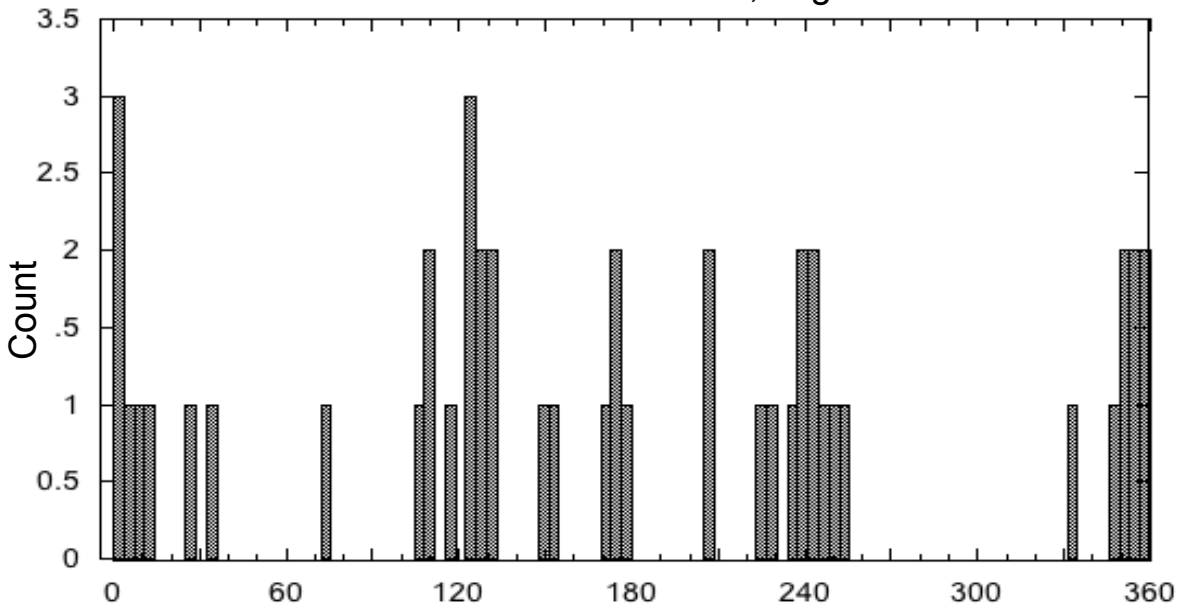


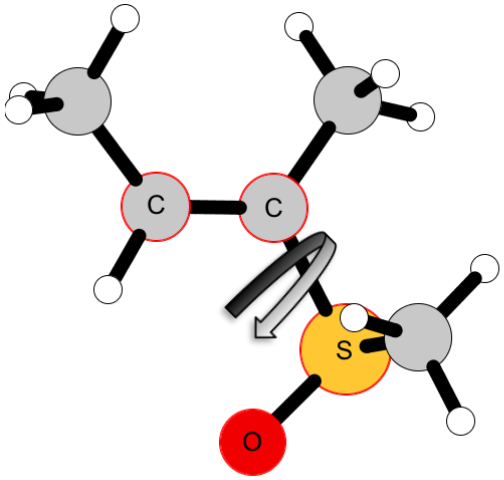
CSD comparison



31 hits

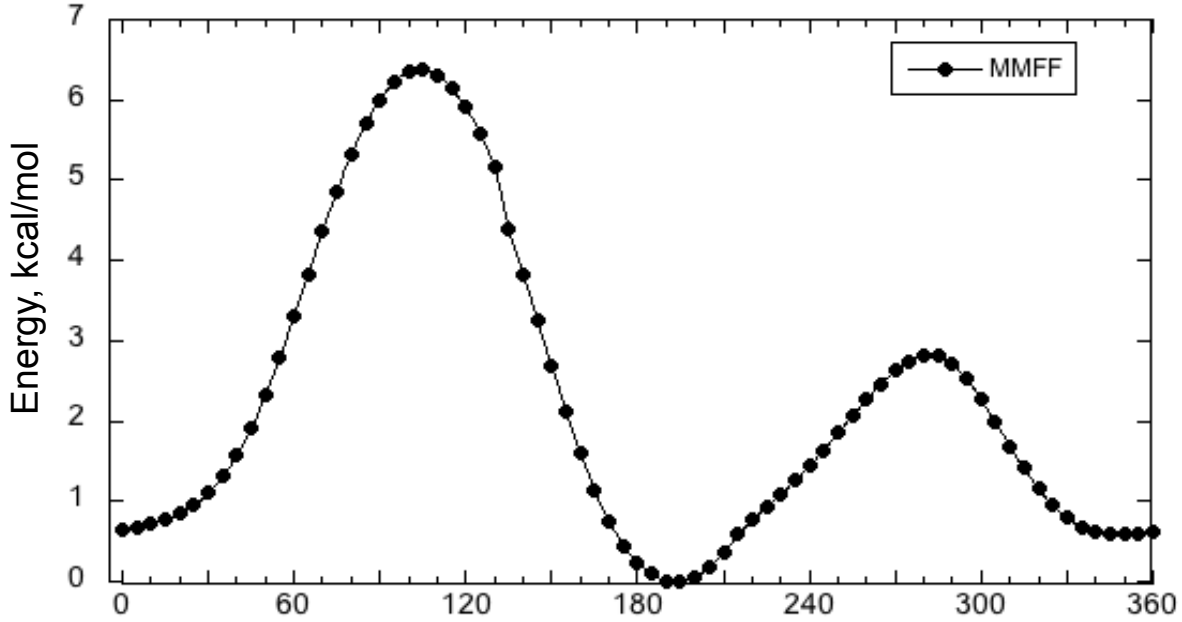
C–P = 1.79 ± 0.02 Å



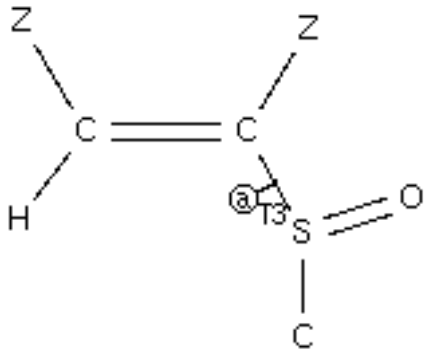


TYPE 2-3:8-2

Φ	E
195.0	0.00
350.0	0.58

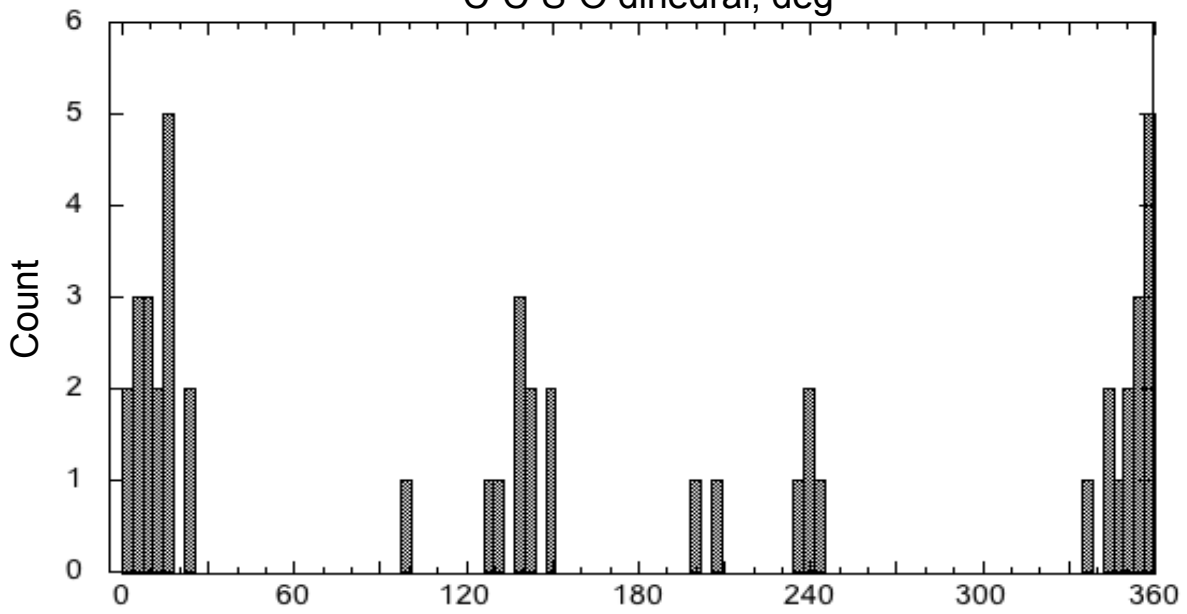


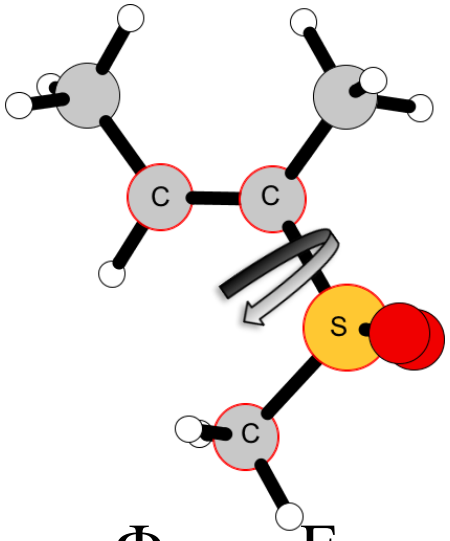
CSD comparison



39 hits

C–S = 1.79 ± 0.02 Å

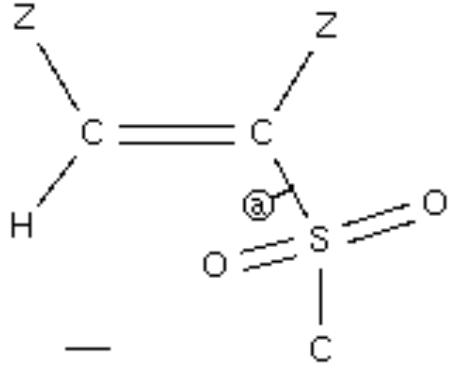




TYPE 2-3:8-4

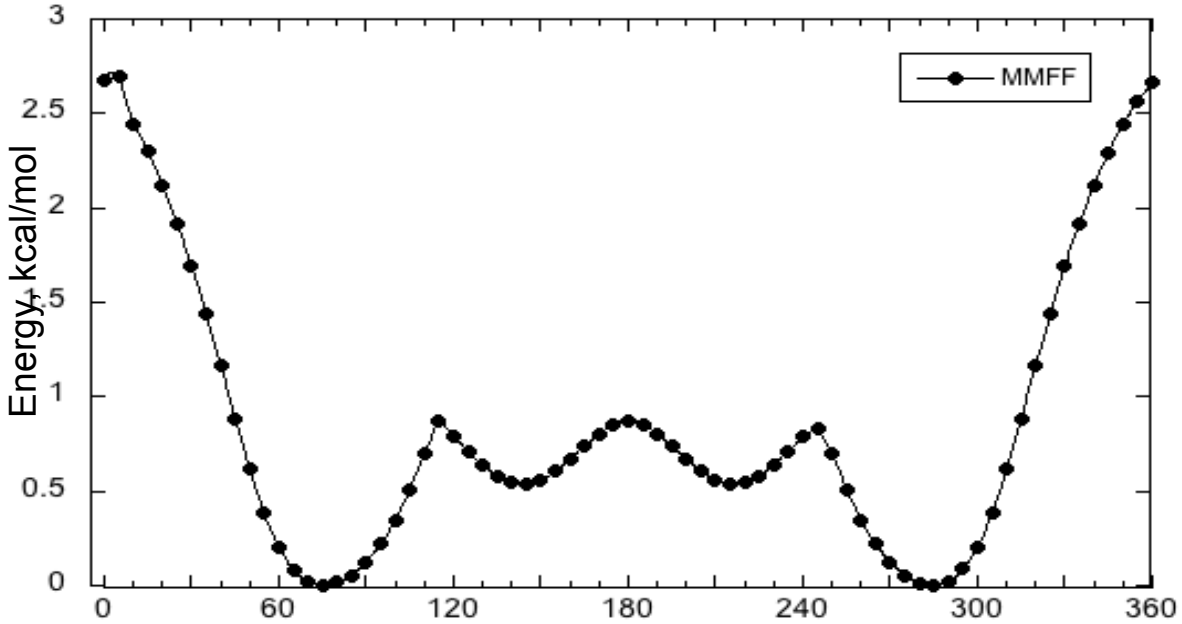
Φ	E
75.0	0.00
145.0	0.54
215.0	0.54
285.0	0.00

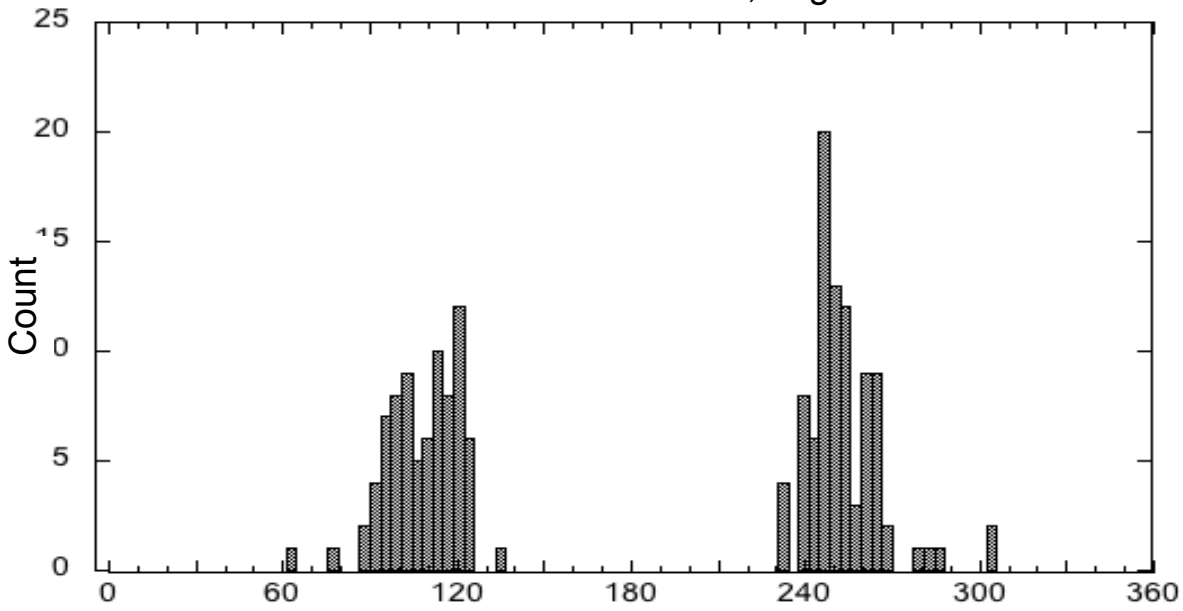
CSD comparison



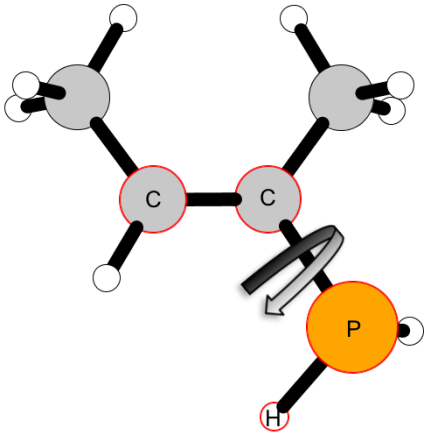
138 hits

C–S = 1.76 ± 0.02 Å

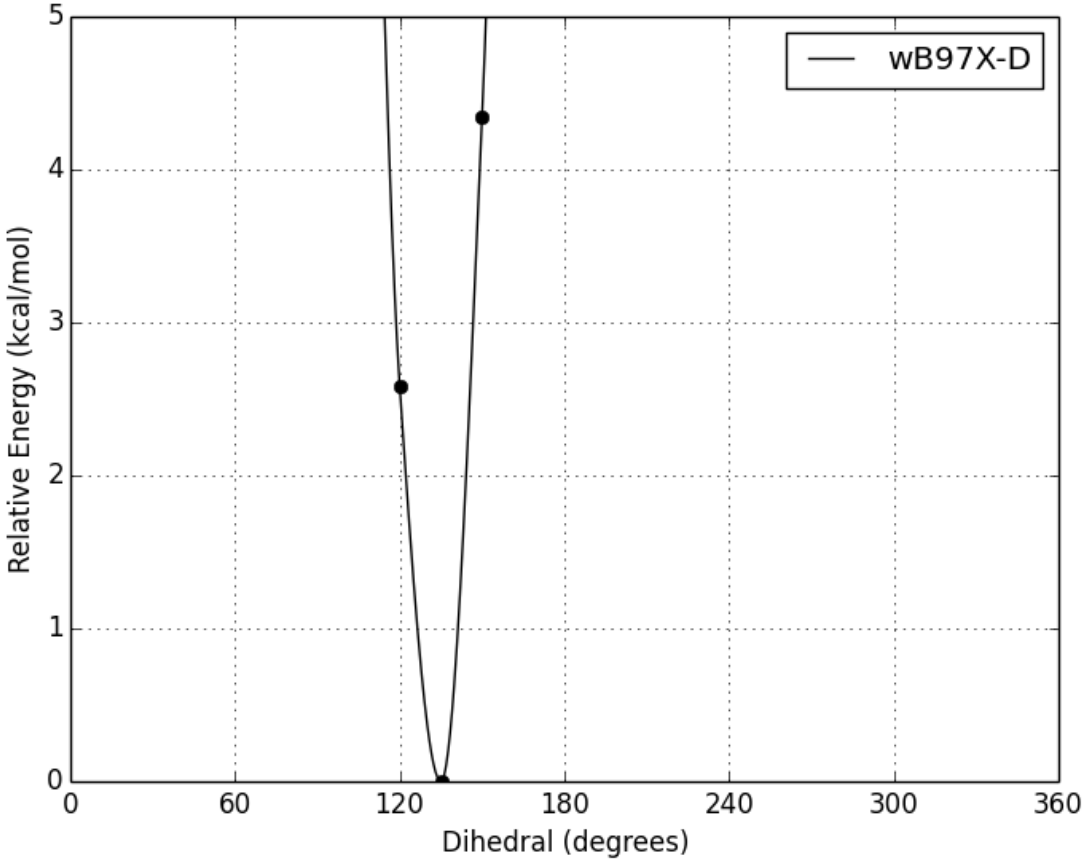




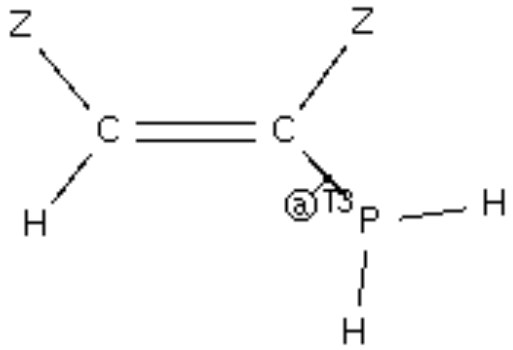
TYPE 2-3:9-1



Φ	E
135.0	0.00

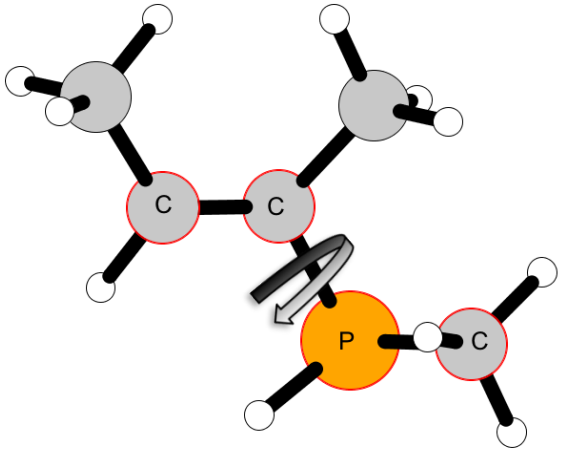


CSD comparison

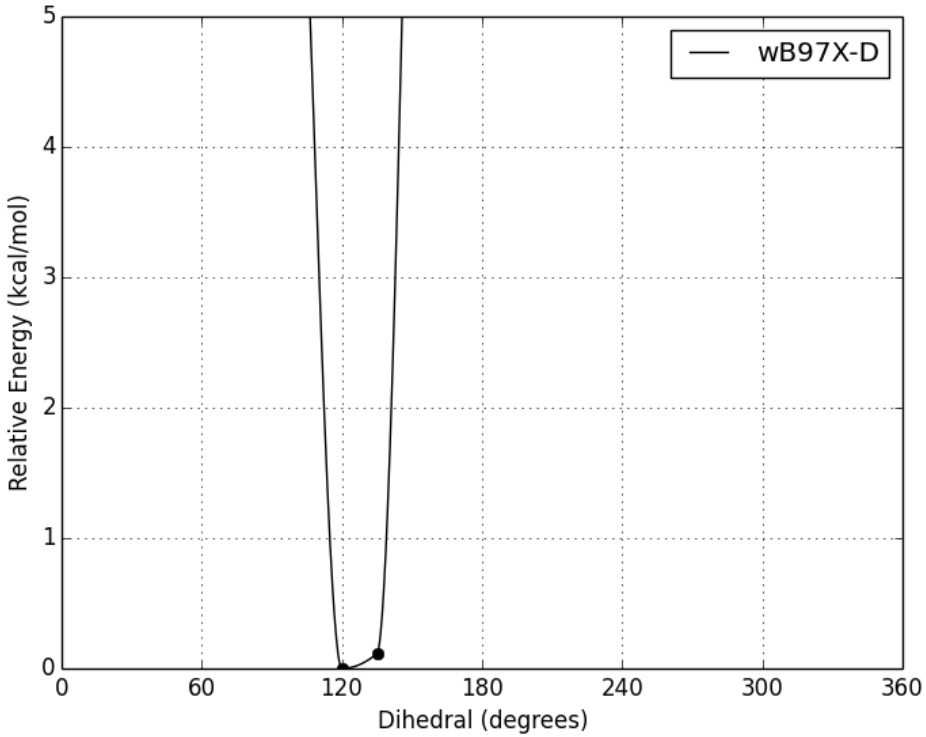


NO CSD HITS

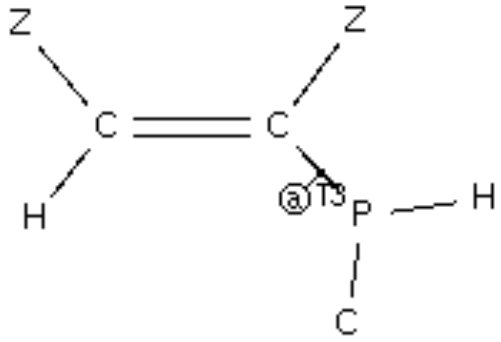
TYPE 2-3:9-2



Φ	E
120.0	0.00

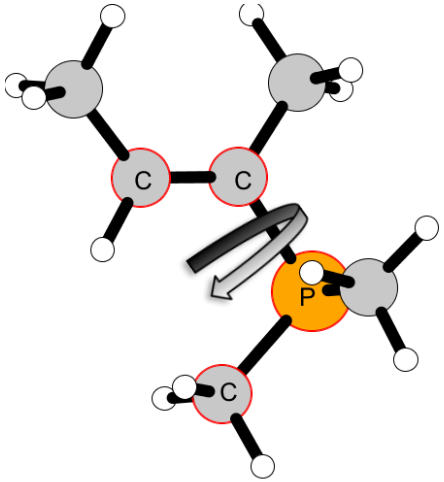


CSD comparison

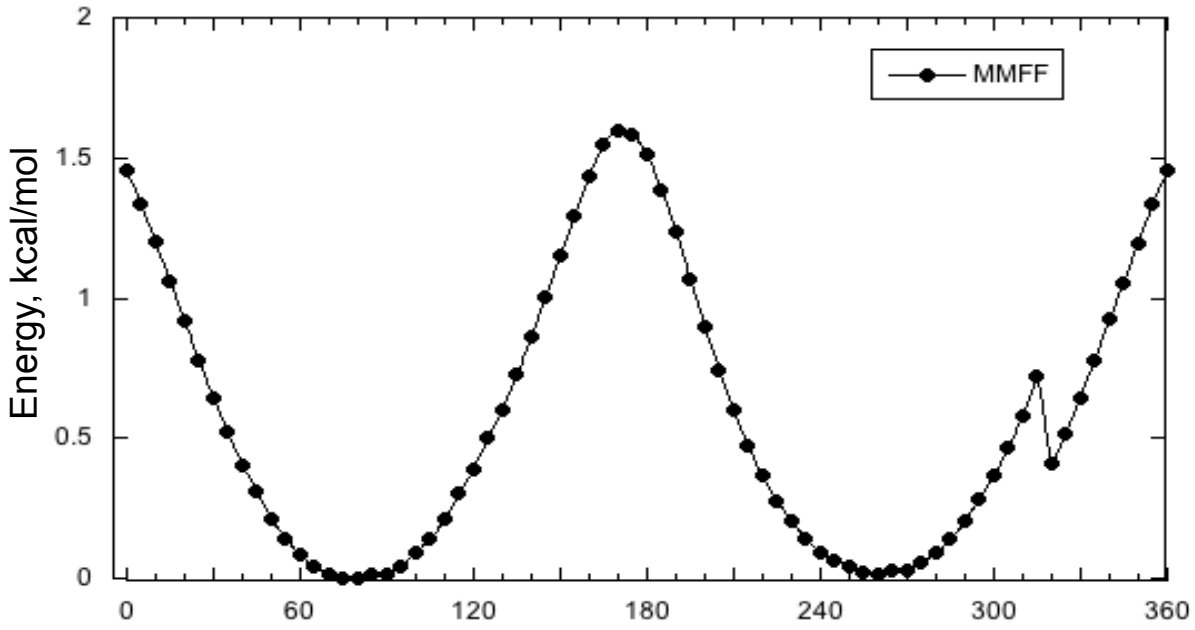


NO CSD HITS

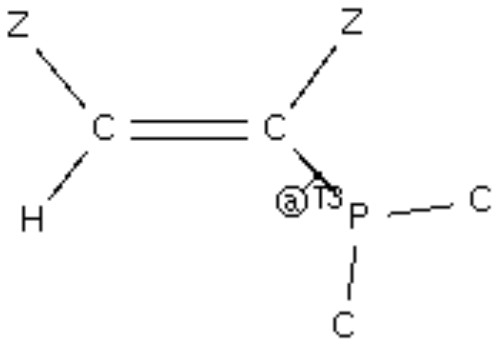
TYPE 2-3:9-4



Φ	E
80.0	0.00
260.0	0.02

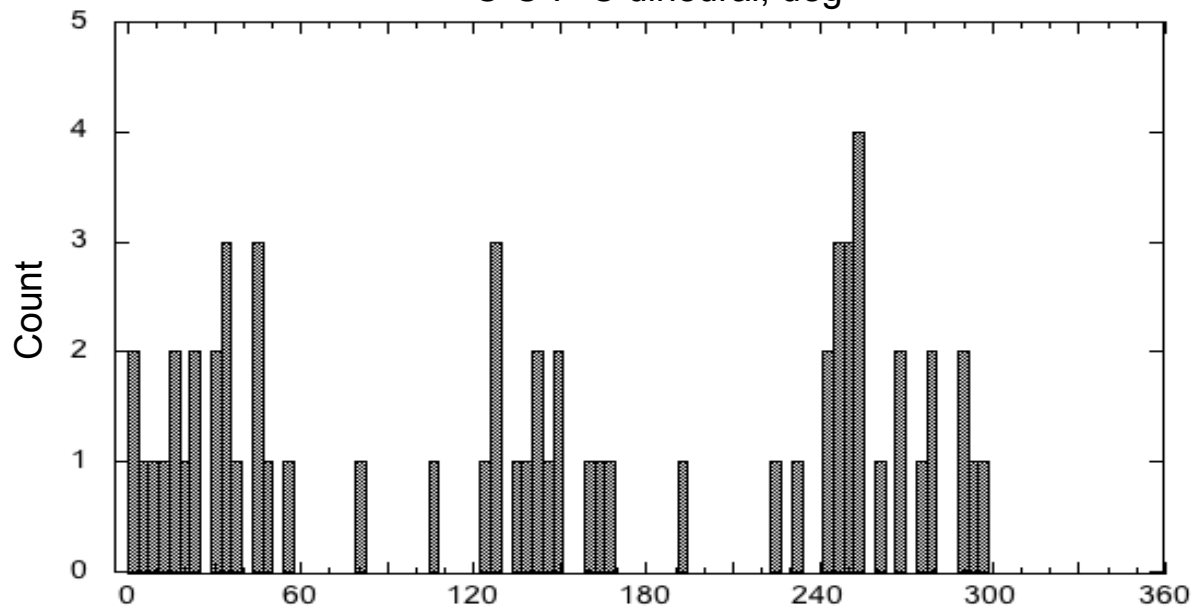


CSD comparison

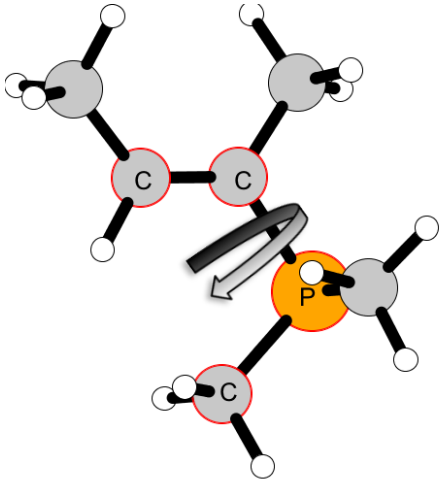


44 hits

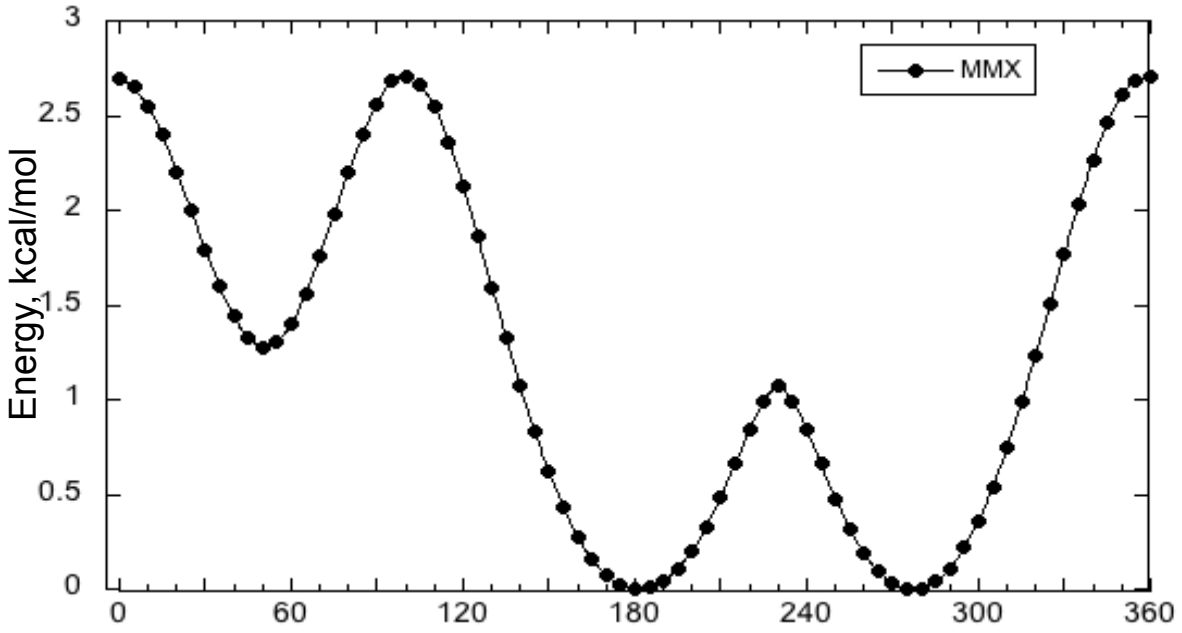
C–P = 1.82 ± 0.01Å



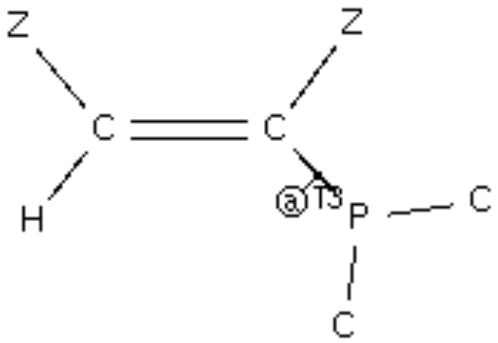
TYPE 2-3:9-4



Φ	E
50.00	1.28
180.00	0.00
275.00	0.00

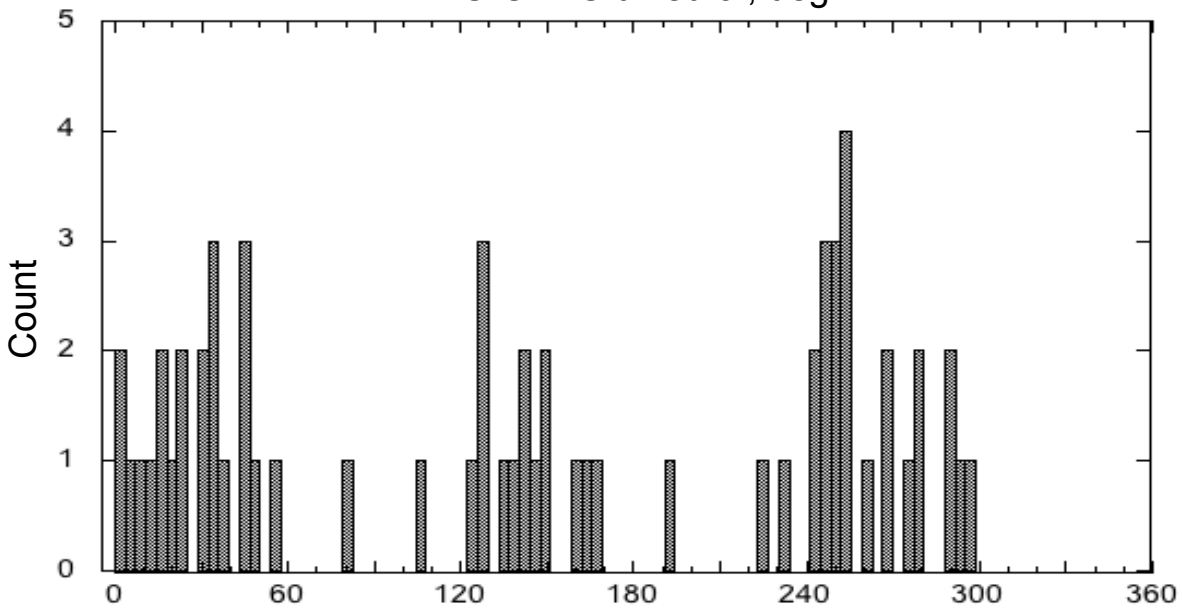


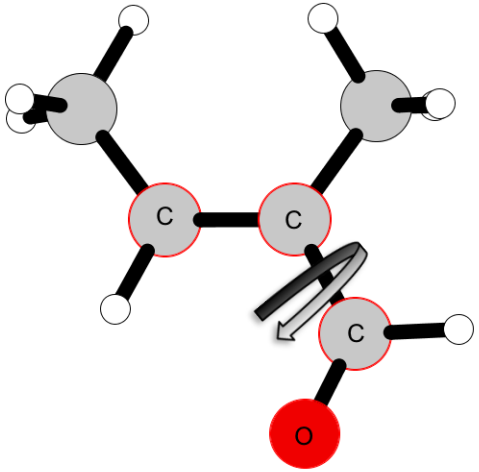
CSD comparison



44 hits

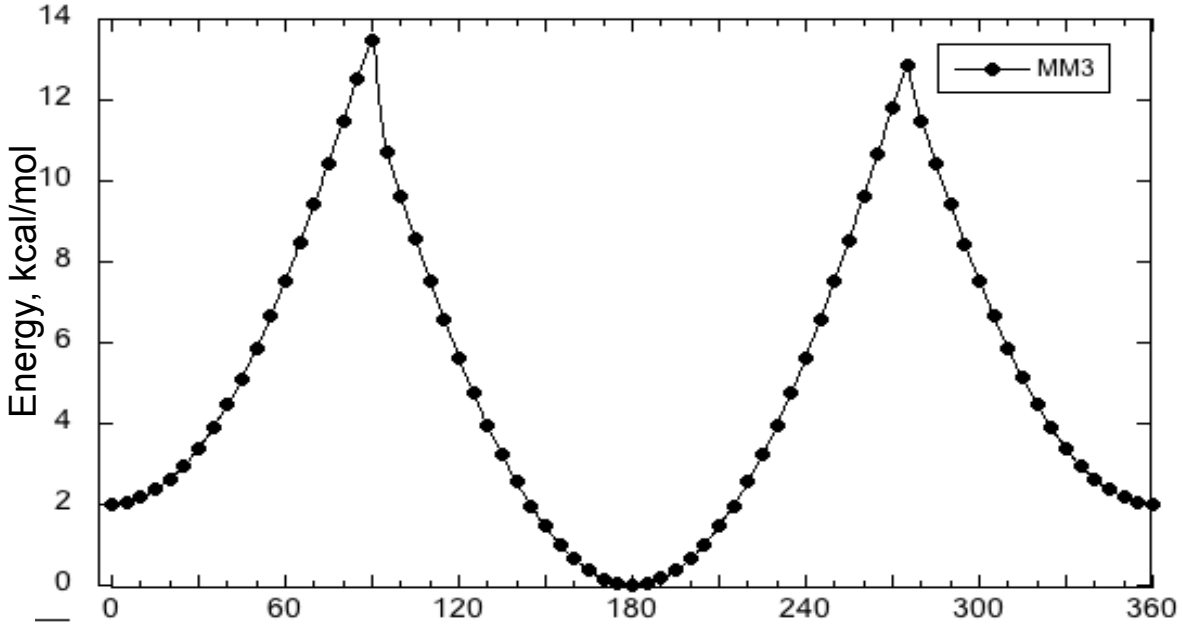
C–P = 1.82 ± 0.01Å



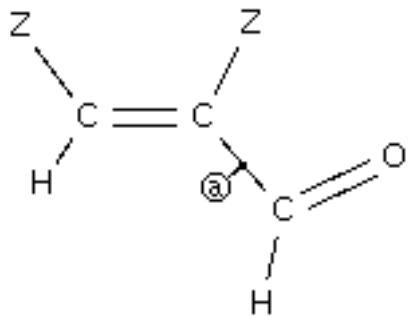


TYPE 2-3:10-1

Φ	E
0.0	2.02
180.0	0.00

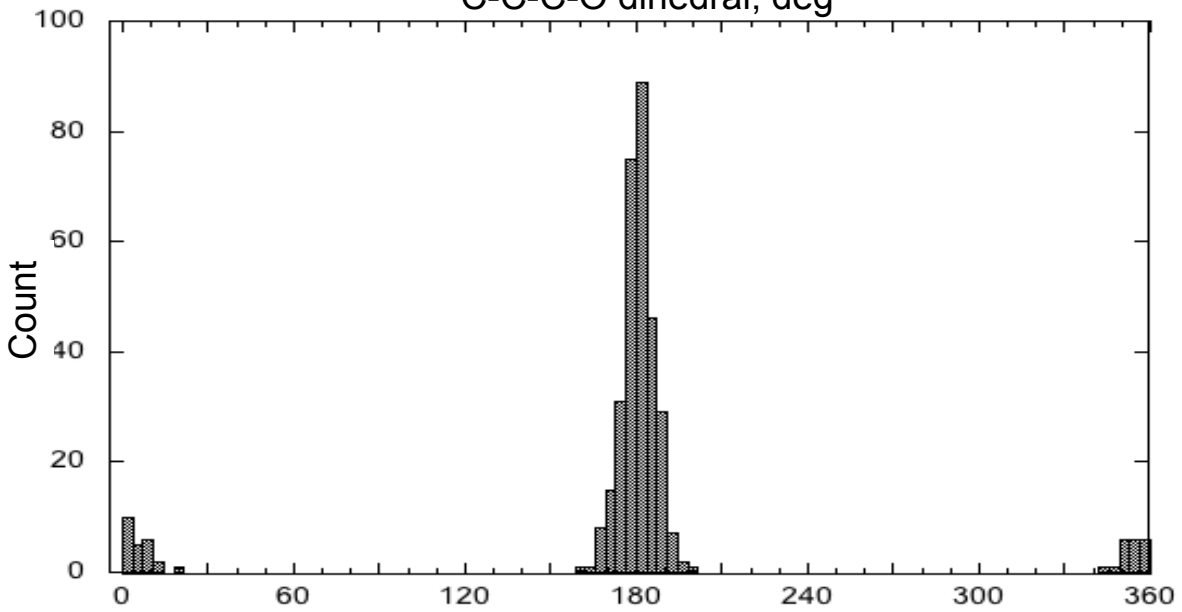


CSD comparison

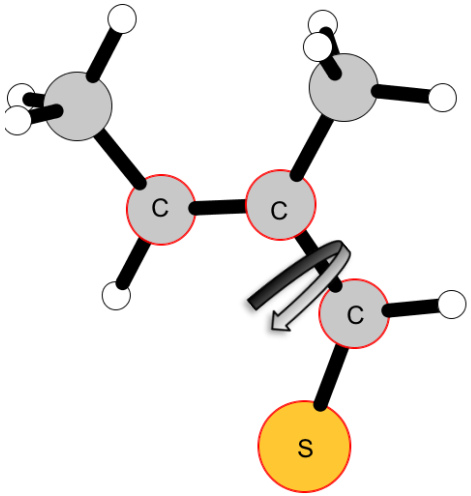


262 hits

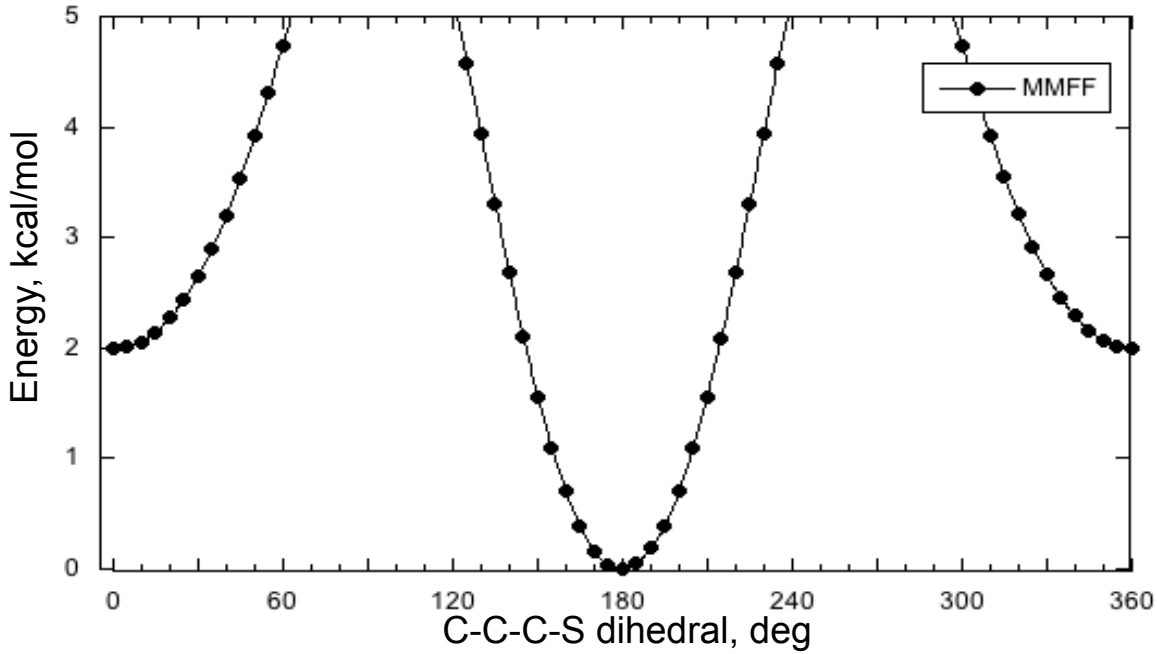
C–C = 1.45 ± 0.02 Å



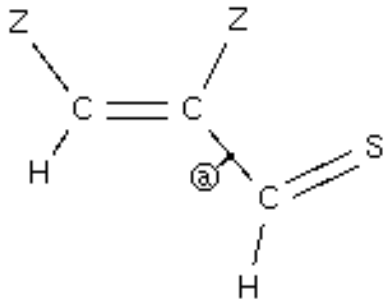
TYPE 2-3:10-2



Φ	E
0.0	2.00
180.0	0.00



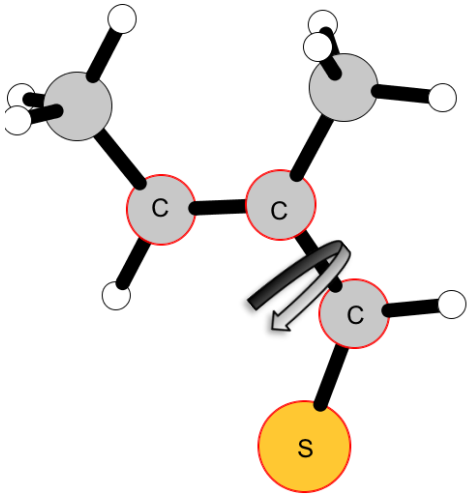
CSD comparison



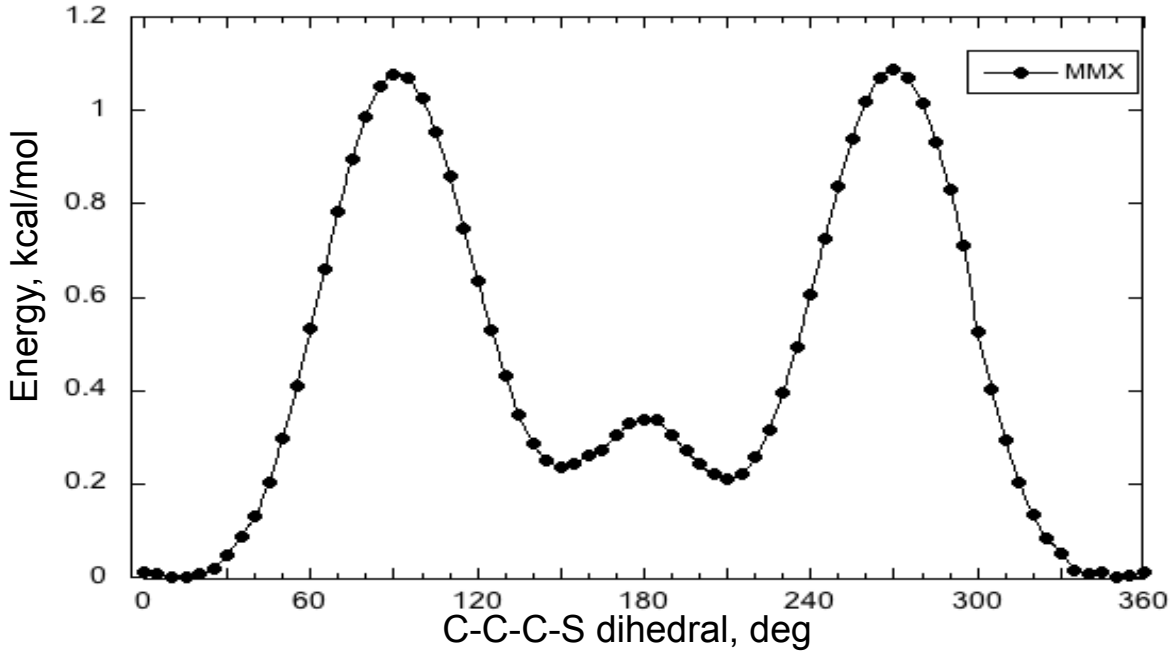
0 hits

NO CSD HITS

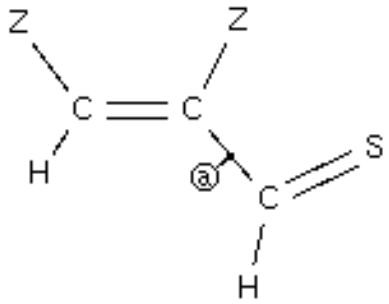
TYPE 2-3:10-2



Φ	E
15.0	0.00
150.0	0.24
210.0	0.21
350.0	0.00



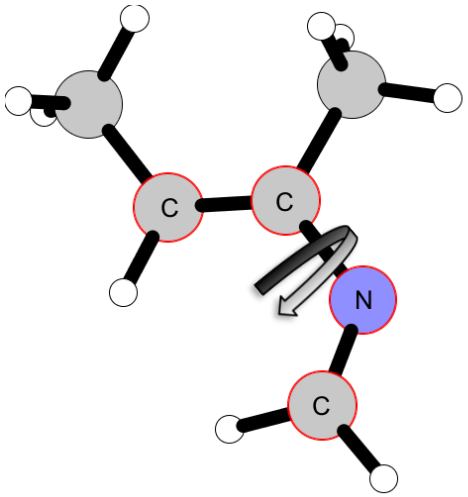
CSD comparison



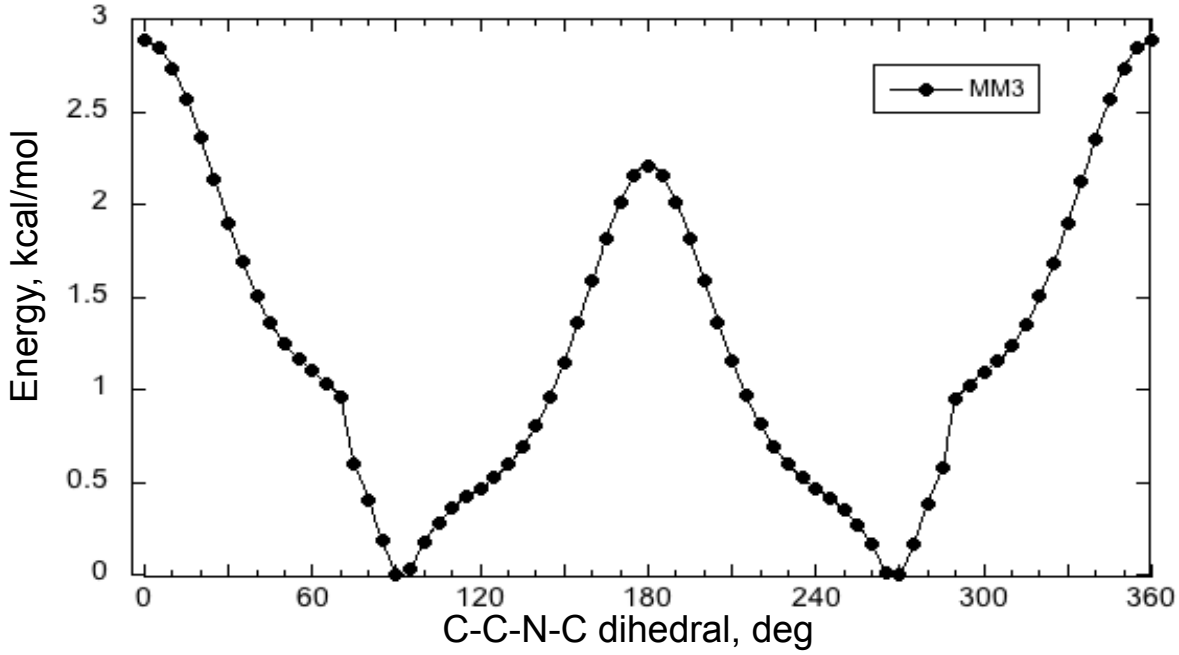
0 hits

NO CSD HITS

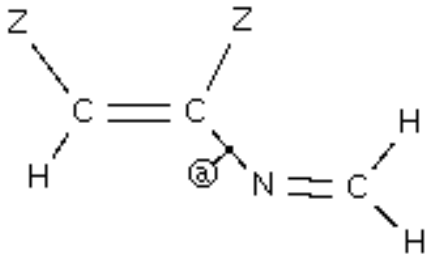
TYPE 2-3:10-3



Φ	E
90.0	0.00
270.0	0.01

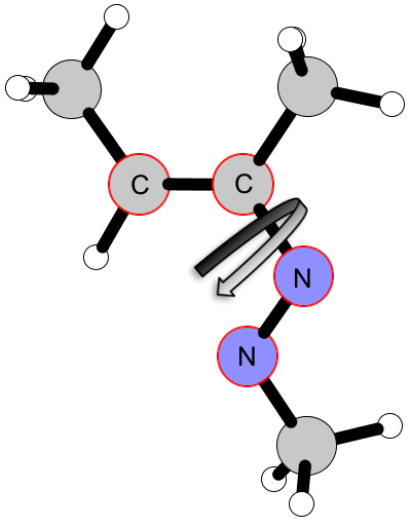


CSD comparison



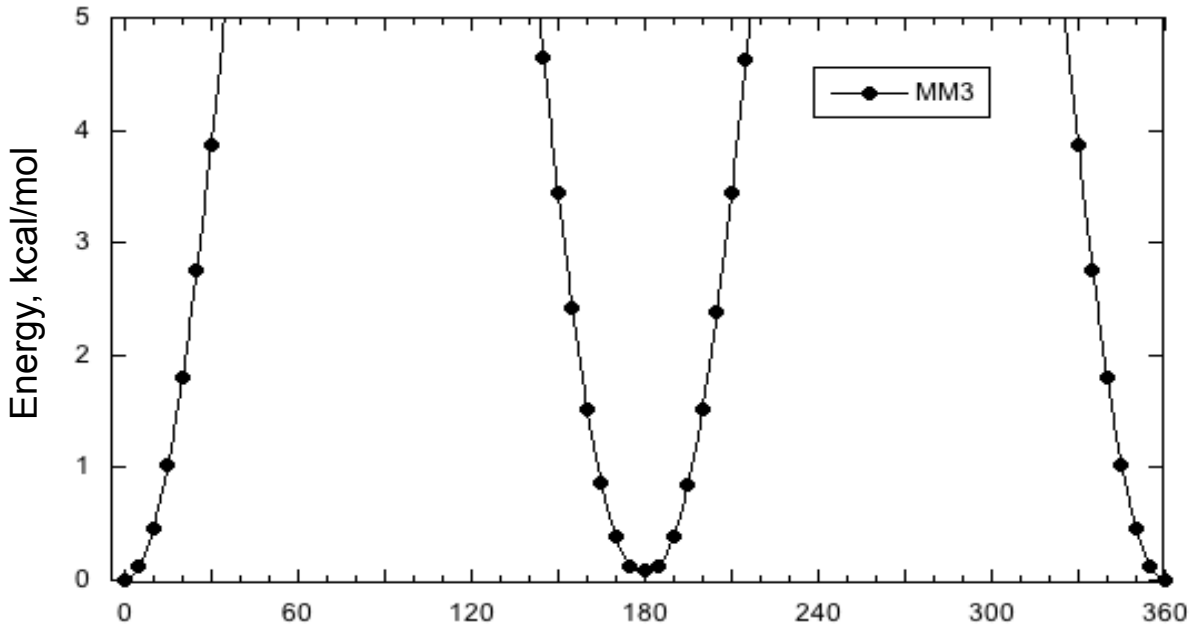
0 hits

NO CSD HITS



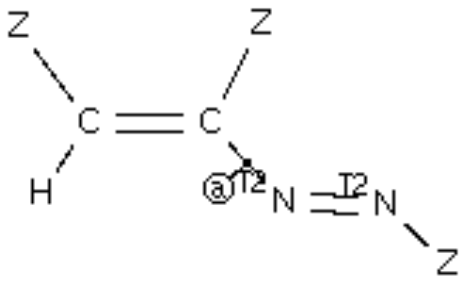
TYPE 2-3:10-4

Φ	E
0.0	0.00
180.0	0.09



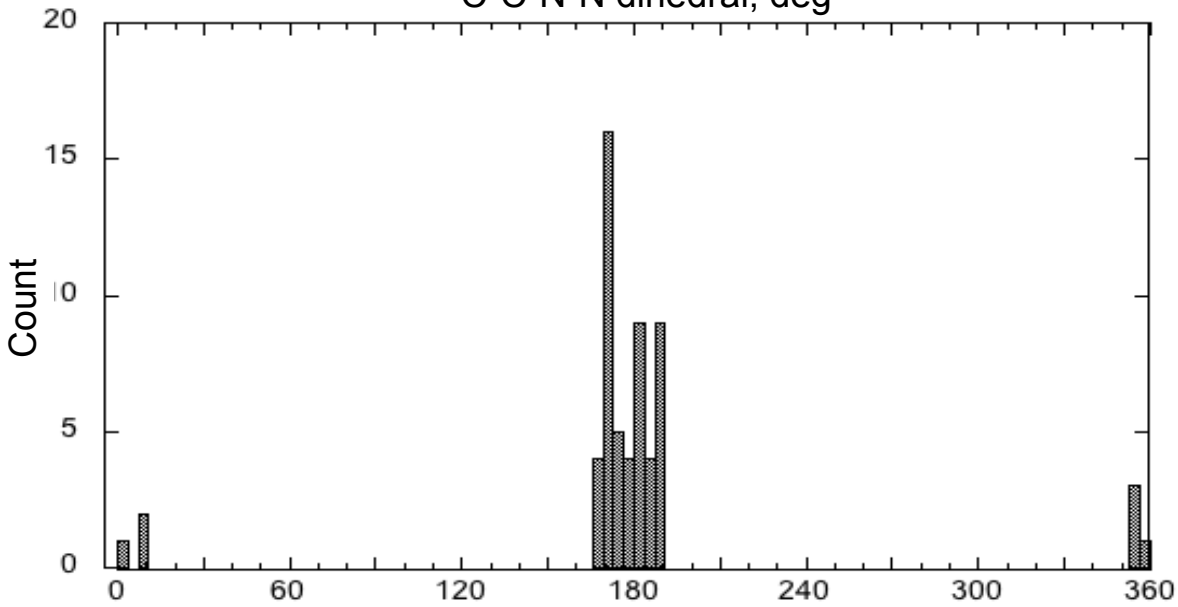
C-C-N-N dihedral, deg

CSD comparison

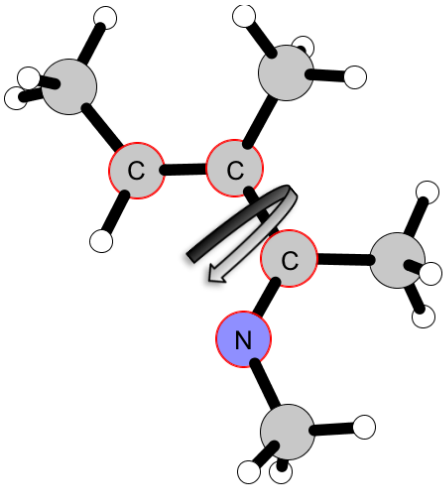


34 hits

C–N = 1.39 ± 0.02 Å

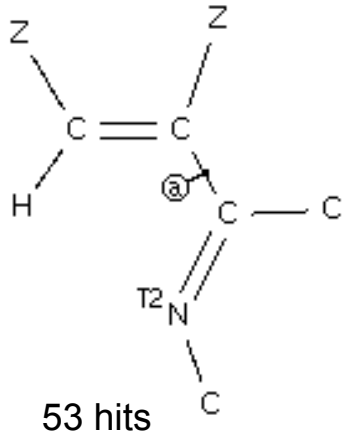


TYPE 2-3:11-1



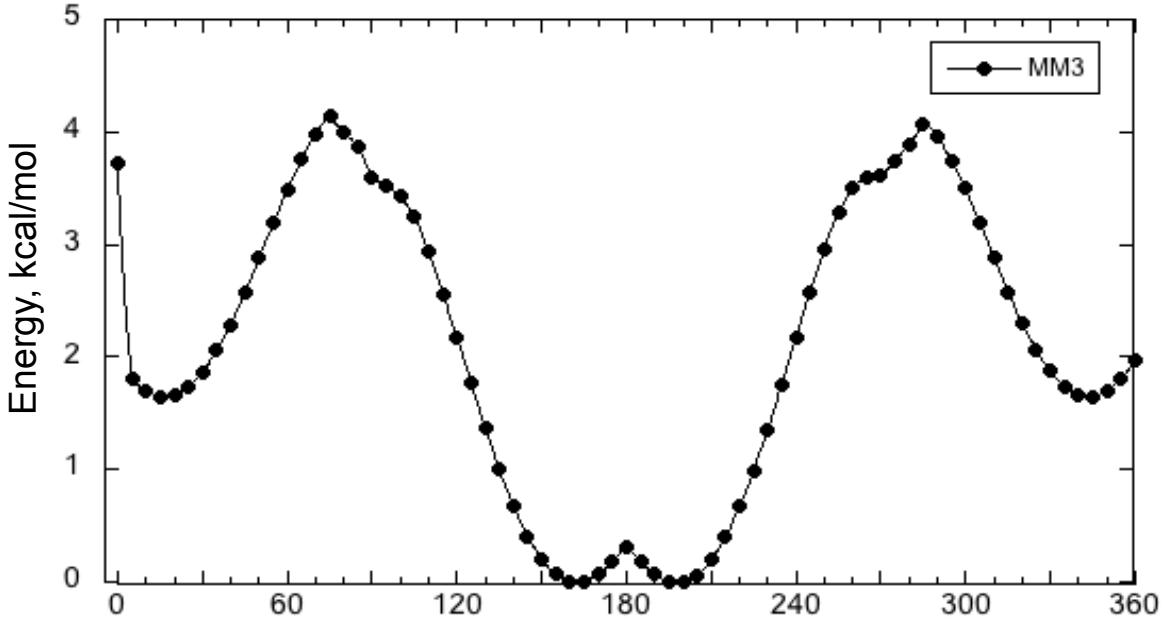
Φ	E
15.0	1.65
165.0	0.00
200.0	0.00
345.0	1.65

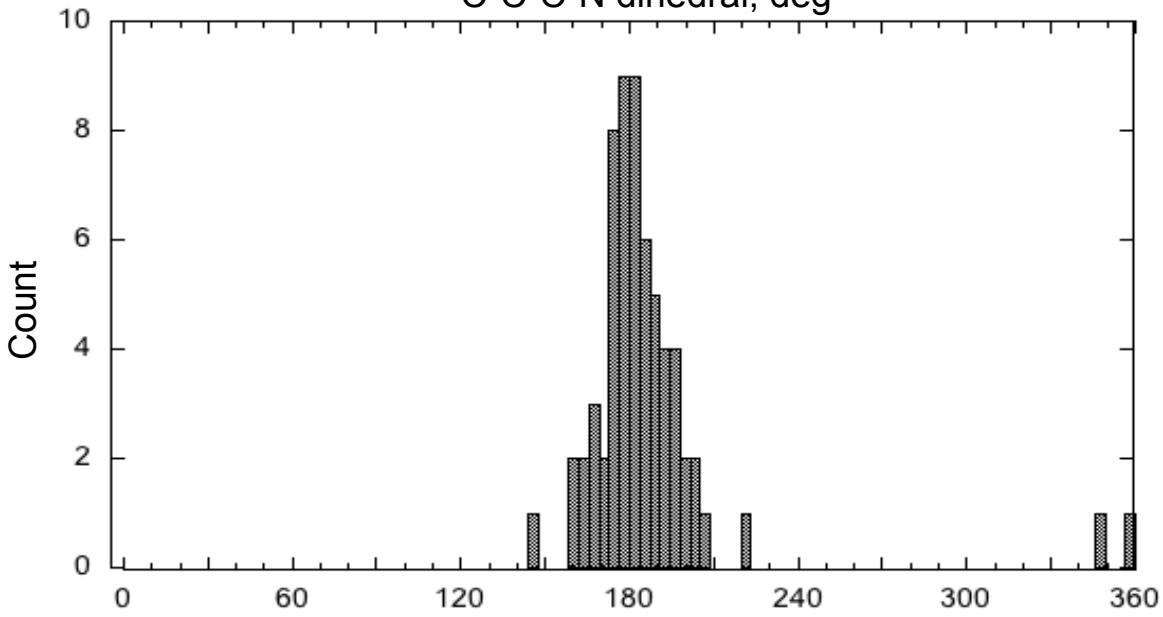
CSD comparison



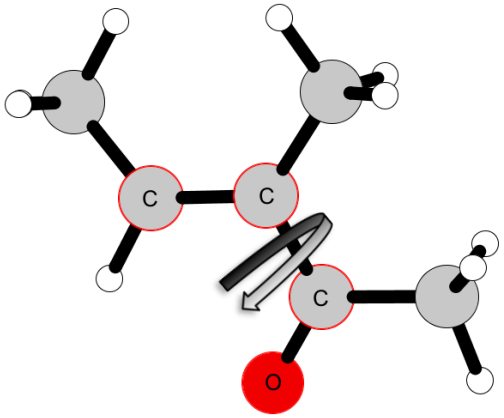
53 hits

C–C = 1.46 ± 0.01 Å

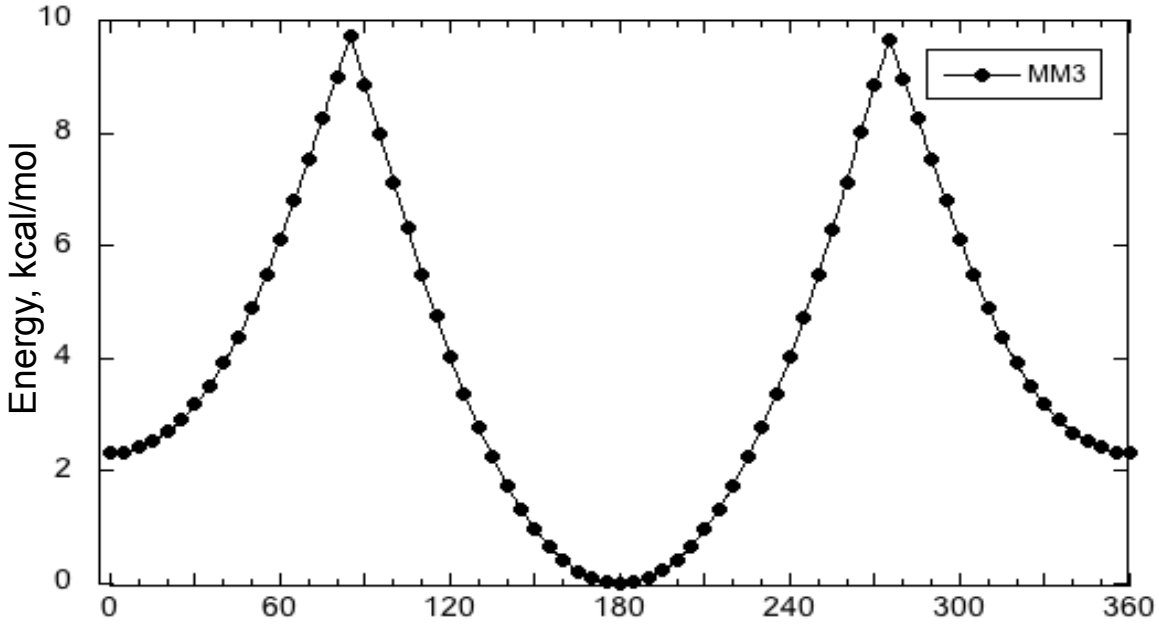




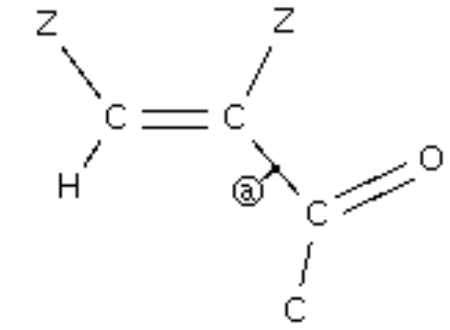
TYPE 2-3:11-2



Φ	E
0.0	2.32
180.0	0.00

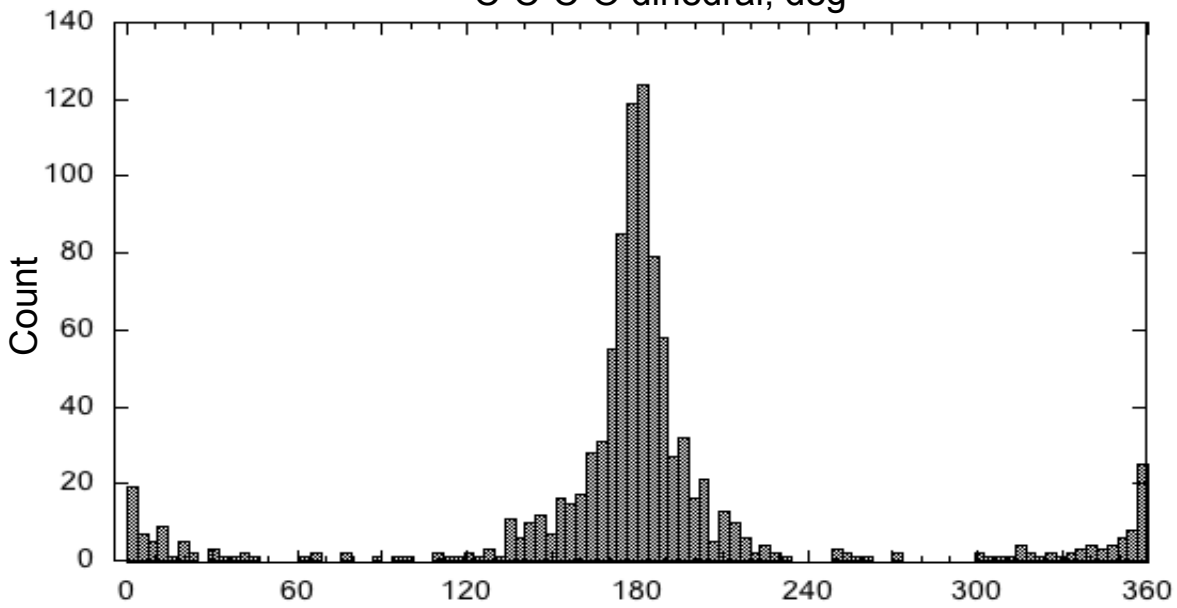


CSD comparison

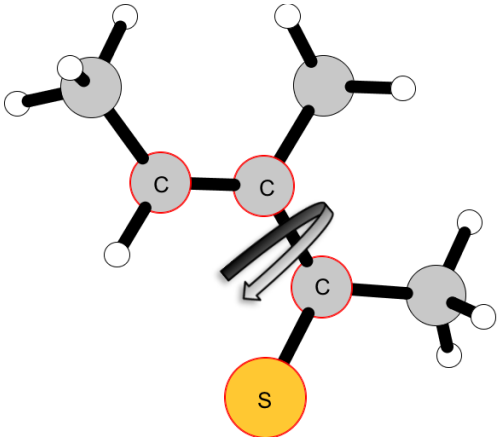


770 hits

C–C = 1.47 ± 0.02 Å

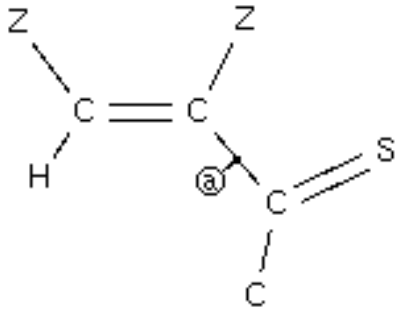


TYPE 2-3:11-3



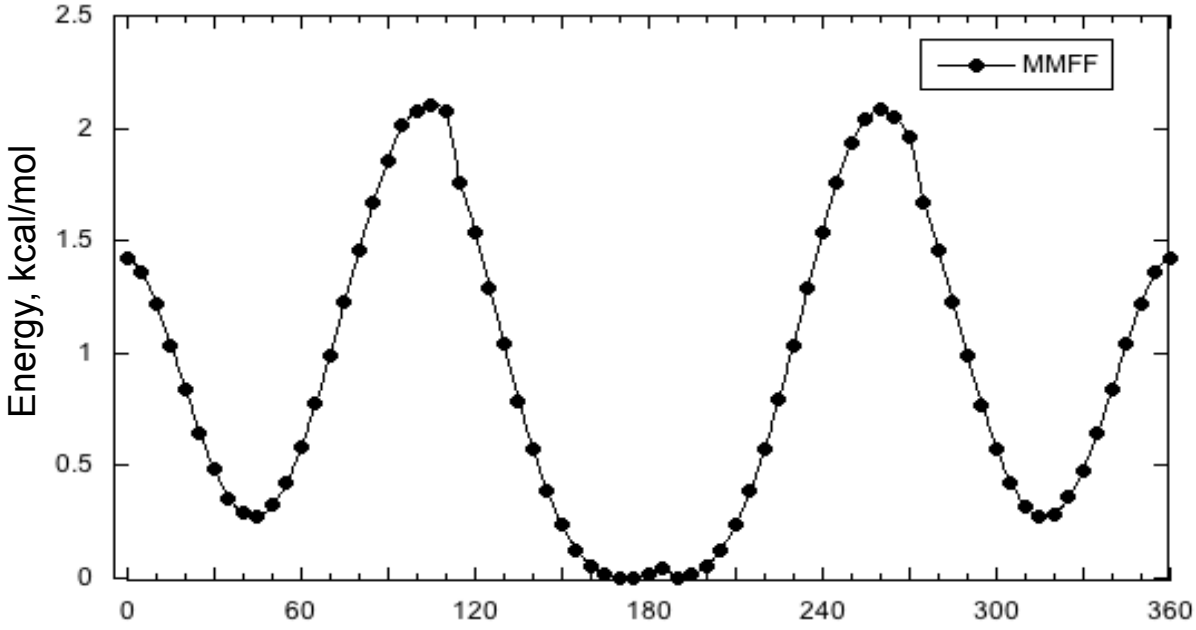
Φ	E
45.0	0.27
170.0	0.00
175.0	0.00
315.0	0.28

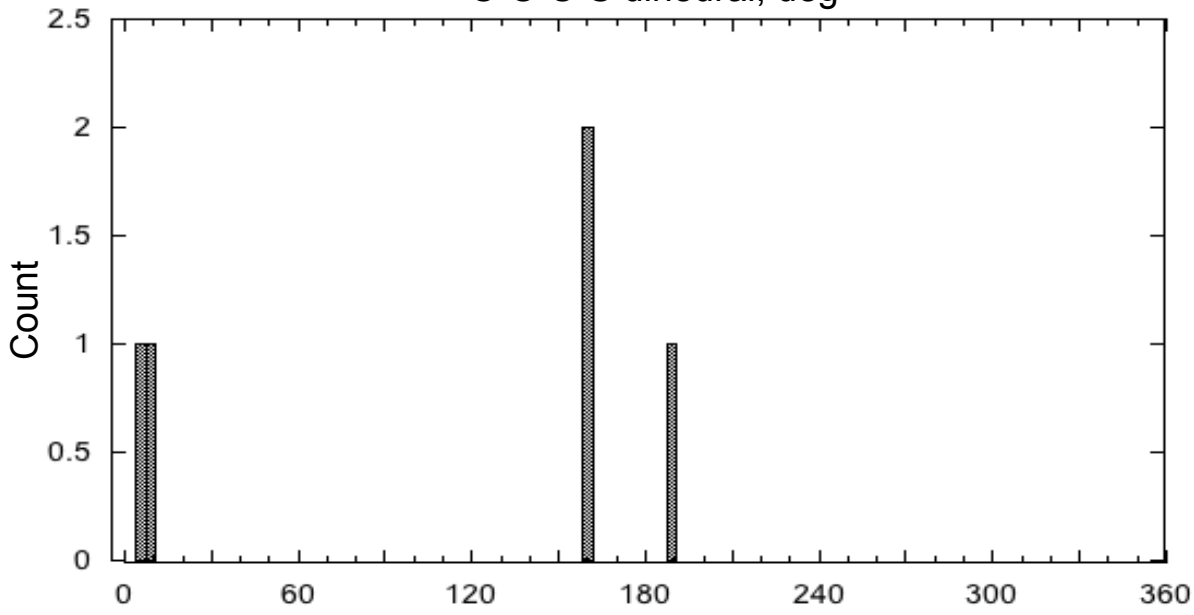
CSD comparison



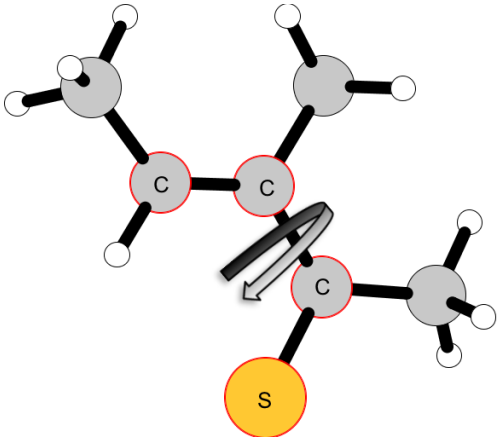
3 hits

C–C = 1.42 ± 0.02 Å



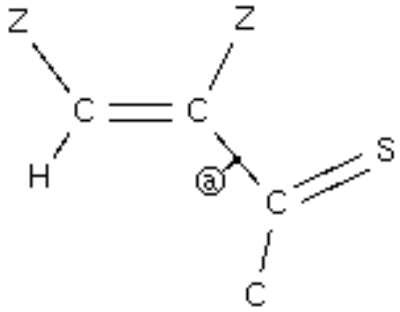


TYPE 2-3:11-3



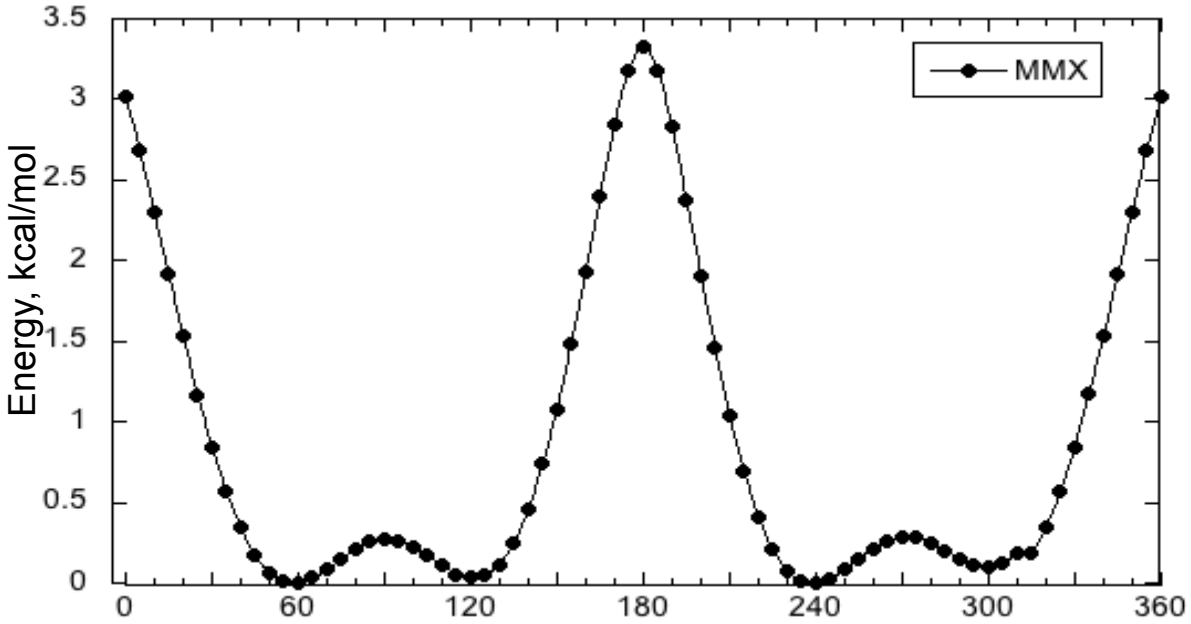
Φ	E
60.0	0.00
120.0	0.03
240.0	0.00
300.0	0.10

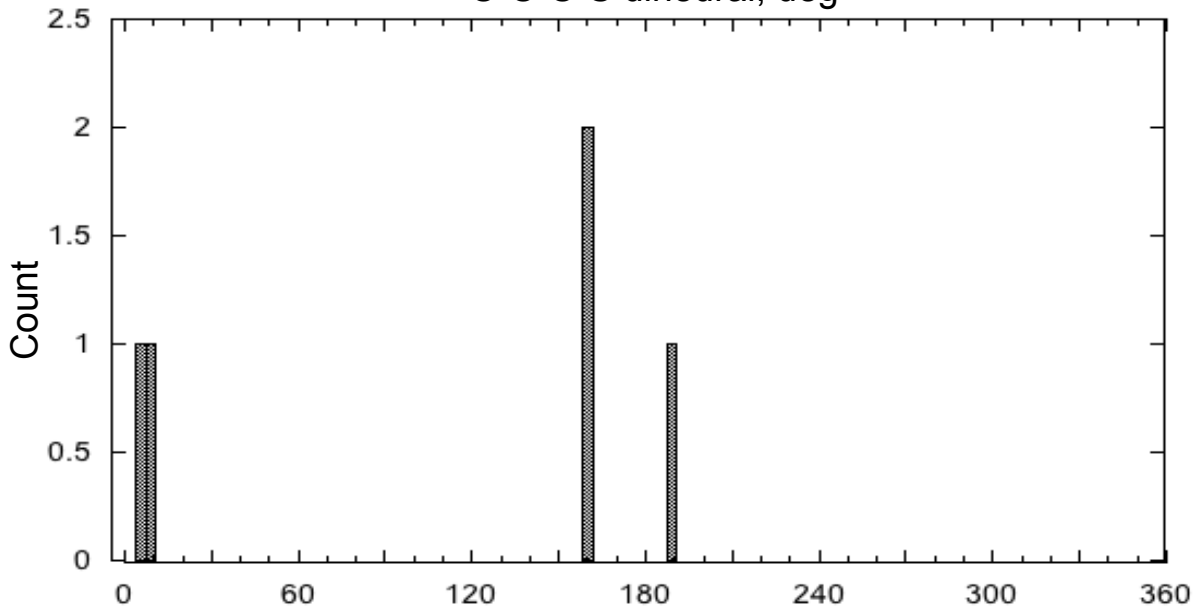
CSD comparison



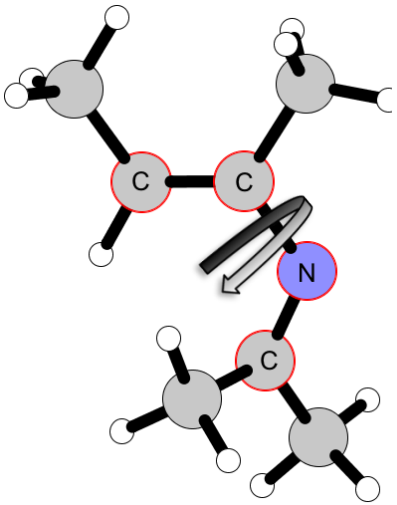
3 hits

C–C = 1.42 ± 0.02 Å

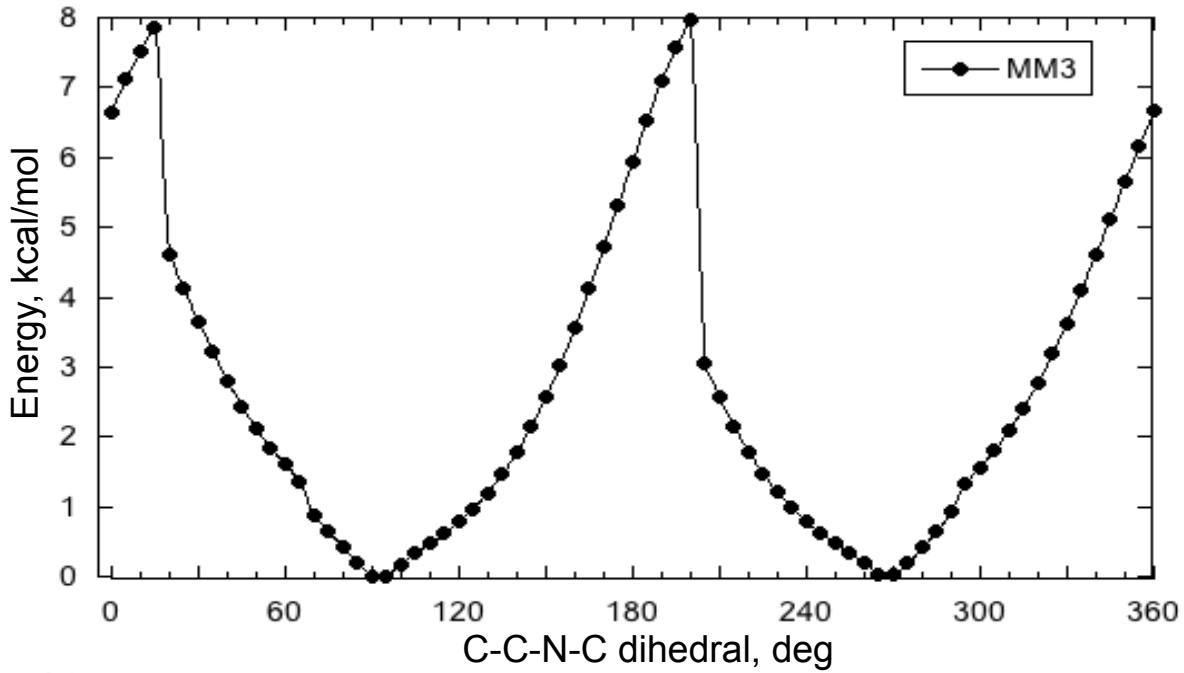




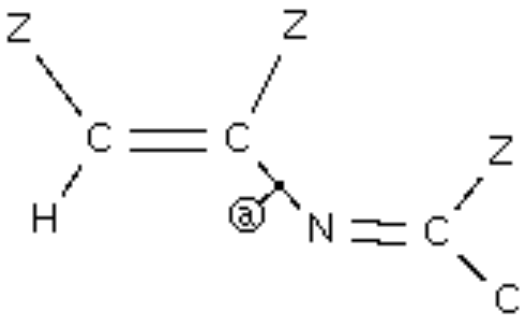
TYPE 2-3:12-1



Φ	E
95.0	0.00
270.0	0.02

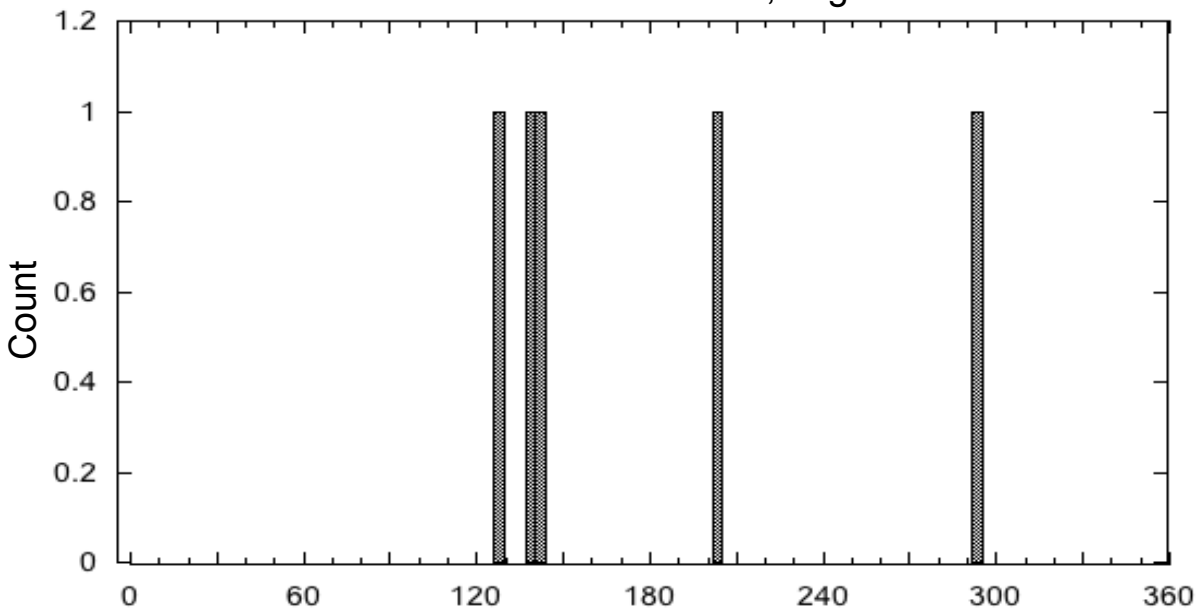


CSD comparison

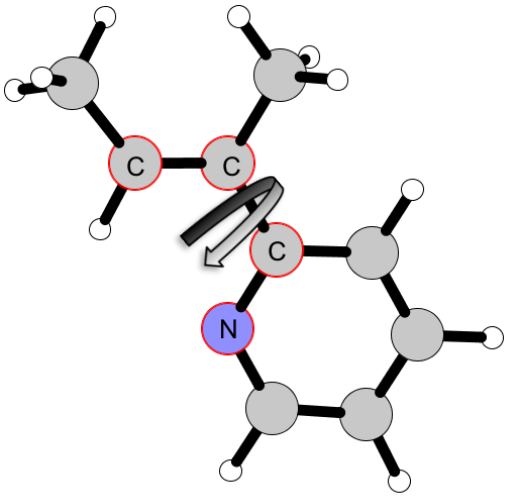


4 hits

C–N = 1.41 ± 0.01 Å

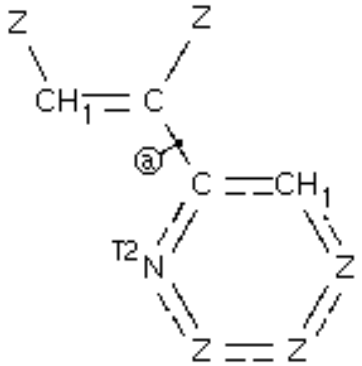


TYPE 2-3:13-1

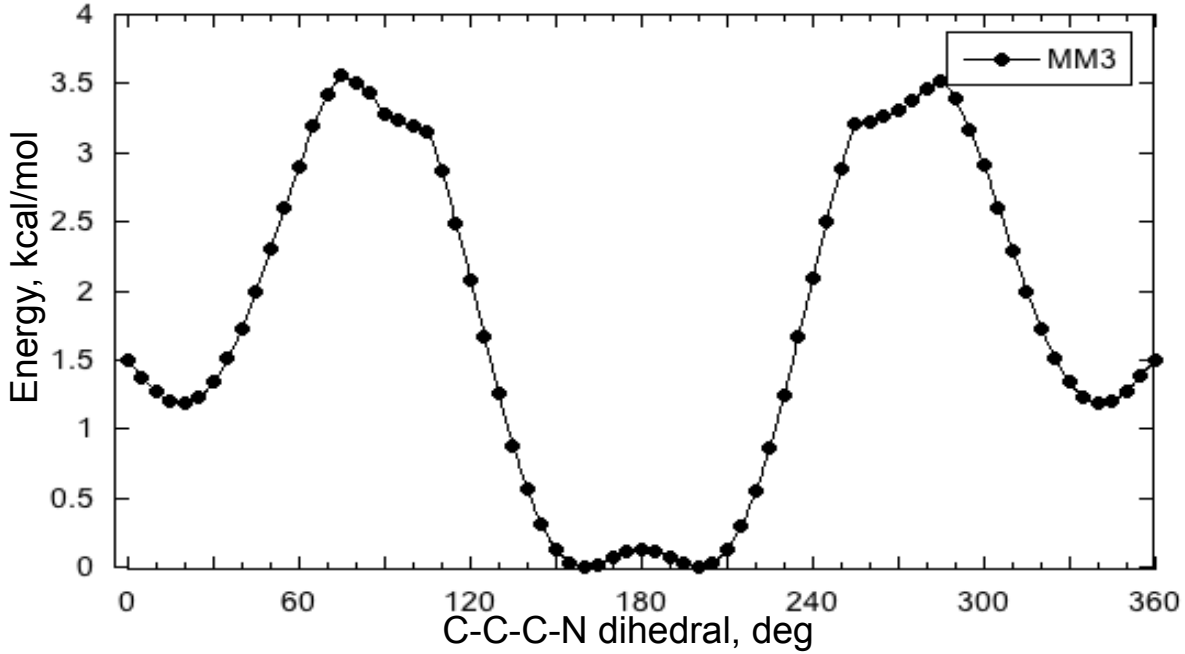


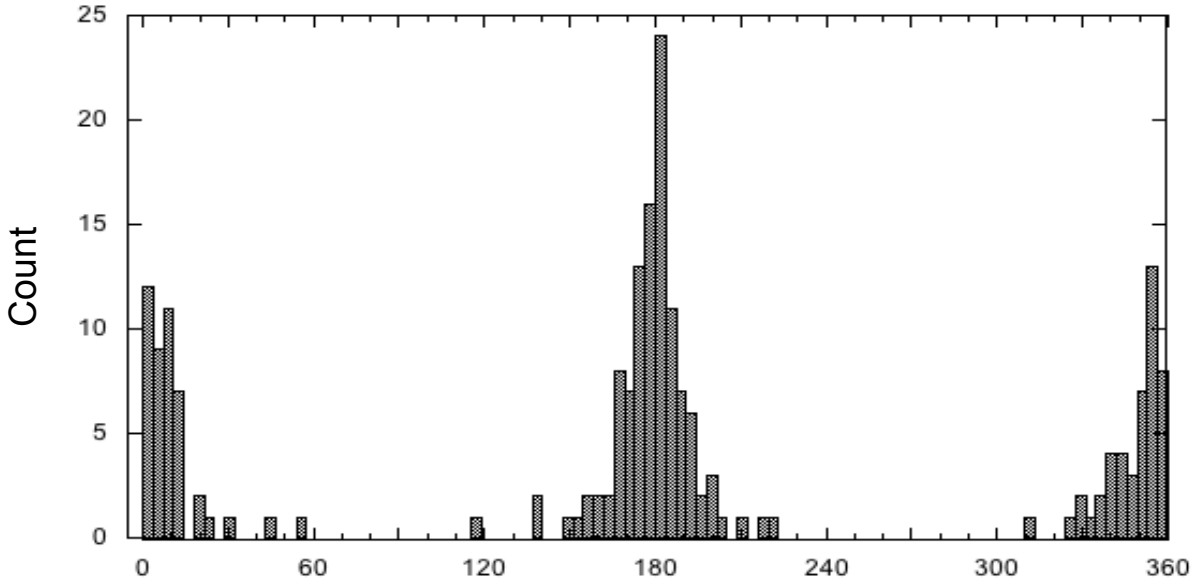
Φ	E
20.0	1.19
160.0	0.00
200.0	0.00
340.0	1.19

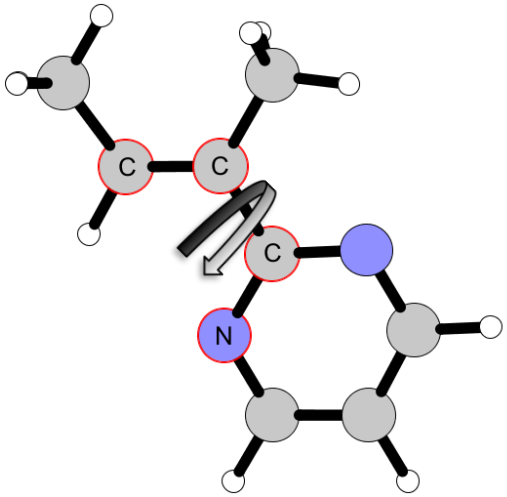
CSD comparison



123 hits
C-C = 1.47 ± 0.01 Å

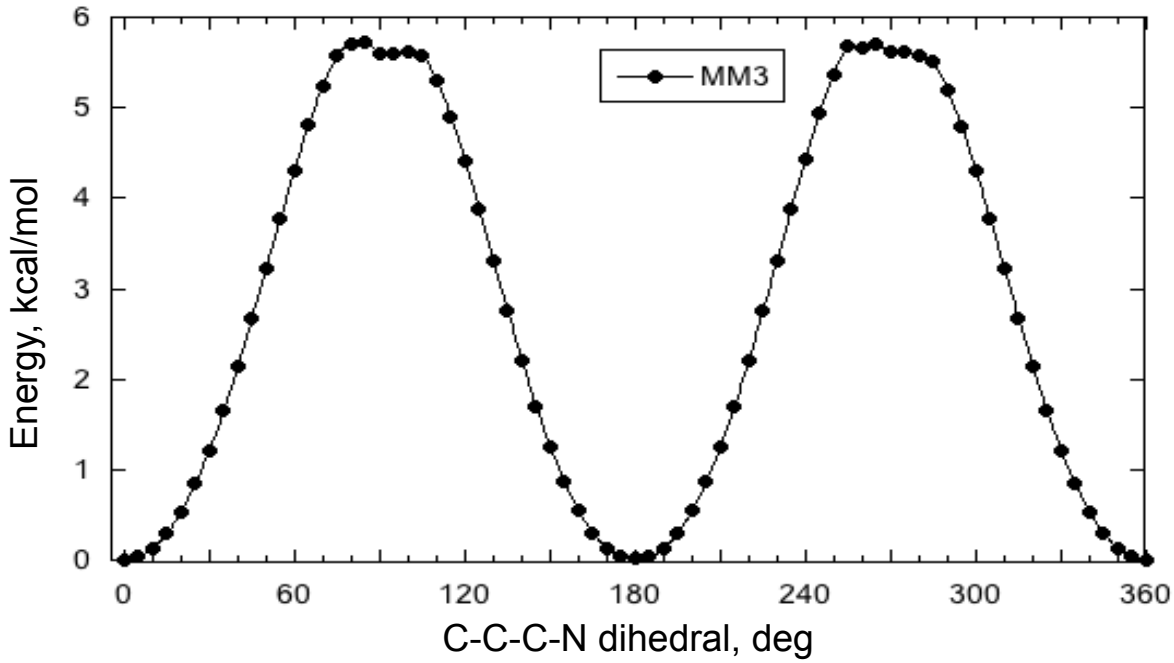




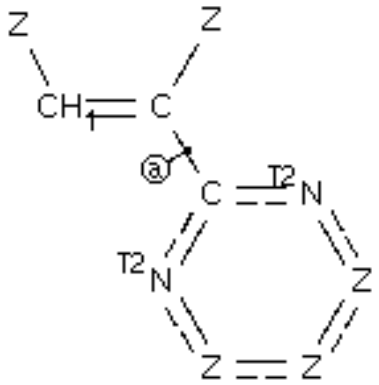


TYPE 2-3:13-2

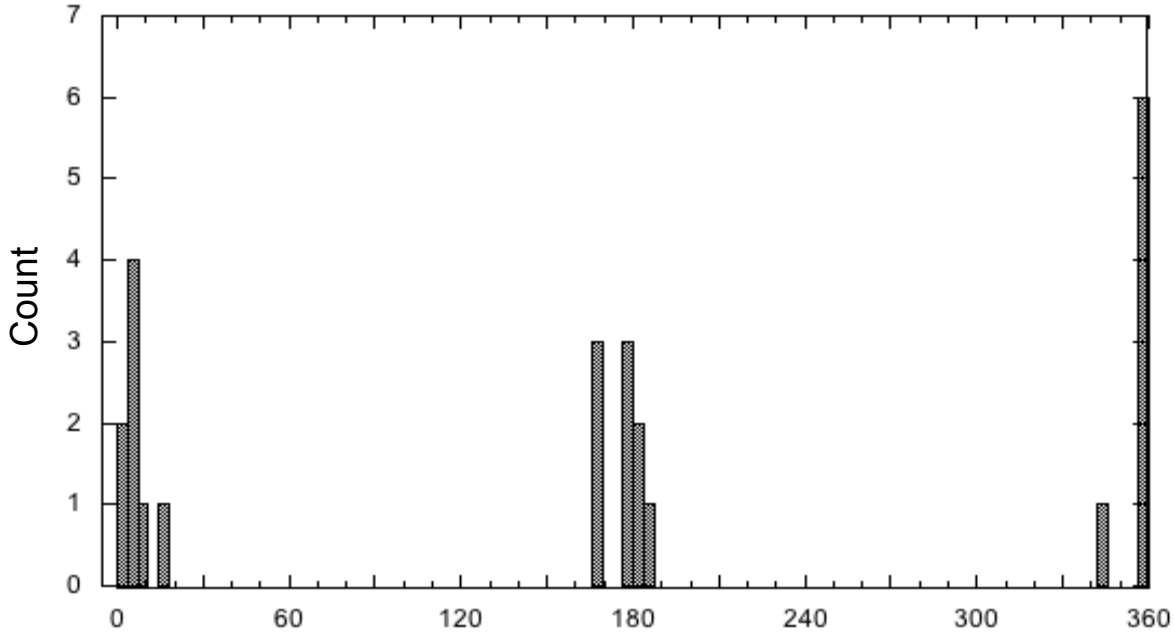
Φ	E
0.0	0.00
180.0	0.01



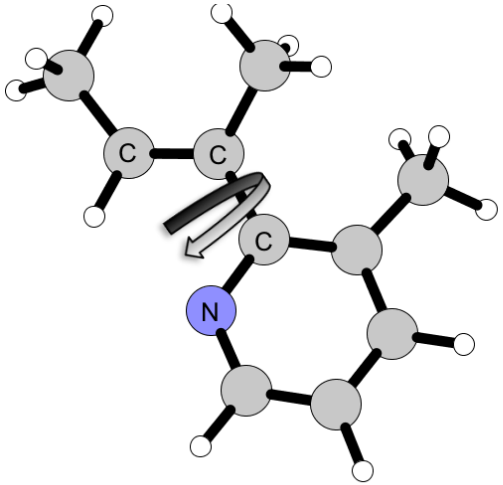
CSD comparison



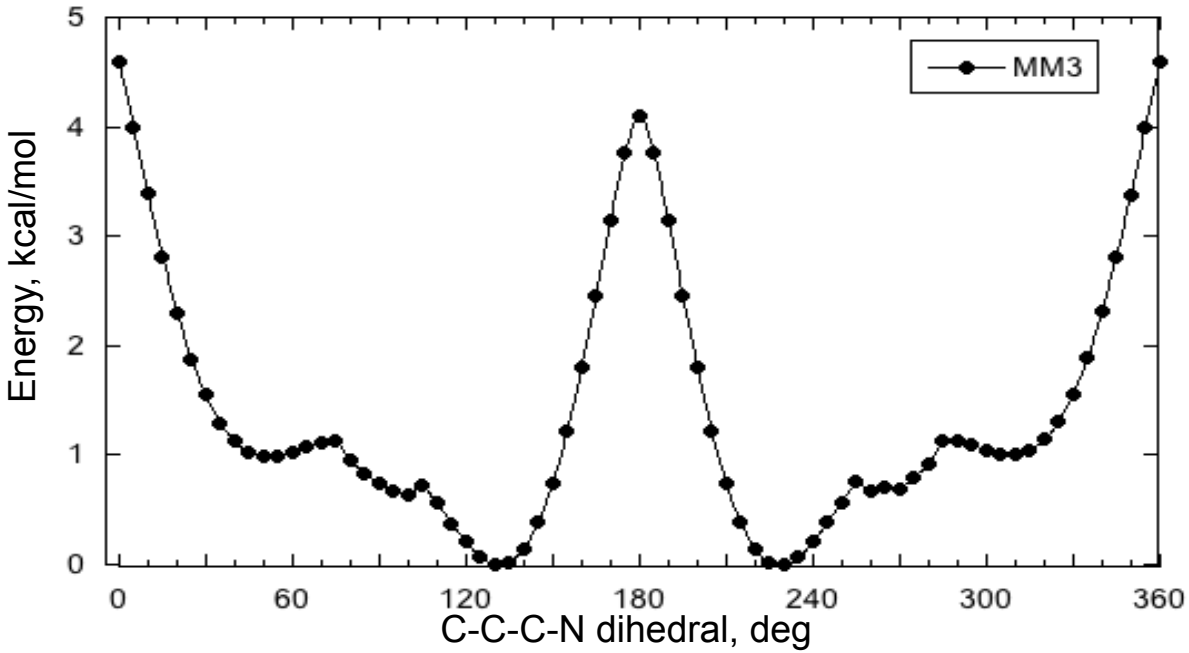
14 hits
C-C = 1.46 ± 0.02 Å



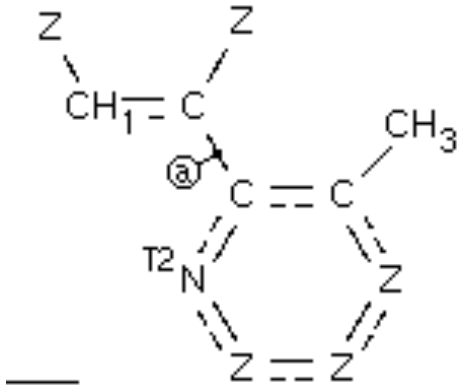
TYPE 2-3:13-3



Φ	E
50.0	0.99
100.0	0.64
130.0	0.00
230.0	0.00
260.0	0.67
310.0	1.00

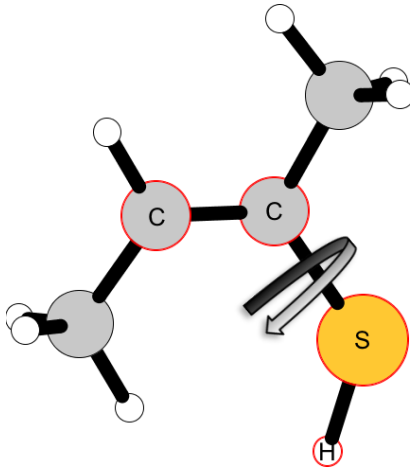


CSD comparison

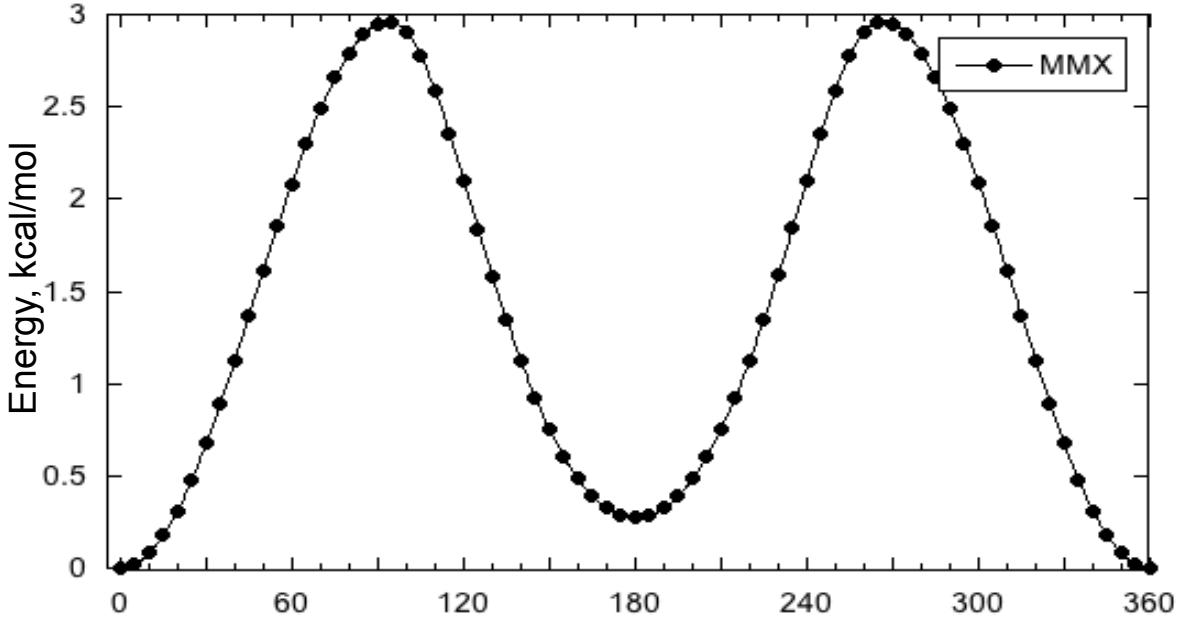


NO CSD HITS

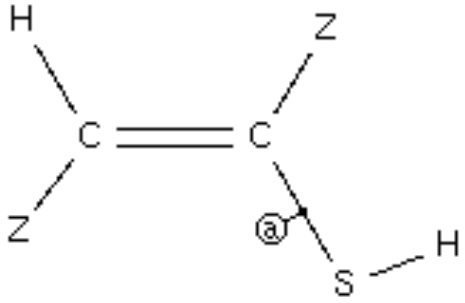
TYPE 2-4:5-4



Φ	E
0.0	0.00
180.0	0.27

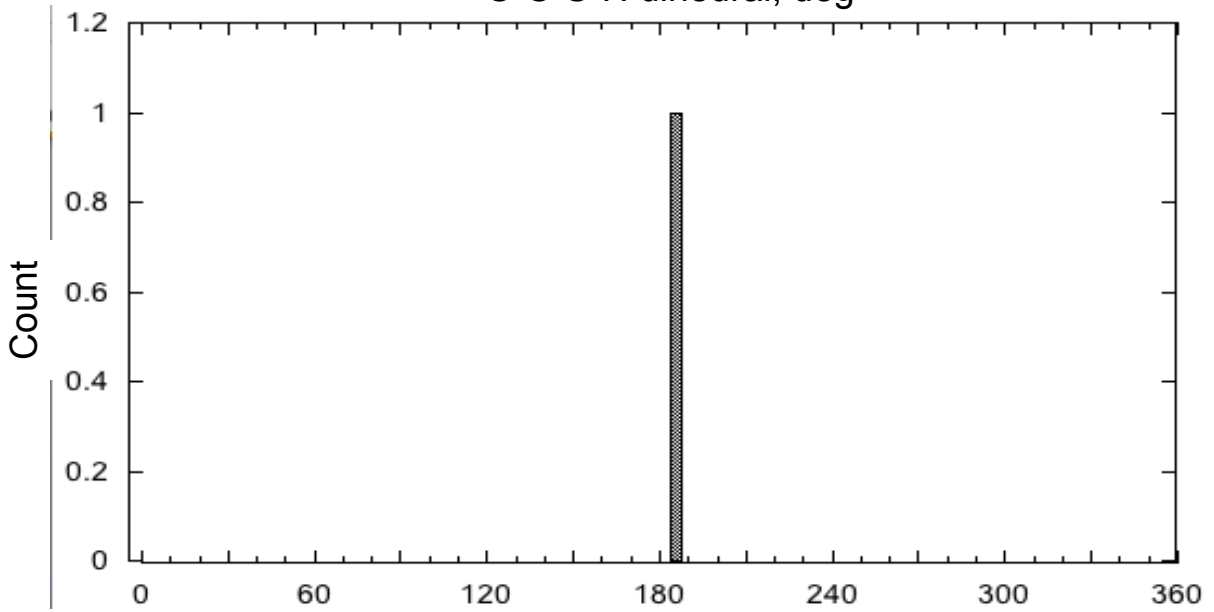


CSD comparison

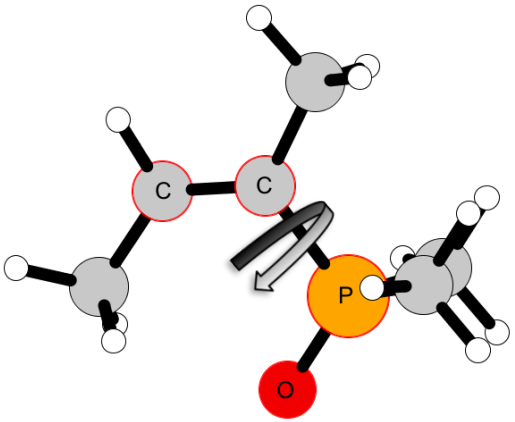


1 hits

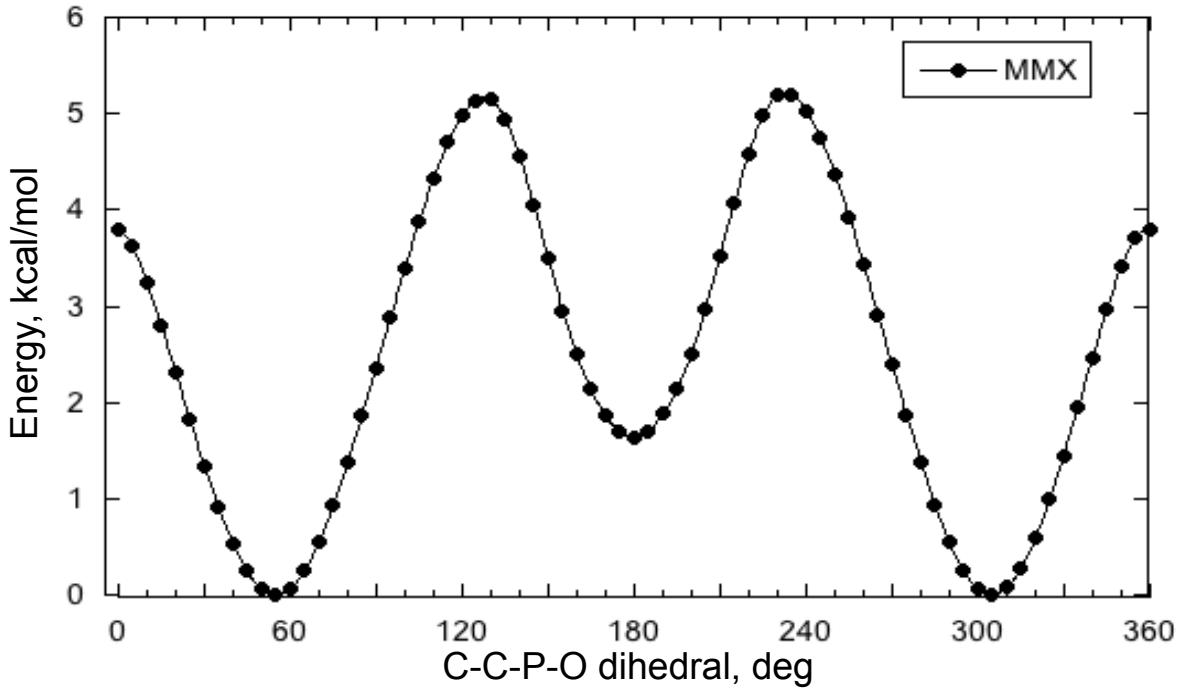
C–S = 1.75 Å



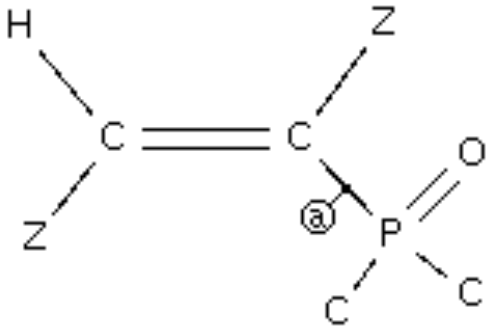
TYPE 2-4:8-1



Φ	E
55.0	0.00
180.0	1.64
305.0	0.01



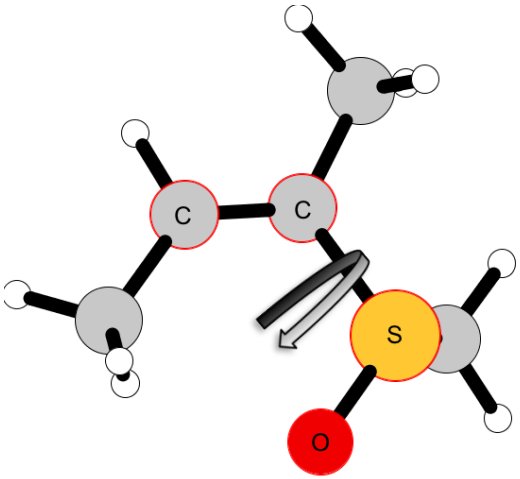
CSD comparison



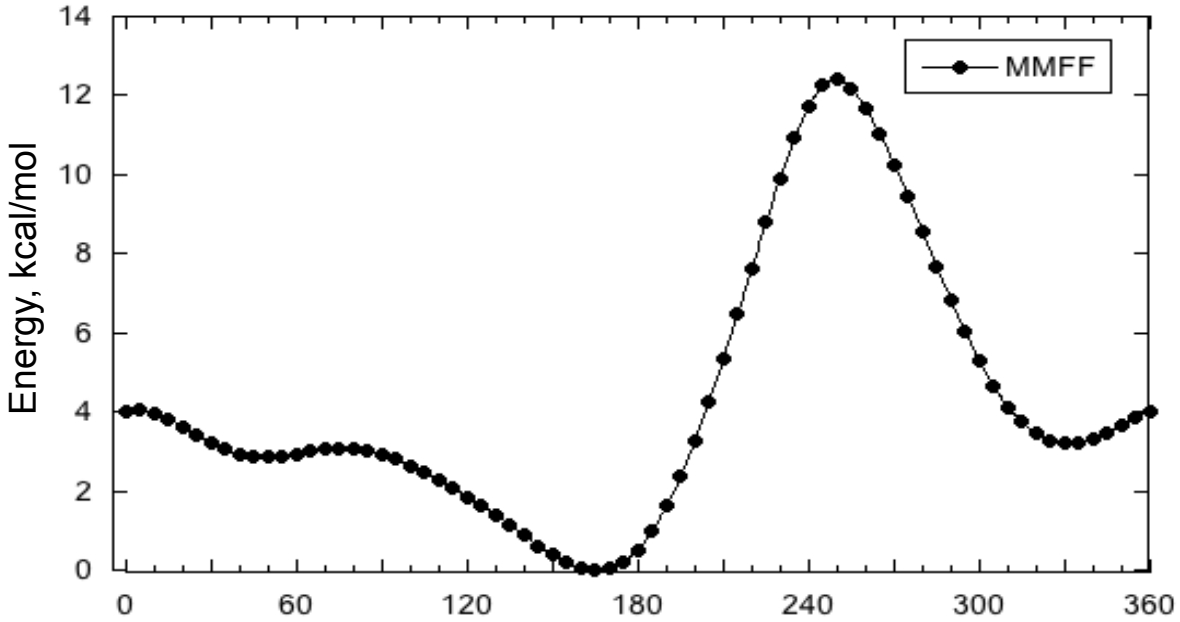
0 hits

NO CSD HITS

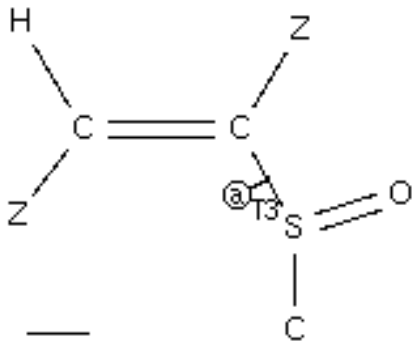
TYPE 2-4:8-2



Φ	E
50.0	2.85
165.0	0.00
330.0	3.20

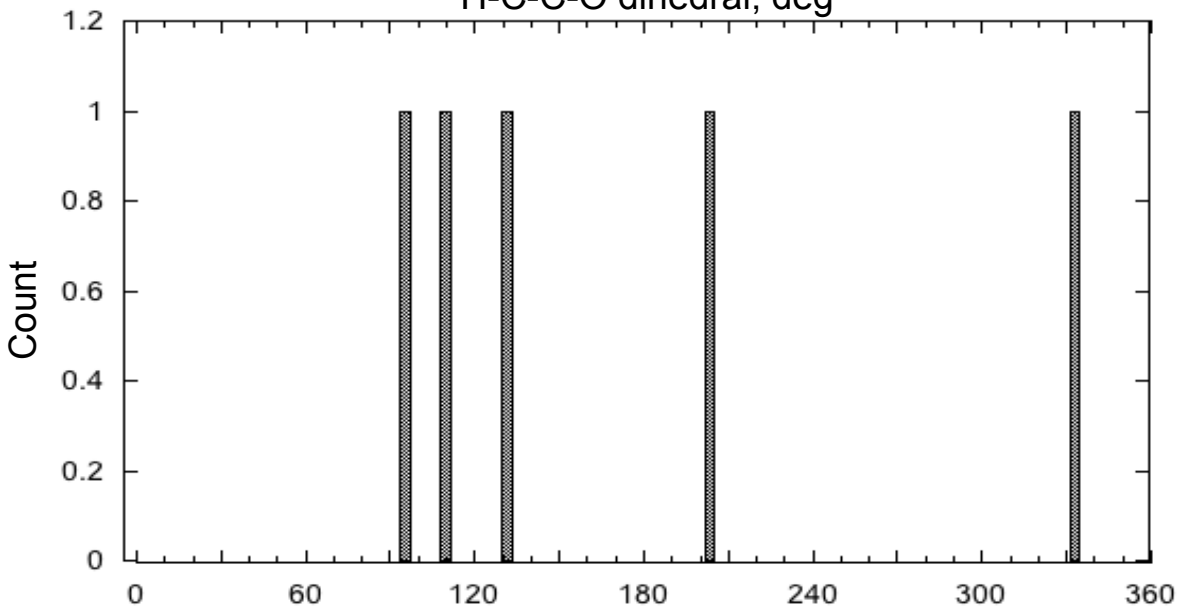


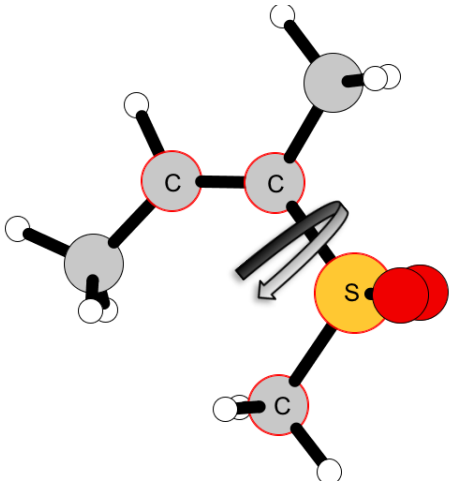
CSD comparison



5 hits

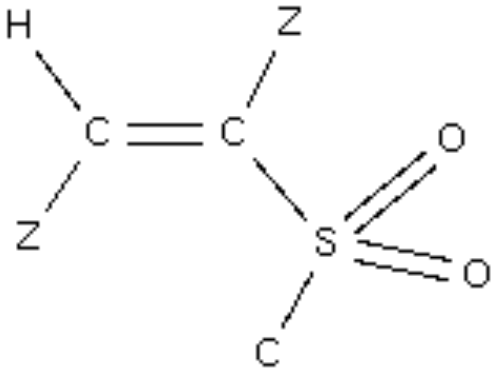
C–S = 1.80 ± 0.01 Å





Φ	E
70.0	0.33
180.0	0.00
290.0	0.33

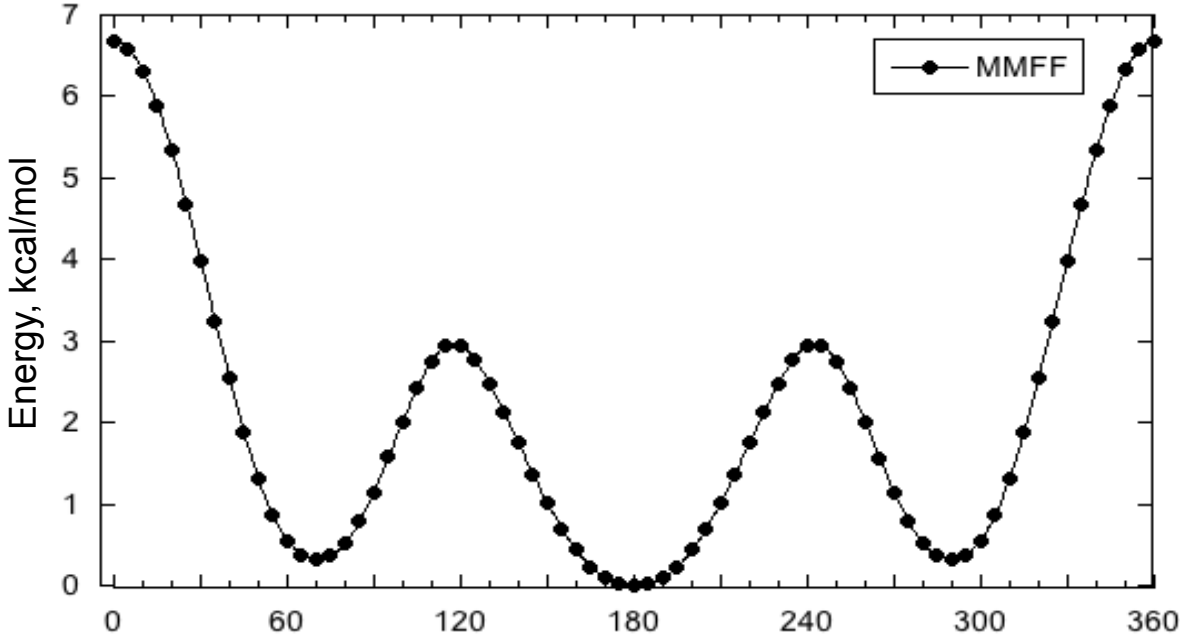
CSD comparison

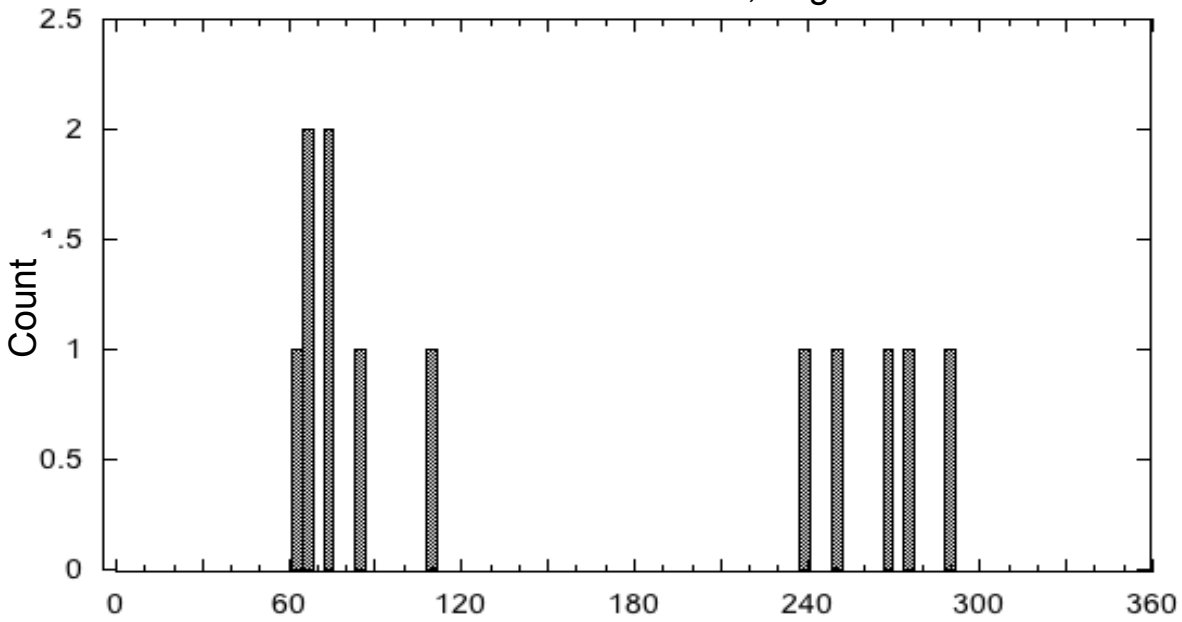


11 hits

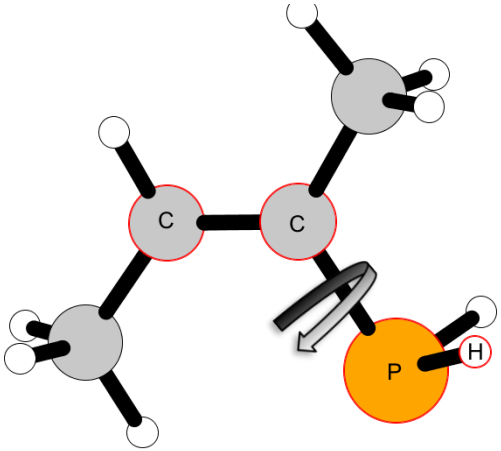
C–S = 1.77 ± 0.02 Å

TYPE 2-4:8-4

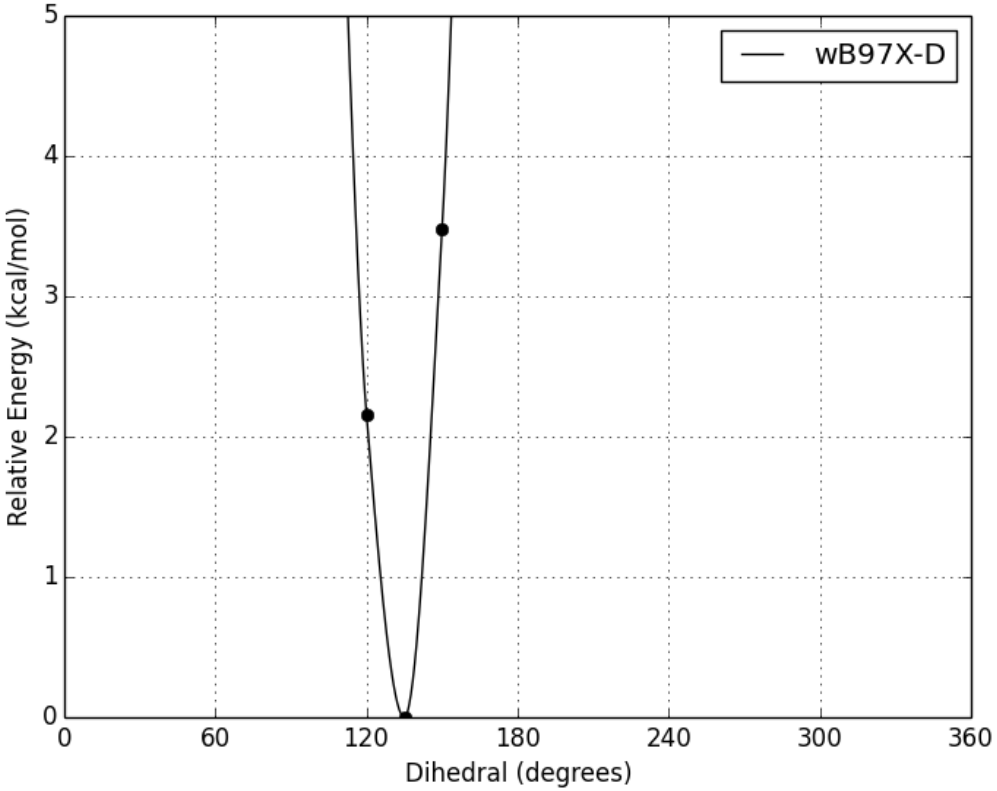




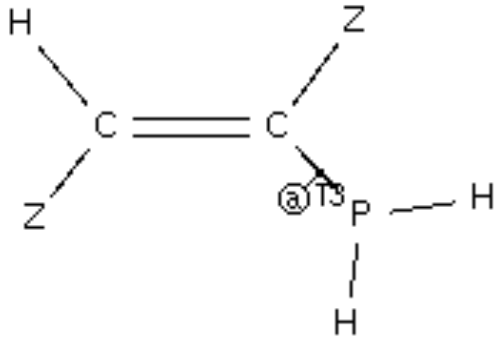
TYPE 2-4:9-1



Φ	E
135.0	0.00

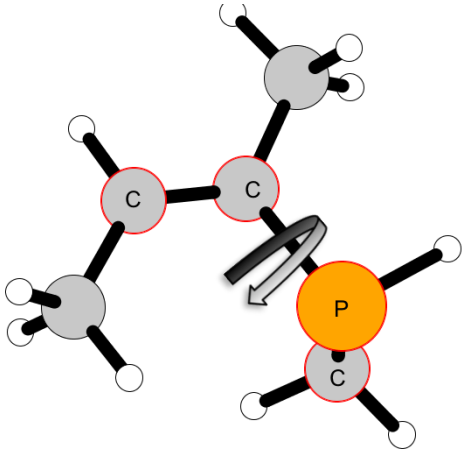


CSD comparison

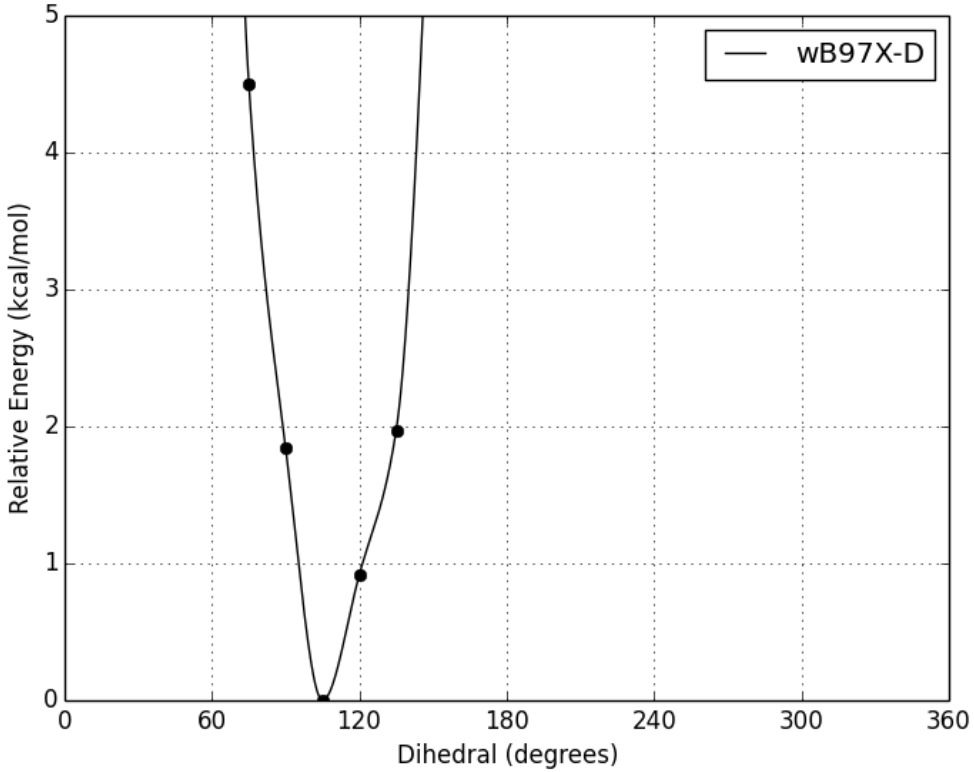


NO CSD HITS

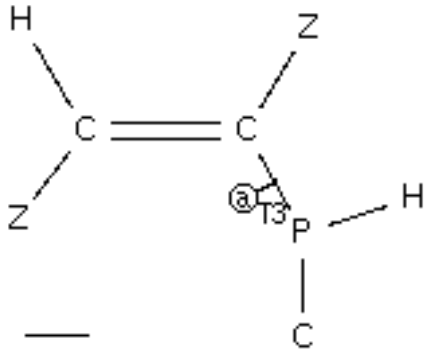
TYPE 2-4:9-2



Φ	E
105.0	0.00

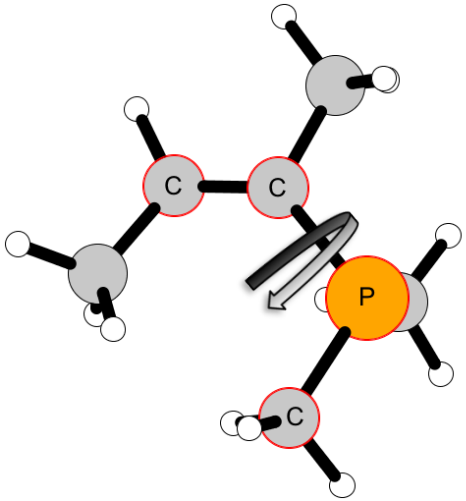


CSD comparison



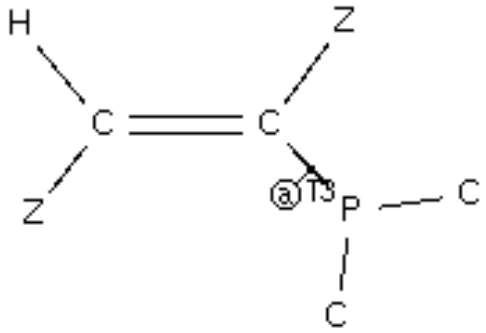
NO CSD HITS

TYPE 2-4:9-4



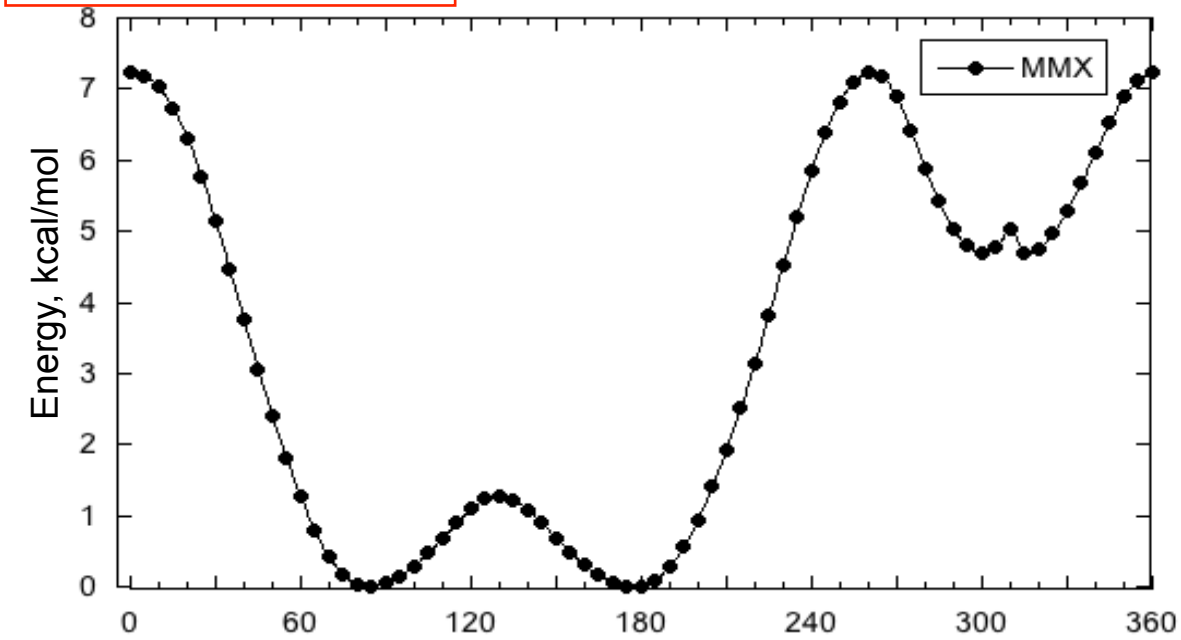
Φ	E
85.0	0.00
175.0	0.01
300.0	4.70
315.0	4.70

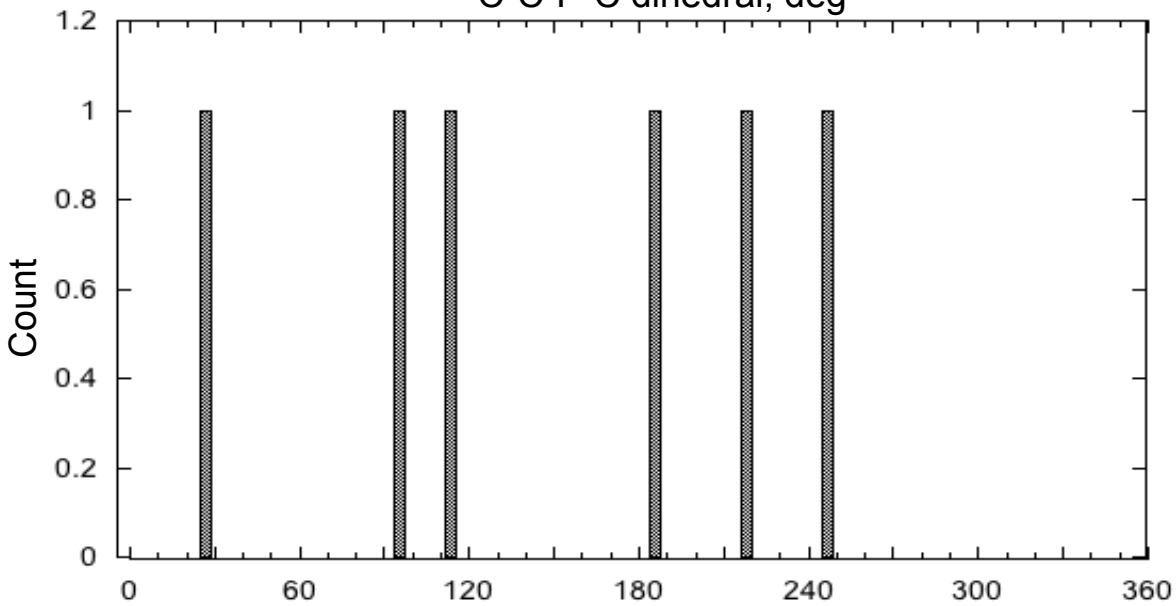
CSD comparison

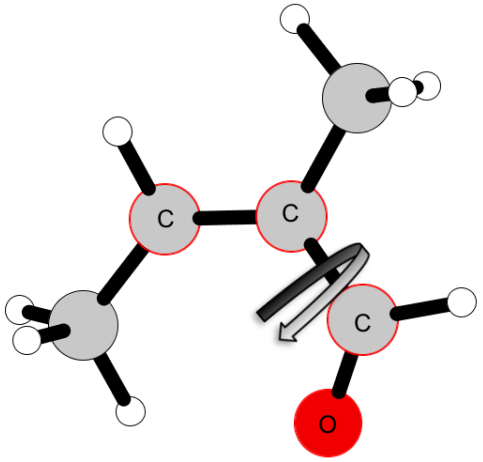


6 hits

C–P = 1.84 ± 0.01 Å

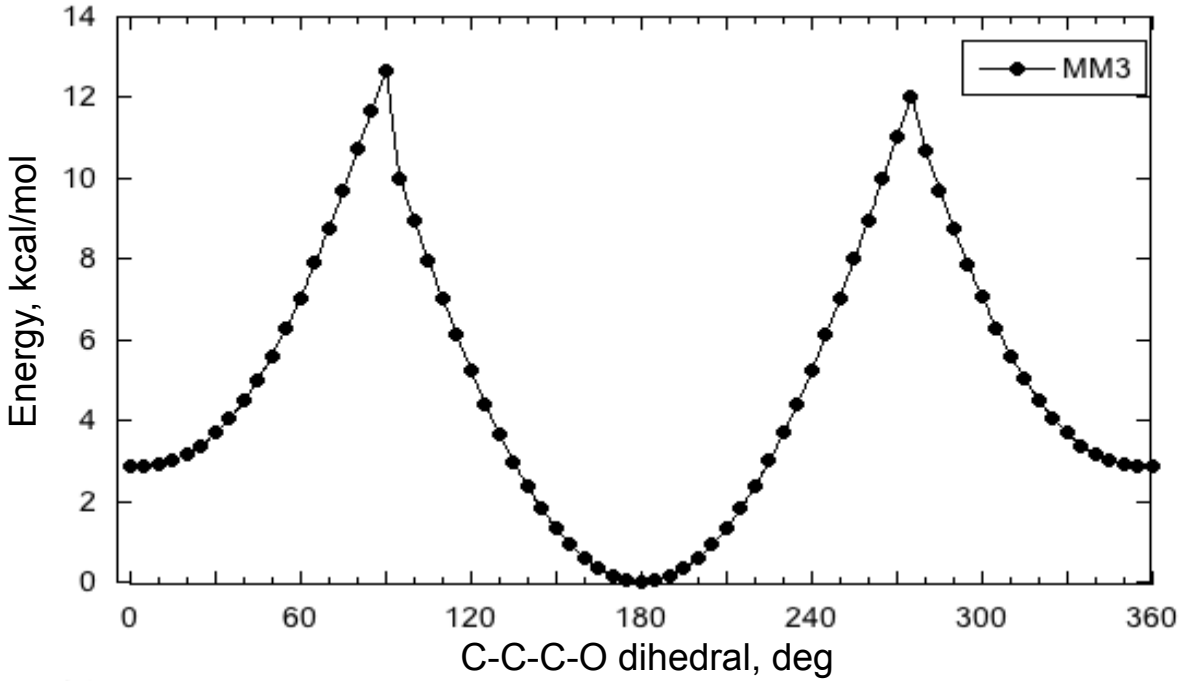




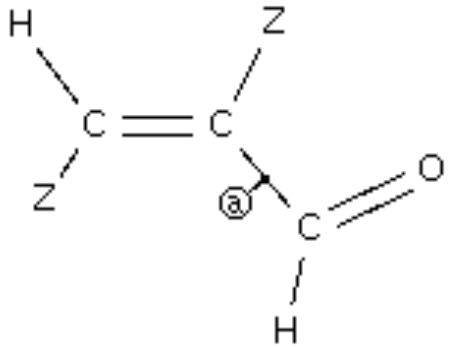


TYPE 2-4:10-1

Φ	E
5.0	2.88
180.0	0.00

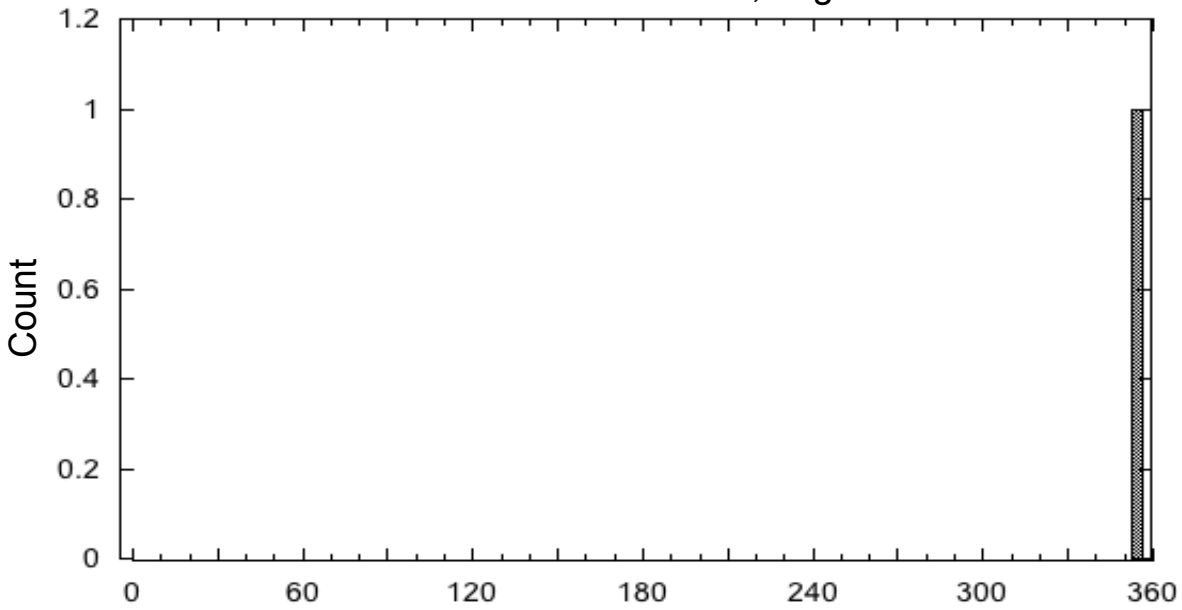


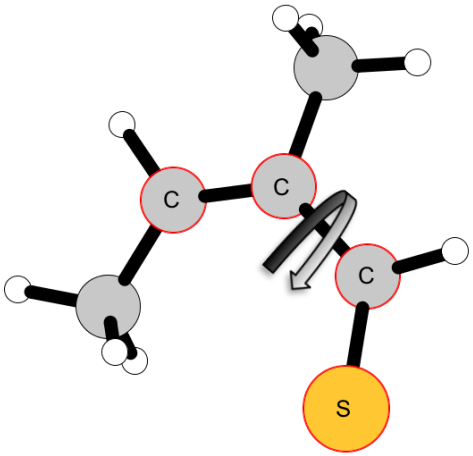
CSD comparison



1 hits

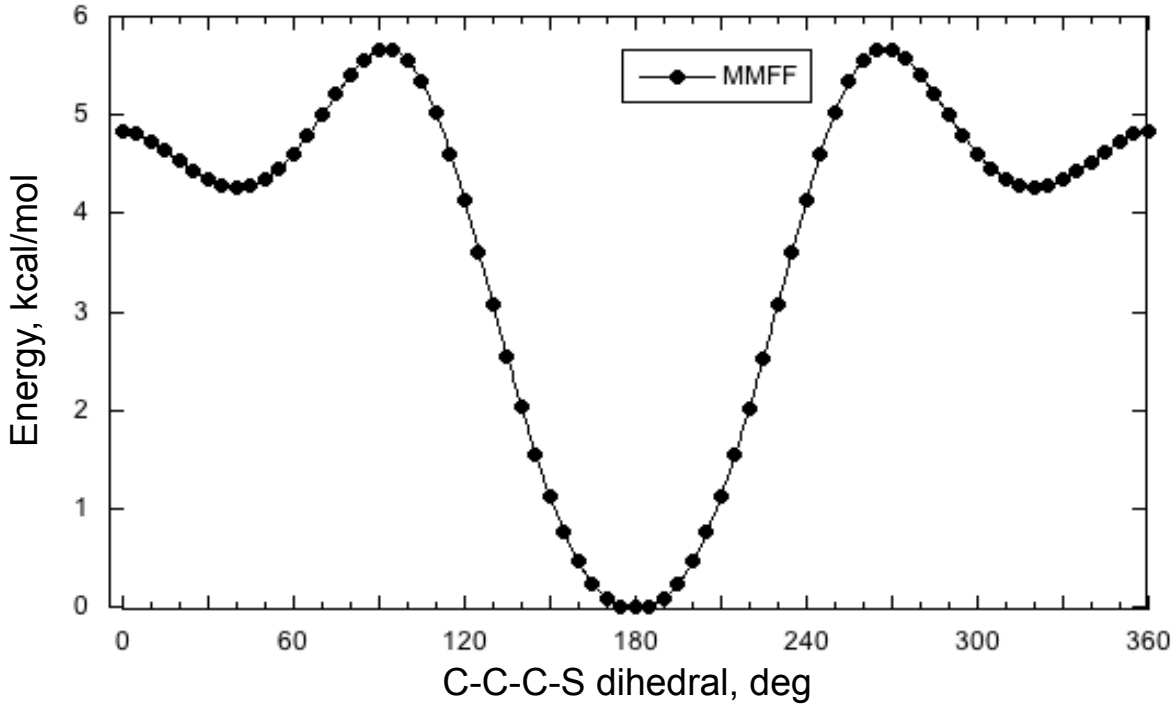
C–C = 1.49 Å



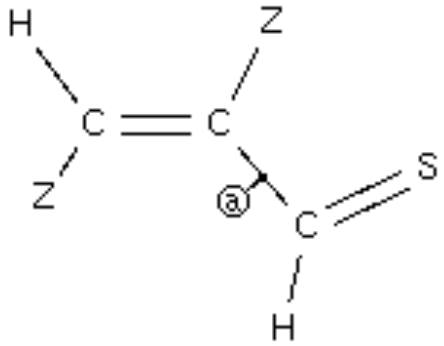


Φ	E
40.0	4.27
180.0	0.00
320.0	4.27

TYPE 2-4:10-2



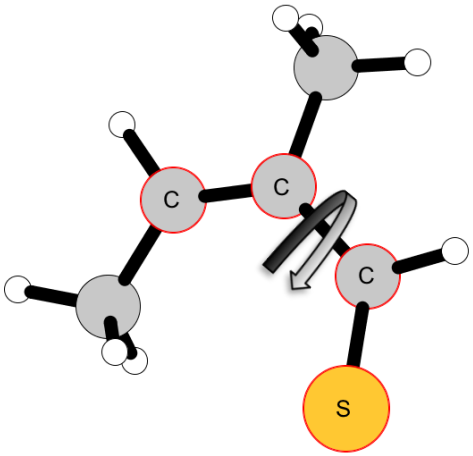
CSD comparison



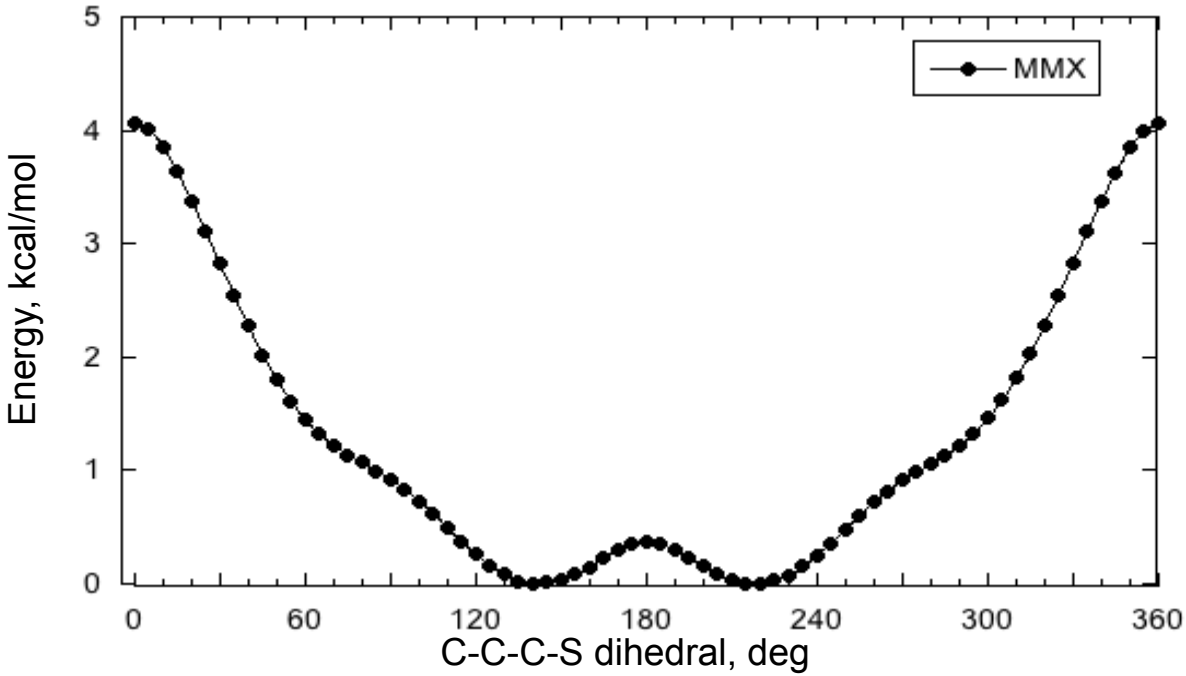
0 hits

NO CSD HITS

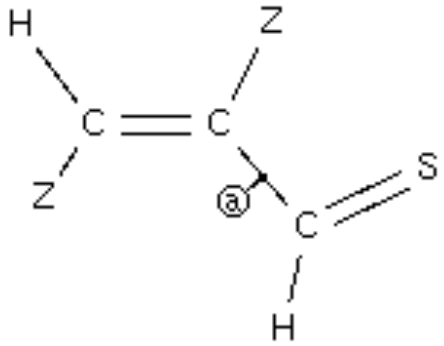
TYPE 2-4:10-2



Φ	E
140.0	0.00
220.0	0.00

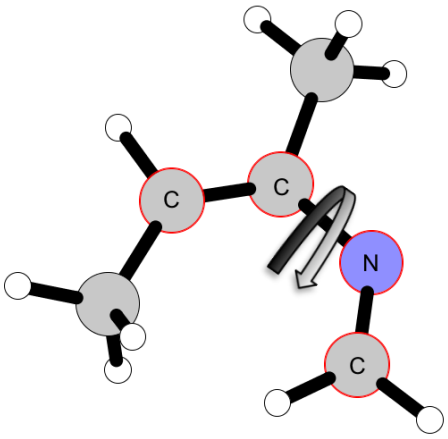


CSD comparison



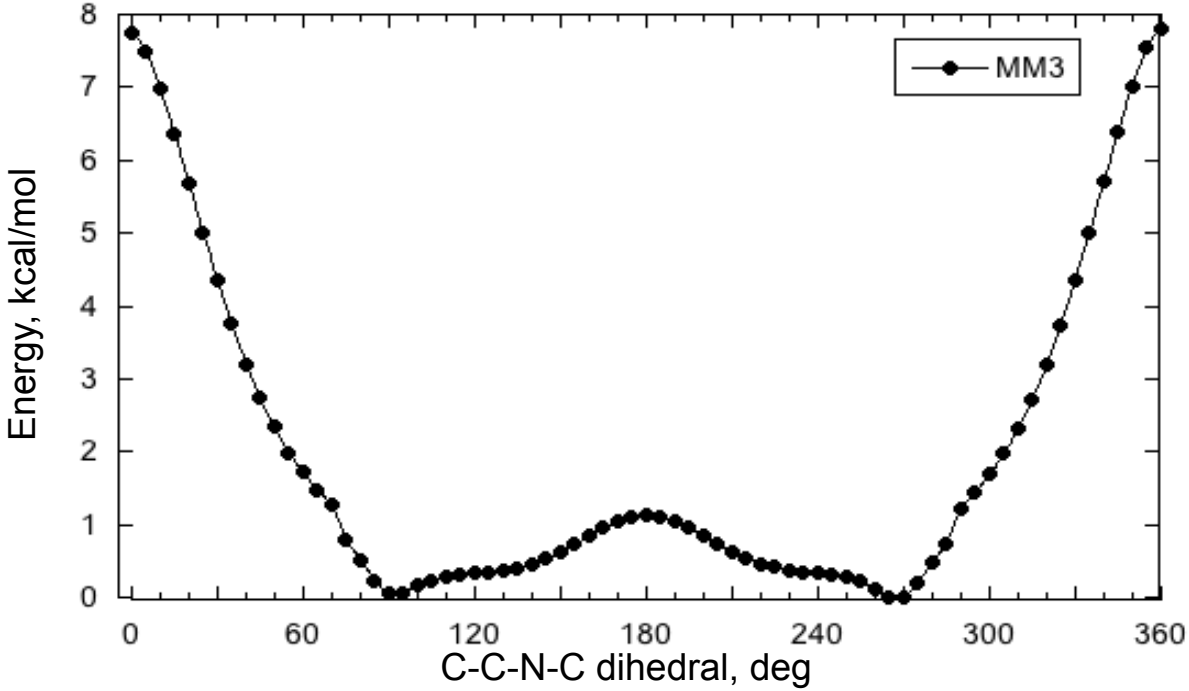
0 hits

NO CSD HITS

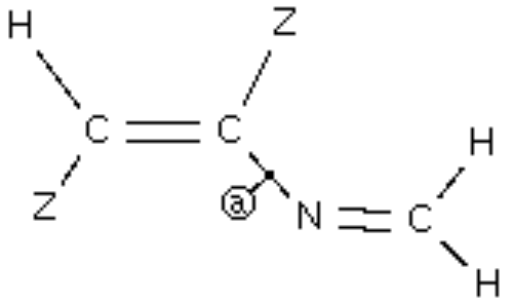


Φ	E
90.0	0.05
265.0	0.00

TYPE 2-4:10-3

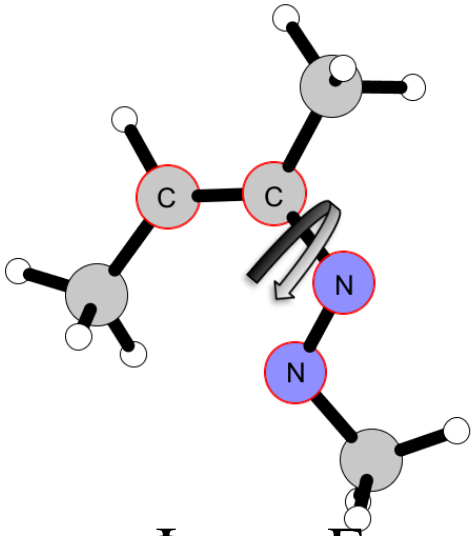


CSD comparison



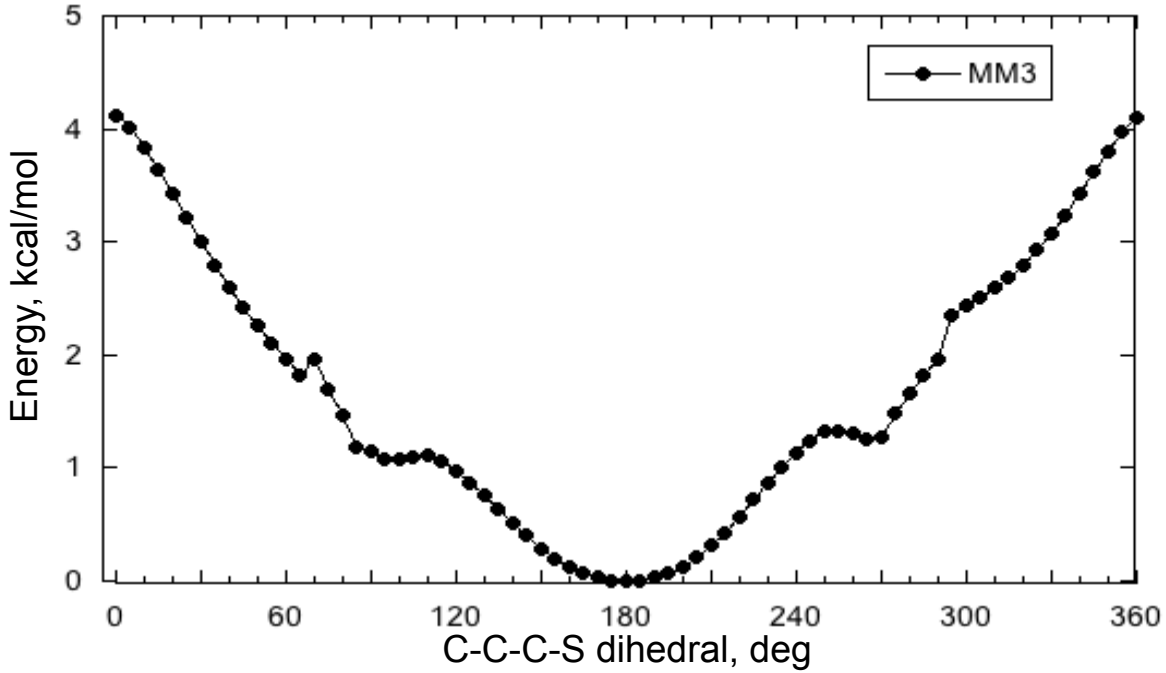
0 hits

NO CSD HITS

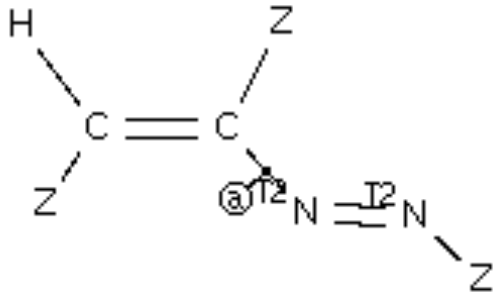


TYPE 2-4:10-4

Φ	E
100.0	1.07
180.0	0.00
265.0	1.26



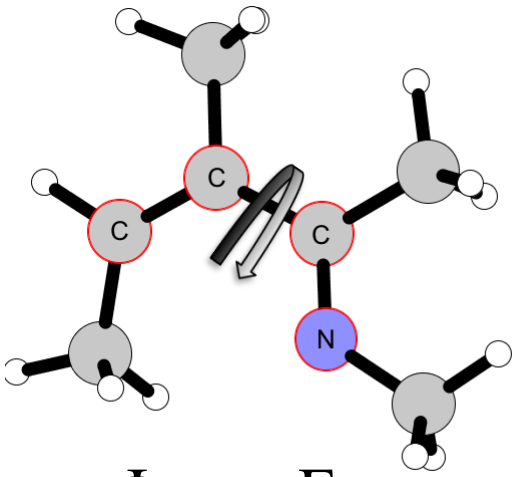
CSD comparison



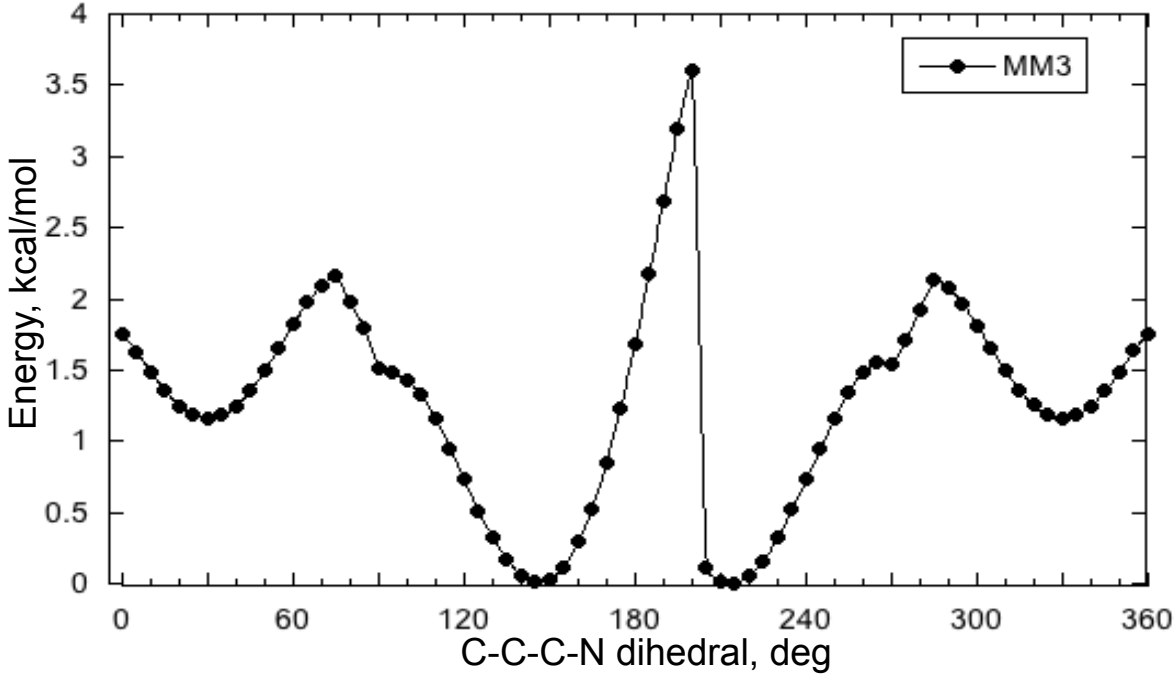
0 hits

NO CSD HITS

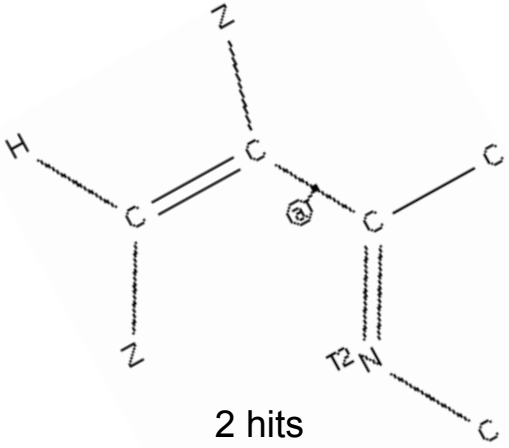
TYPE 2-4:11-1



Φ	E
30.0	1.16
140.0	0.06
215.0	0.00
330.0	1.16

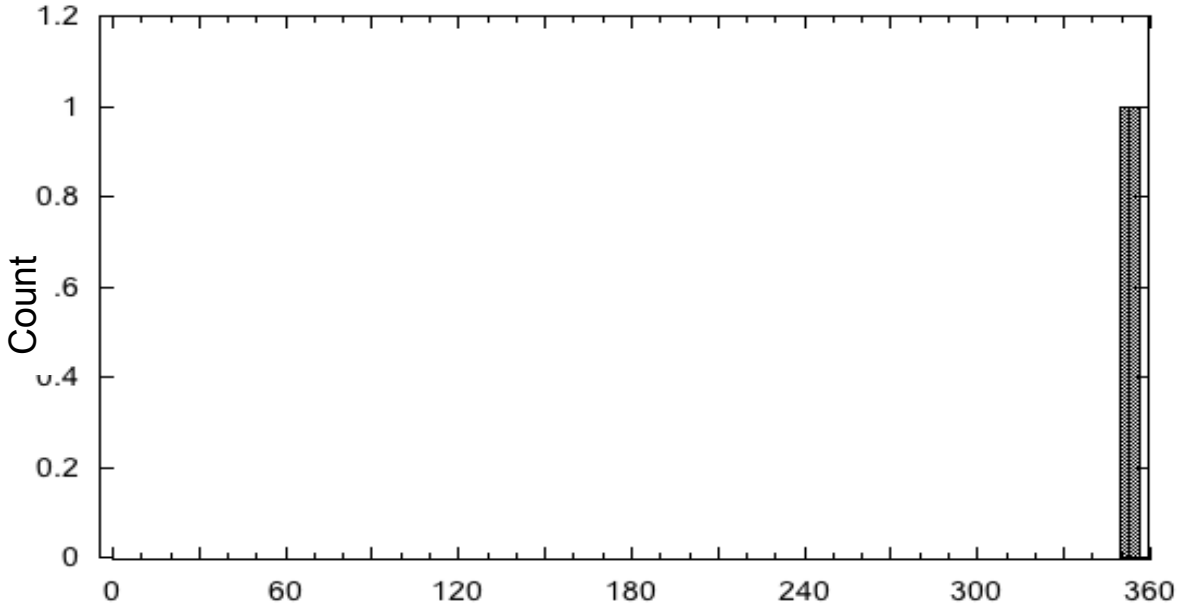


CSD comparison

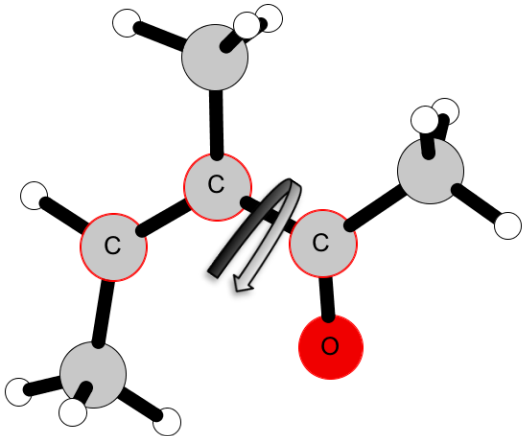


2 hits

C–C = 1.46 ± 0.00 Å

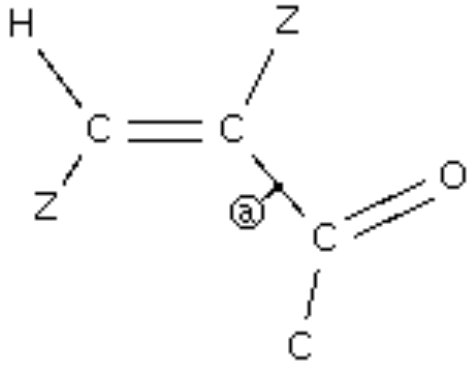


TYPE 2-4:11-2



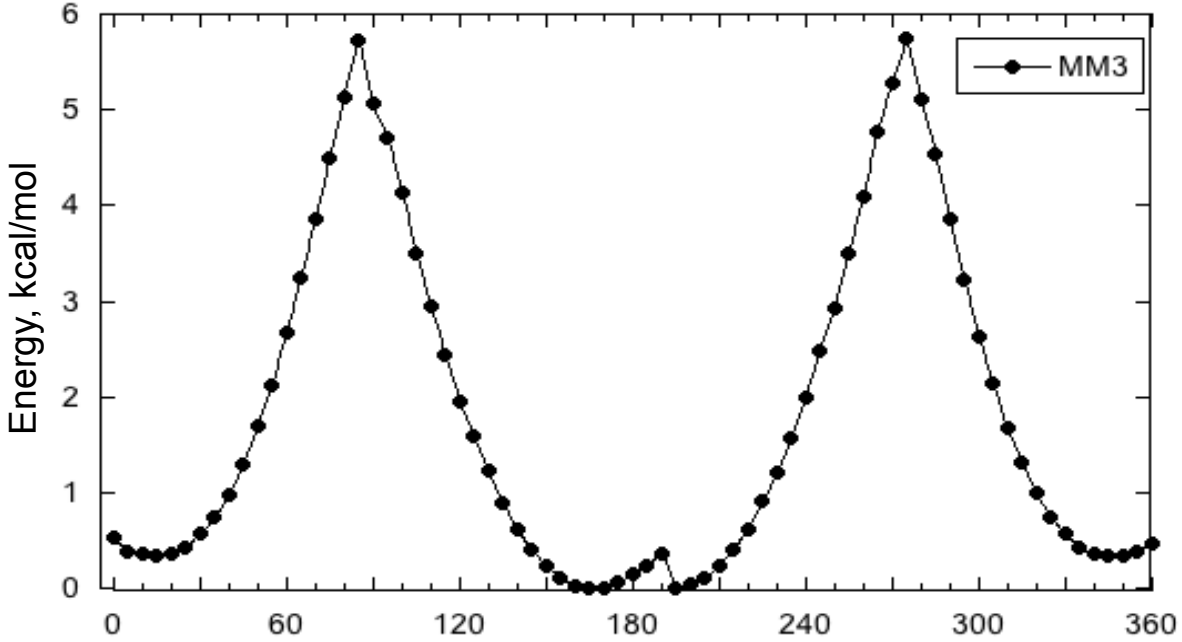
Φ	E
15.0	0.35
165.0	0.00
195.0	0.00
345.0	0.34

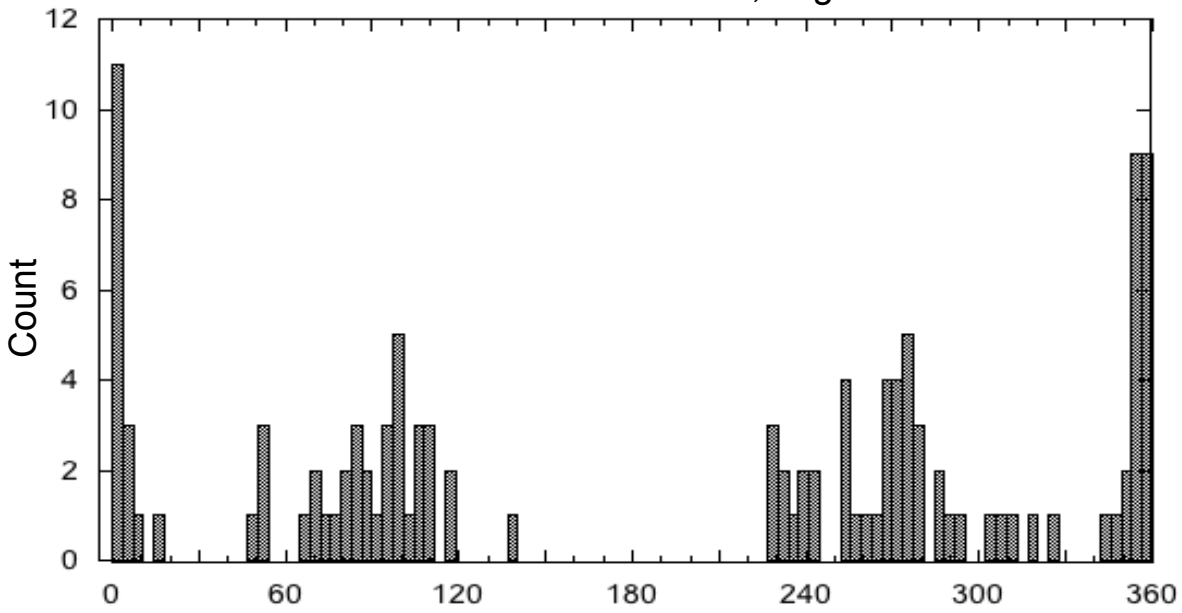
CSD comparison



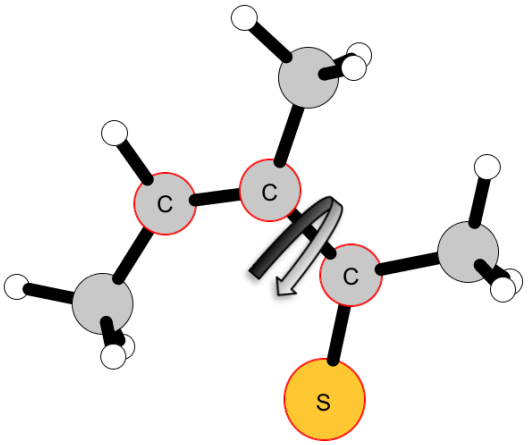
98 hits

C–C = 1.49 ± 0.03 Å

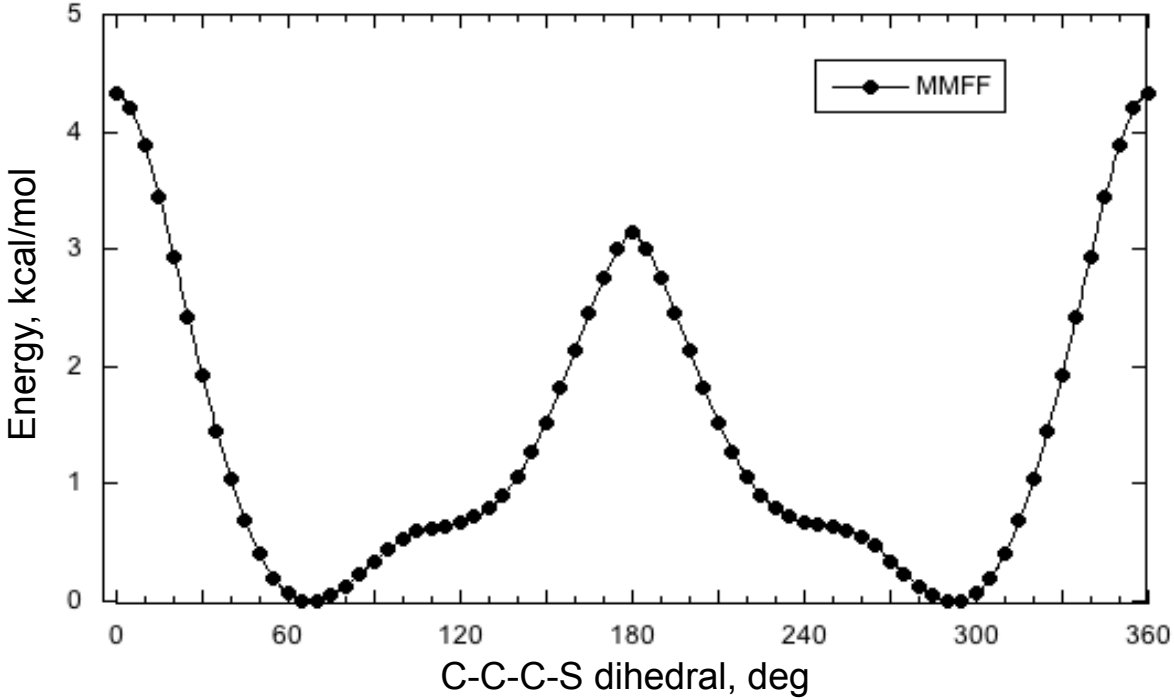




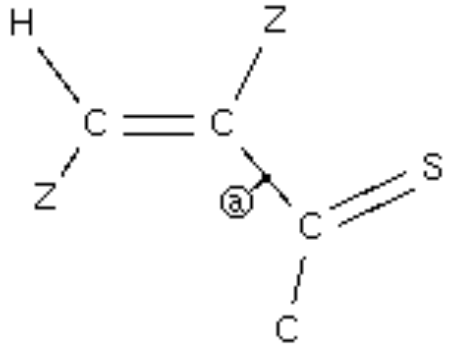
TYPE 2-4:11-3



Φ	E
70.0	0.00
290.0	0.00



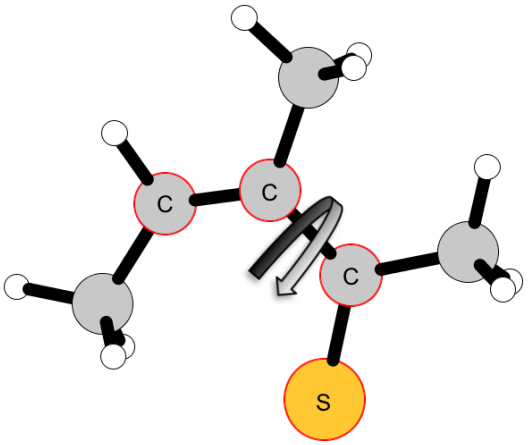
CSD comparison



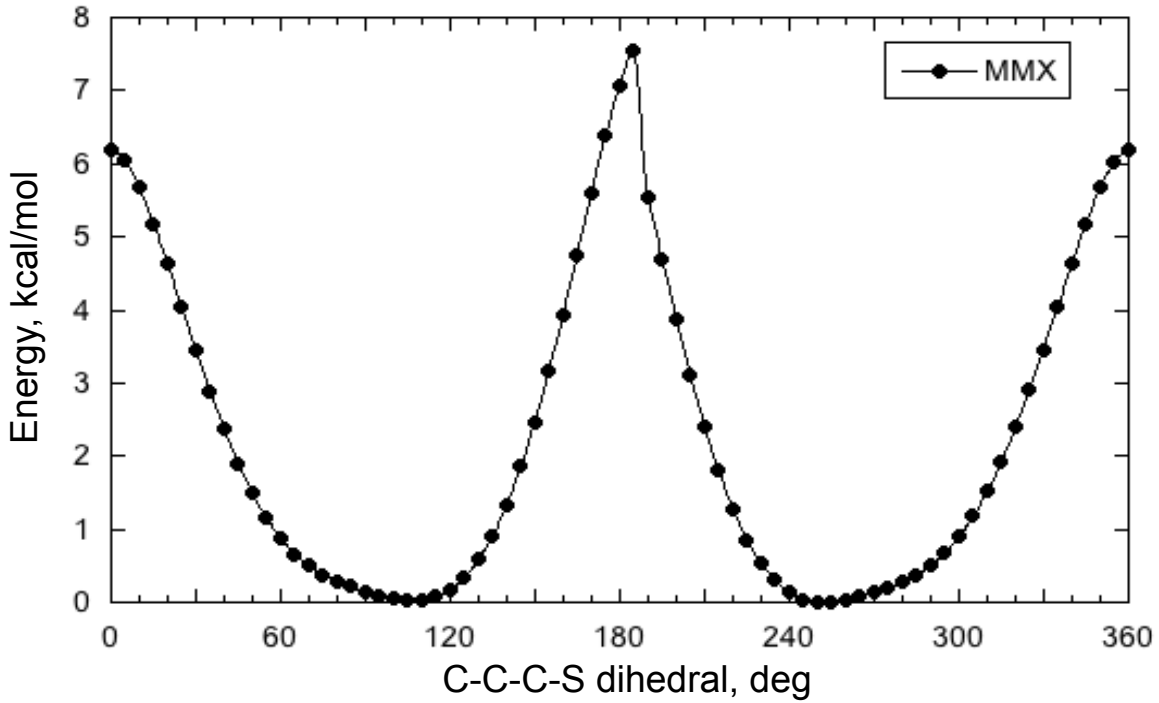
0 hits

NO CSD HITS

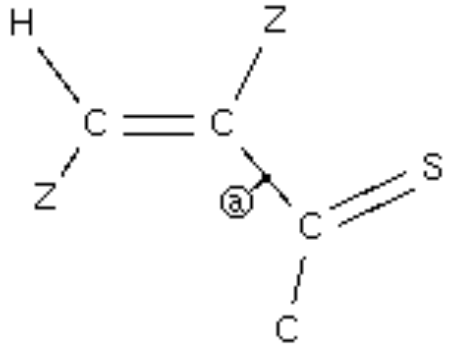
TYPE 2-4:11-3



Φ	E
105.0	0.03
255.0	0.00

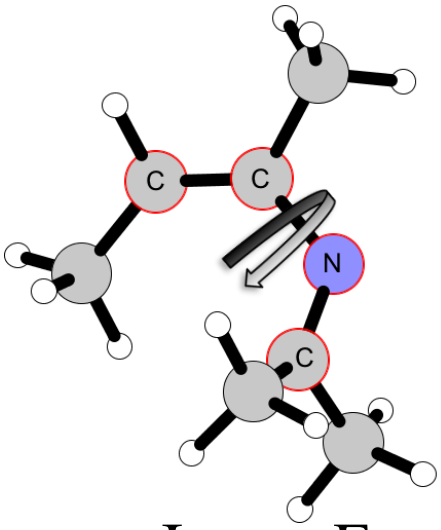


CSD comparison



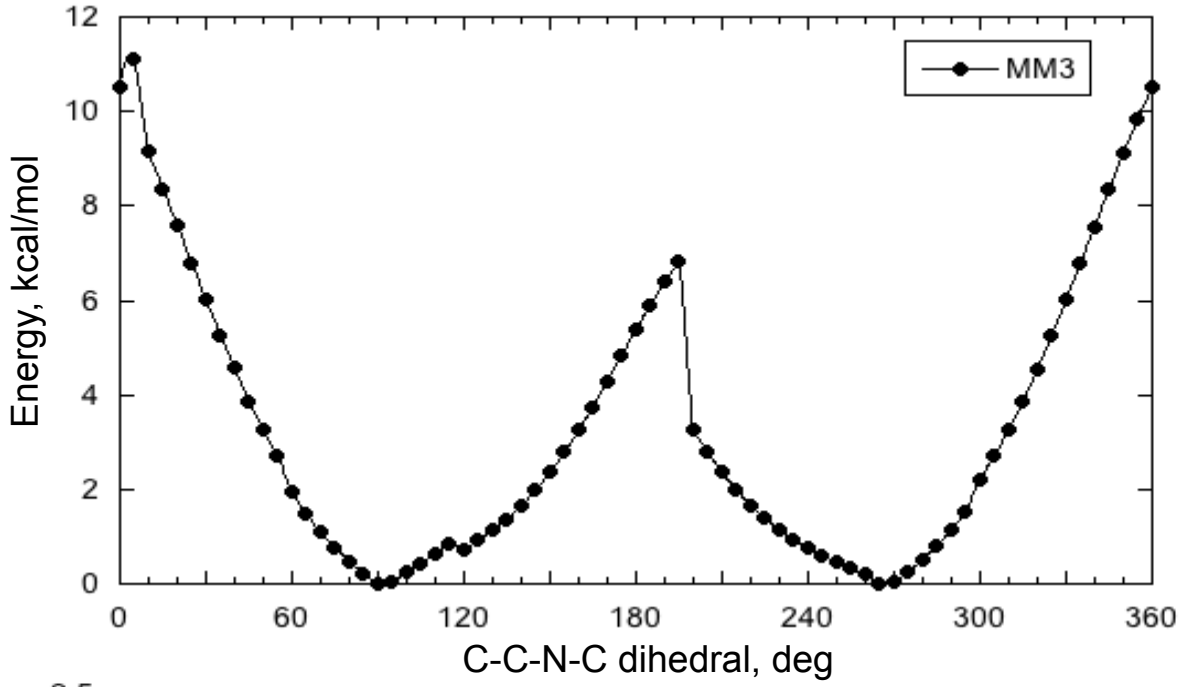
0 hits

NO CSD HITS

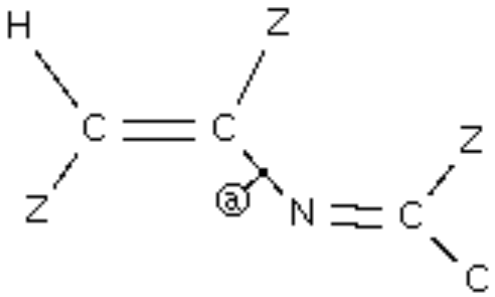


TYPE 2-4:12-1

Φ	E
90.0	0.00
270.0	0.04

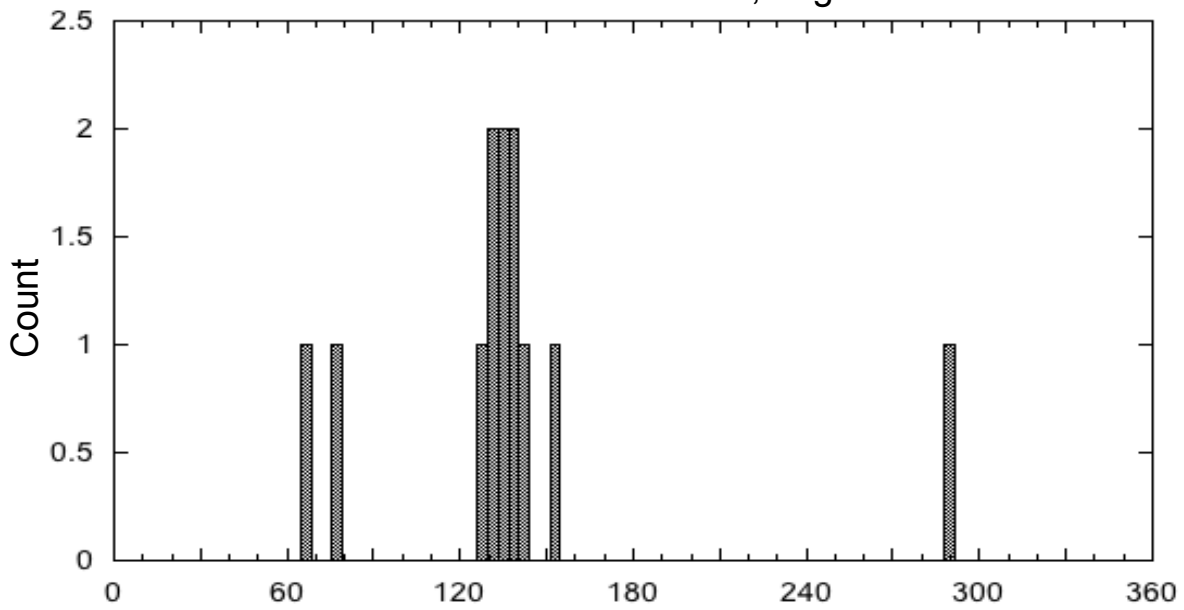


CSD comparison

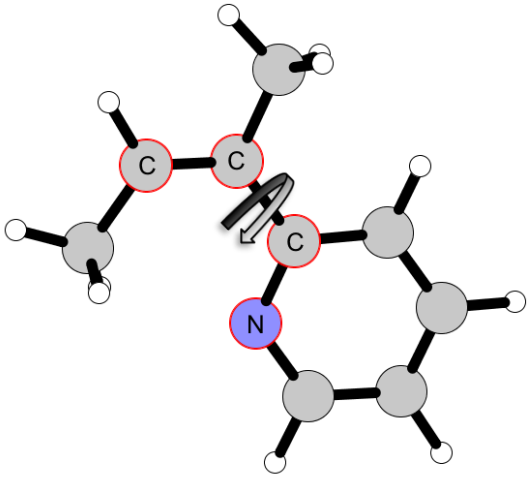


11 hits

C–N = 1.40 ± 0.01 Å

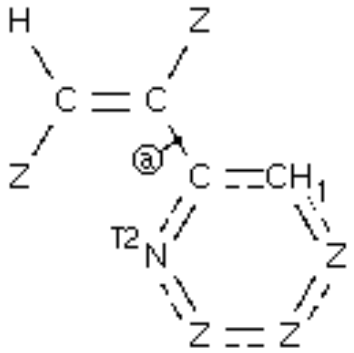


TYPE 2-4:13-1

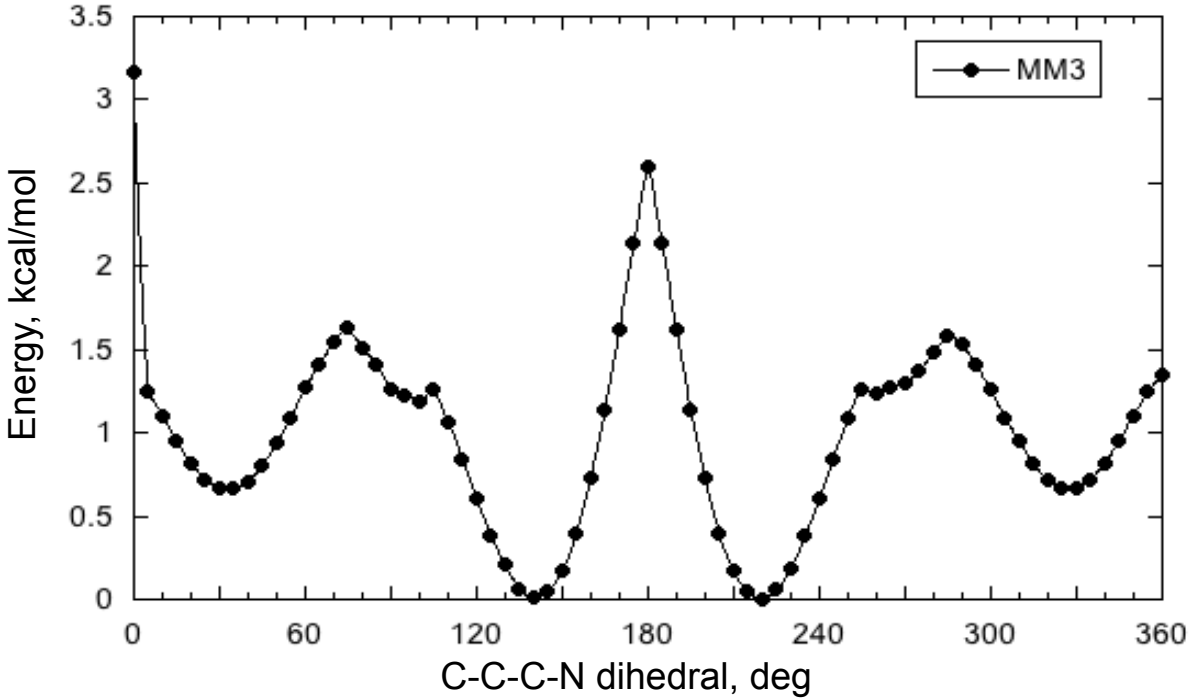


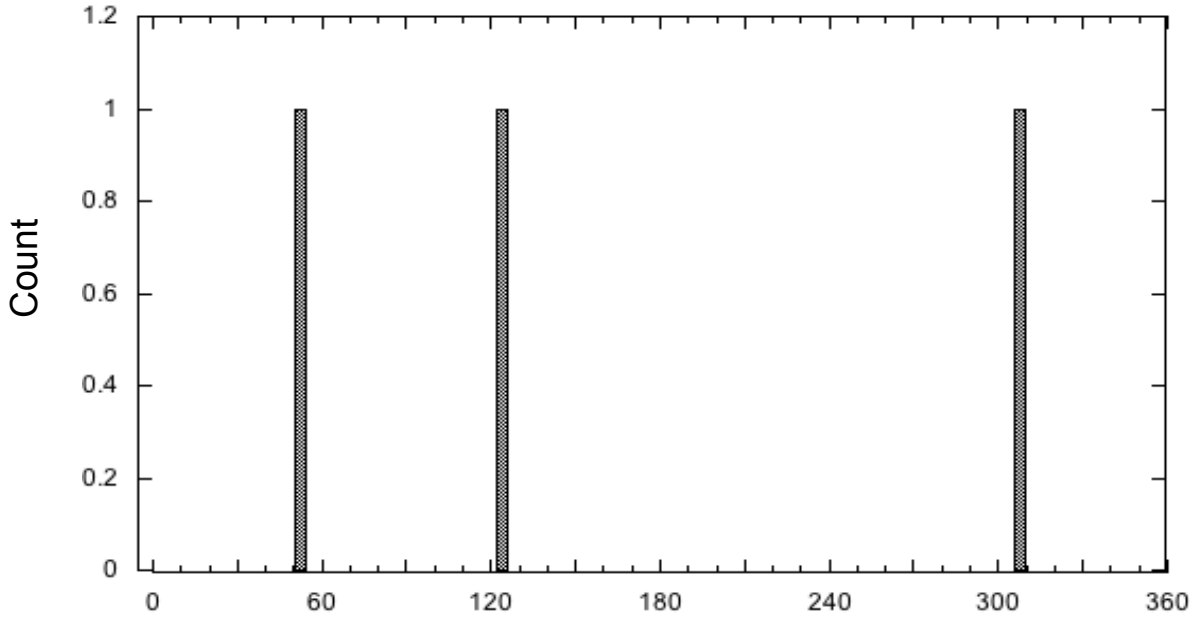
Φ	E
35.0	0.67
100.0	1.19
140.0	0.01
220.0	0.00
260.0	1.24
325.0	0.67

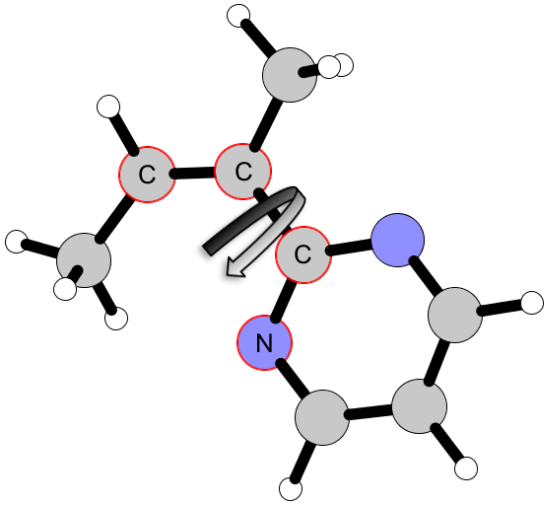
CSD comparison



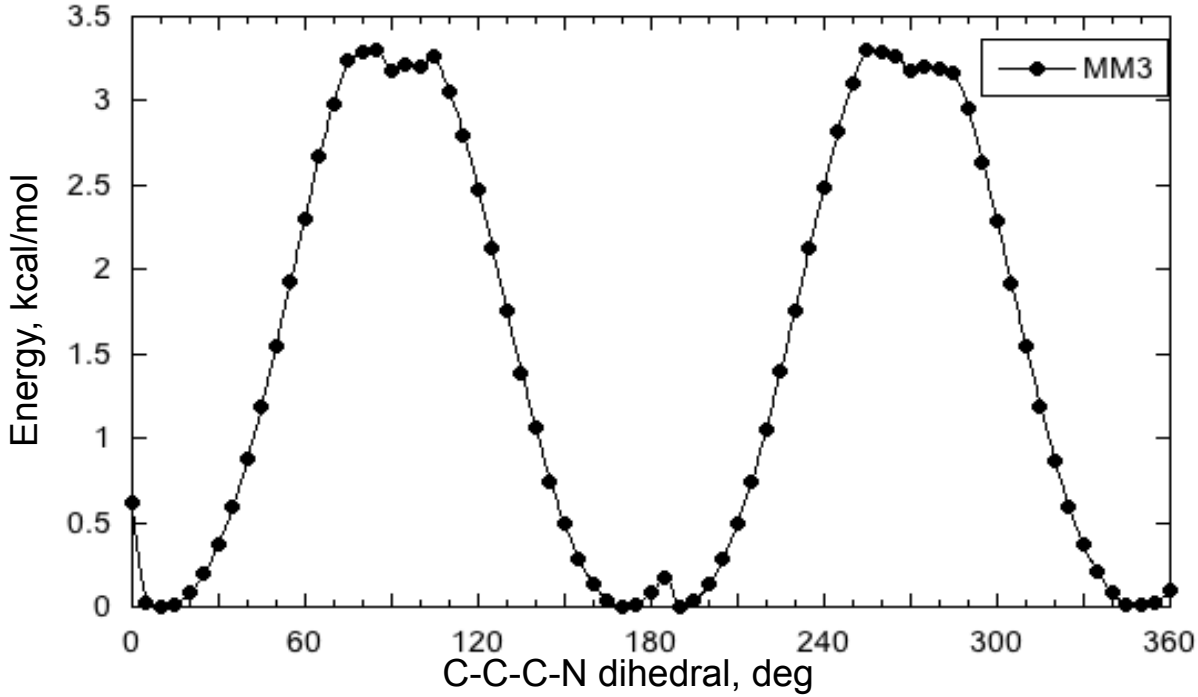
3 hits
C-C= 1.49 ± 0.01 Å





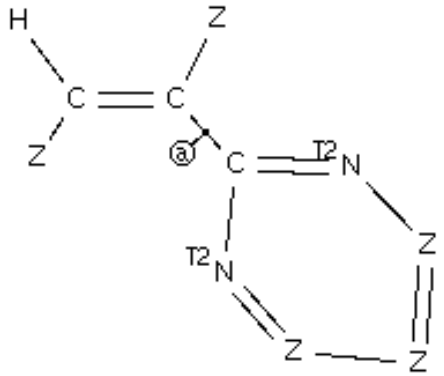


TYPE 2-4:13-2

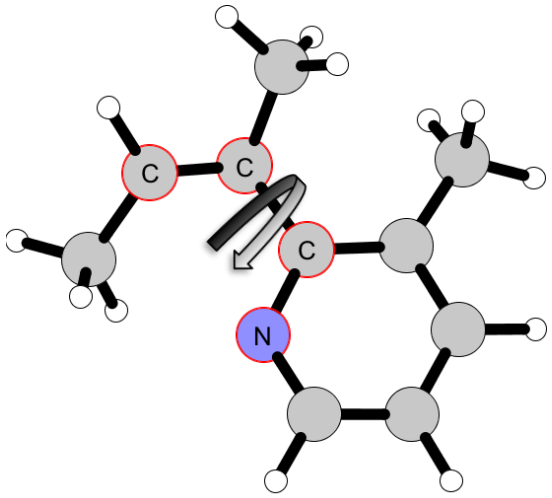


Φ	E
10.0	0.00
90.0	3.18
170.0	0.00
190.0	0.00
270.0	3.18
350.0	0.01

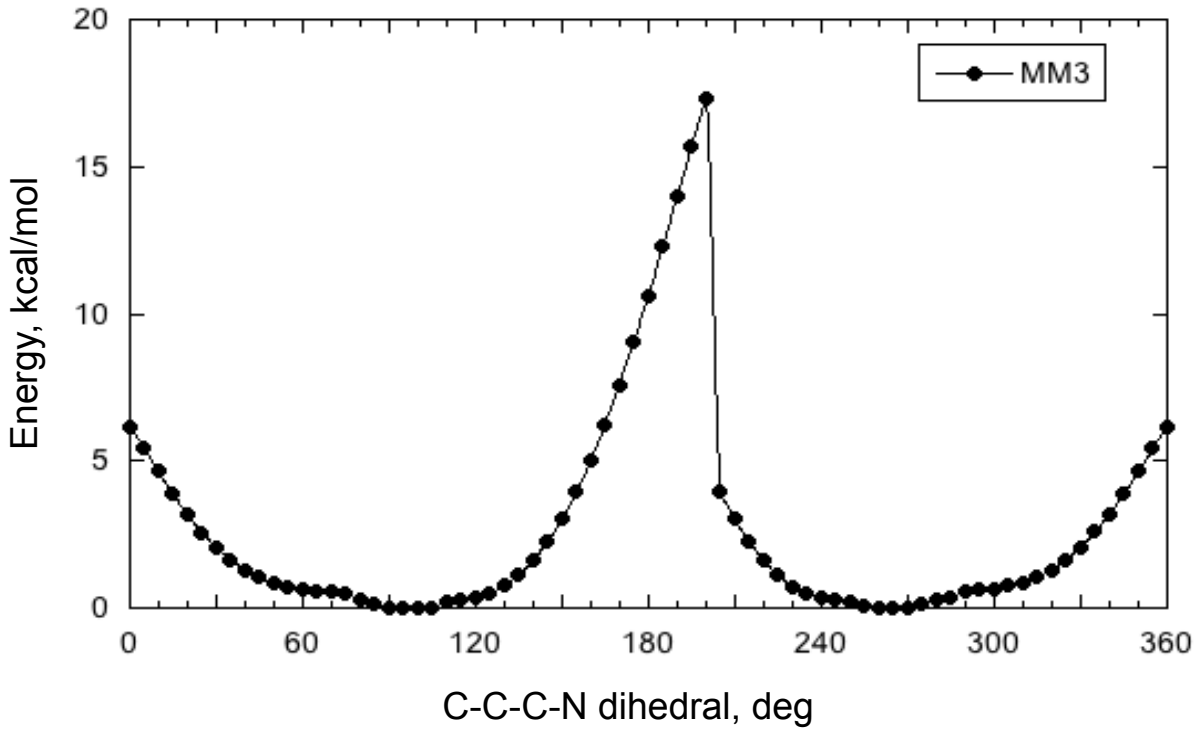
CSD comparison



NO CSD HITS

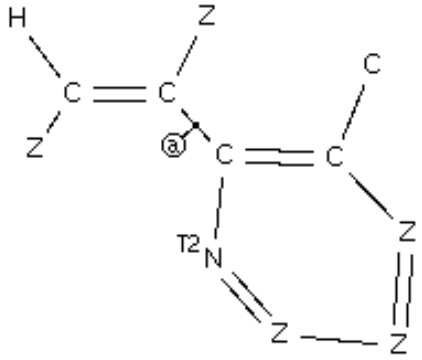


TYPE 2-4:13-3



Φ	E
95.0	0.00
265.0	0.01

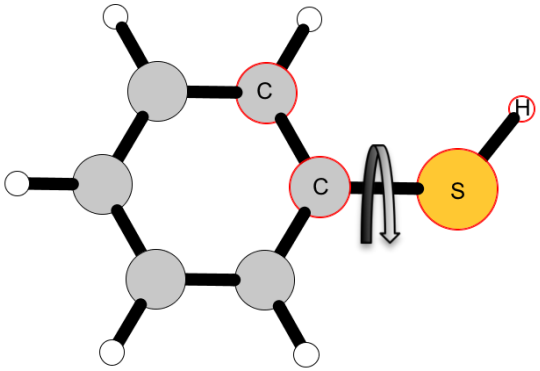
CSD comparison



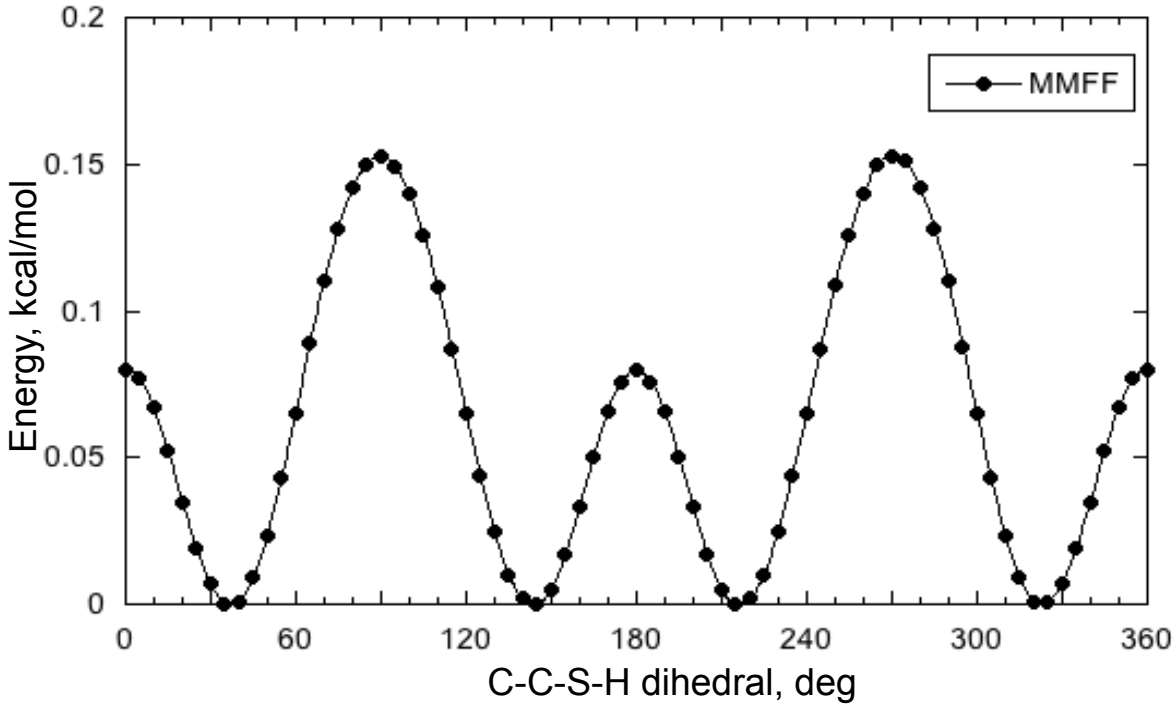
0 hits

NO CSD HITS

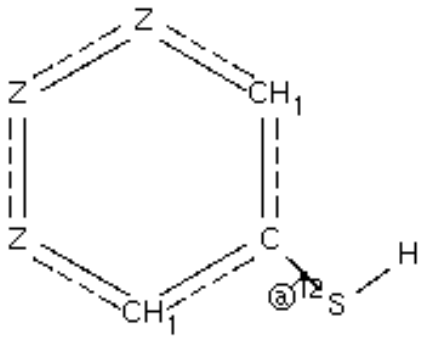
TYPE 3-1:5-4



Φ	E
35.0	0.00
145.0	0.00
215.0	0.00
325.0	0.00

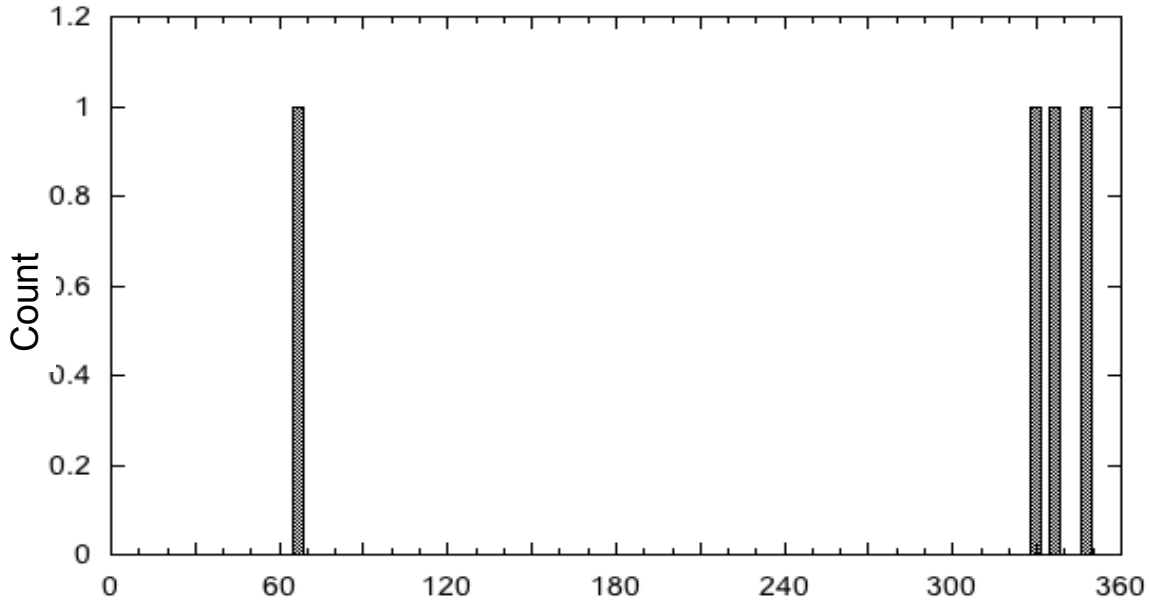


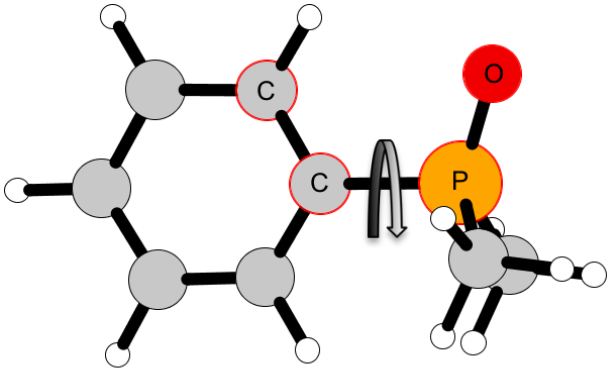
CSD comparison



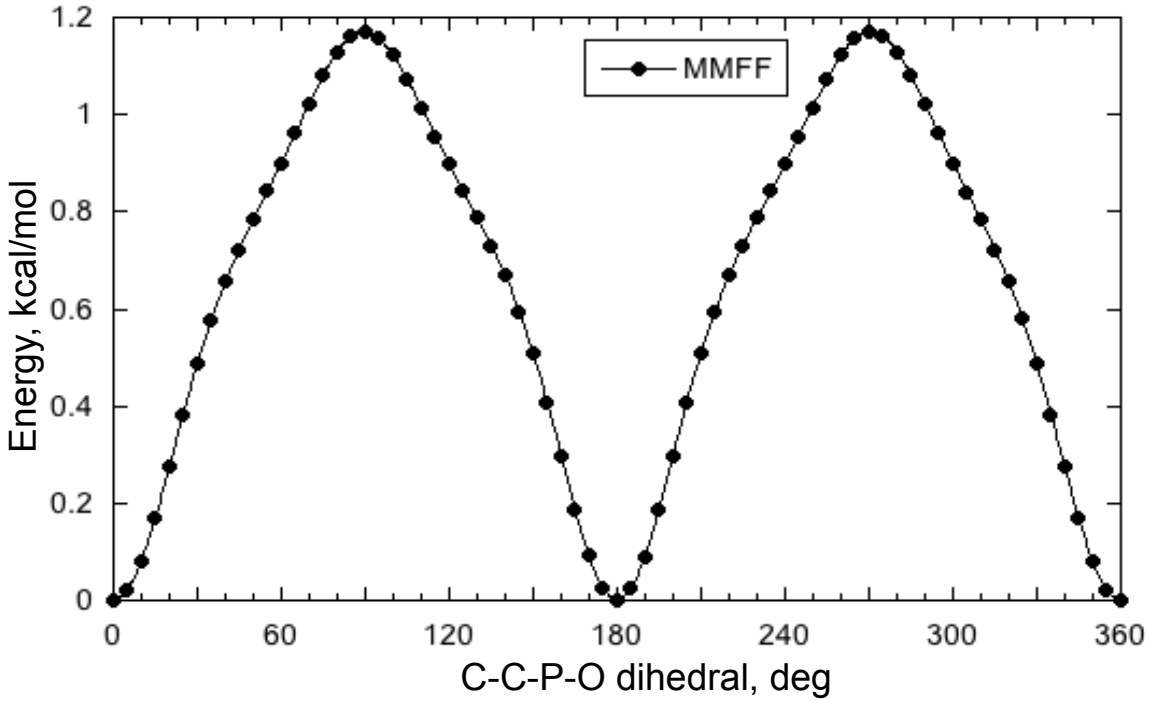
4 hits

C–S = 1.75 ± 0.02 Å



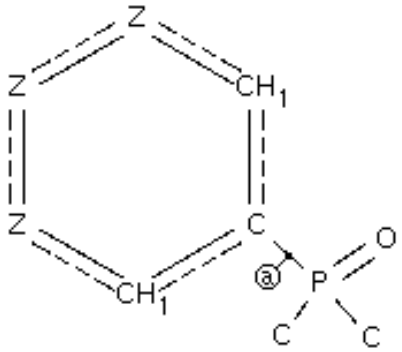


TYPE 3-1:8-1



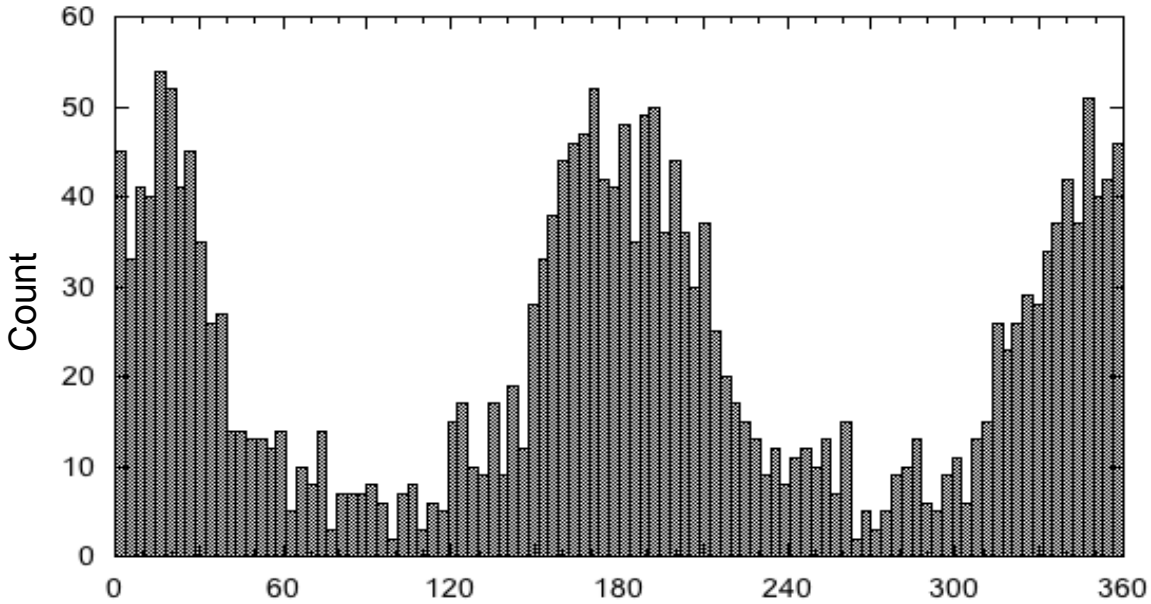
Φ	E
0.0	0.00
180.0	0.00

CSD comparison

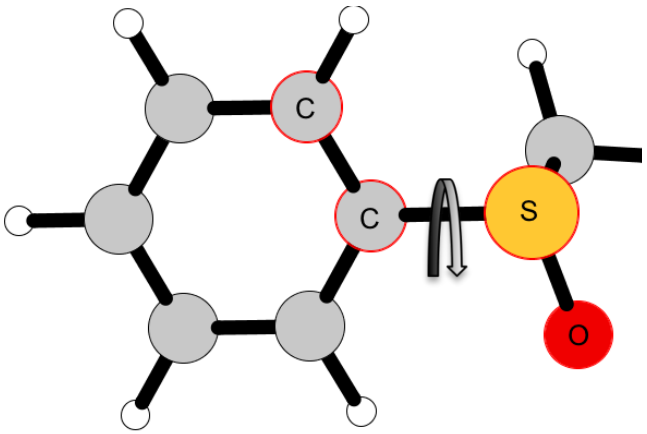


847 hits

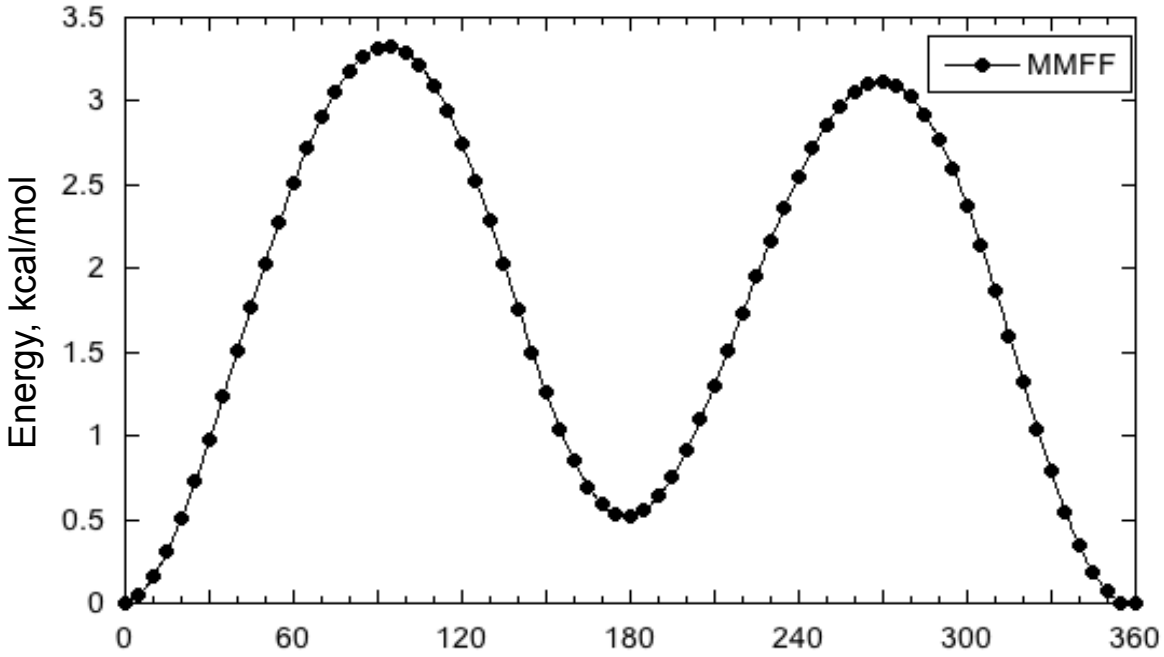
C–P = 1.46 ± 0.02 Å



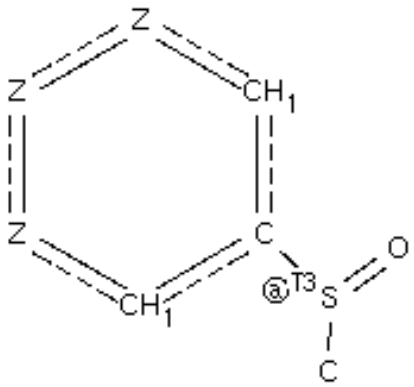
TYPE 3-1:8-2



Φ	E
0.0	0.00
180.0	0.52

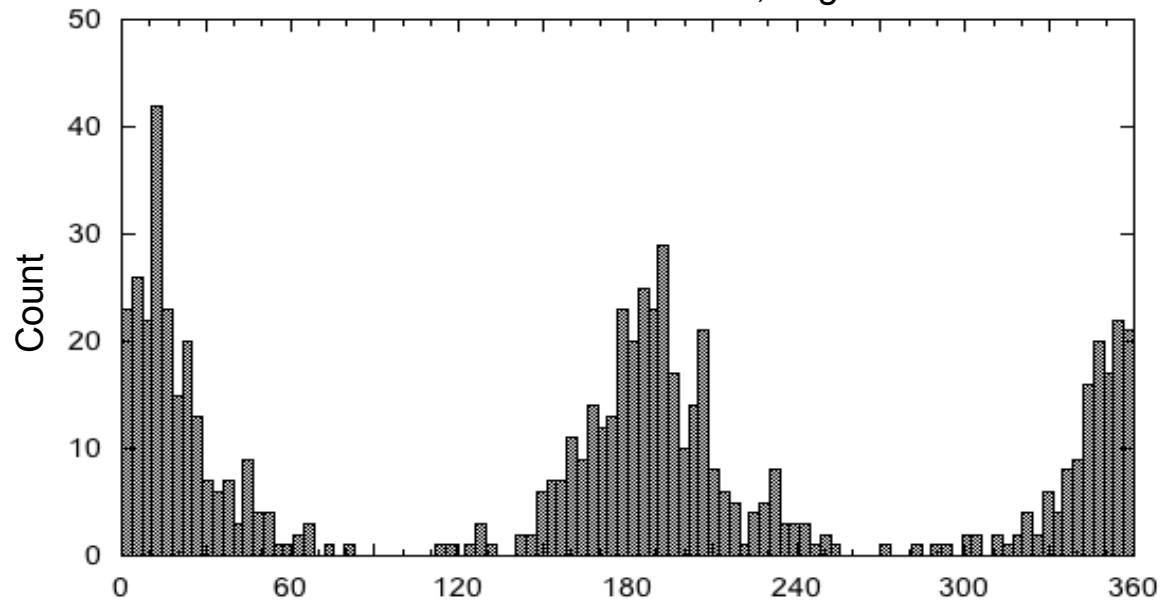


CSD comparison

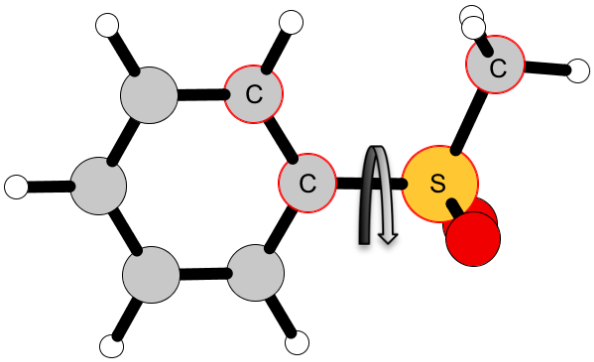


575 hits

C–S = 1.79 ± 0.01 Å

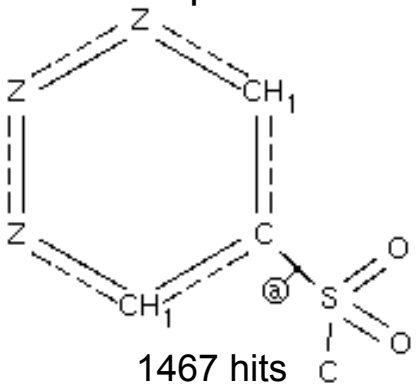


TYPE 3-1:8-4



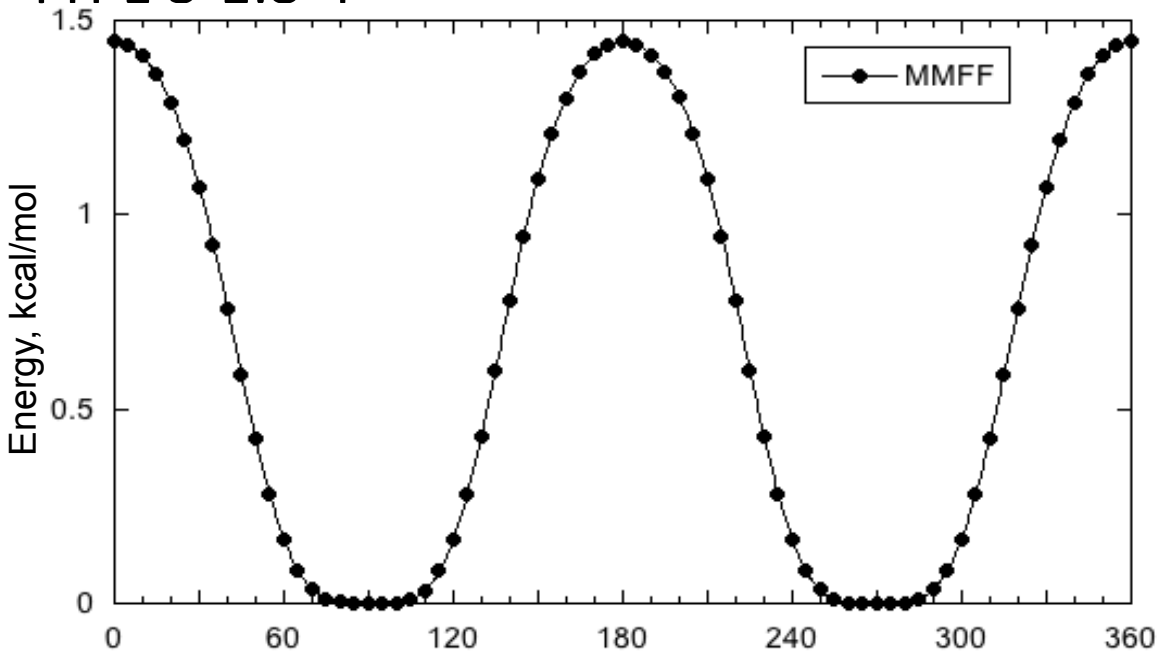
Φ	E
85.0	0.00
90.0	0.00
95.0	0.00
265.0	0.00
270.0	0.00
275.0	0.00

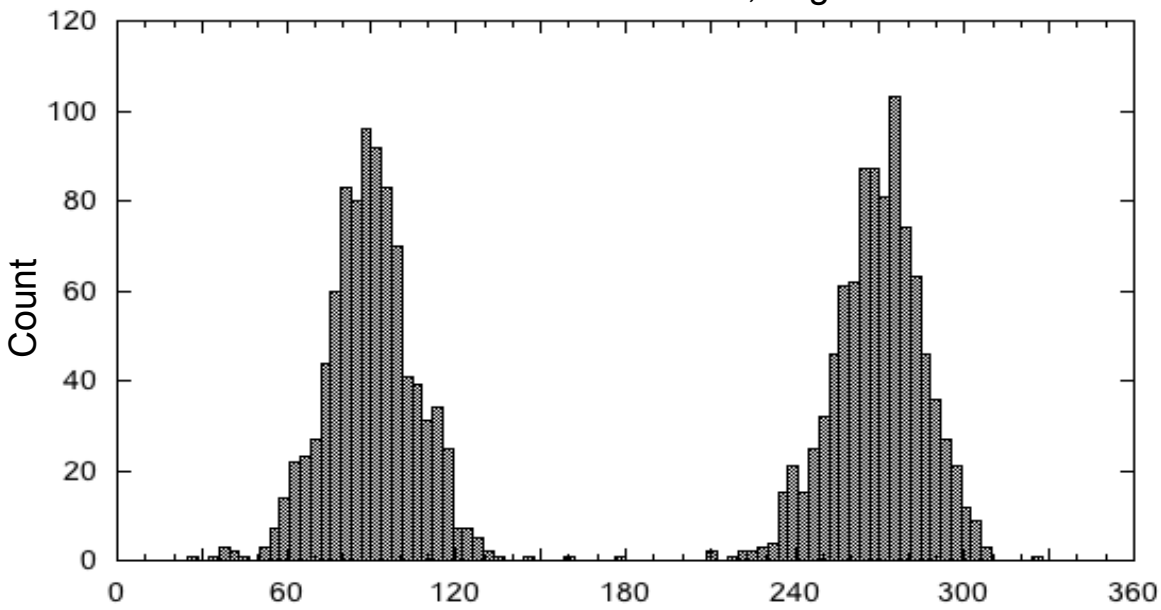
CSD comparison



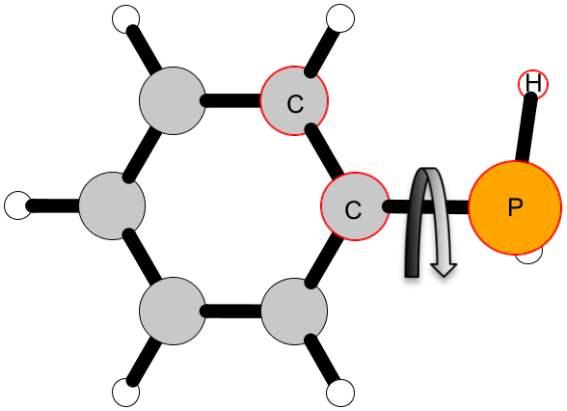
1467 hits

C–S = 1.76 ± 0.01 Å

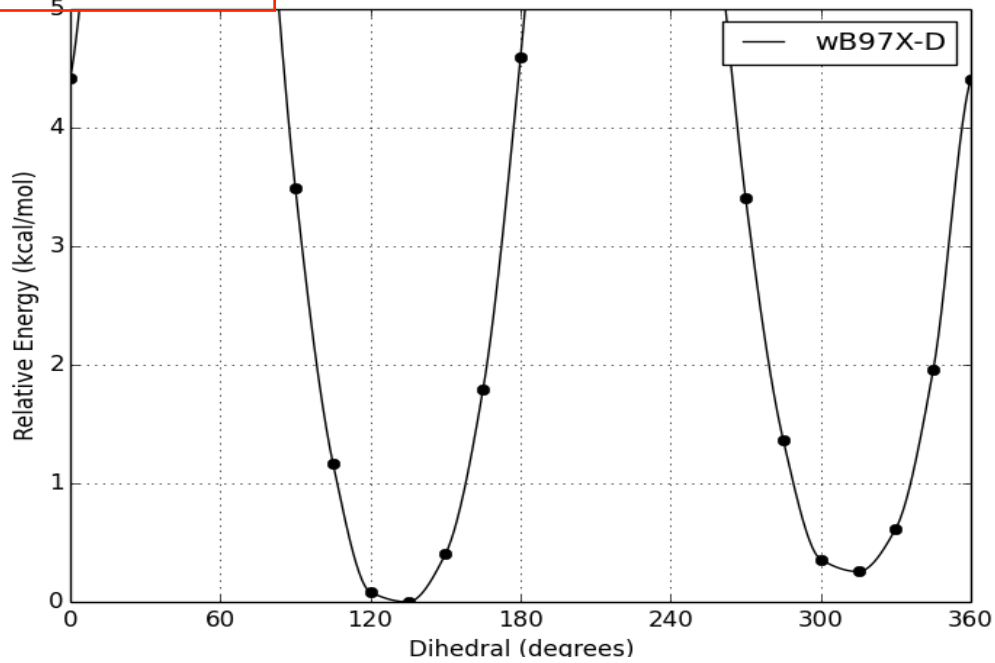




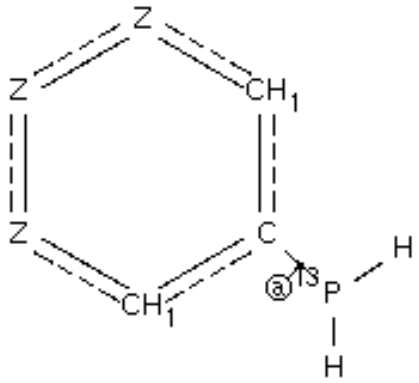
TYPE 3-1:9-1



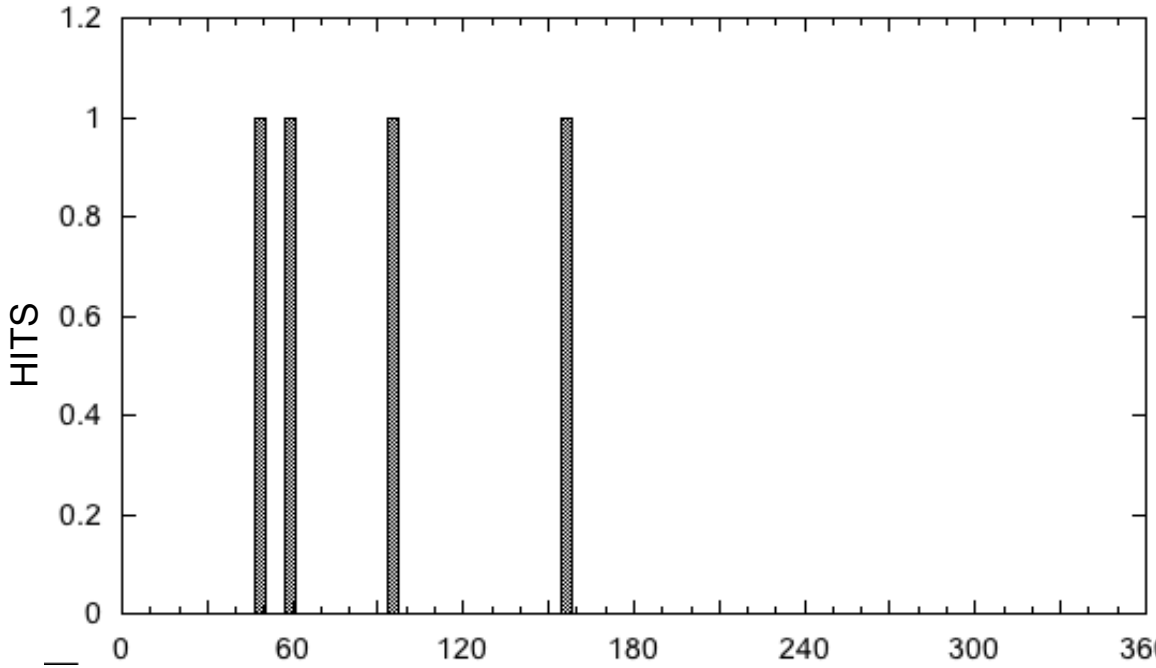
Φ	E
135.0	0.00
315.0	0.26



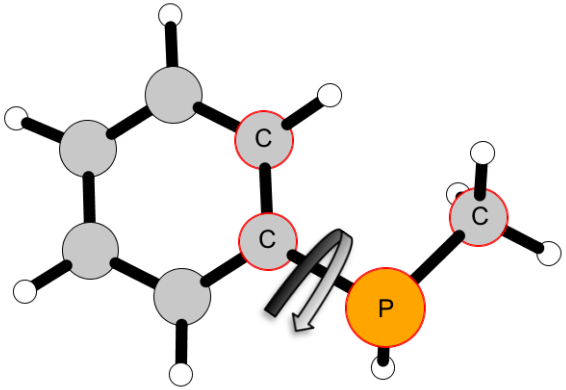
CSD comparison



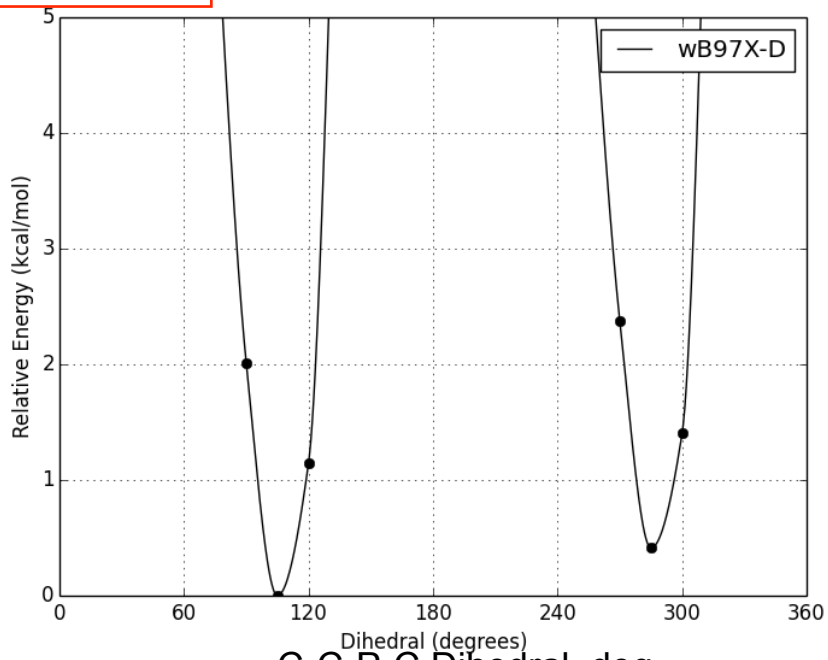
2 hits



TYPE 3-1:9-2

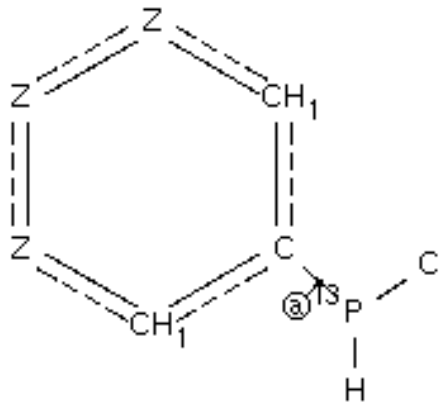


Φ	E
105.0	0.00
285.0	0.41



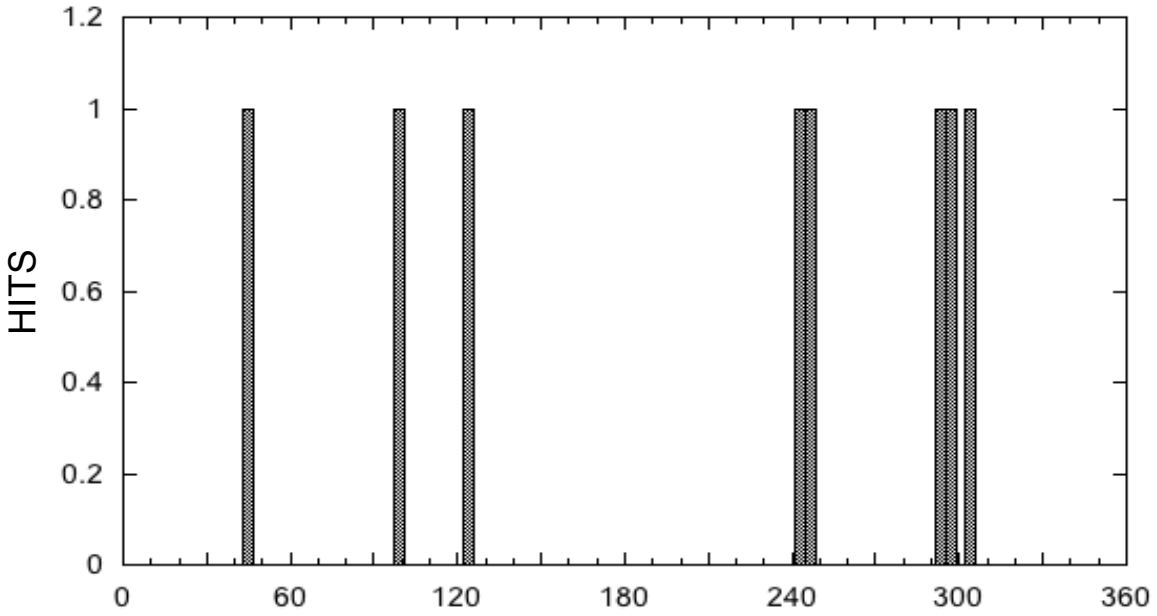
C-C-P-C Dihedral, deg.

CSD comparison

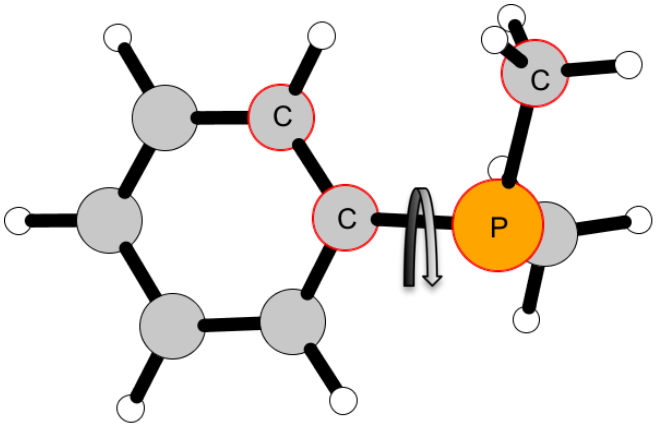


8 hits

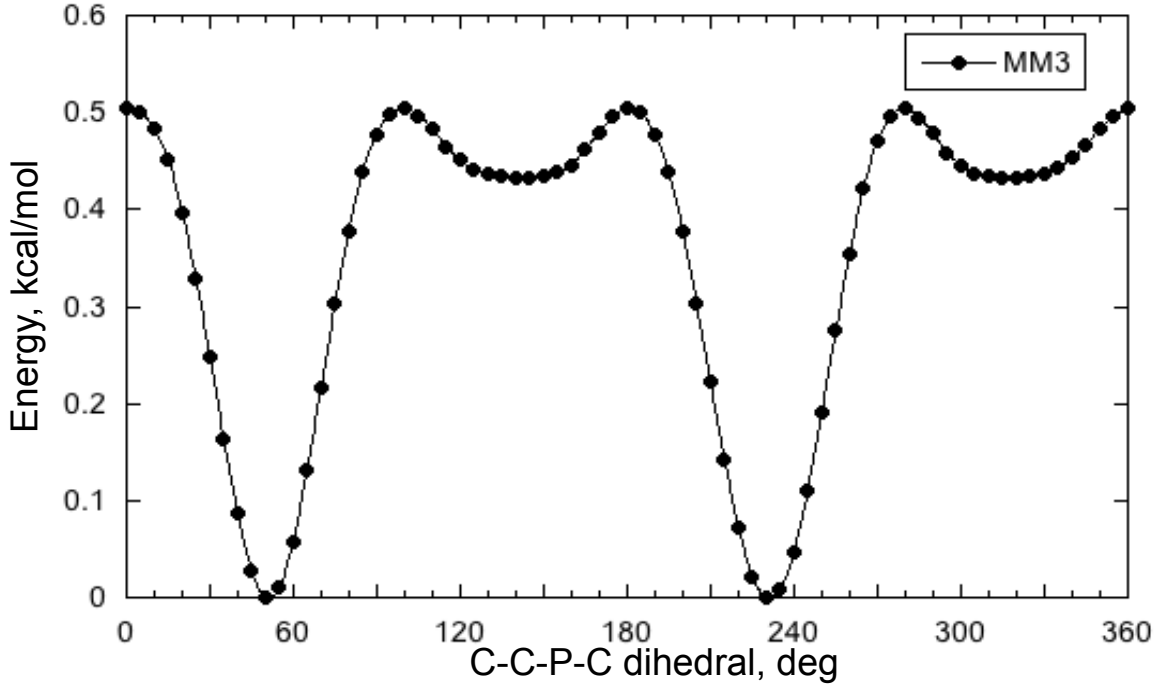
C-P = 1.83 ± 0.01 Å



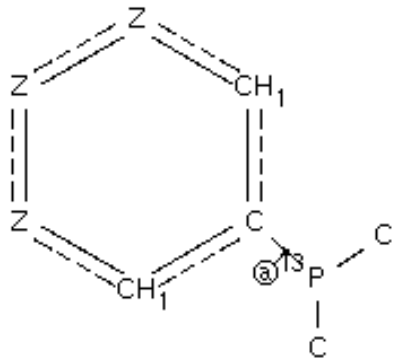
TYPE 3-1:9-4



Φ	E
50.0	0.00
140.0	0.43
230.0	0.00
315.0	0.43

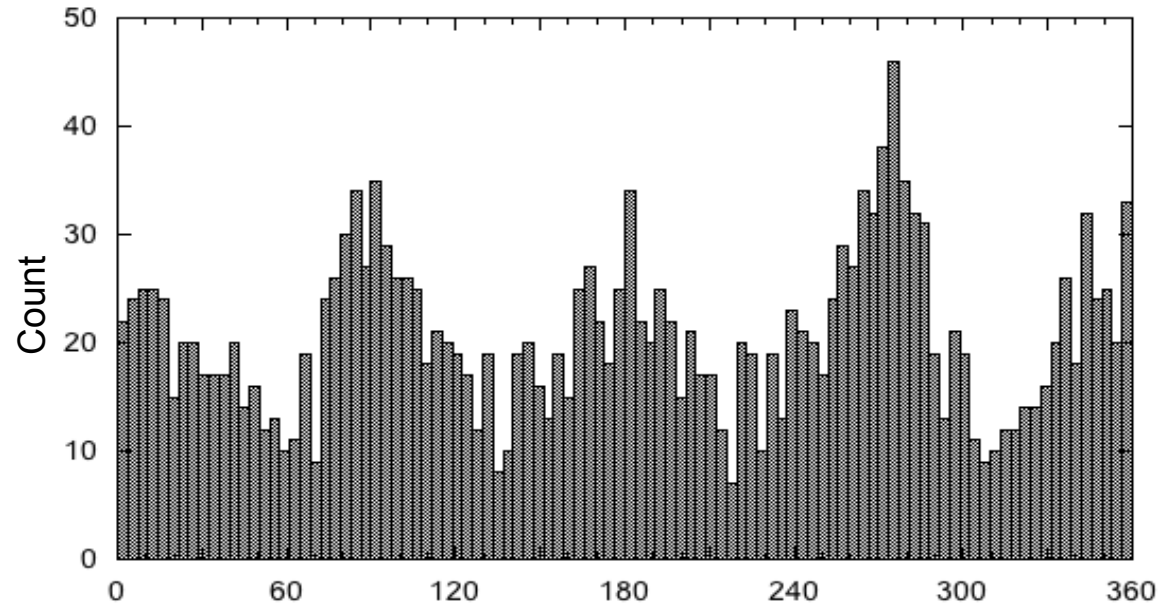


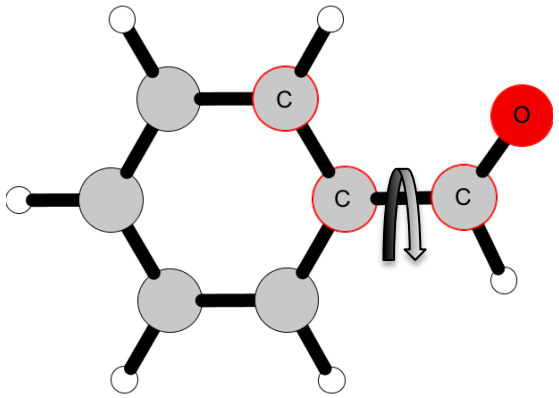
CSD comparison



768 hits

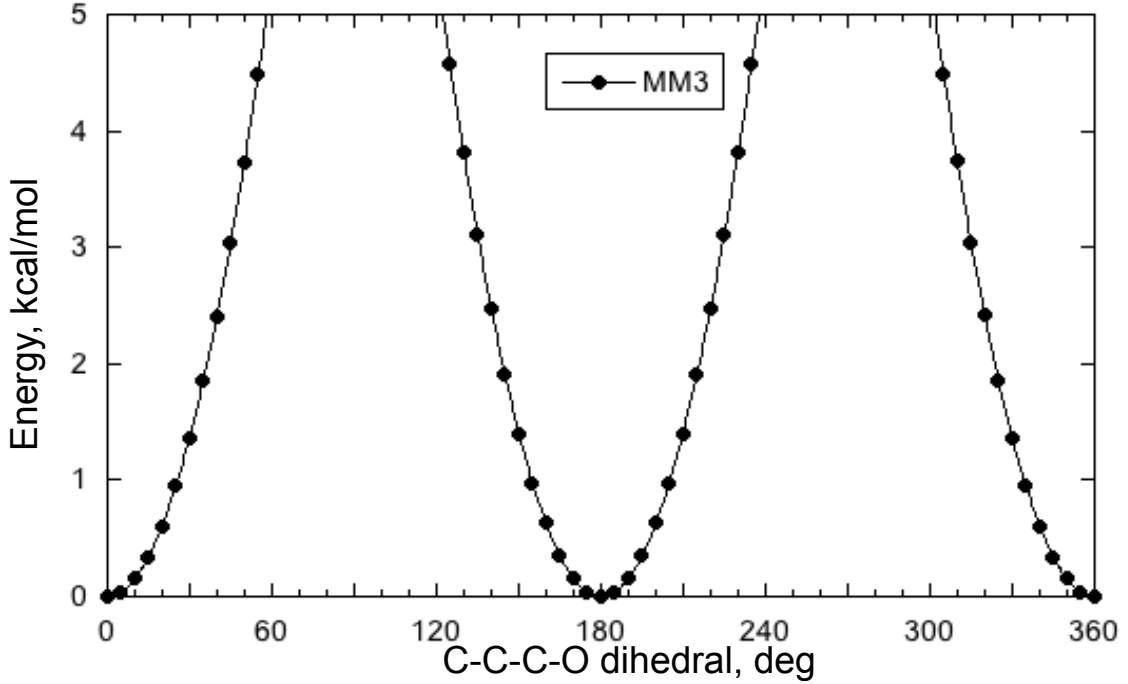
C–N = 1.83 ± 0.01 Å



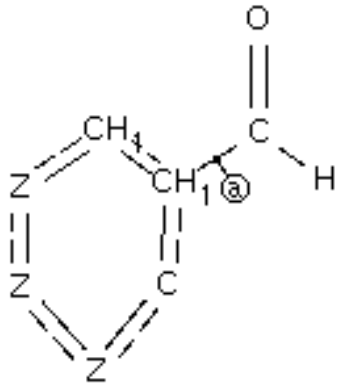


TYPE 3-1:10-1

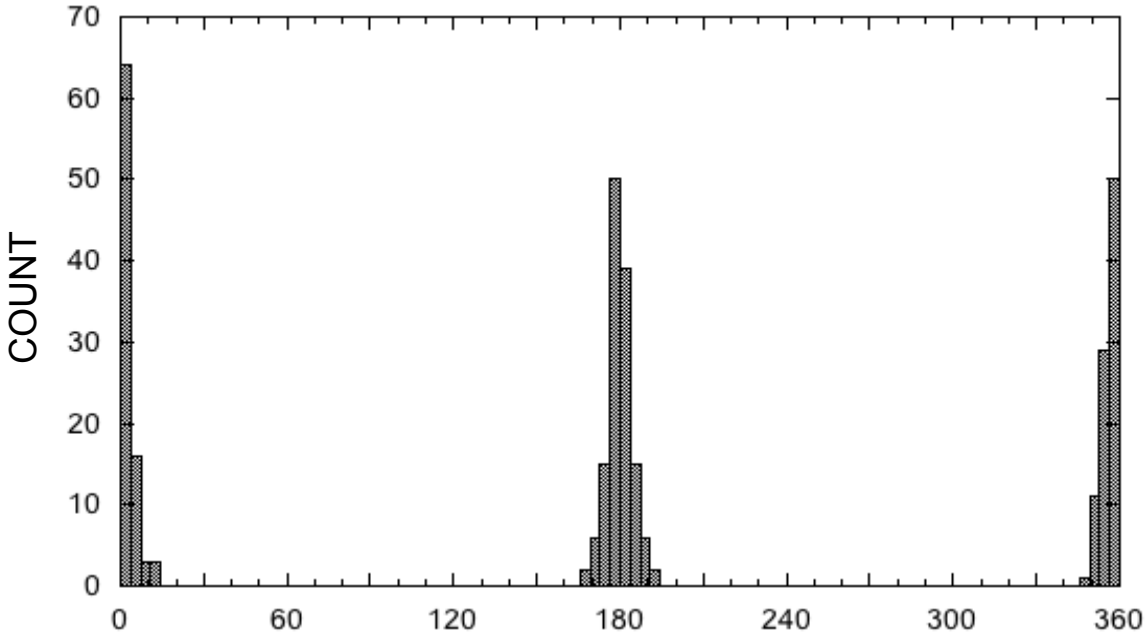
Φ	E
0.0	0.00
180.0	0.00



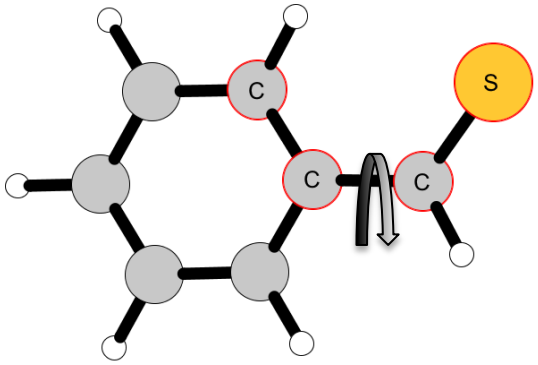
CSD comparison



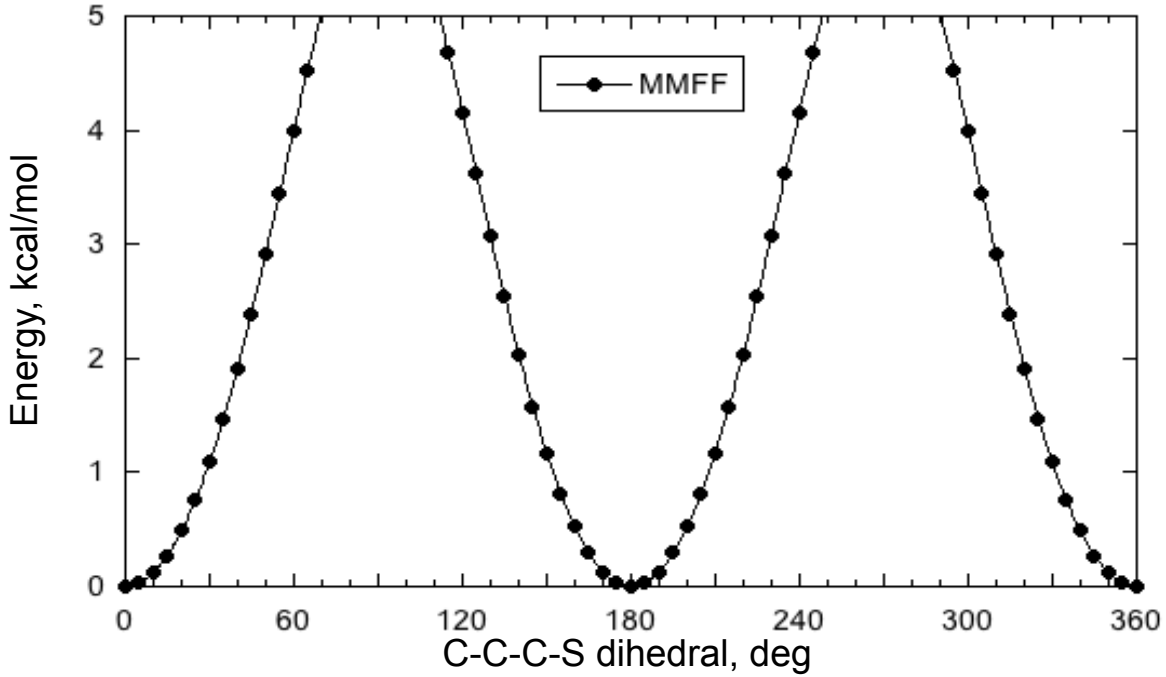
232 hits
C-C = 1.47 ± 0.02 Å



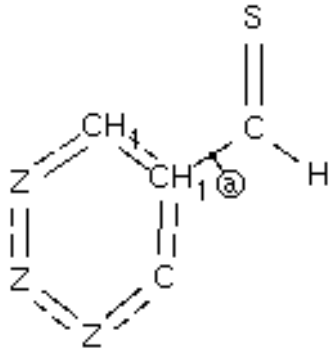
TYPE 3-1:10-2



Φ	E
0.0	0.00
180.0	0.00



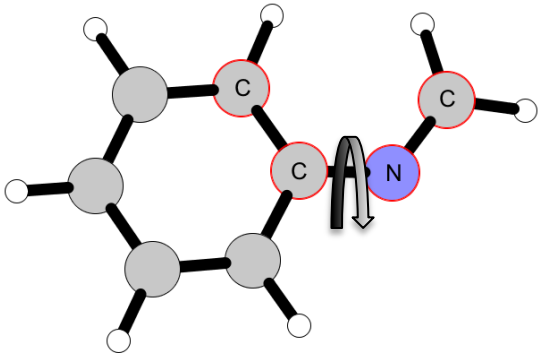
CSD comparison



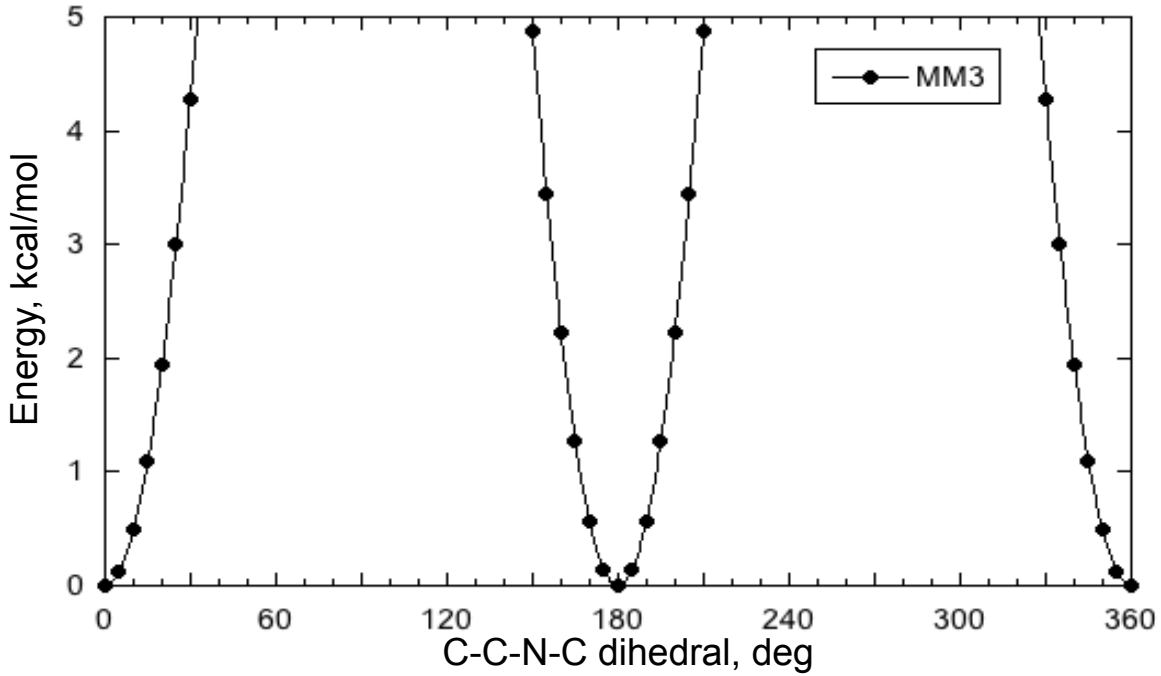
0 hits

NO CSD HITS

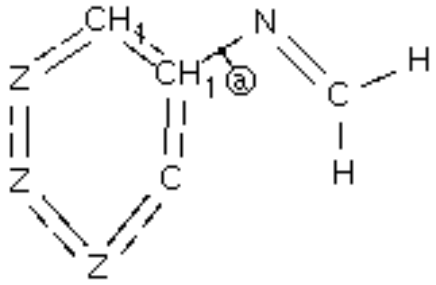
TYPE 3-1:10-3



Φ	E
0.0	0.00
180.0	0.00



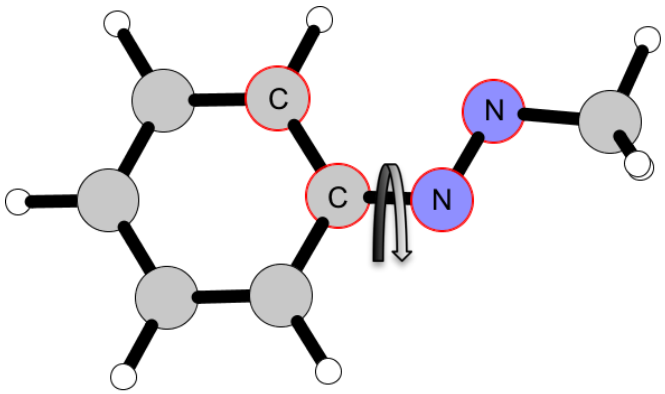
CSD comparison



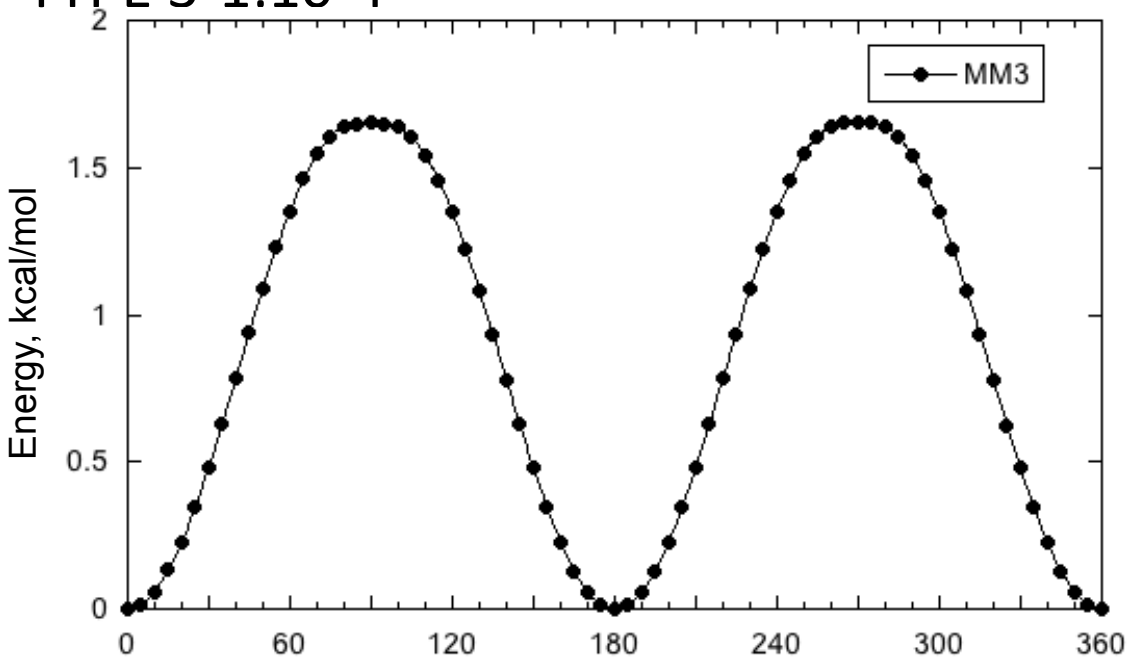
0 hits

NO CSD HITS

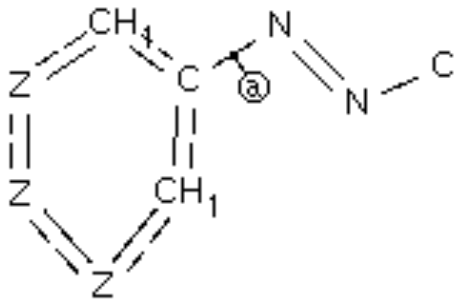
TYPE 3-1:10-4



Φ	E
0.0	0.00
180.0	0.00

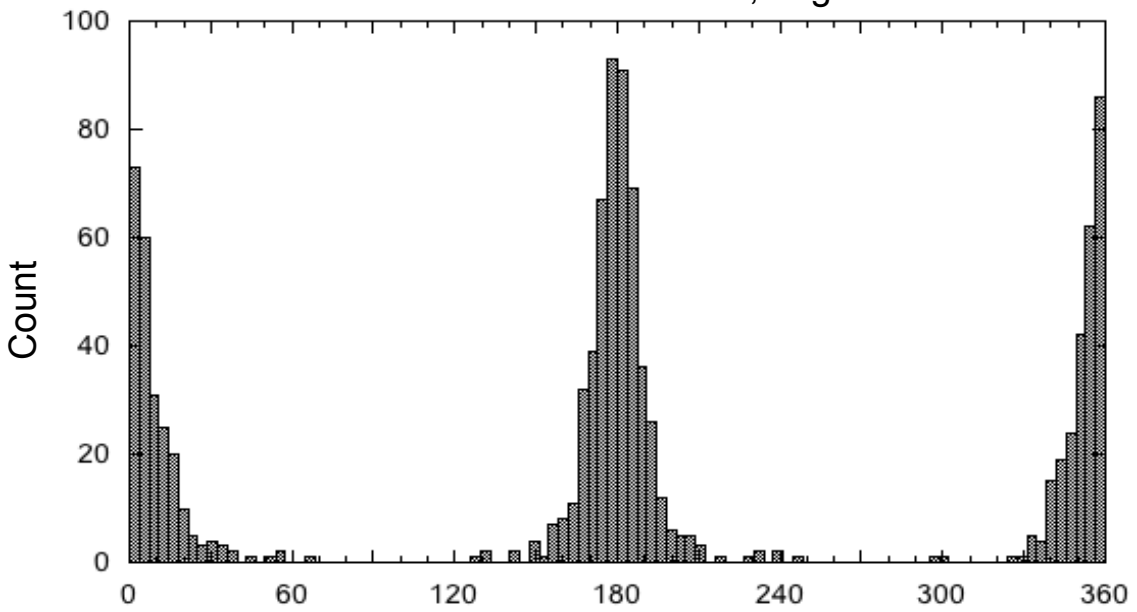


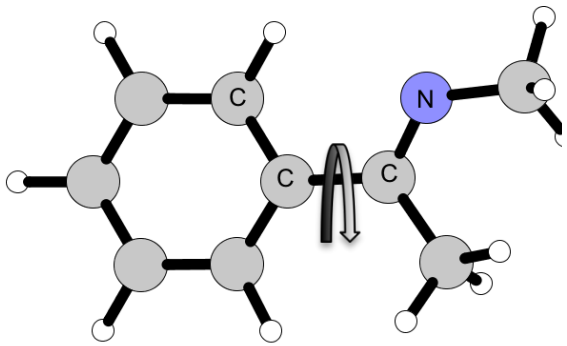
CSD comparison



632 hits

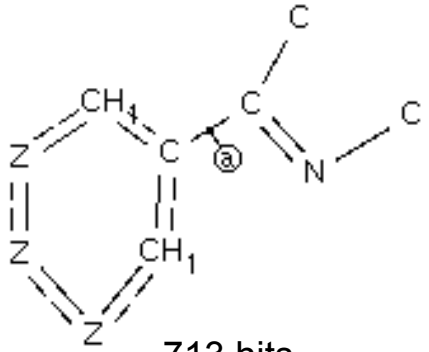
C–N = 1.43 ± 0.03 Å





Φ	E
15.0	0.00
95.0	2.95
165.0	0.00
195.0	0.00
270.0	2.94
345.0	0.00

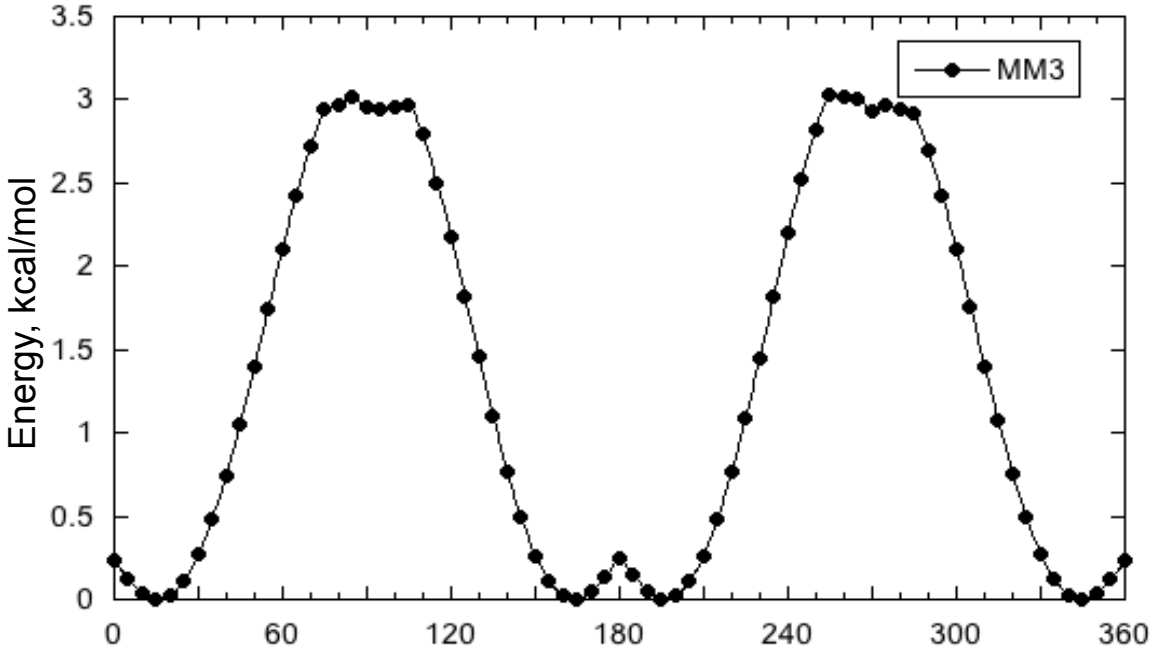
CSD comparison

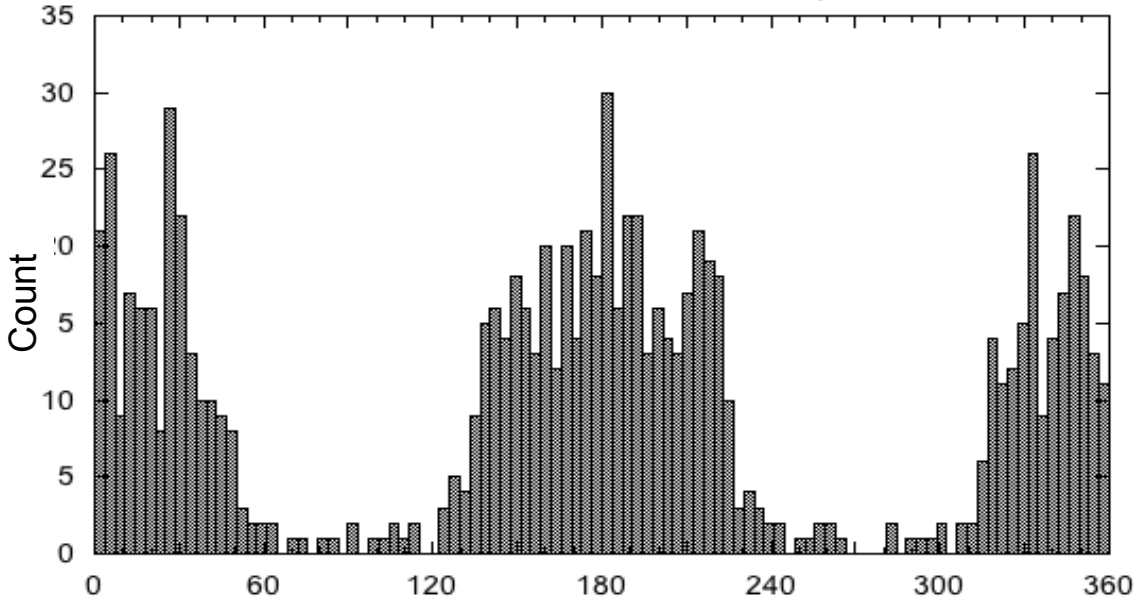


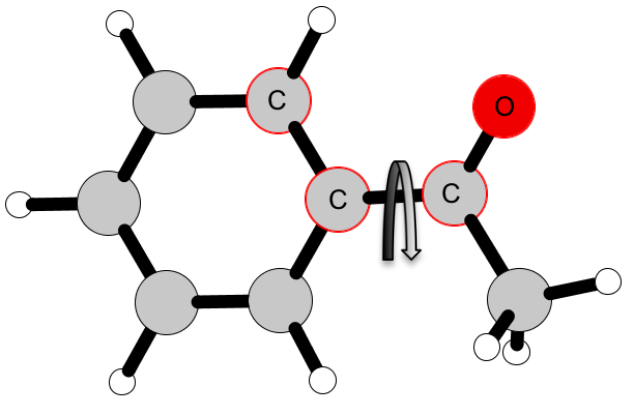
713 hits

C–C = 1.48 ± 0.02 Å

TYPE 3-1:11-1

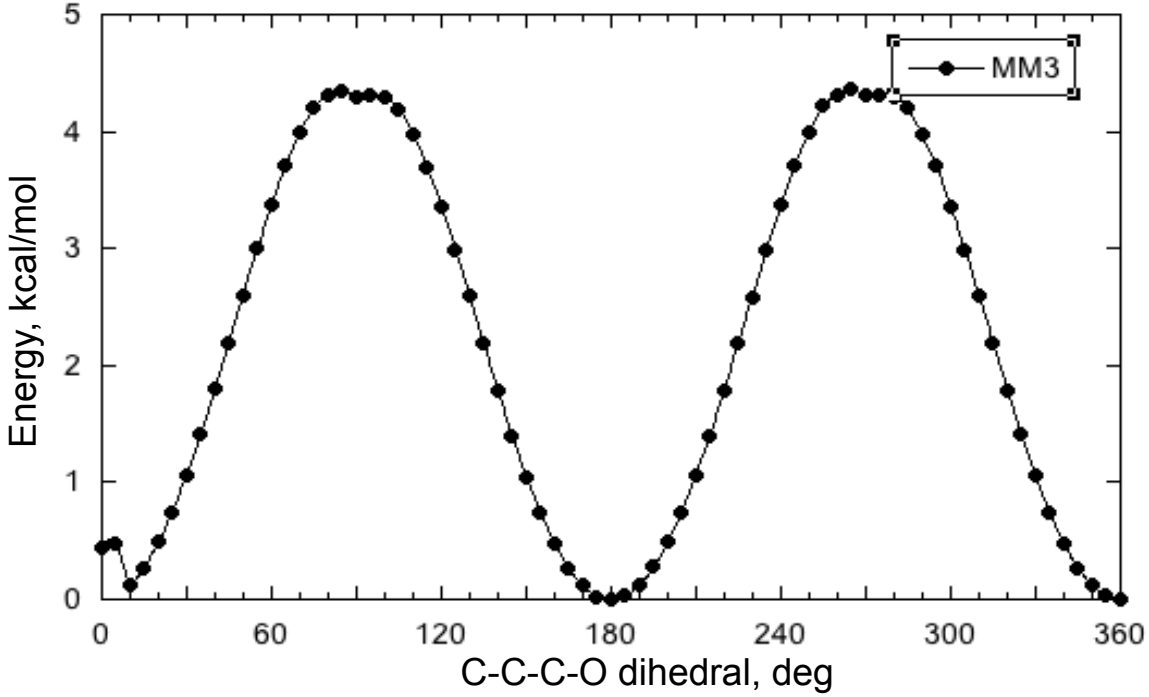




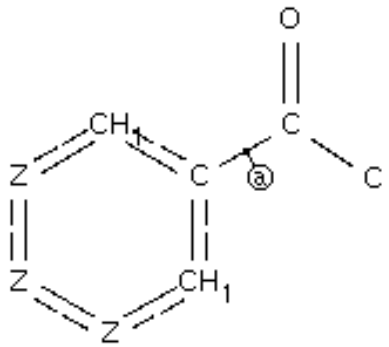


TYPE 3-1:11-2

Φ	E
0.0	0.00
90.0	4.30
180.0	0.00
270.0	4.30

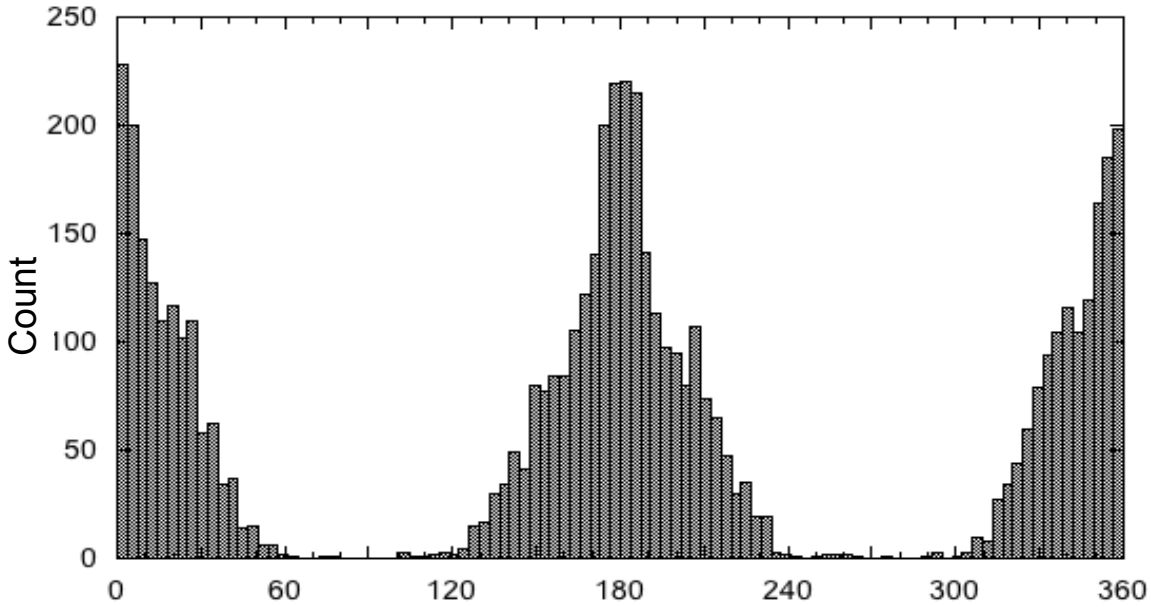


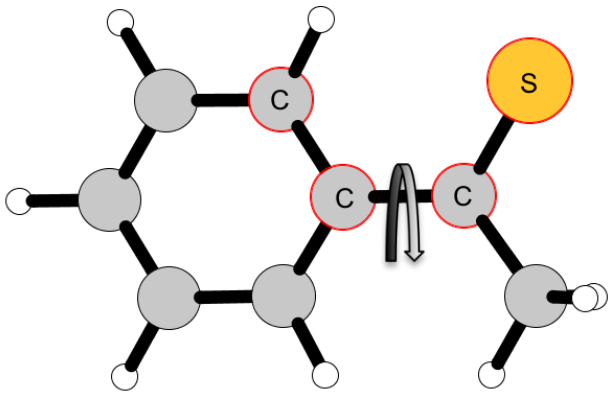
CSD comparison



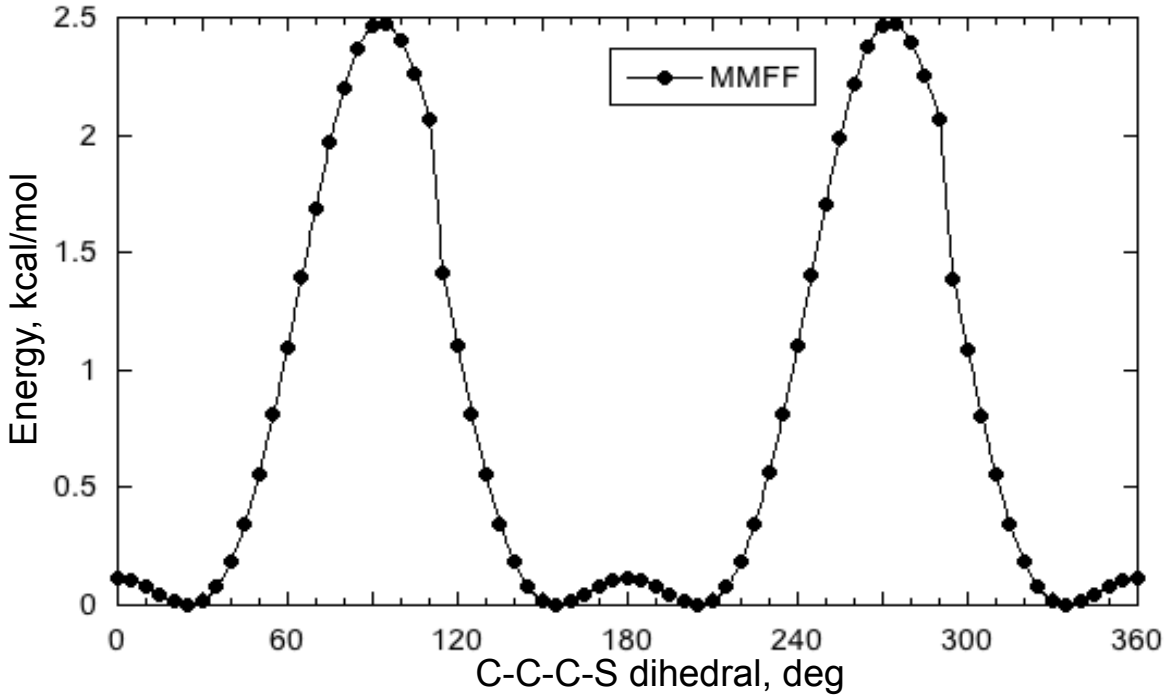
4046 hits

C–N = 1.49 ± 0.01 Å



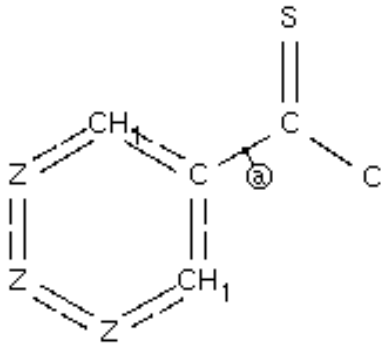


TYPE 3-1:11-3



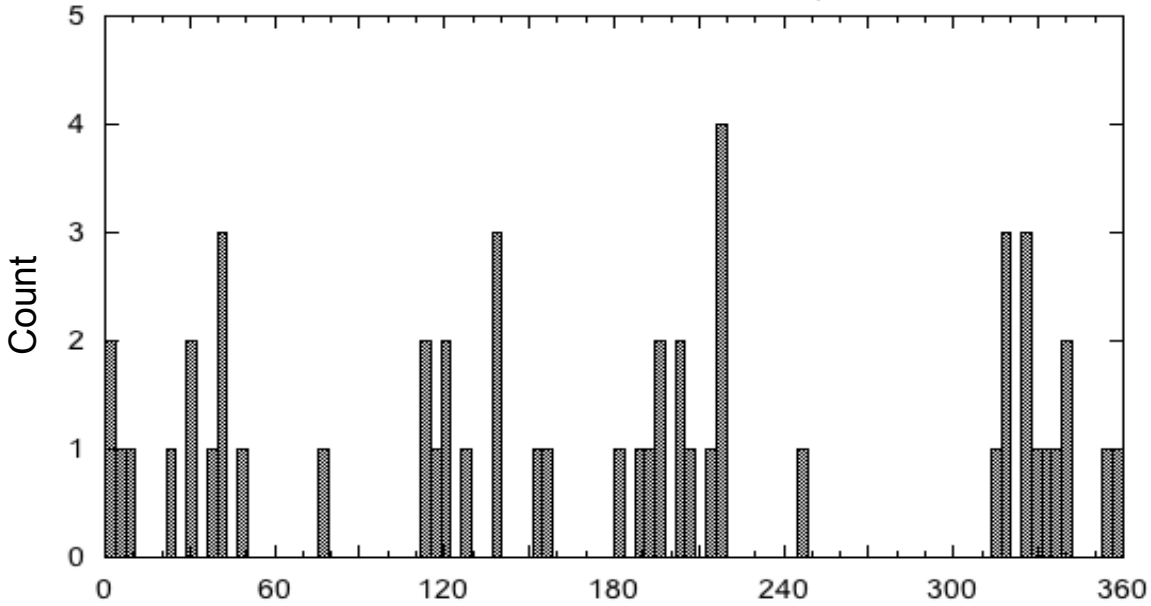
Φ	E
25.0	0.00
155.0	0.00
205.0	0.00
335.0	0.00

CSD comparison

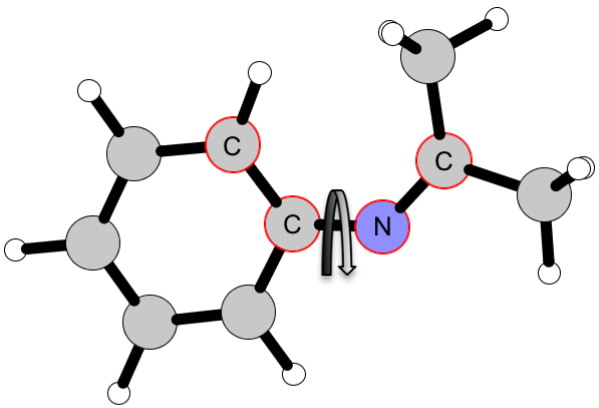


43 hits

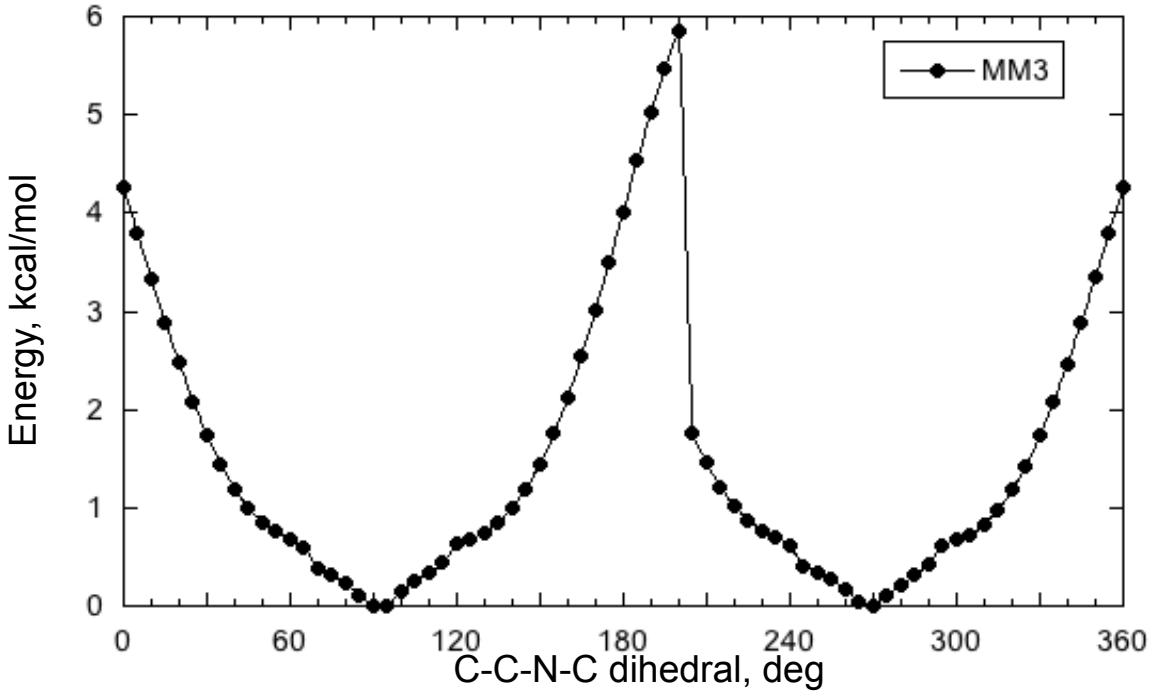
C–C = 1.48 ± 0.02 Å



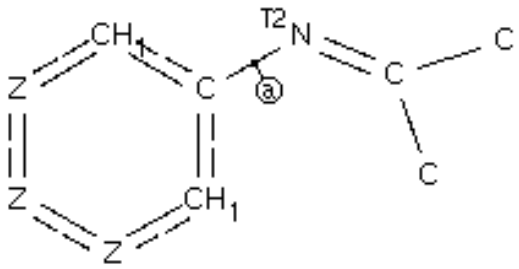
TYPE 3-1:12-1



Φ	E
90.0	0.00
270.0	0.01

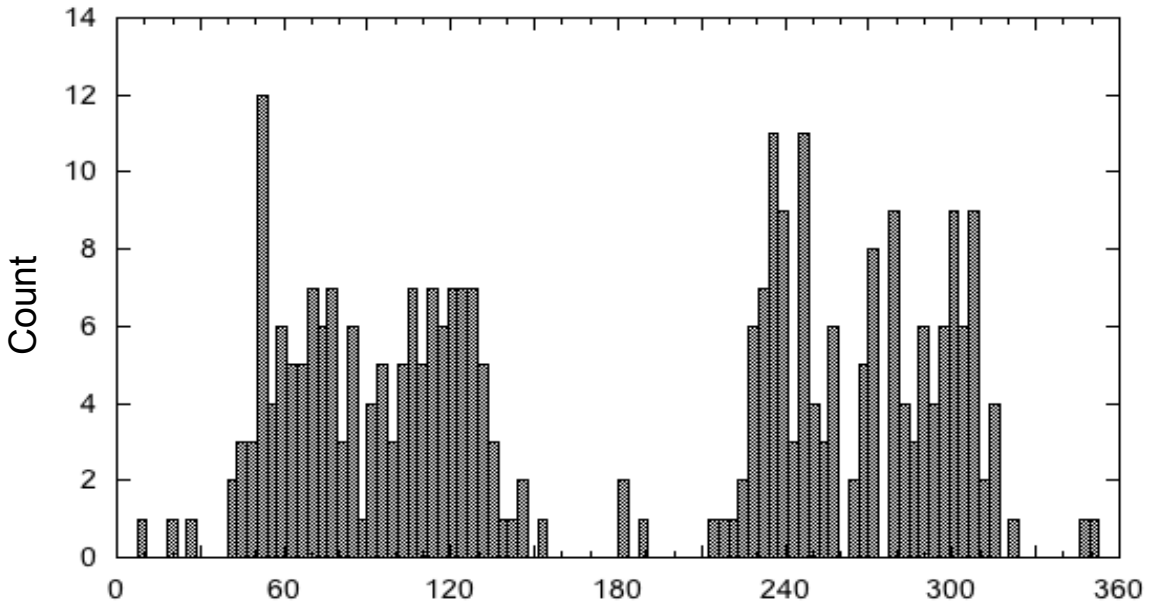


CSD comparison

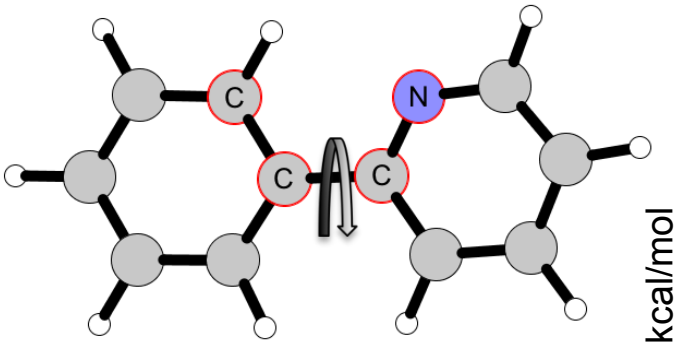


229 hits

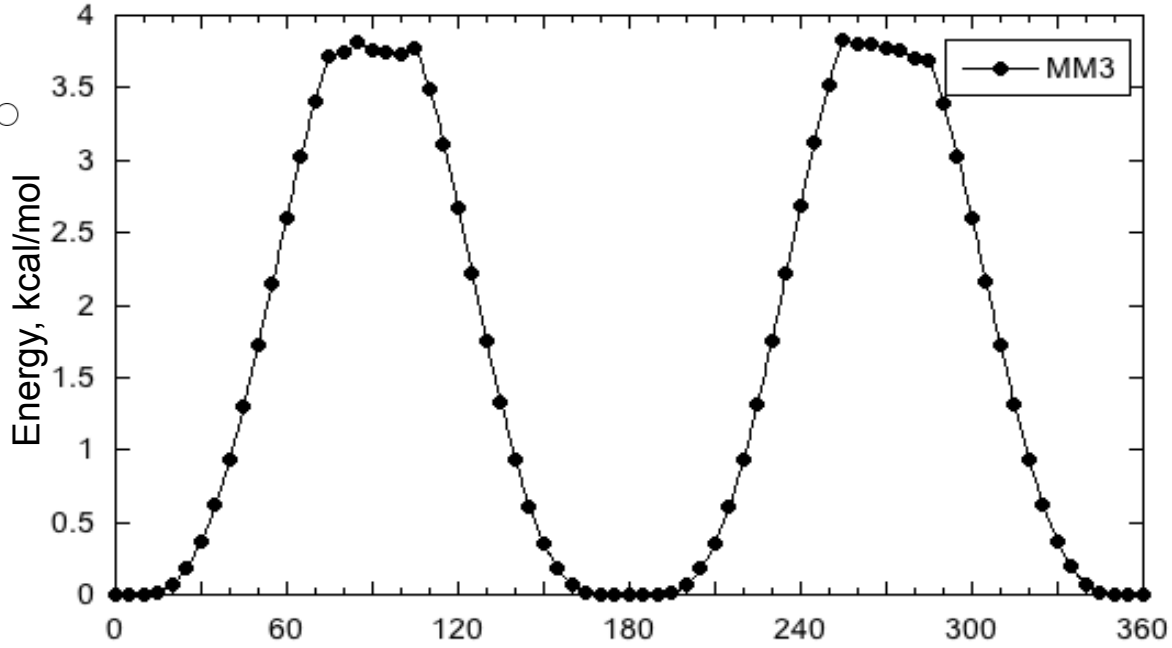
C–N = 1.42 ± 0.01 Å



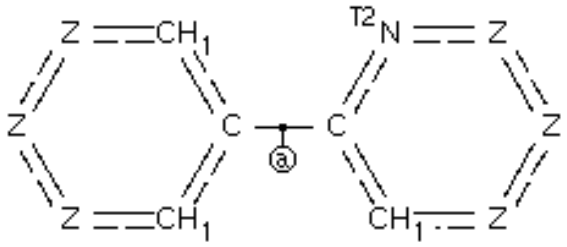
TYPE 3-1:13-1



Φ	E
10.0	0.00
190.0	0.00
350.0	0.00

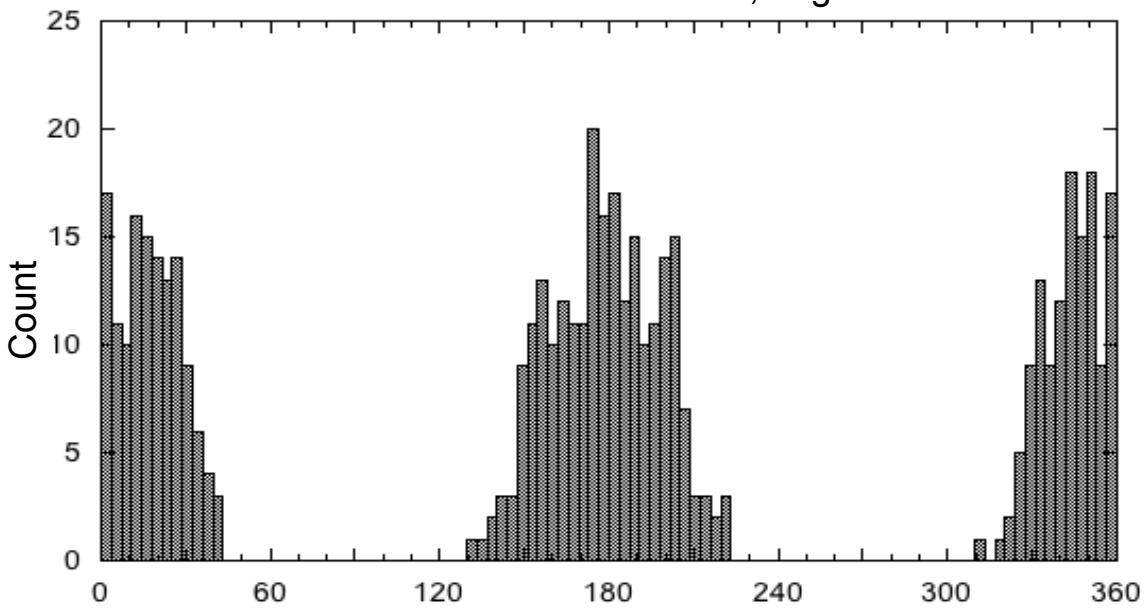


CSD comparison

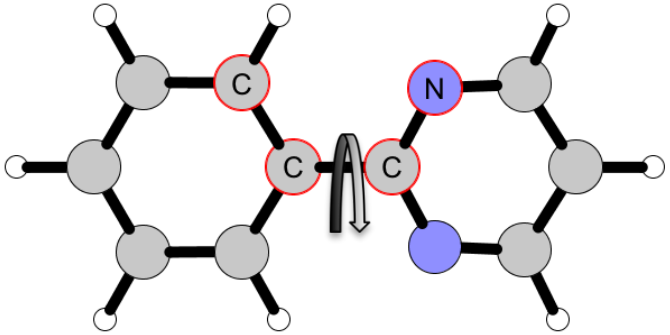


355 hits

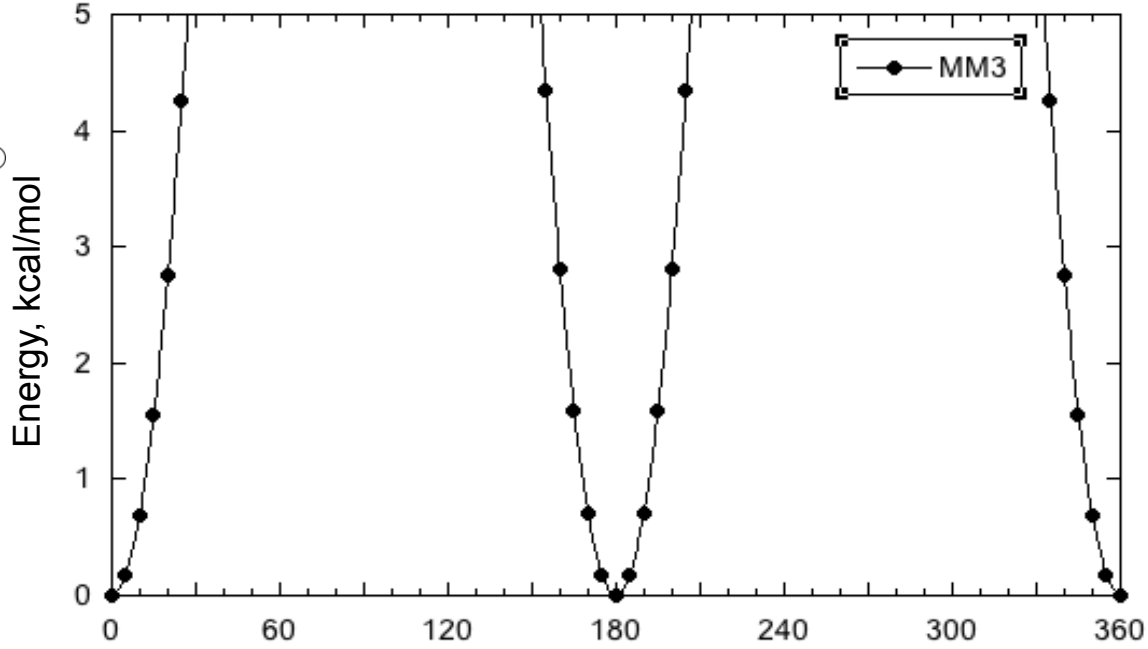
C–C = 1.48 ± 0.01 Å



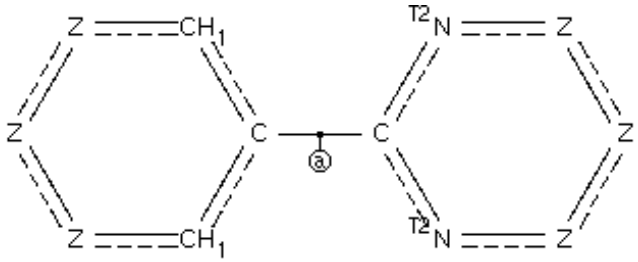
TYPE 3-1:13-2



Φ	E
0.0	0.00
180.0	0.00

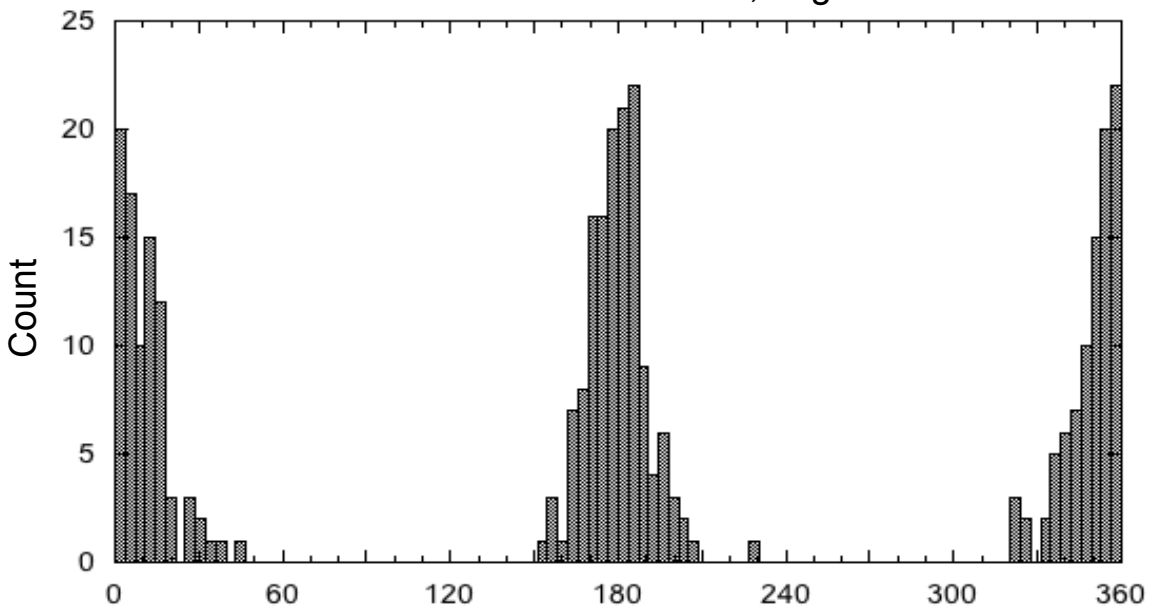


CSD comparison

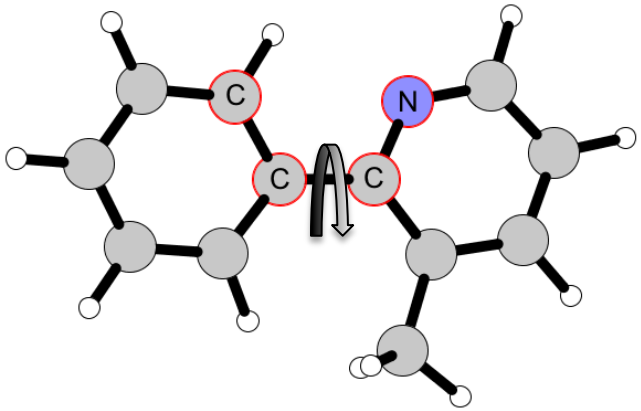


206 hits

C–N = 1.48 ± 0.01 Å

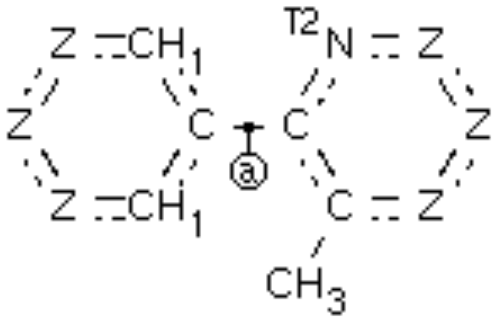


TYPE 3-1:13-3

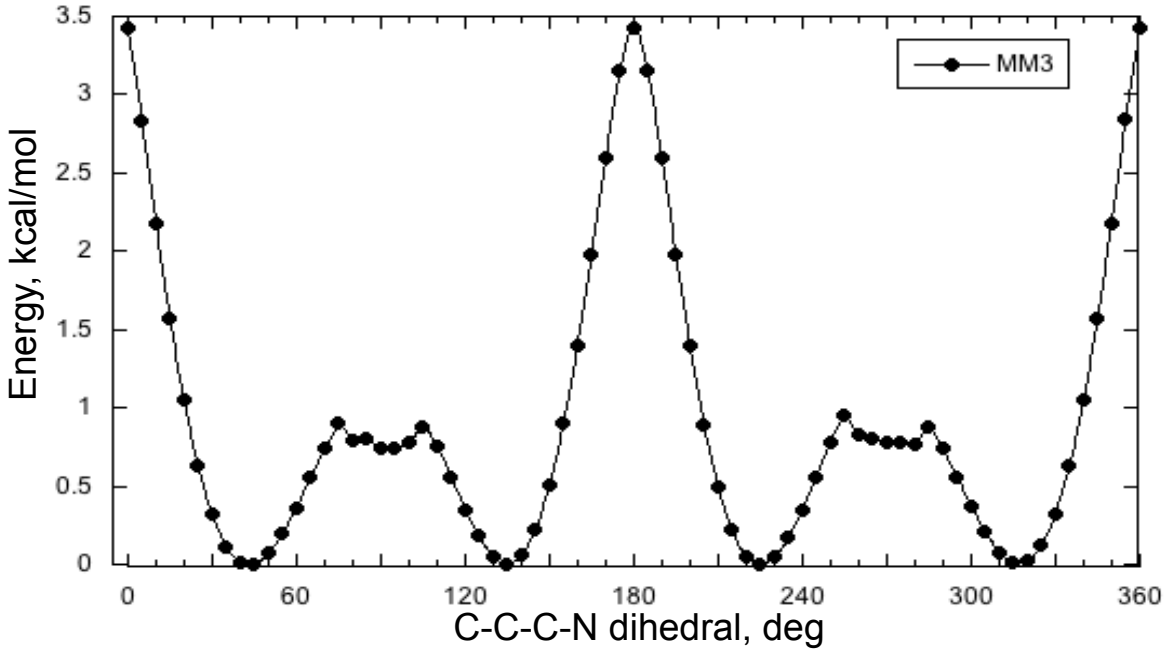


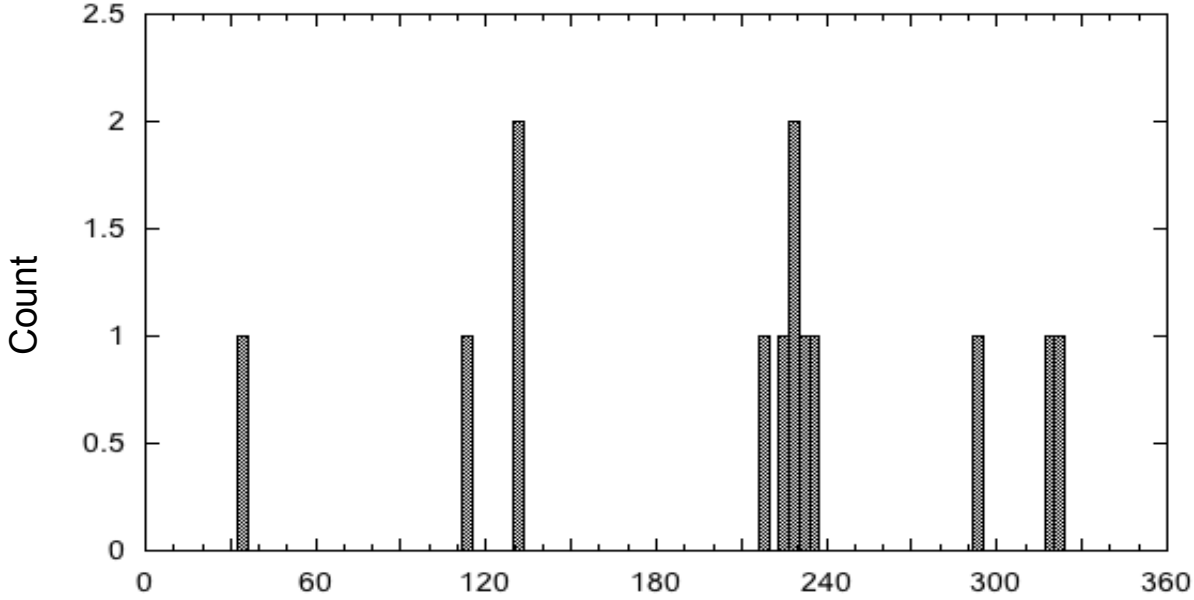
Φ	E
45.0	0.00
90.0	0.74
135.0	0.00
225.0	0.00
270.0	0.77
315.0	0.01

CSD comparison

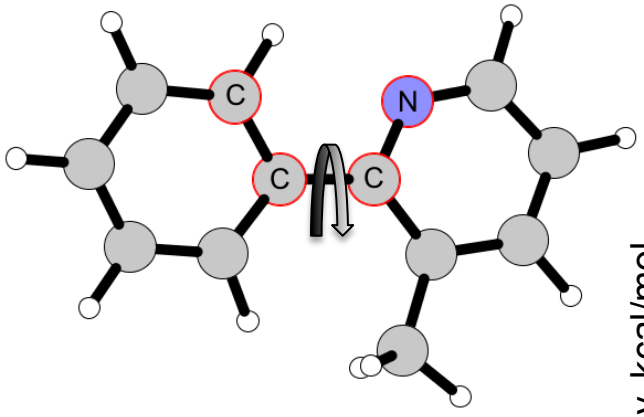


10 hits
C-C = 1.49 ± 0.00 Å

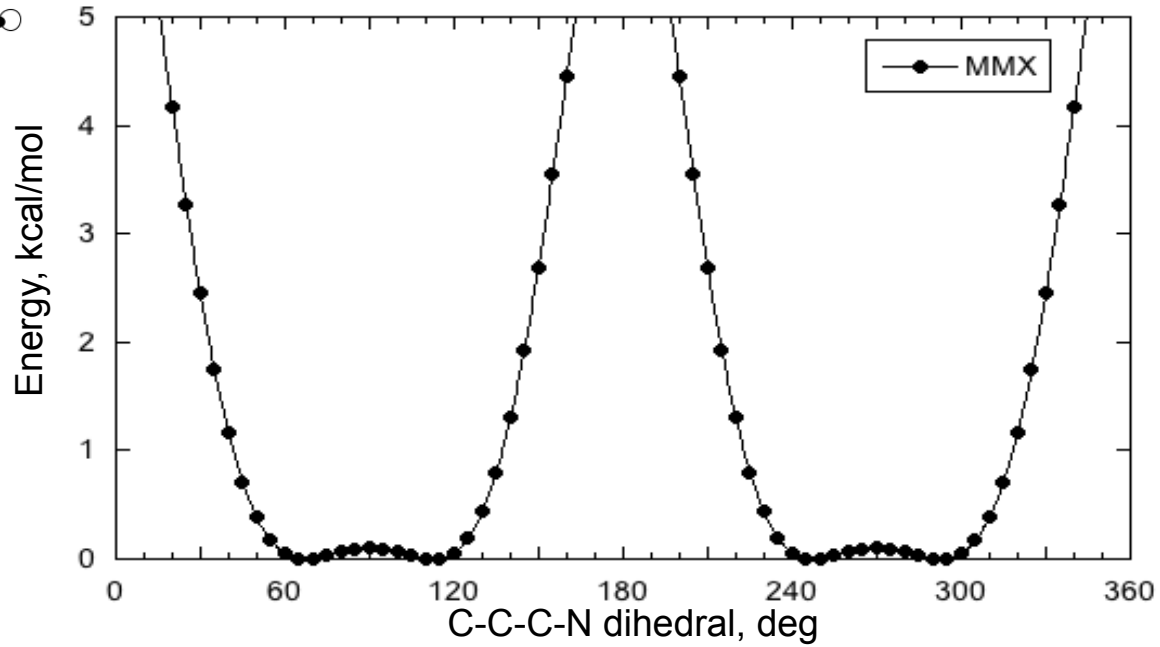




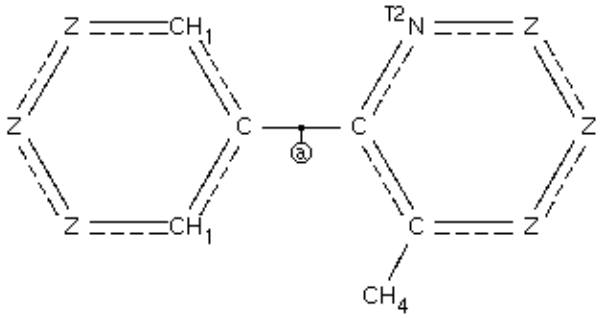
TYPE 3-1:13-3



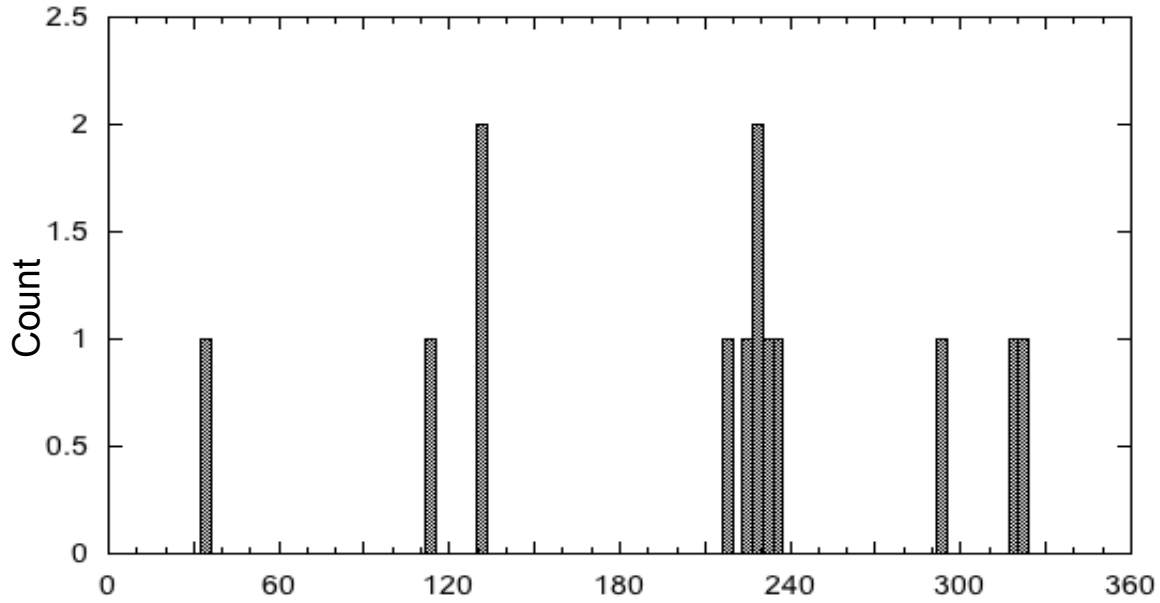
Φ	E
65.0	0.00
115.0	0.00
250.0	0.00
295.0	0.00



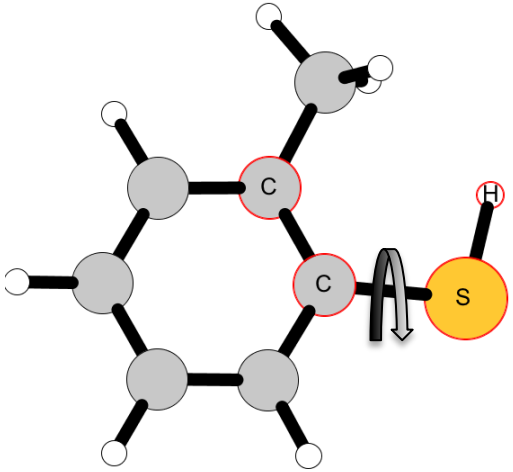
CSD comparison



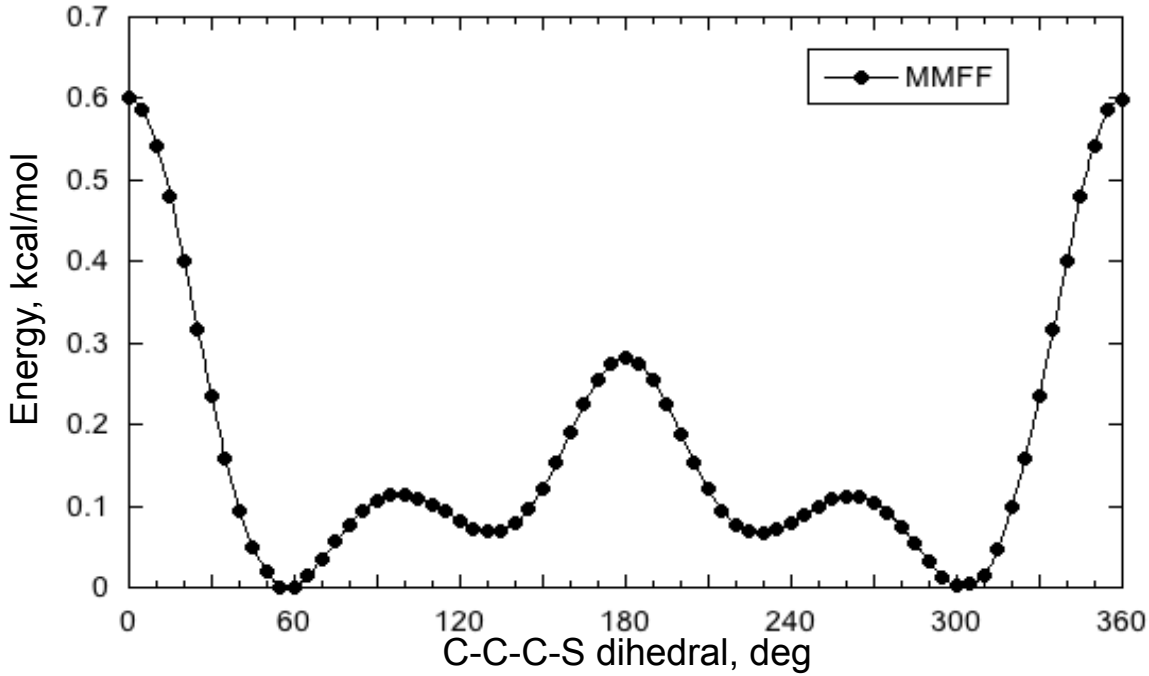
11 hits
C-C = 1.49 ± 0.00 Å



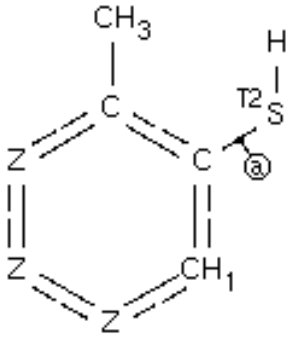
TYPE 3-2:5-4



Φ	E
55.0	0.00
130.0	0.07
230.0	0.07
305.0	0.00



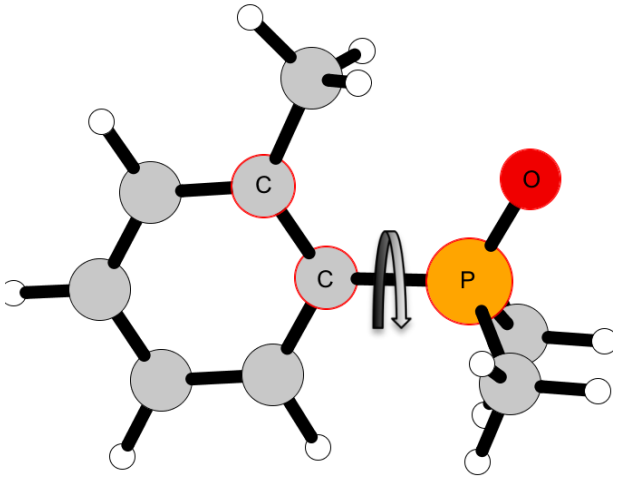
CSD comparison



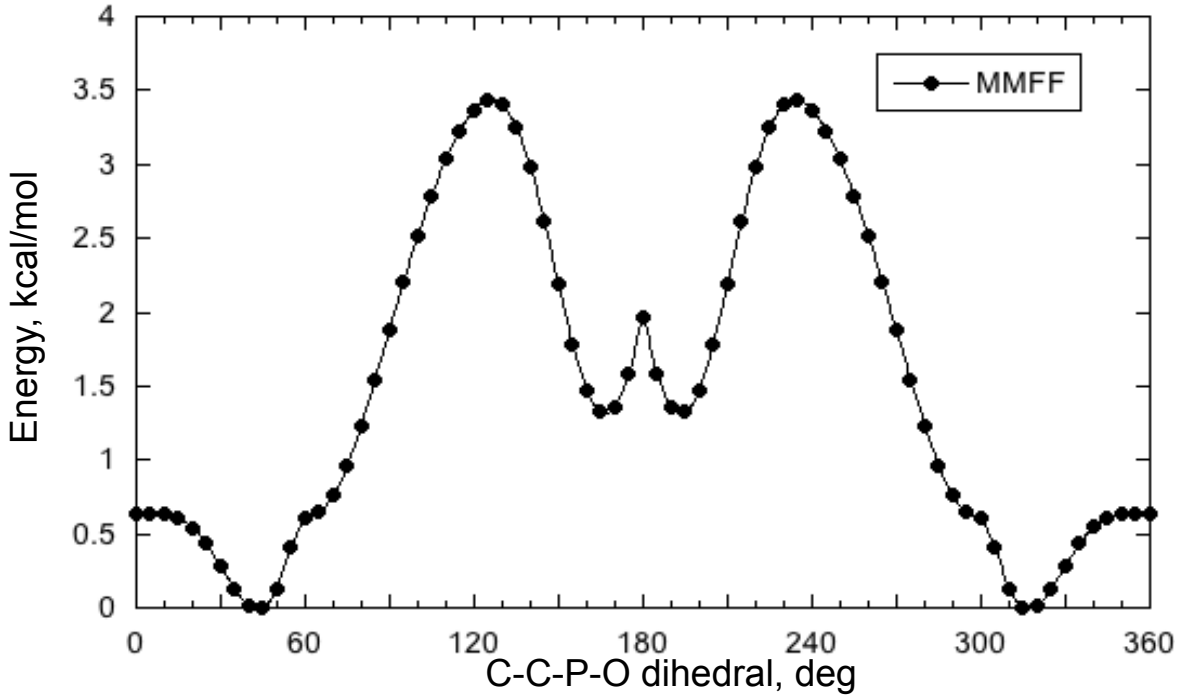
0 hits

NO CSD HITS

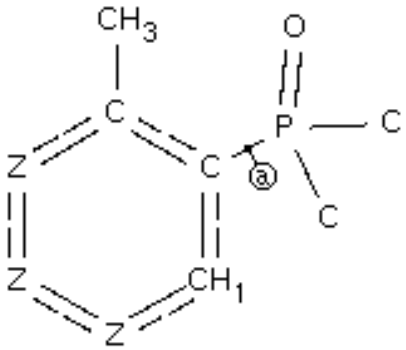
TYPE 3-2:8-1



Φ	E
45.0	0.00
165.0	1.32
195.0	1.32
315.0	0.00

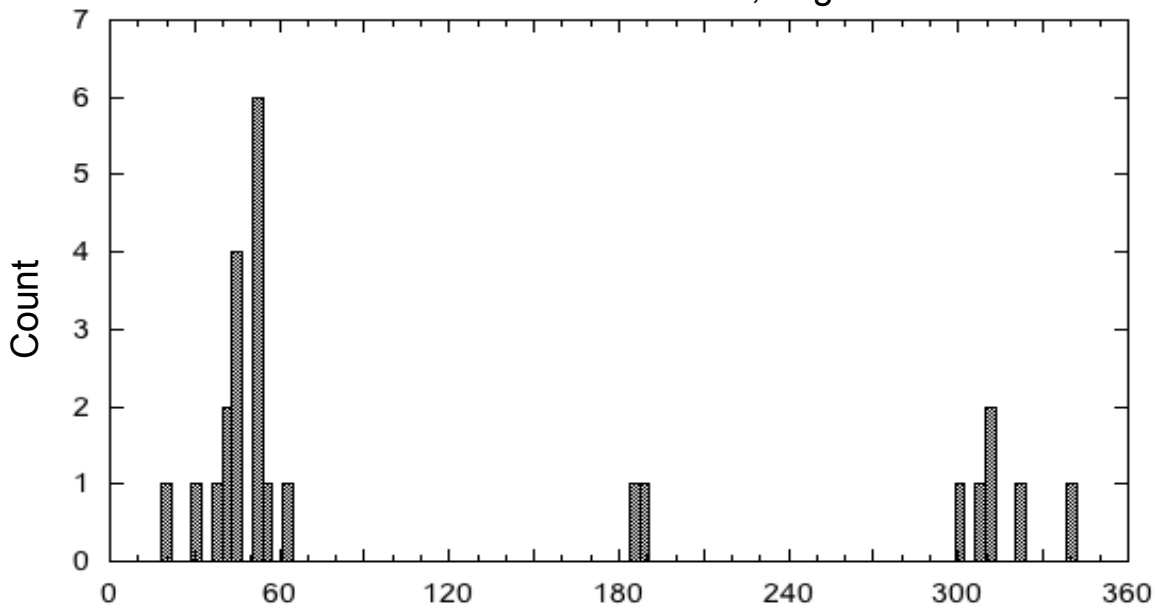


CSD comparison

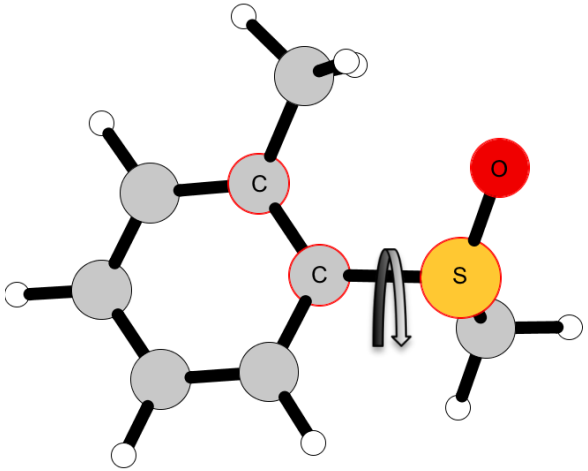


10 hits

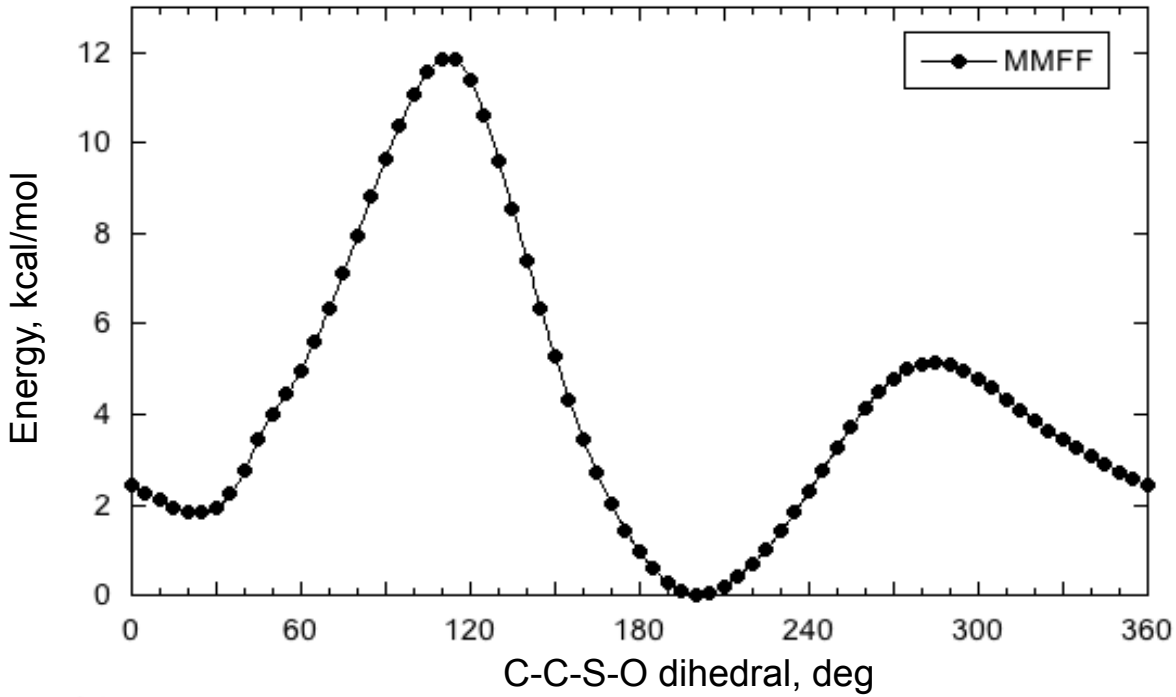
C–P = 1.82 ± 0.01 Å



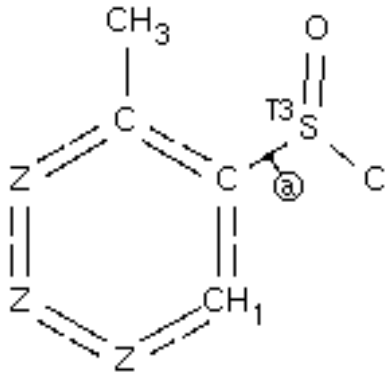
TYPE 3-2:8-2



Φ	E
25.0	1.82
200.0	0.00

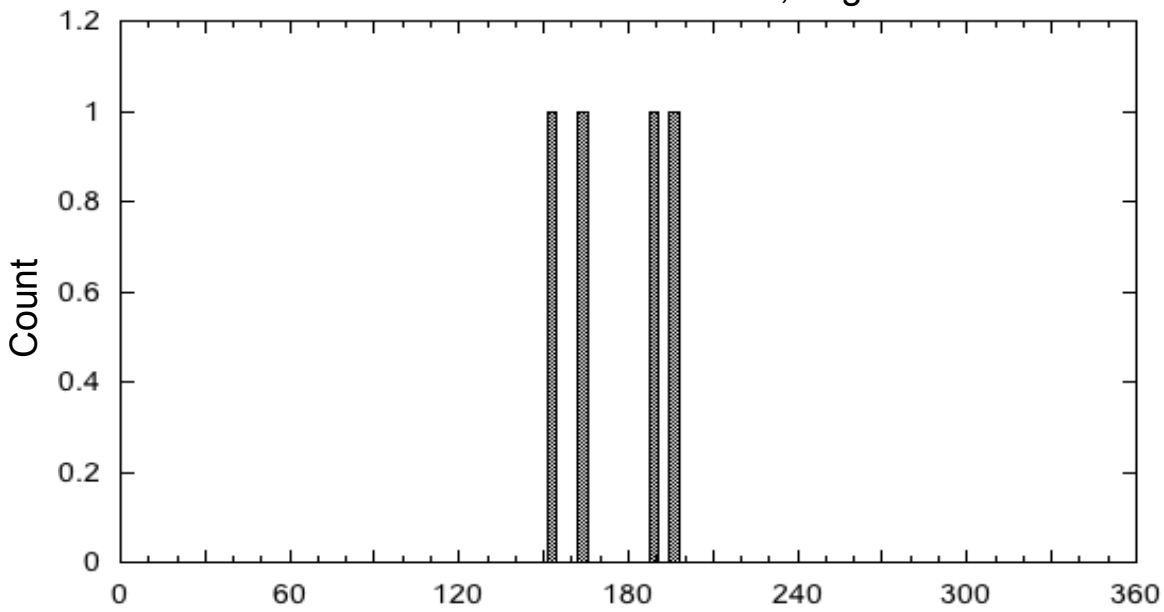


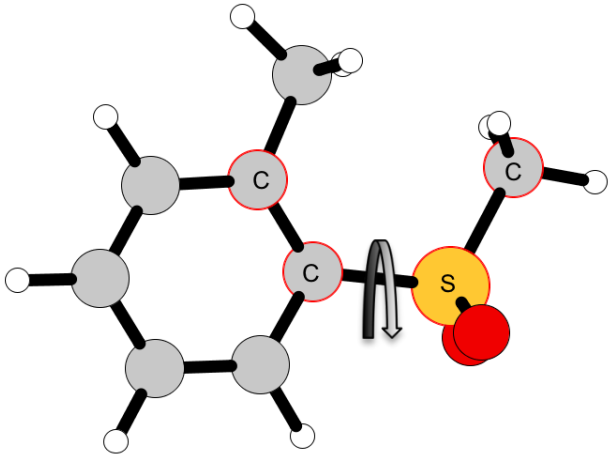
CSD comparison



3 hits

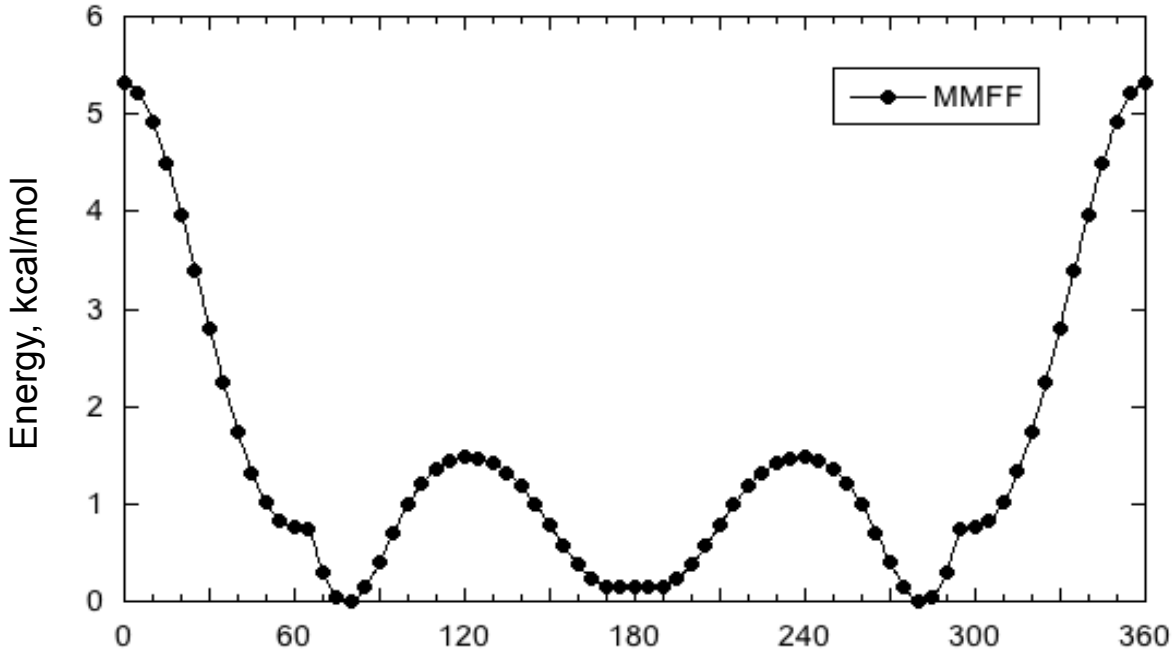
C–S = 1.80 ± 0.00 Å



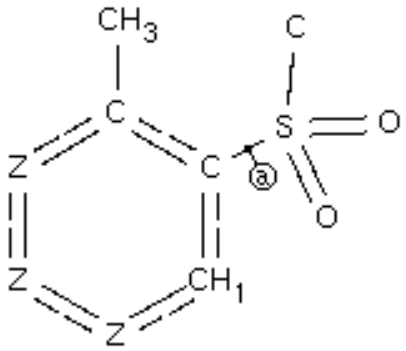


TYPE 3-2:8-4

Φ	E
80.0	0.00
175.0	0.14
185.0	0.14
280.0	0.00

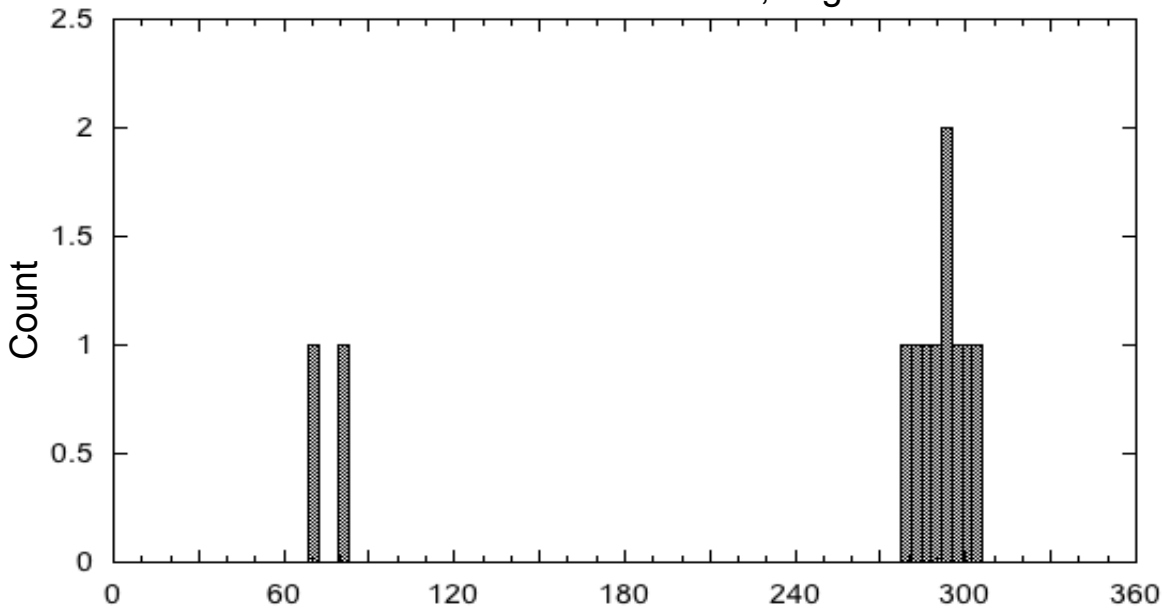


CSD comparison

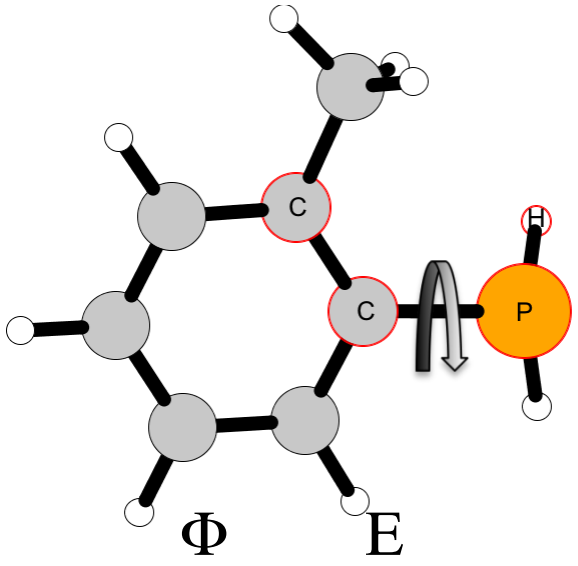


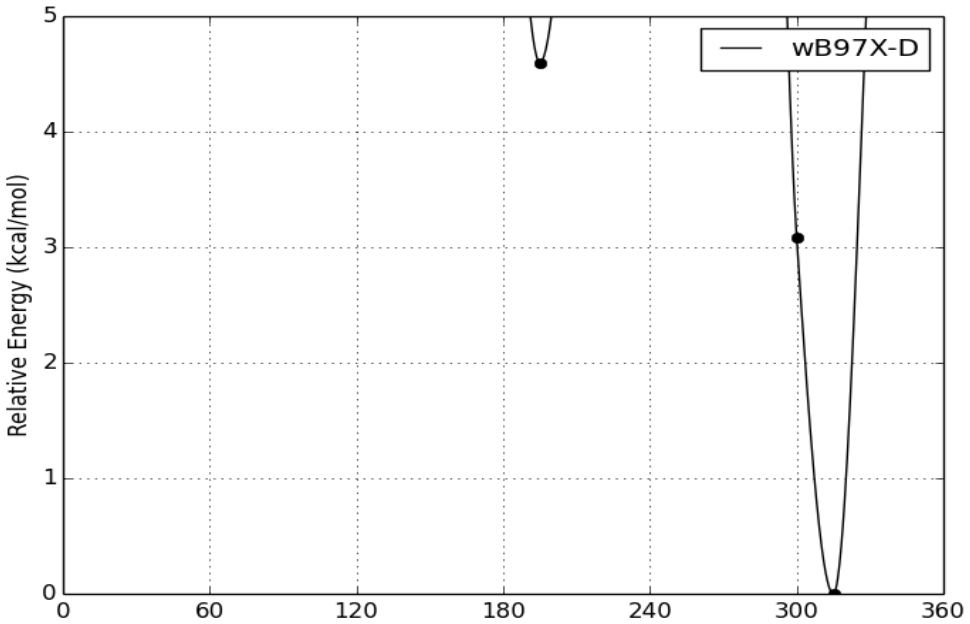
10 hits

C–S = 1.78 ± 0.02 Å

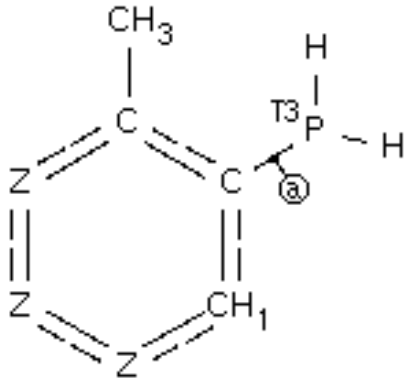


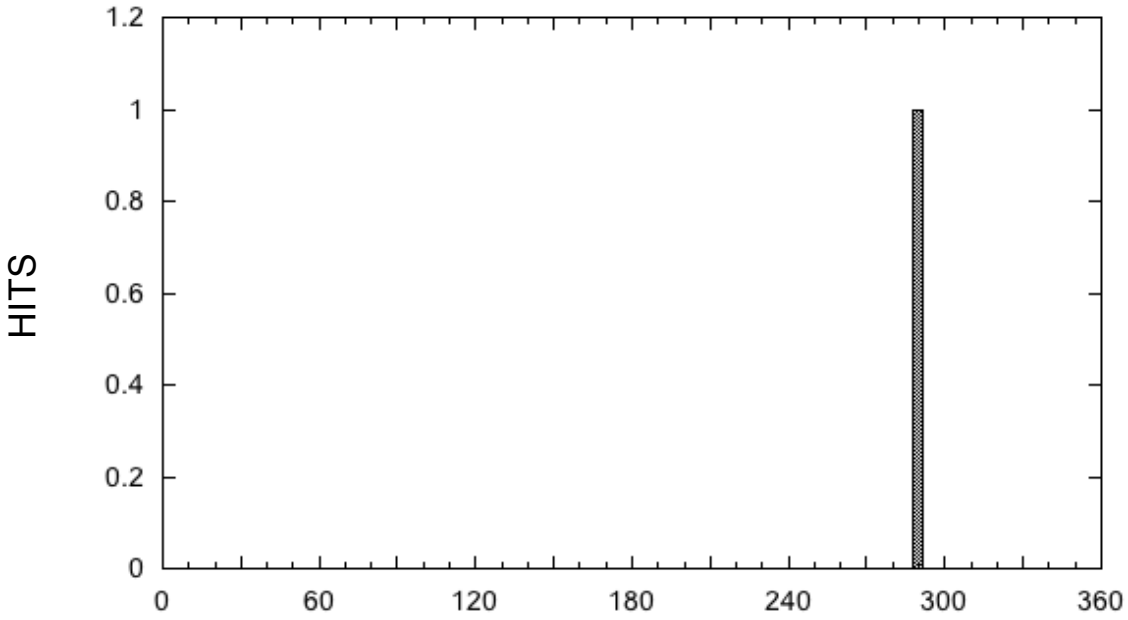
TYPE 3-2:9-1



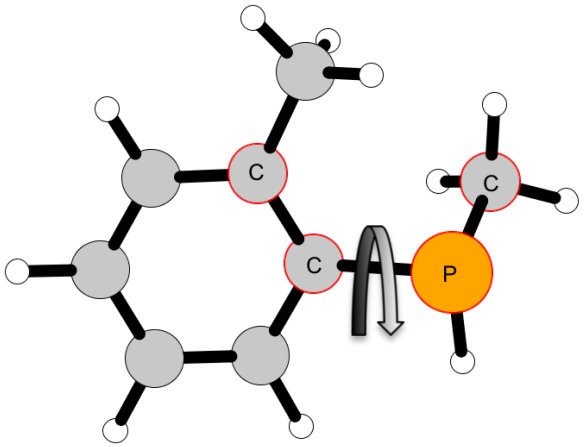


CSD comparison

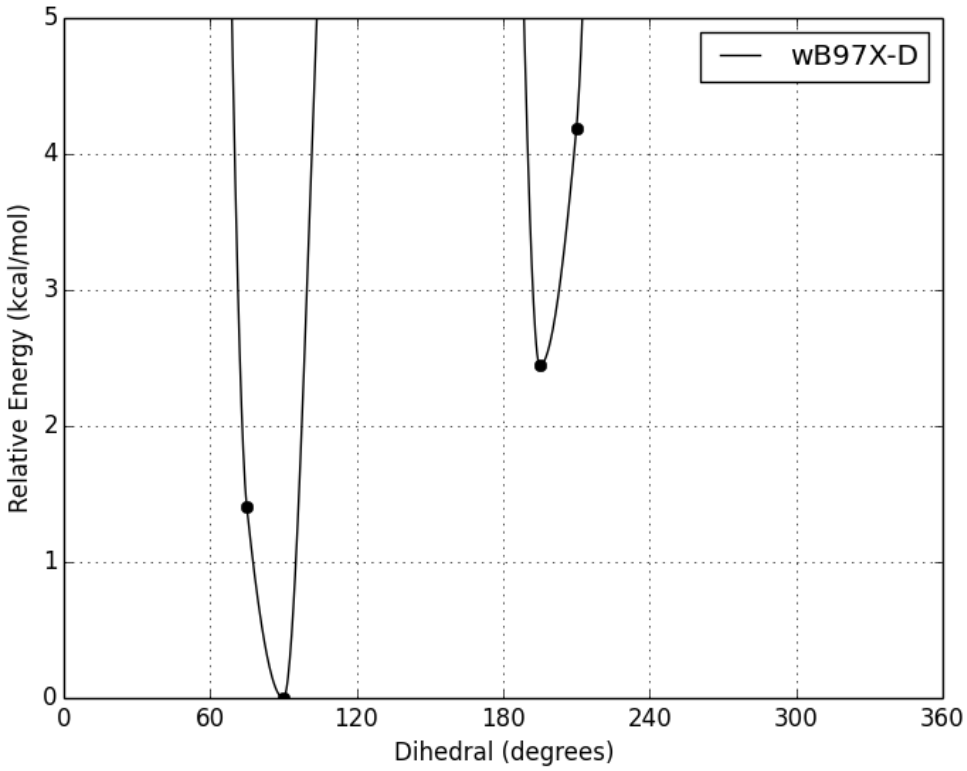




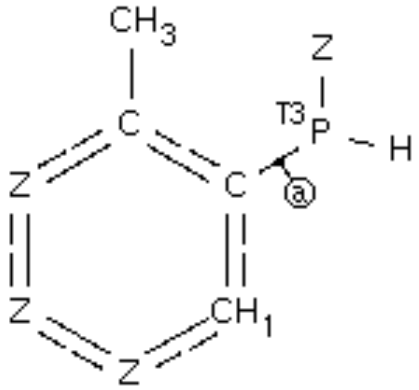
TYPE 3-2:9-2



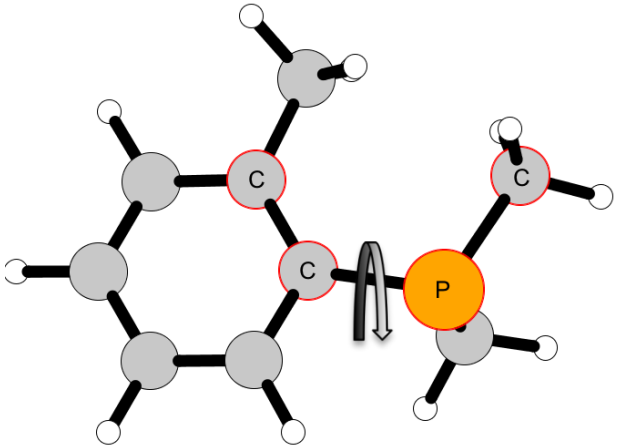
Φ	E
90.0	0.00
195.0	2.44



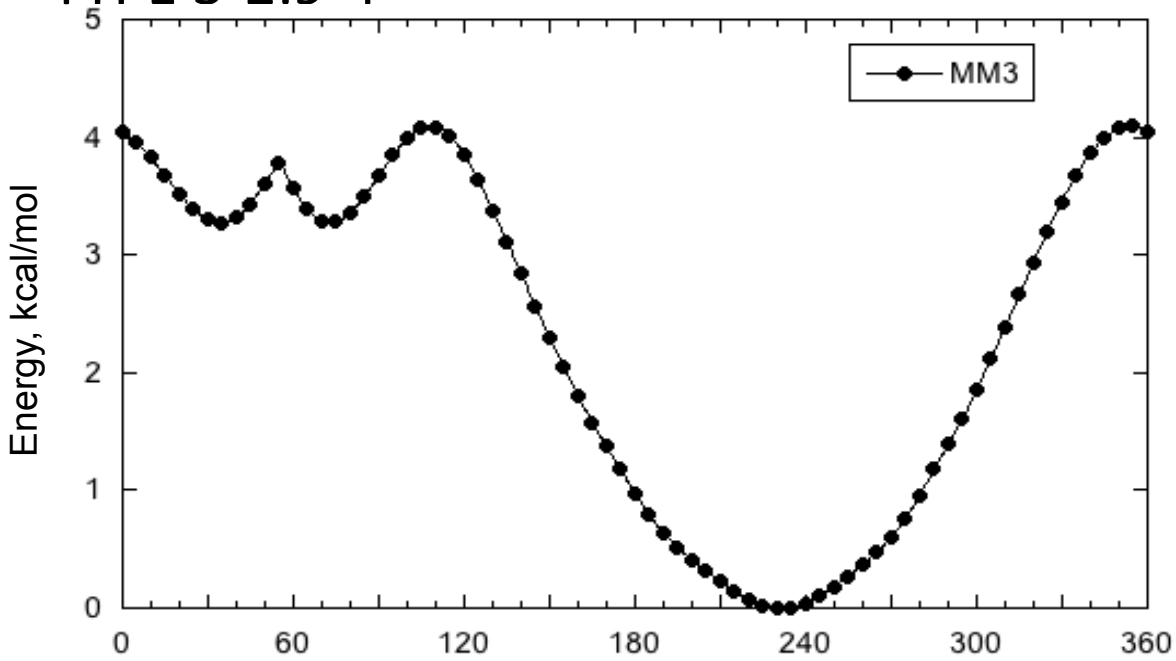
CSD comparison



NO CSD HITS

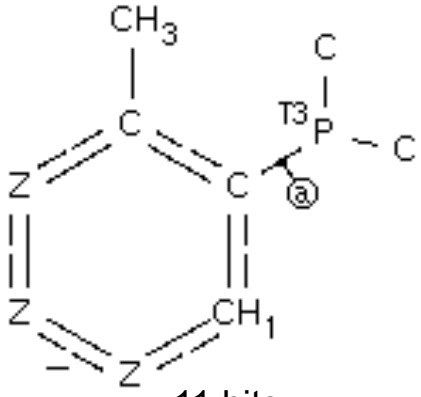


TYPE 3-2:9-4



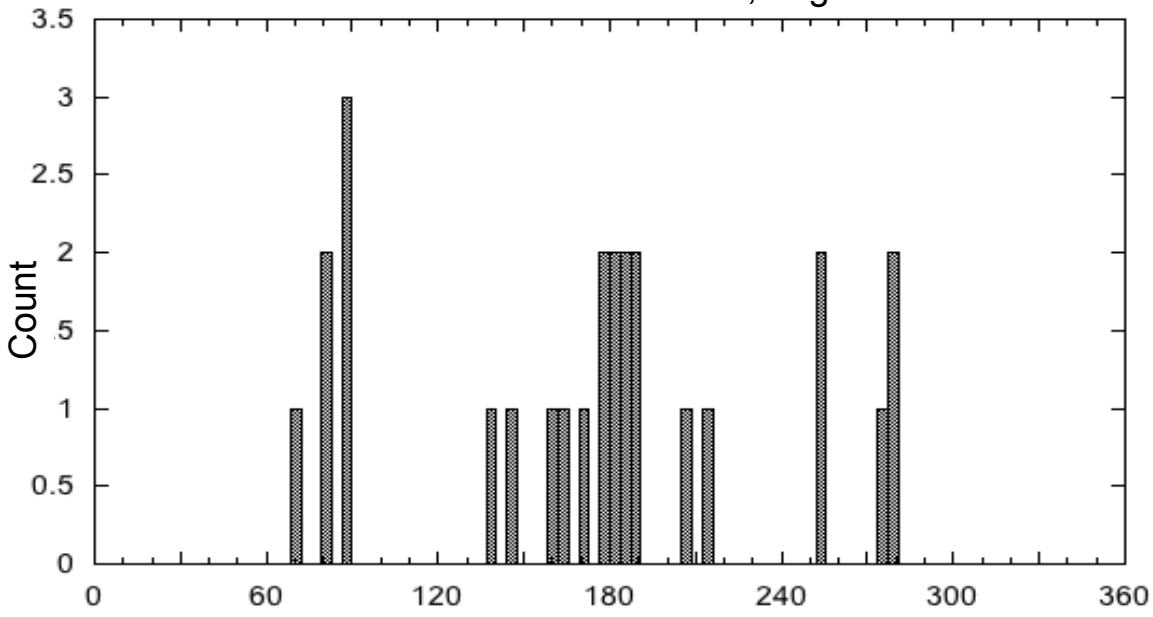
Φ	E
35.0	3.27
75.0	3.28
230.0	0.00

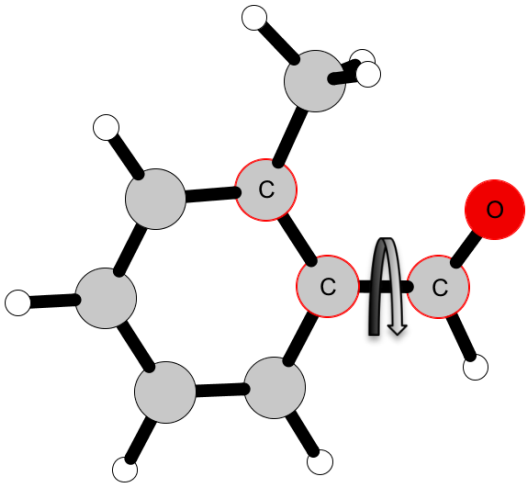
CSD comparison



11 hits

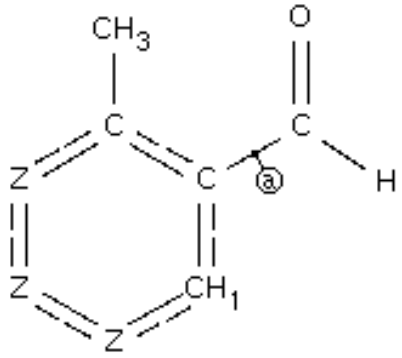
C–P = 1.84 ± 0.01 Å





Φ	E
0.0	0.62
180.0	0.00

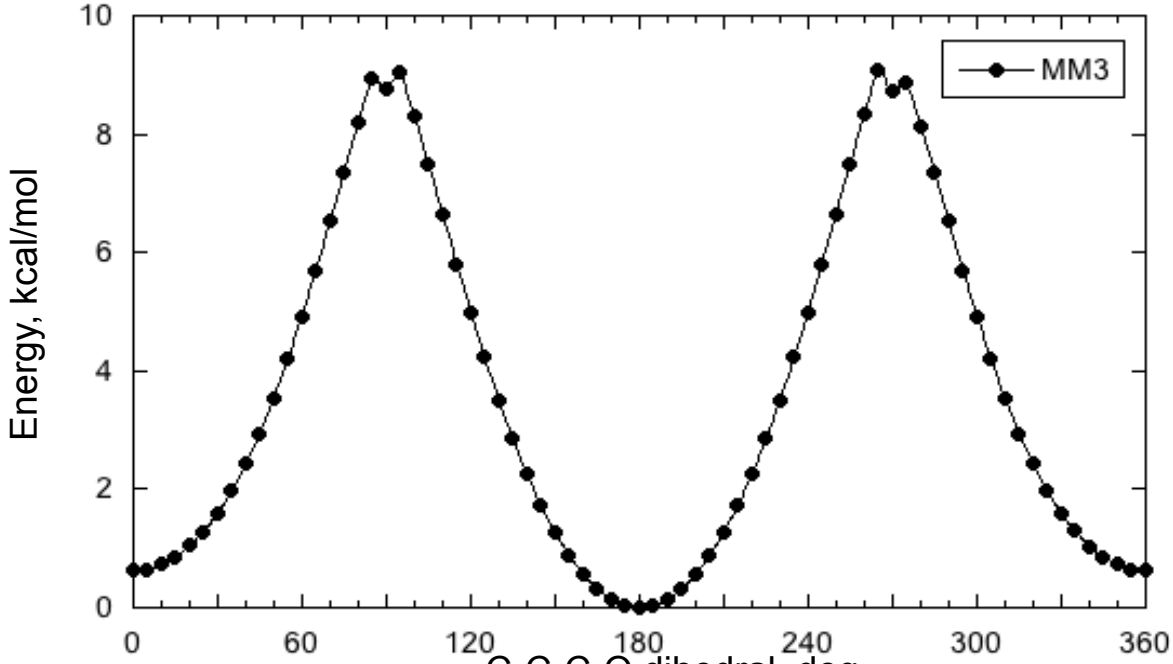
CSD comparison

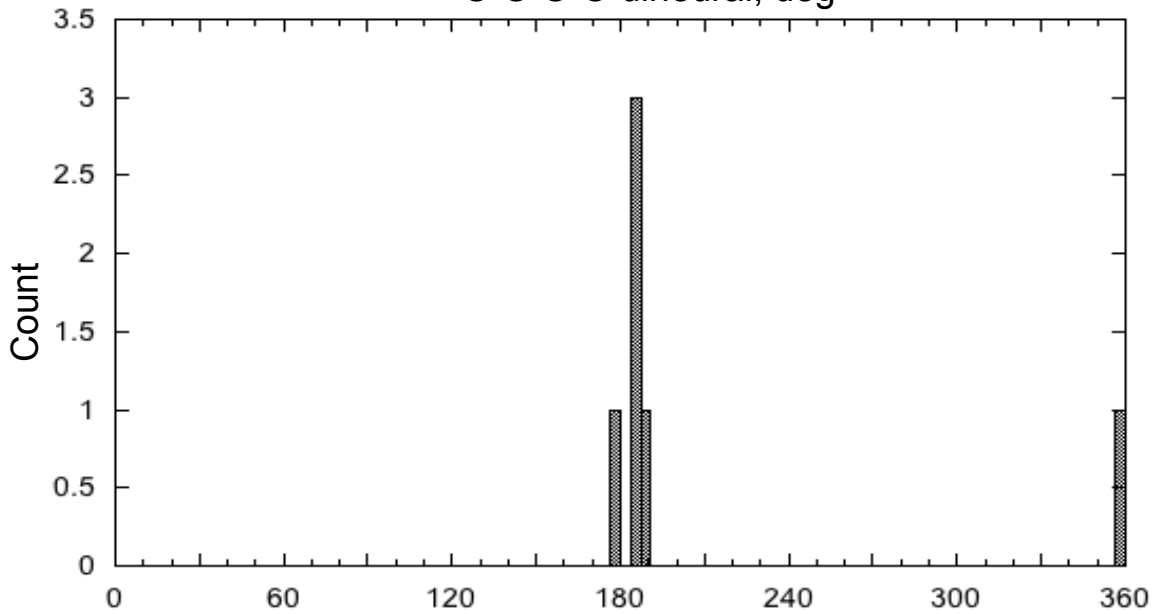


5 hits

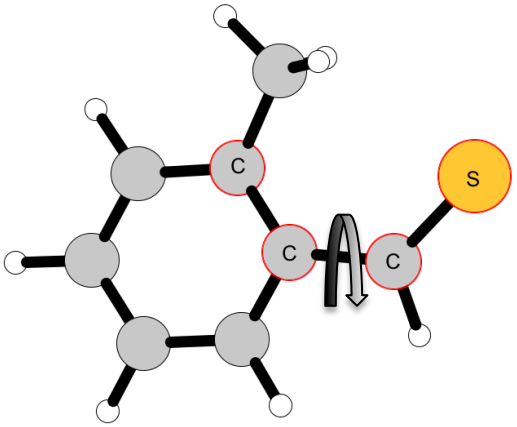
C–N = 1.47 ± 0.01 Å

TYPE 3-2:10-1

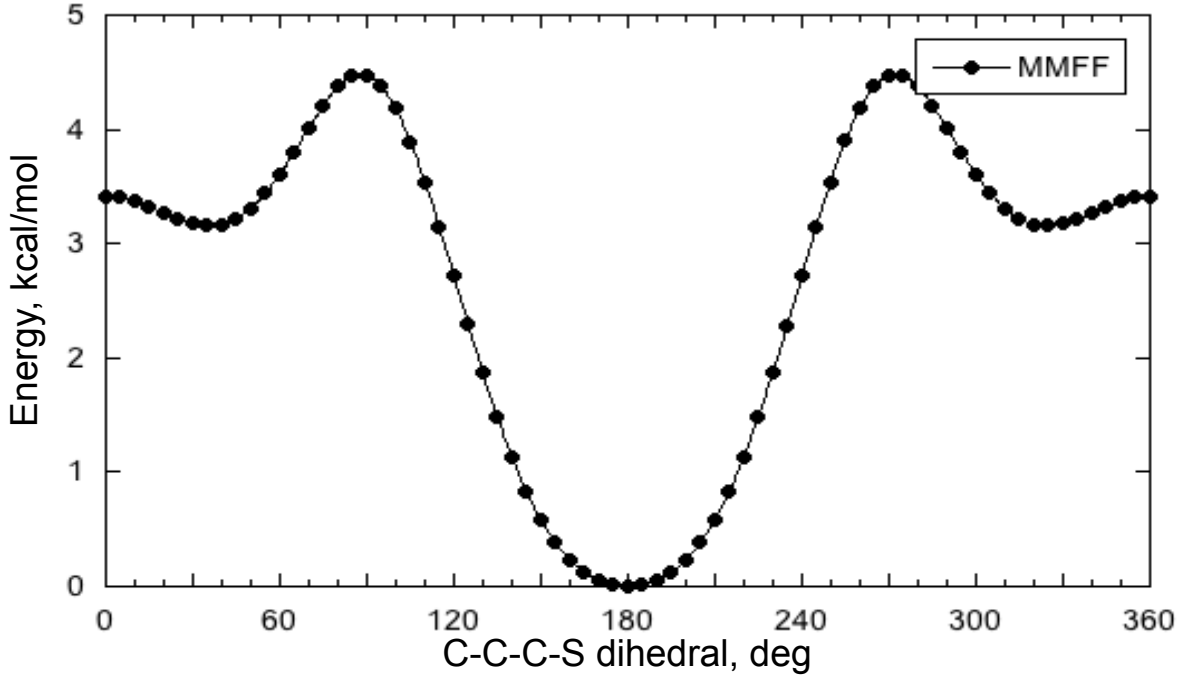




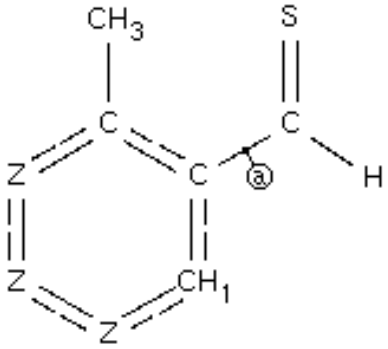
TYPE 3-2:10-2



Φ	E
35.0	3.16
180.0	0.00



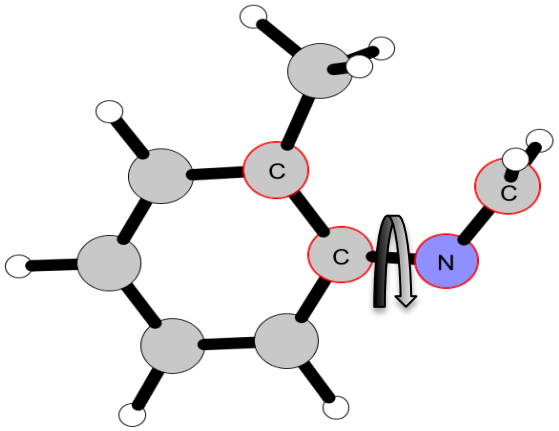
CSD comparison



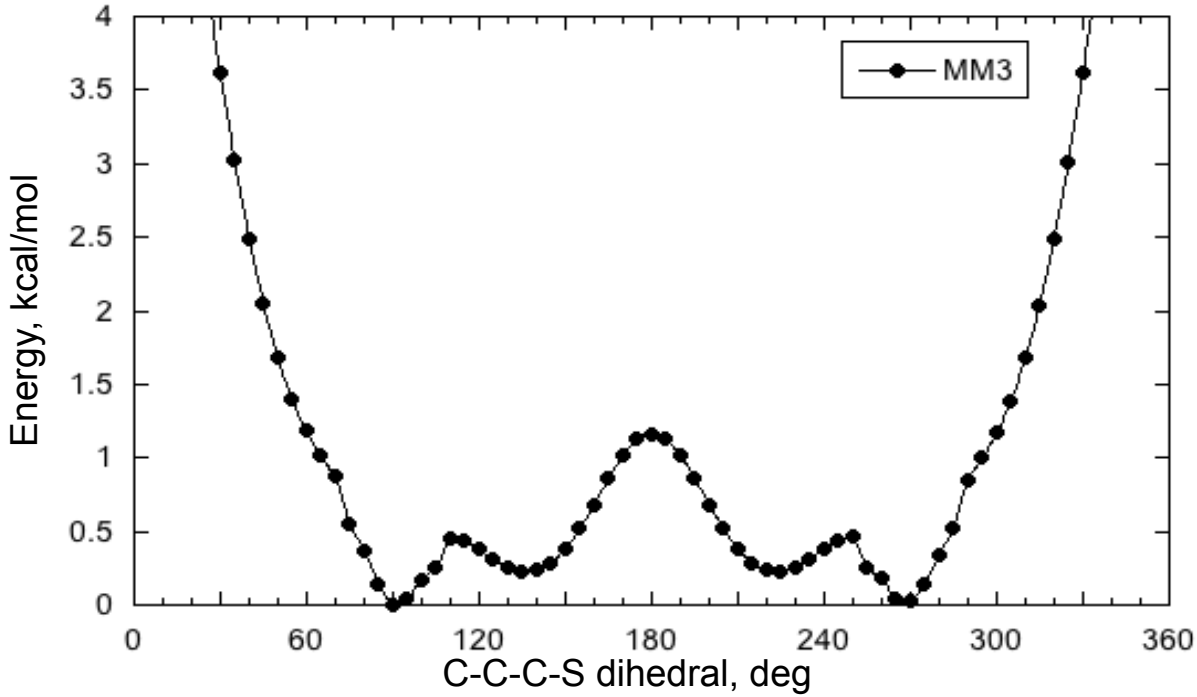
0 hits

NO CSD HITS

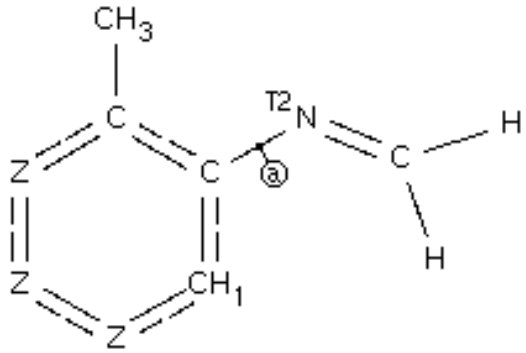
TYPE 3-2:10-3



Φ	E
90.0	0.00
135.0	0.23
225.0	0.23
270.0	0.03



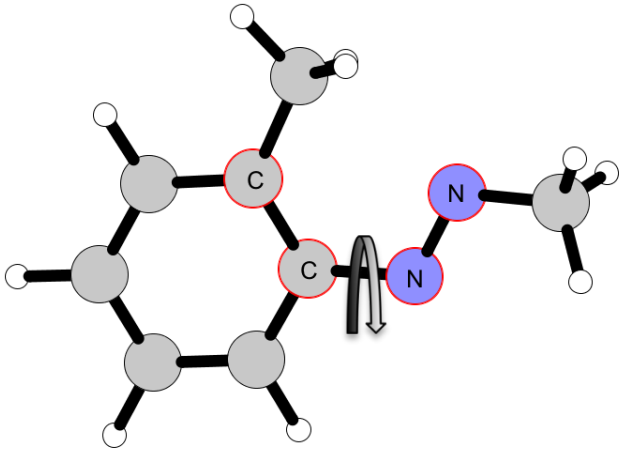
CSD comparison



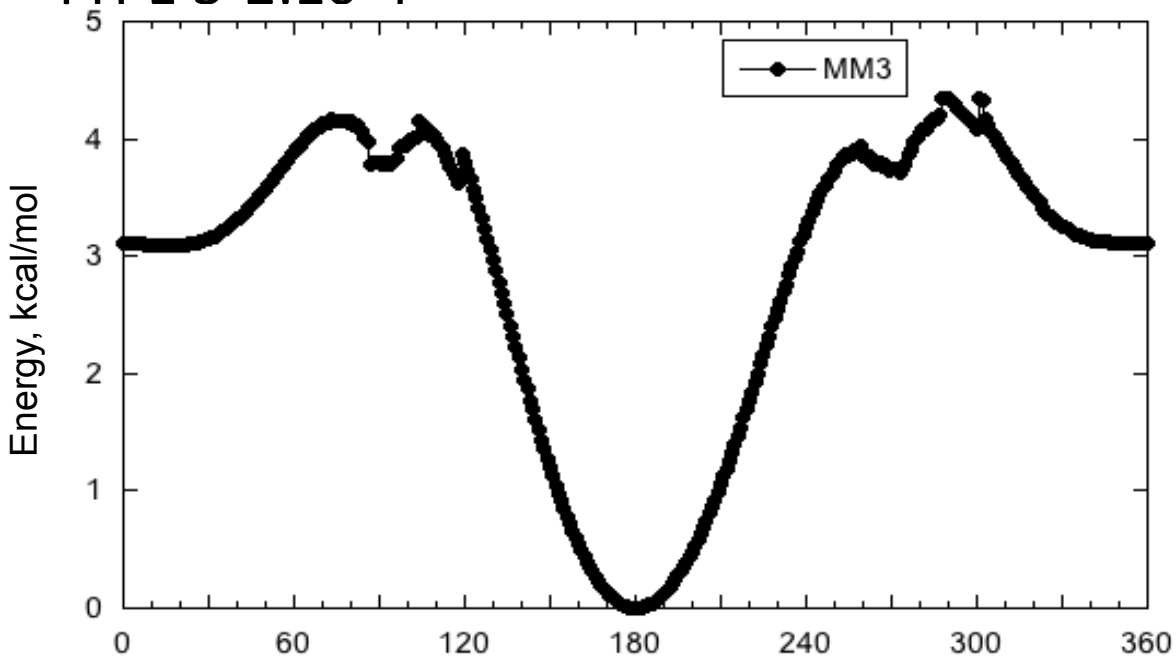
0 hits

NO CSD HITS

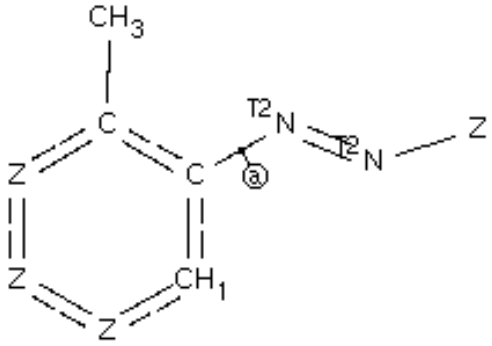
TYPE 3-2:10-4



Φ	E
94.0	3.78
180.0	0.00
269.0	3.74

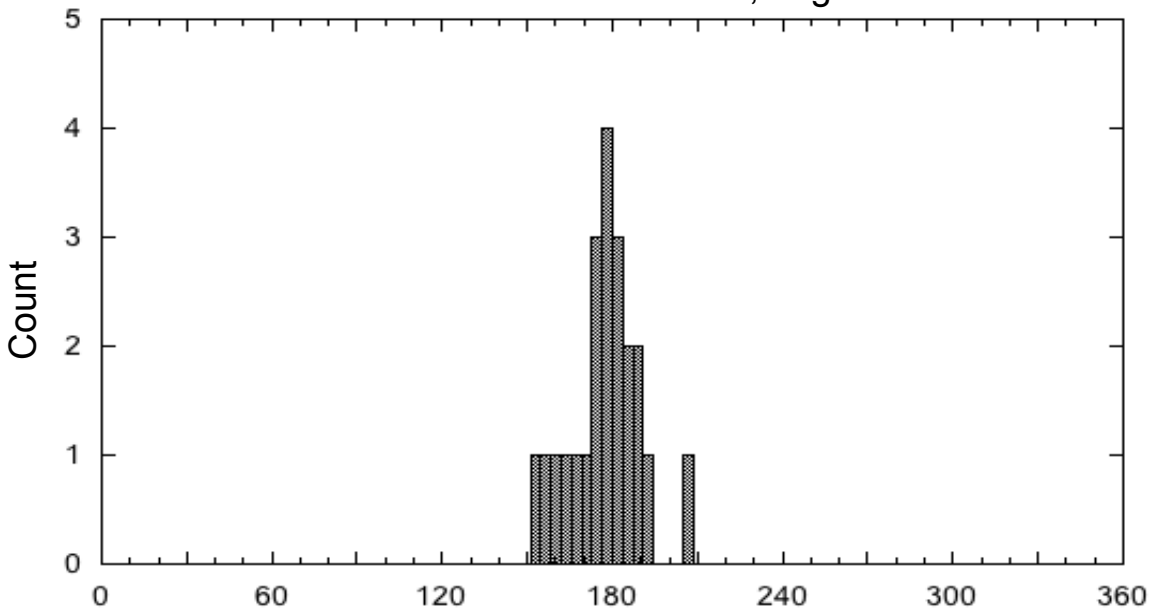


CSD comparison

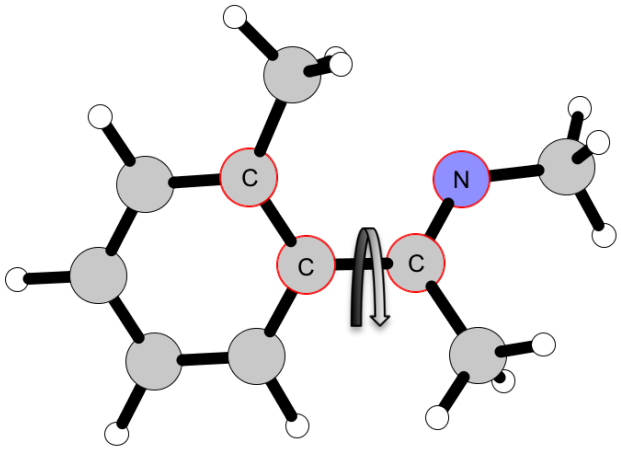


19 hits

C–N = 1.42 ± 0.02 Å

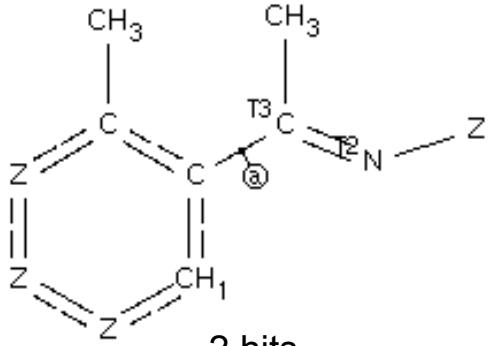


TYPE 3-2:11-1



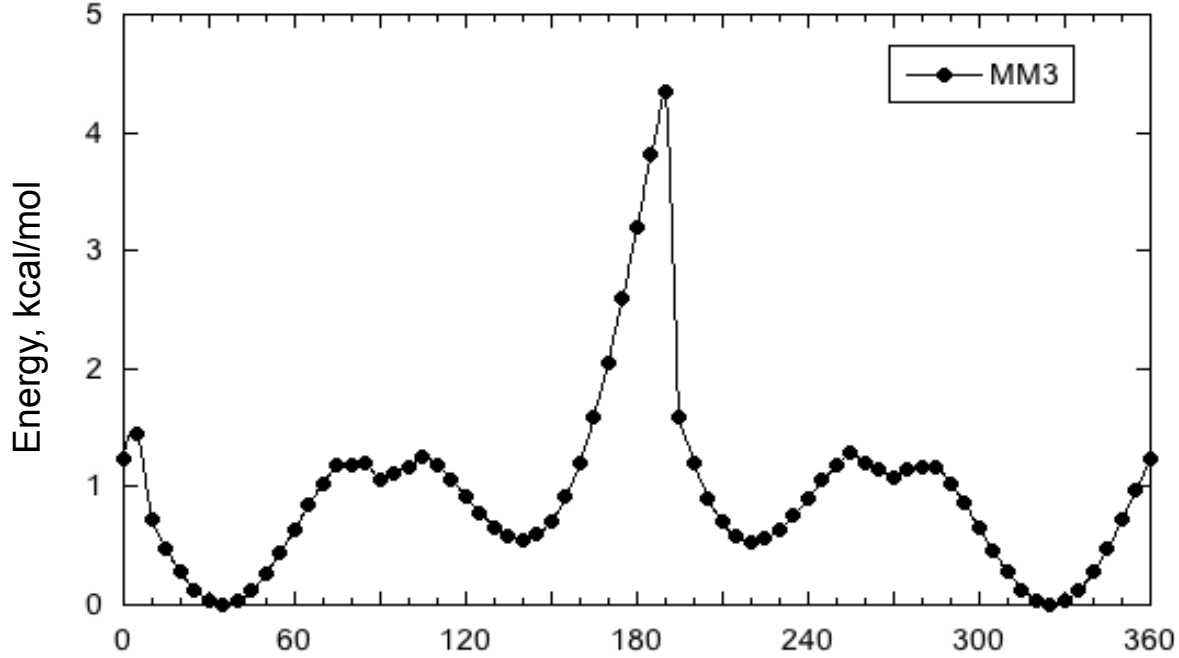
Φ	E
35.0	0.00
90.0	1.06
140.0	0.55
220.0	0.53
270.0	1.07
325.0	0.04

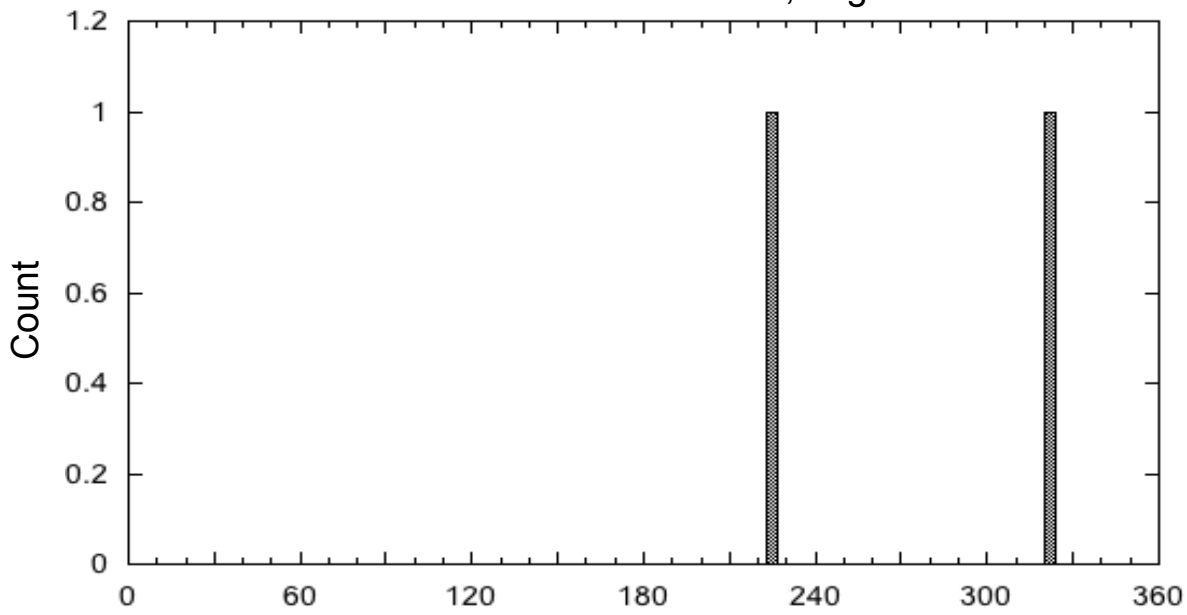
CSD comparison

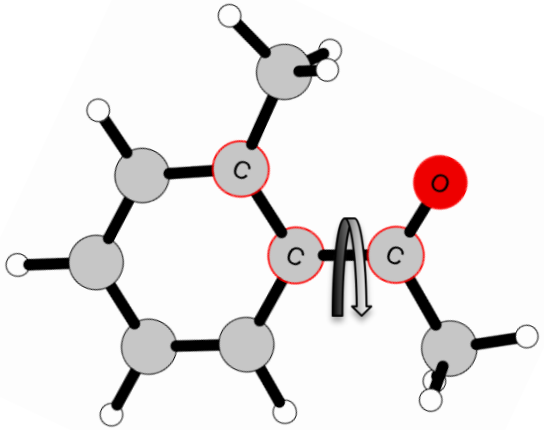


2 hits

C–C = 1.48 ± 0.01 Å



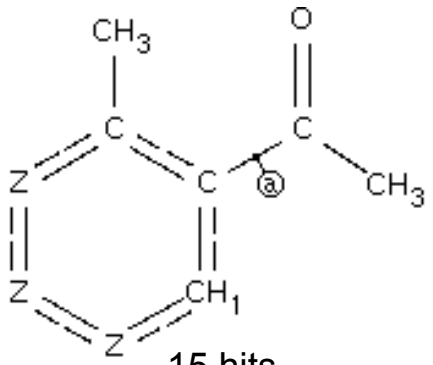




TYPE 3-2:11-2

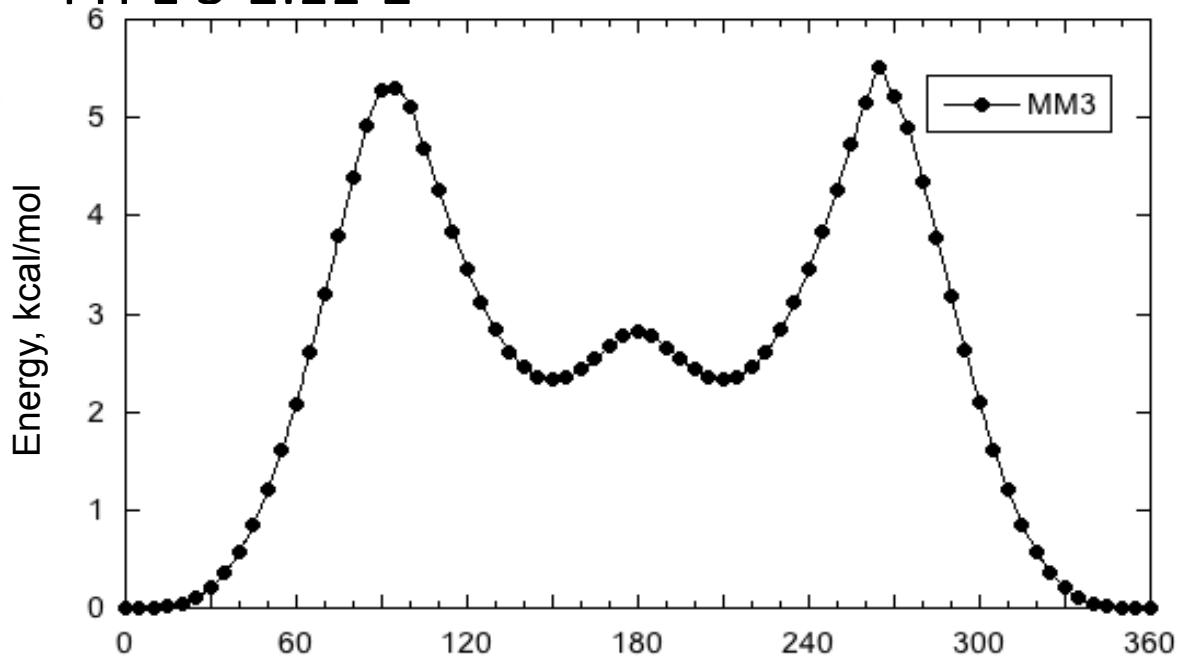
Φ	E
0.0	0.00
5.0	0.00
10.0	0.00
150.0	2.33
210.0	2.33
350.0	0.00
355.0	0.00

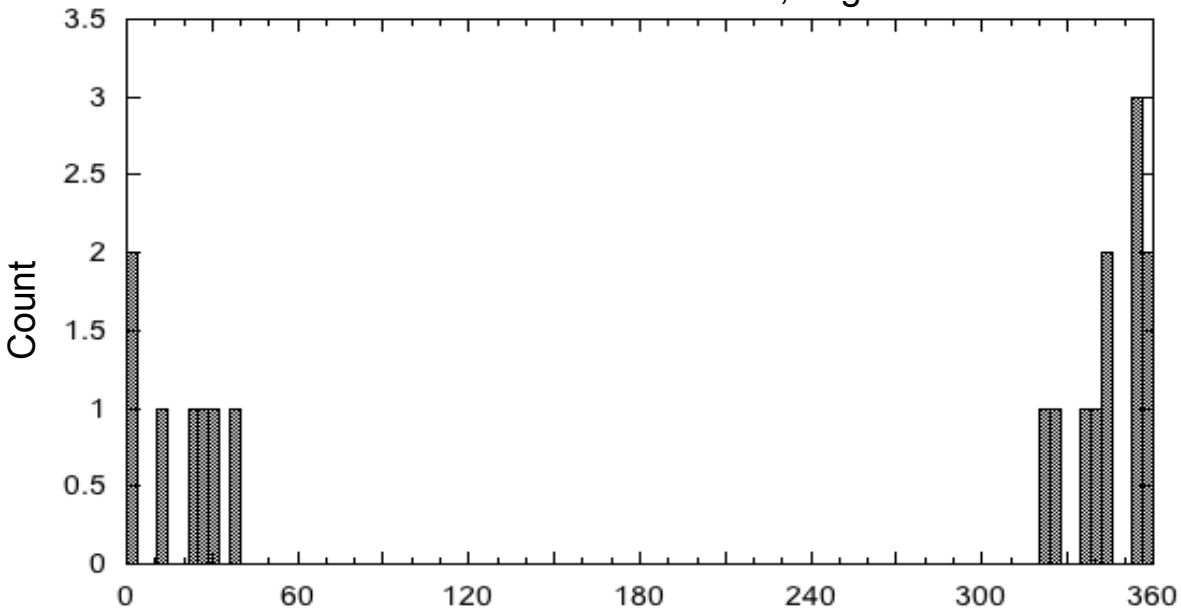
CSD comparison



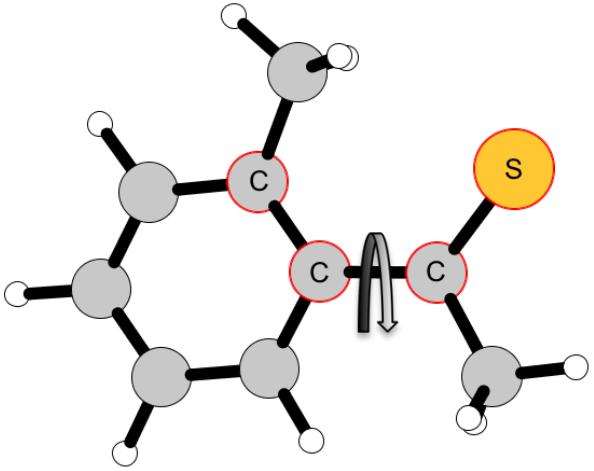
15 hits

C–C = 1.49 ± 0.01 Å

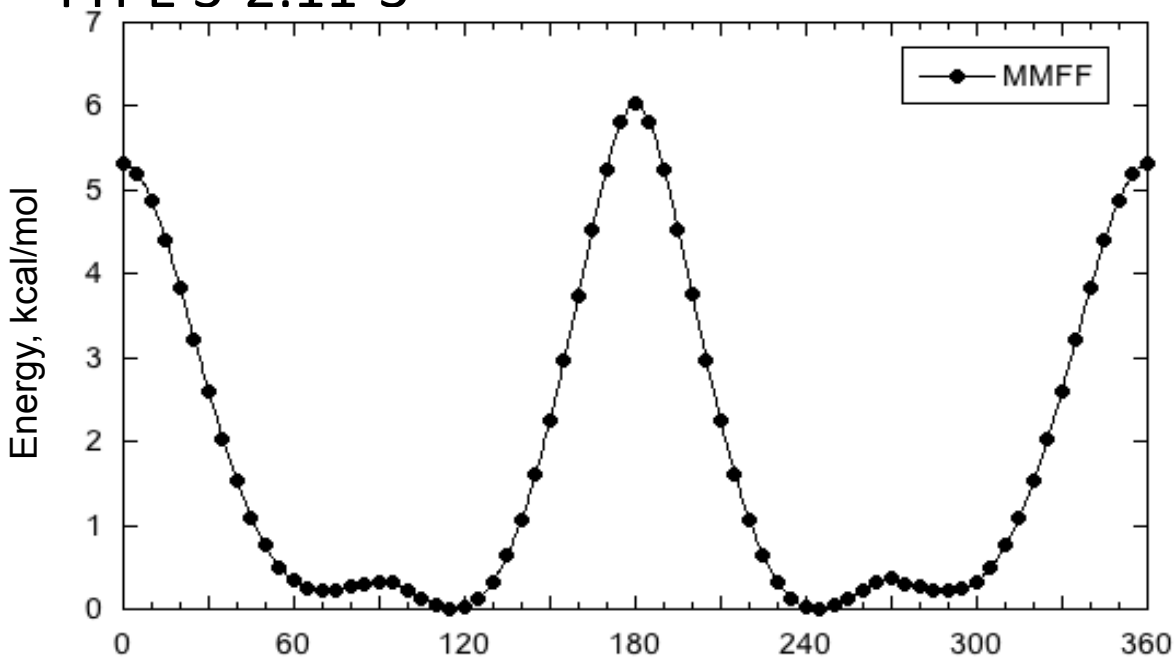




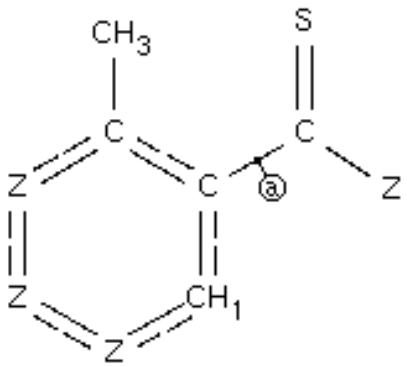
TYPE 3-2:11-3



Φ	E
70.0	0.22
115.0	0.00
245.0	0.00
290.0	0.22

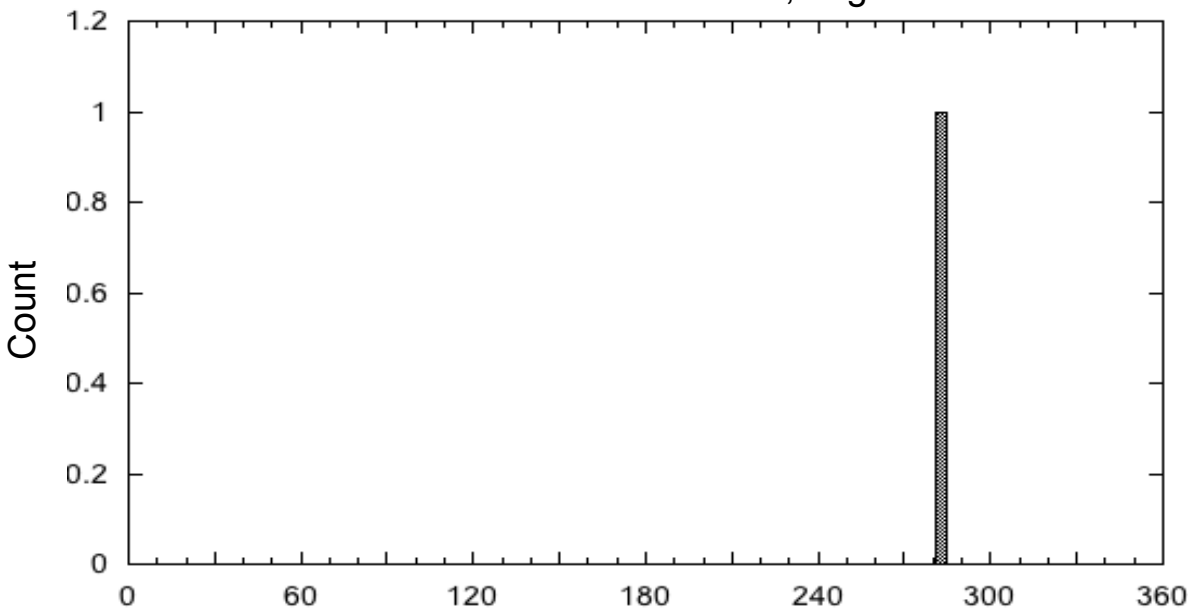


CSD comparison

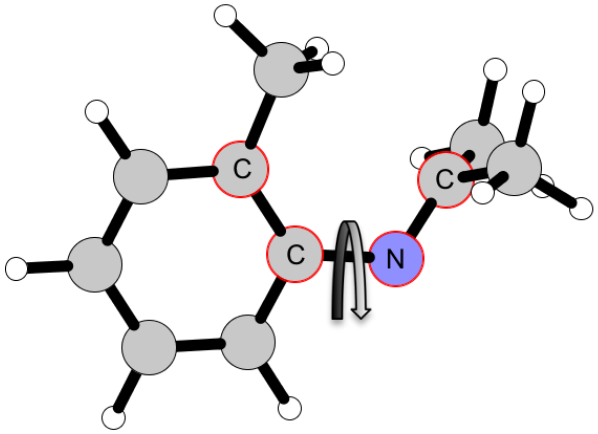


1 hit

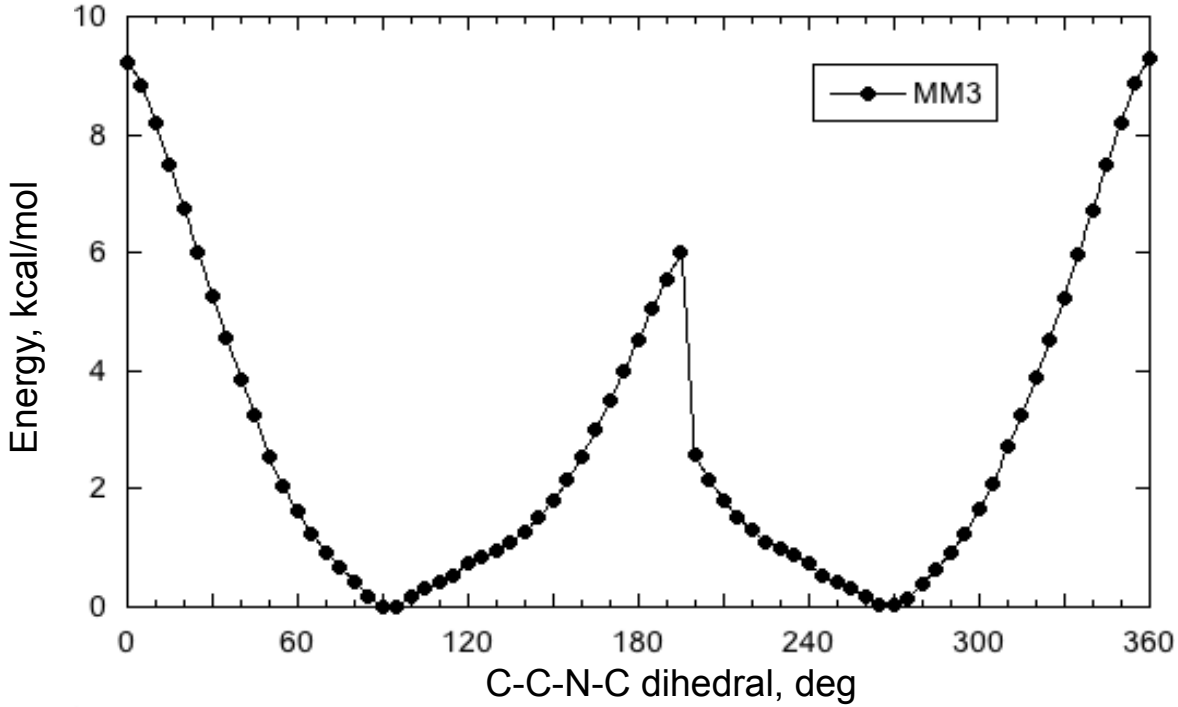
C–C = 1.50 Å



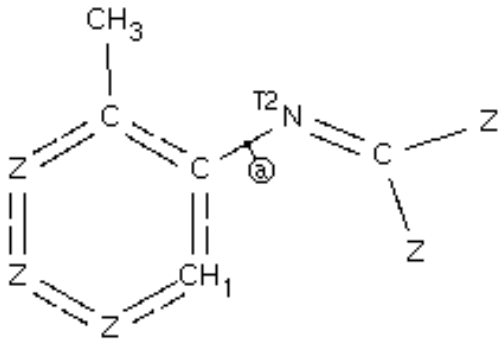
TYPE 3-2:12-1



Φ	E
95.0	0.00
265.0	0.03

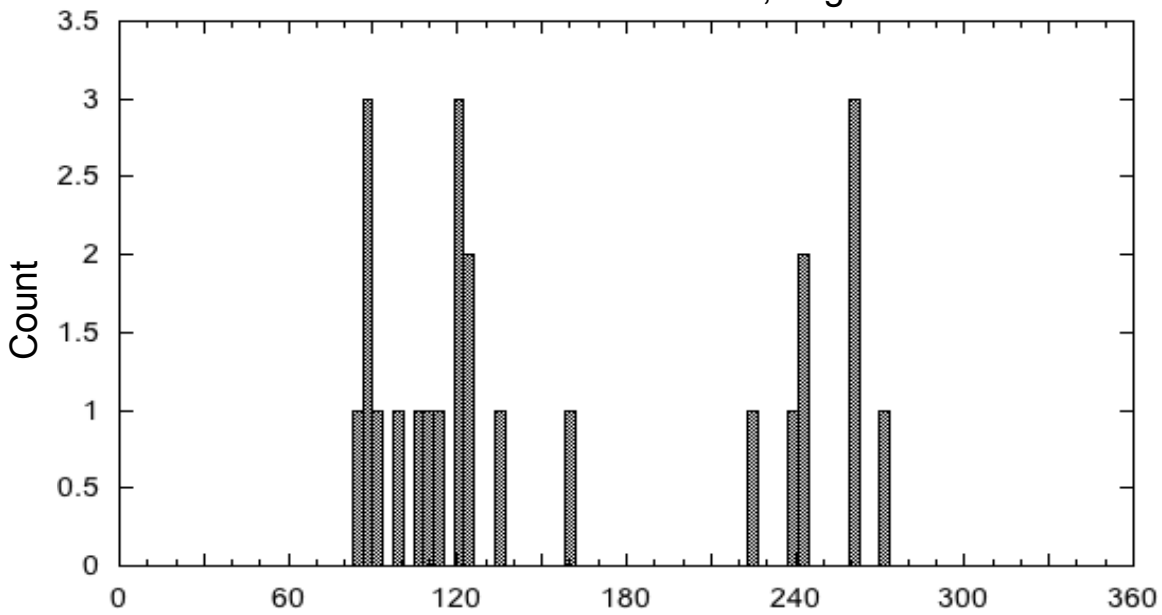


CSD comparison

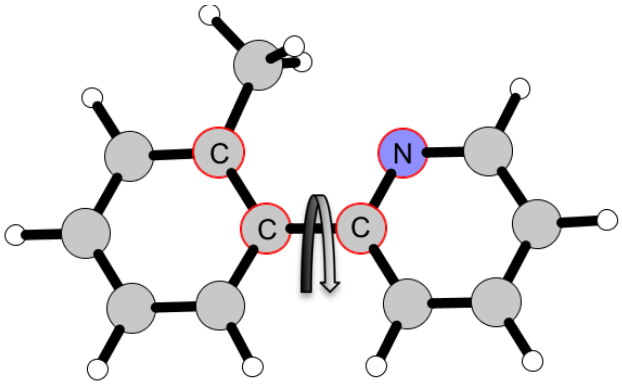


17 hits

C–N = 1.42 ± 0.01 Å

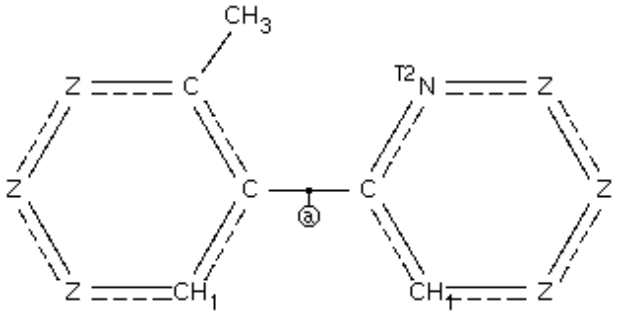


TYPE 3-2:13-1



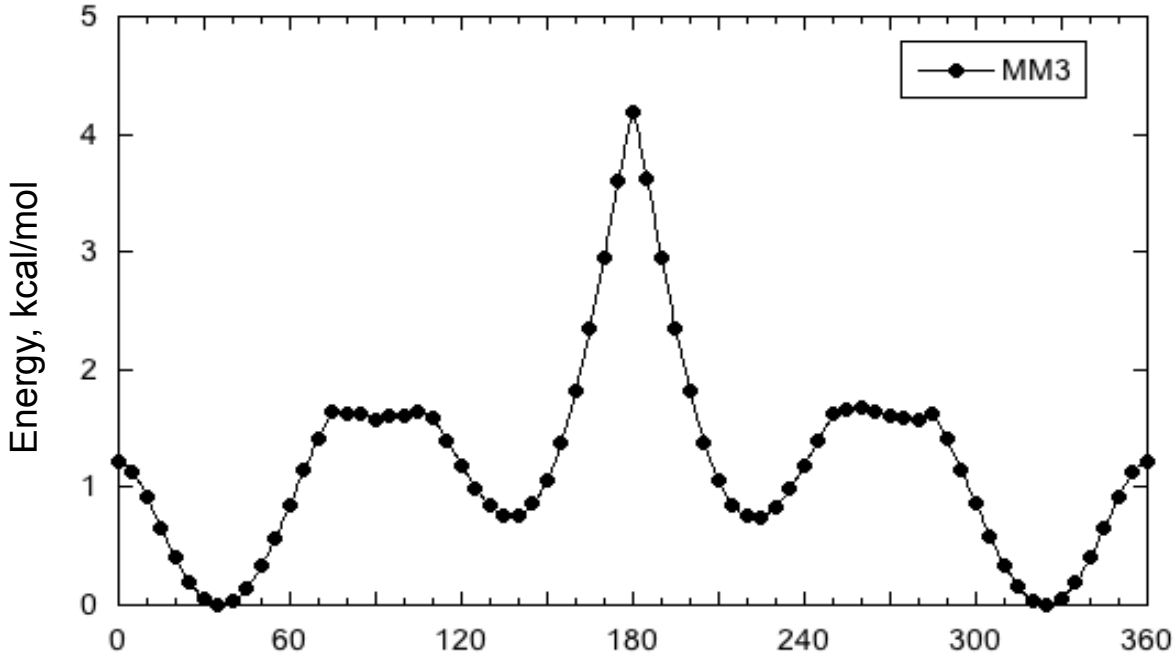
Φ	E
35.0	0.00
135.0	0.76
140.0	0.76
220.0	0.75
225.0	0.75
325.0	0.01

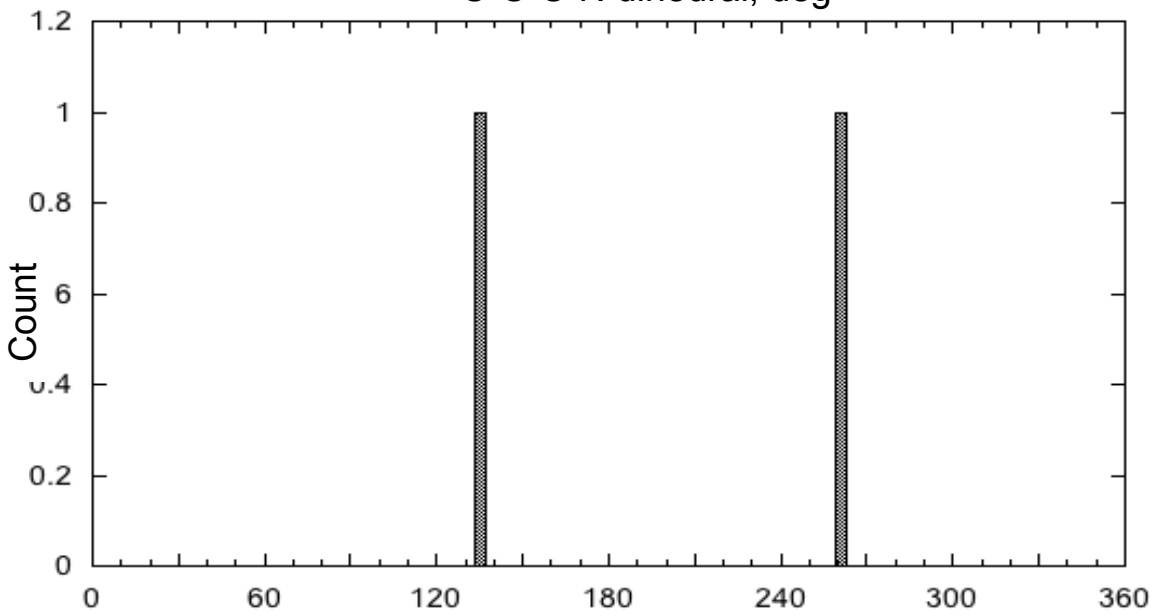
CSD comparison

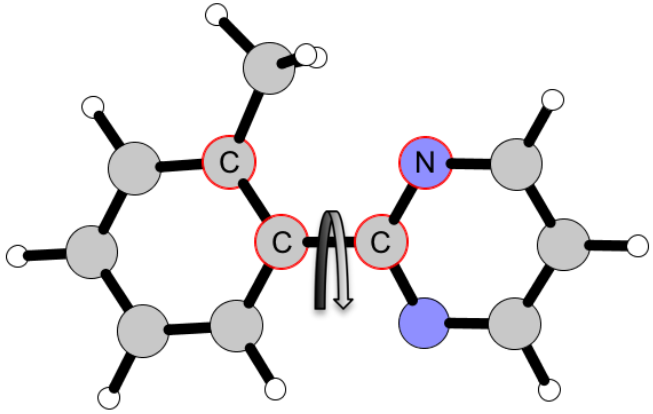


2 hits

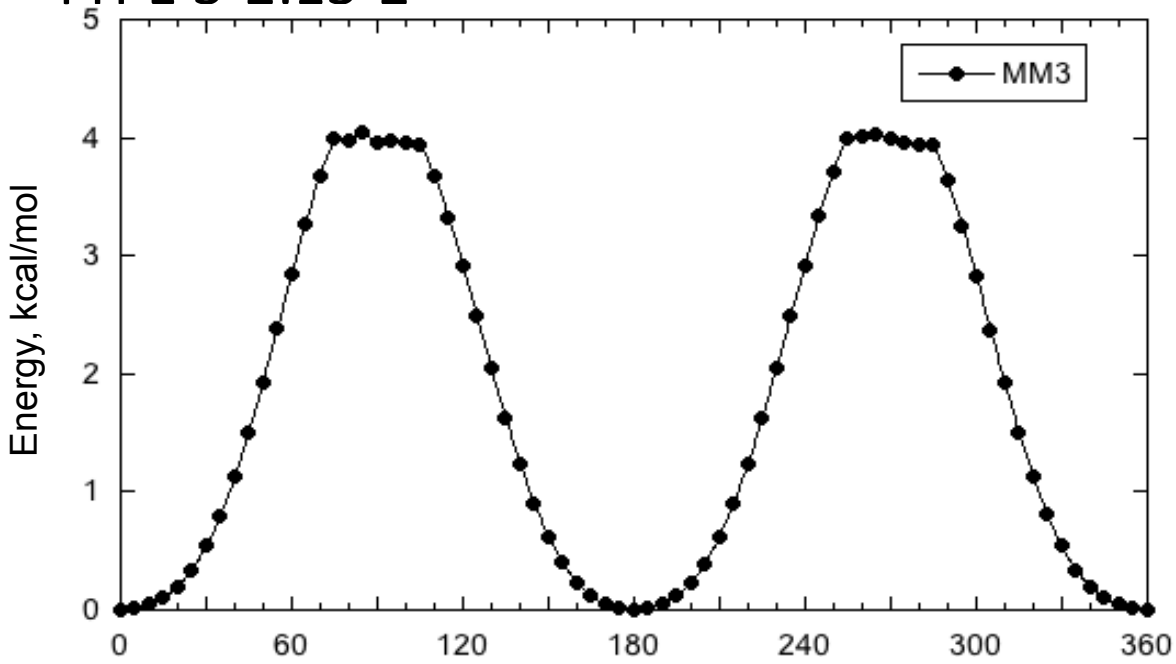
C–C = 1.49 ± 0.02 Å





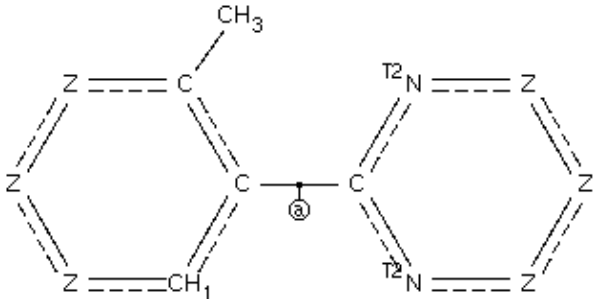


TYPE 3-2:13-2



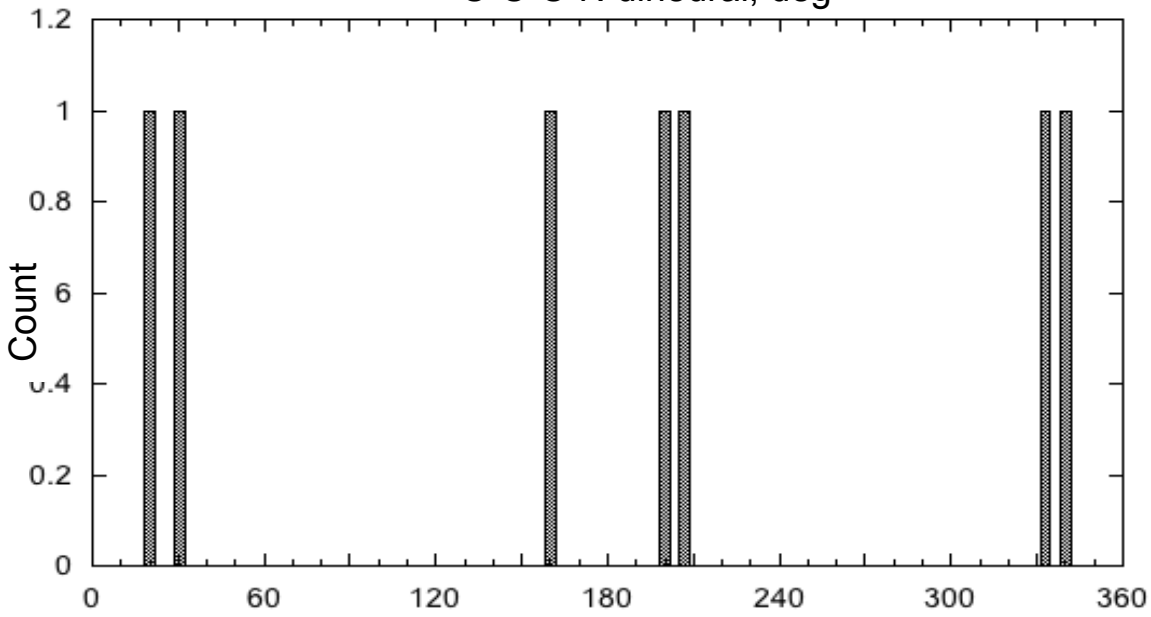
Φ	E
0.00	0.00
180.0	0.01

CSD comparison

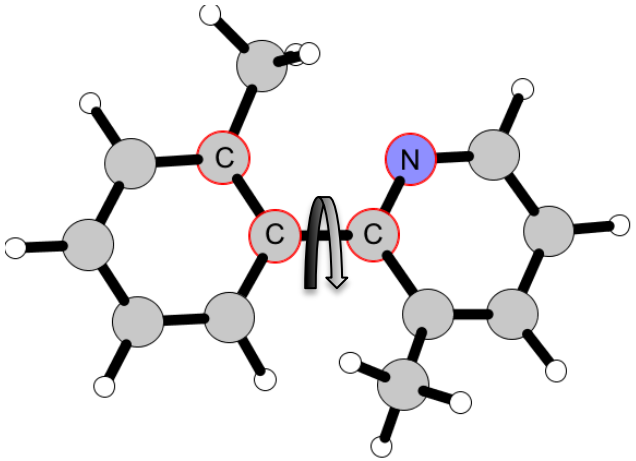


2 hits

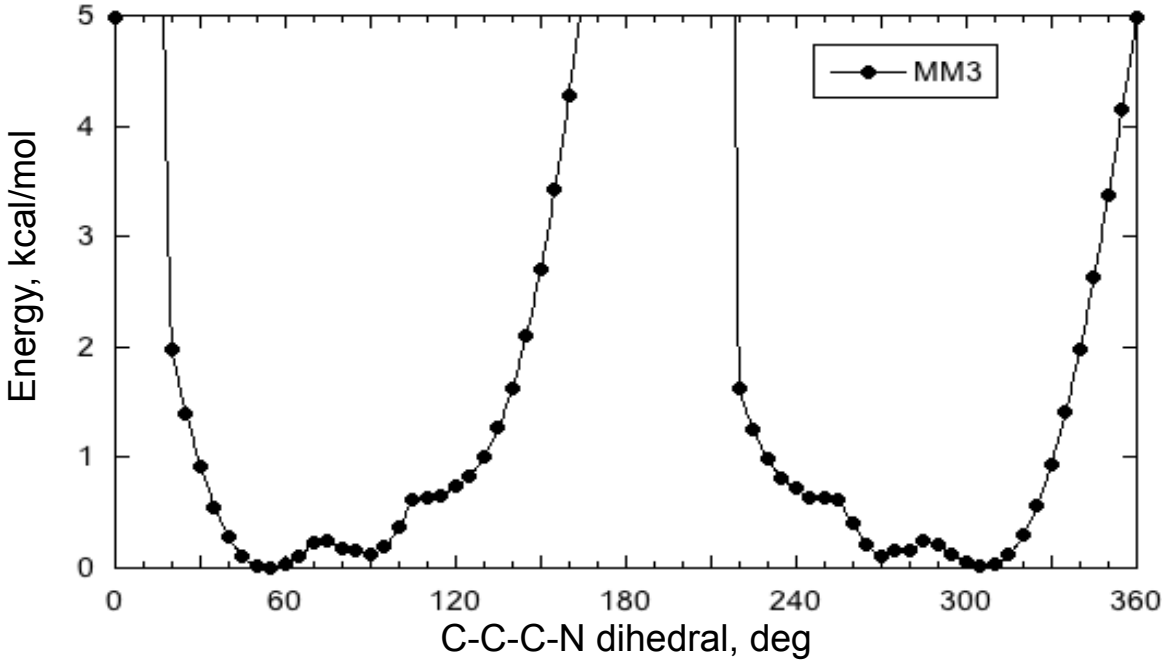
C–N = 1.48 ± 0.00 Å



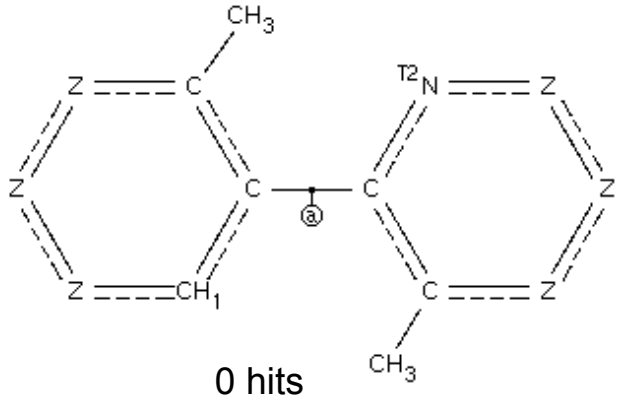
TYPE 3-2:13-3



Φ	E
55.0	0.00
90.0	0.12
270.0	0.11
305.0	0.03



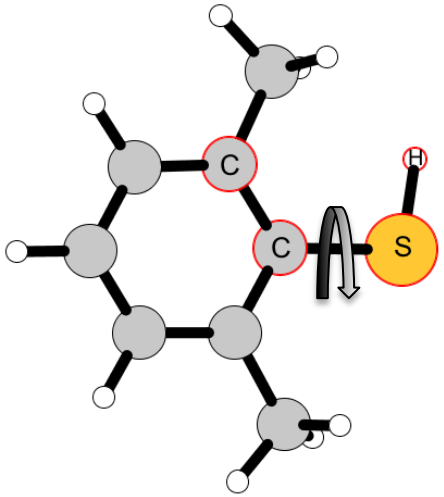
CSD comparison



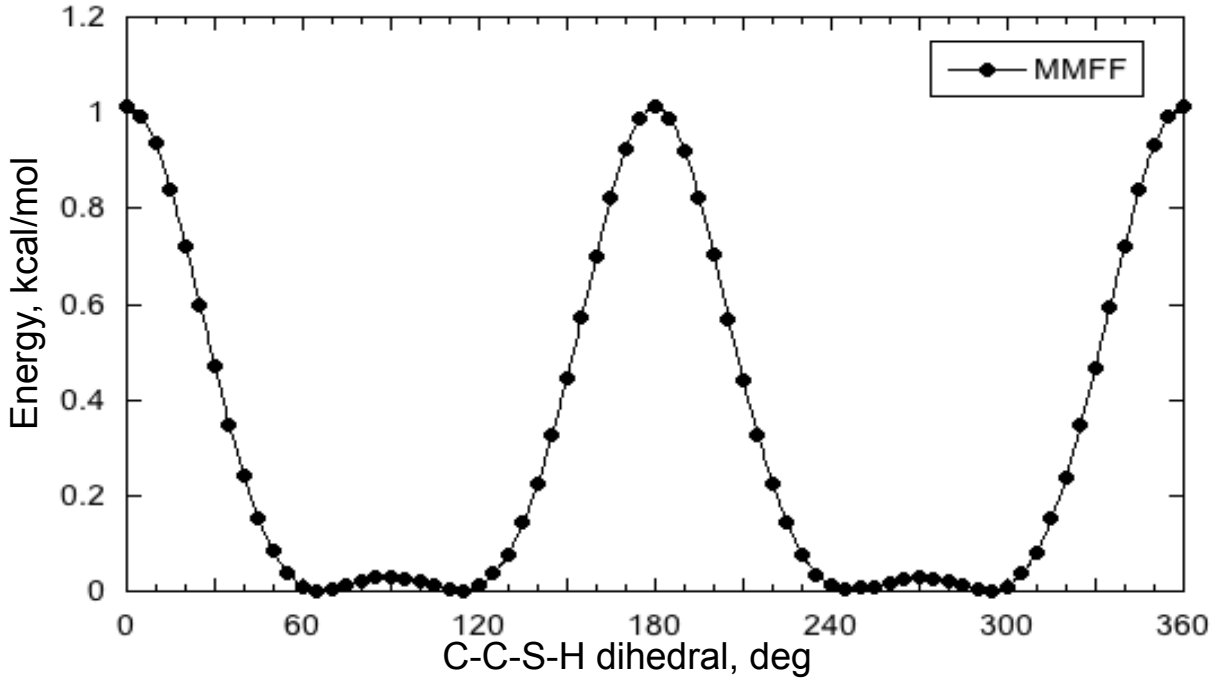
0 hits

NO CSD HITS

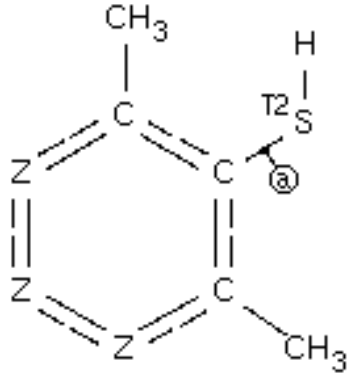
TYPE 3-3:5-4



Φ	E
65.0	0.00
115.0	0.00
245.0	0.01
295.0	0.00

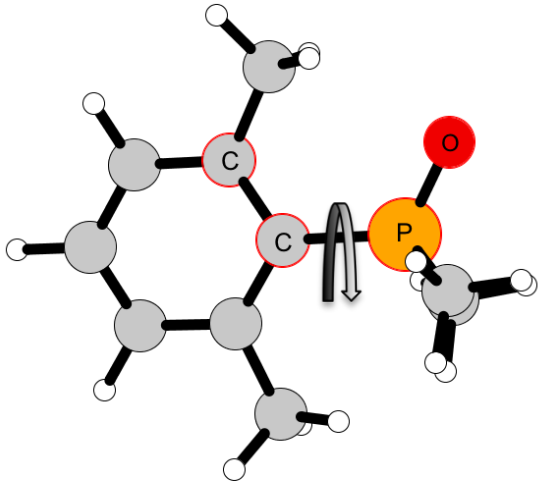


CSD comparison

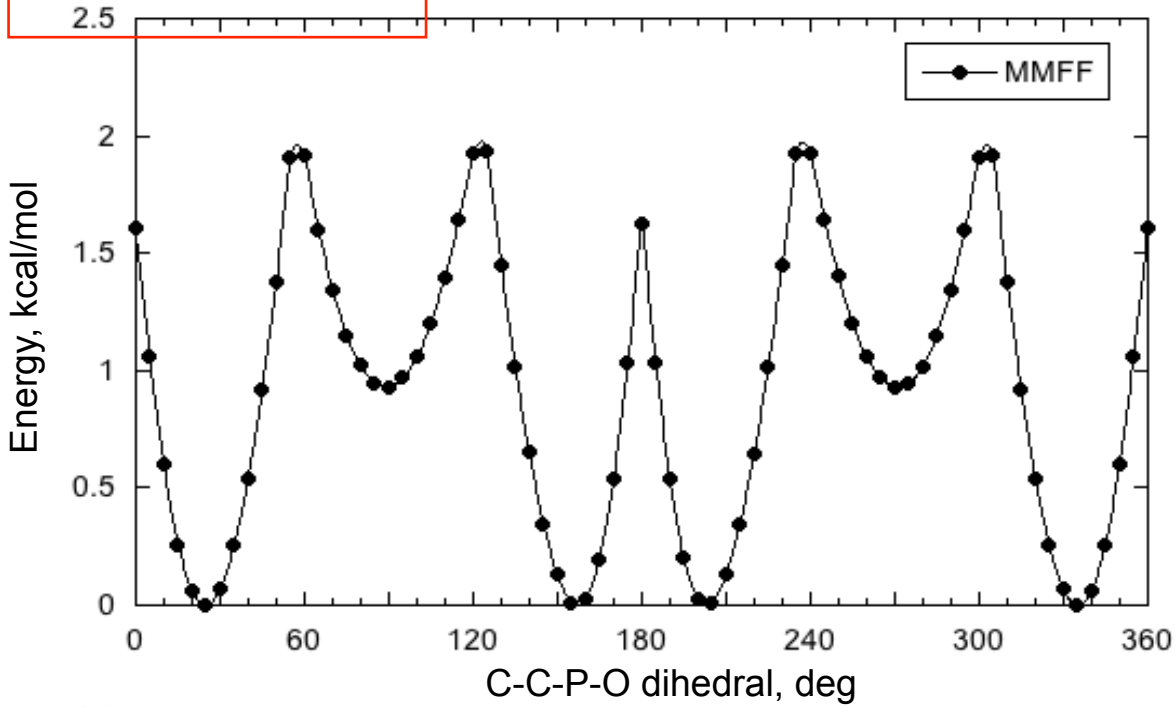


0 hits

NO CSD HITS

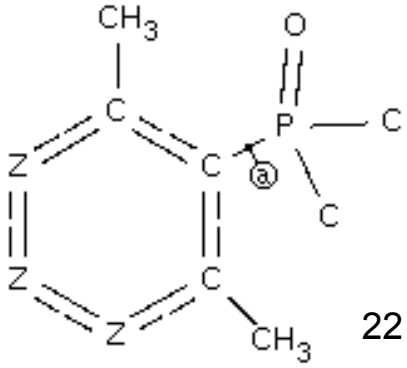


TYPE 3-3:8-1



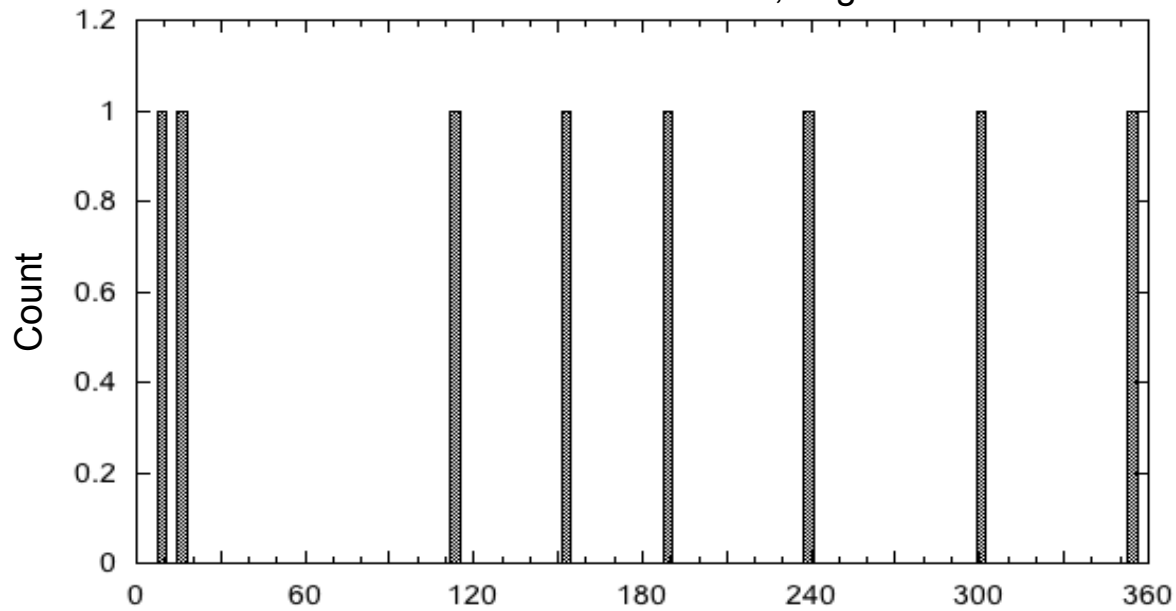
Φ	E
25.0	0.00
90.0	0.93
160.0	0.03
205.0	0.01
270.0	0.93
335.0	0.00

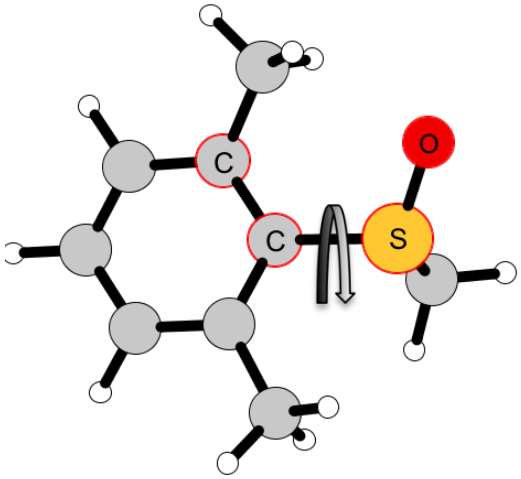
CSD comparison



22 hits

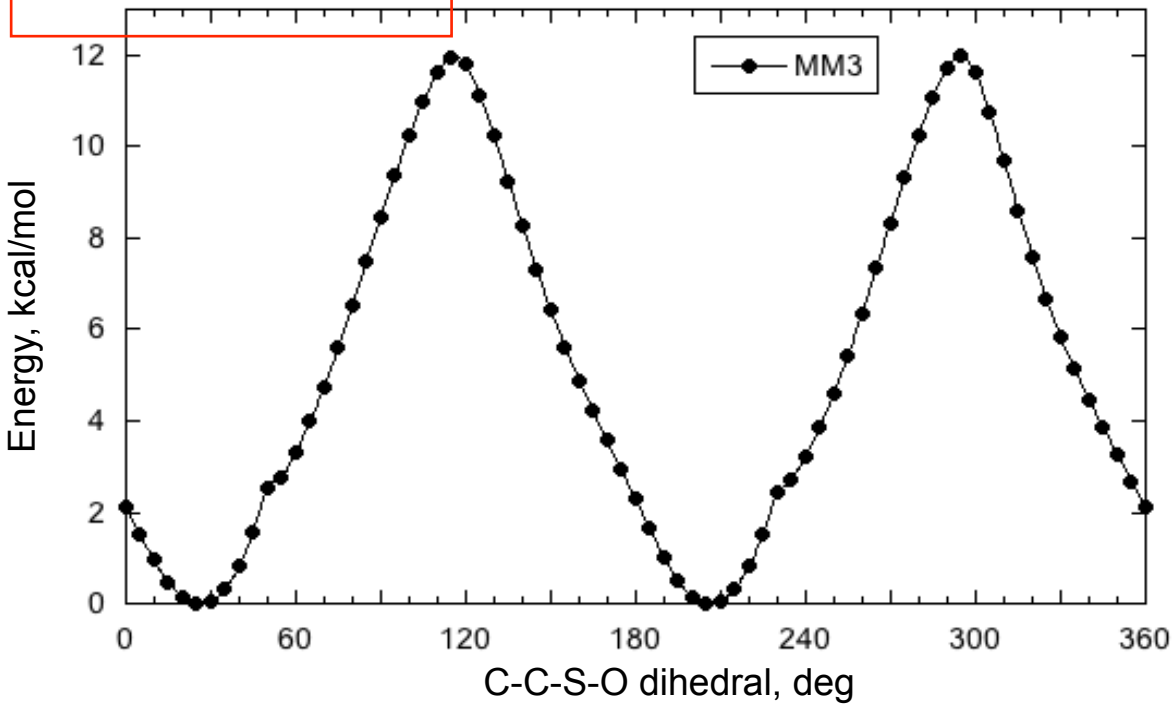
C–P = 1.82 ± 0.01 Å



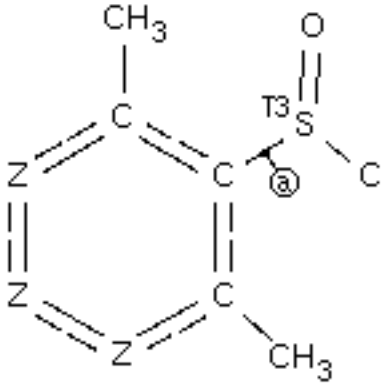


Φ	E
25.0	0.00
205.0	0.00

TYPE 3-3:8-2

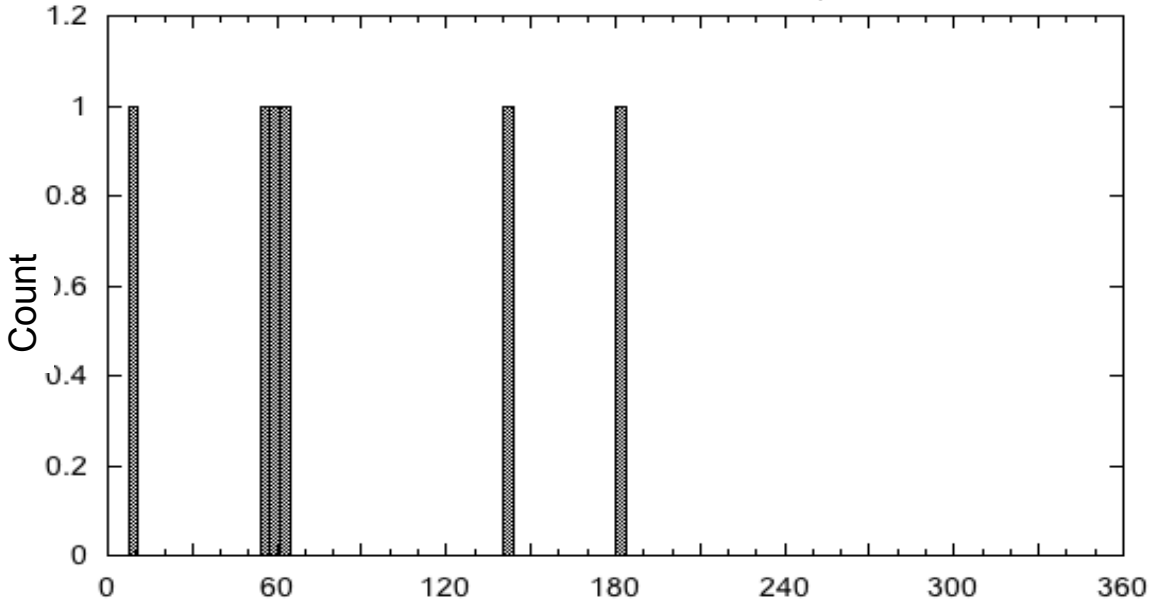


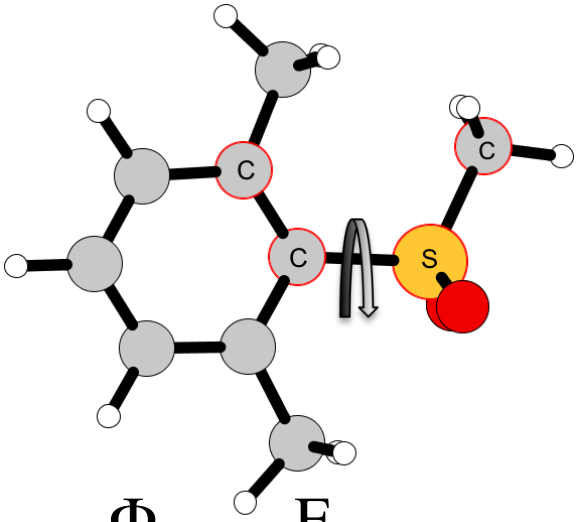
CSD comparison



3 hits

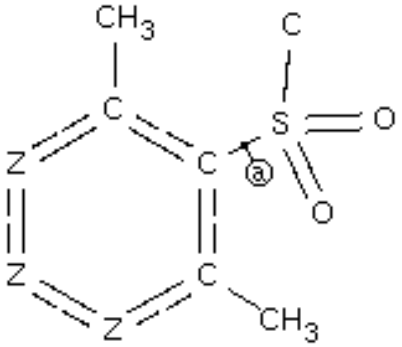
C–S = 1.81 ± 0.01 Å





Φ	E
45.0	1.12
90.0	0.00
135.0	1.12
225.0	1.12
270.0	0.00
315.0	1.12

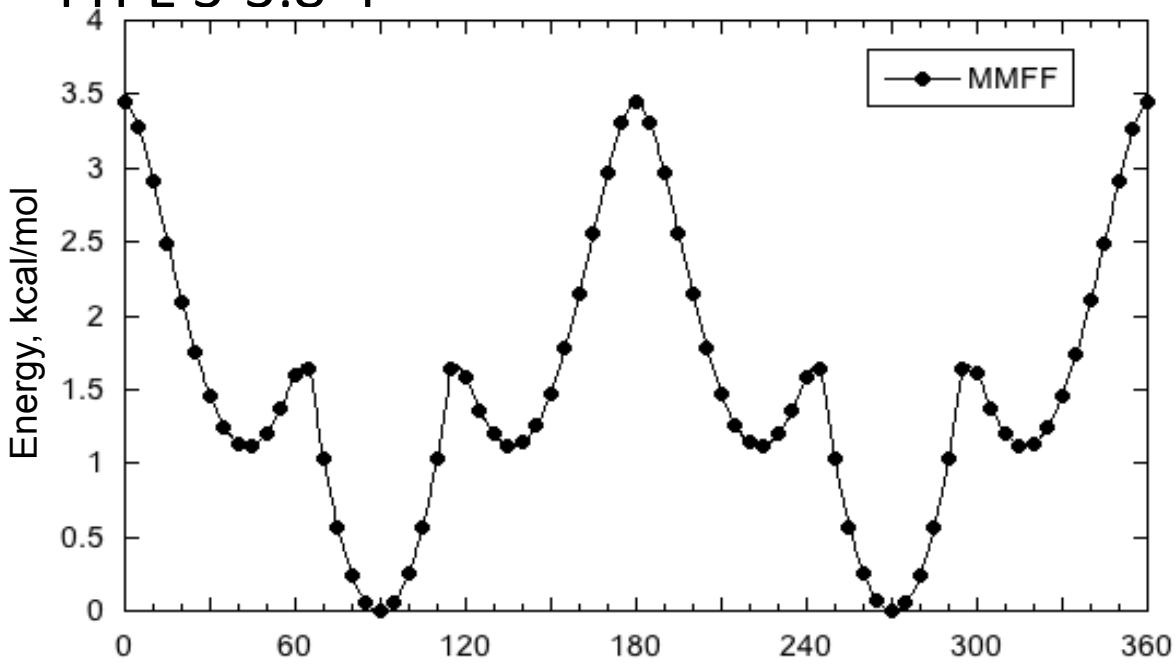
CSD comparison

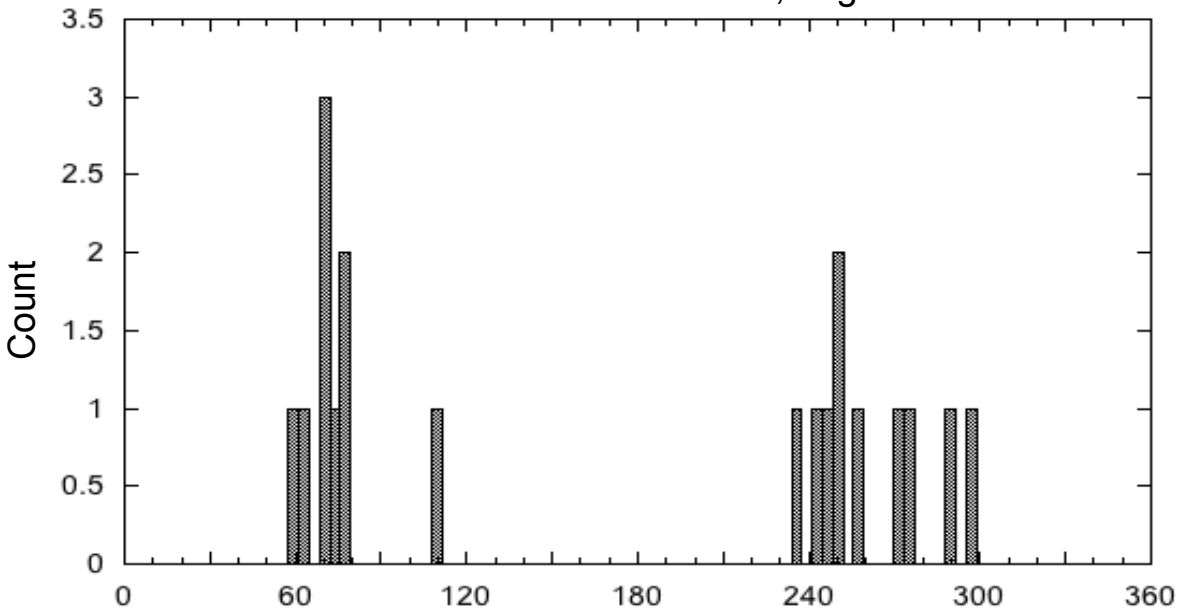


22 hits

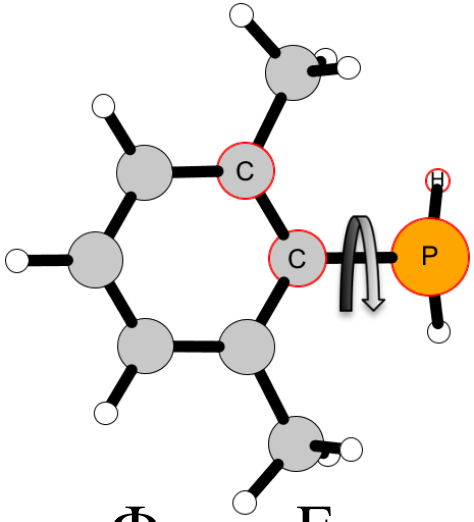
C–N = 1.78 ± 0.01 Å

TYPE 3-3:8-4

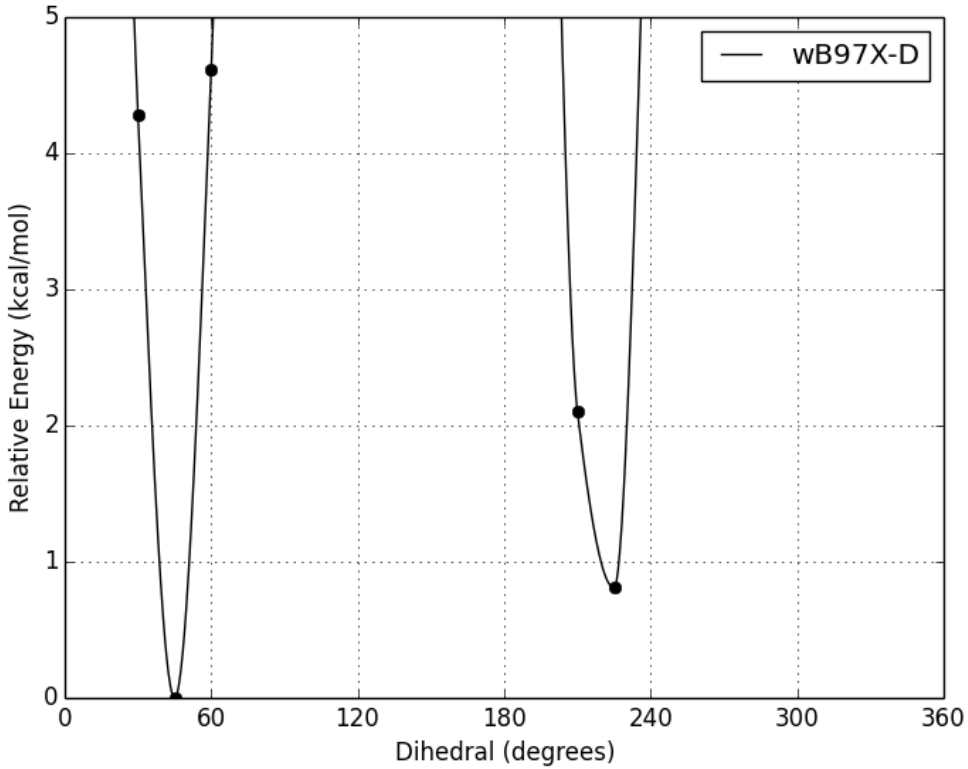




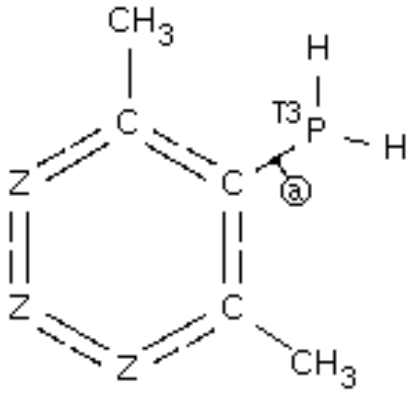
TYPE 3-3:9-1



Φ	E
45.0	0.00
225.0	0.81

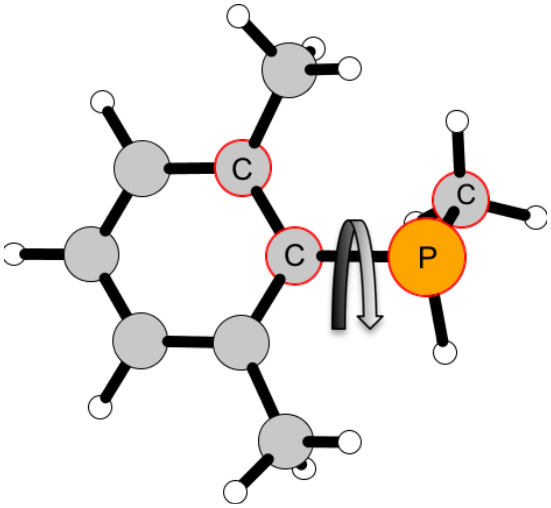


CSD comparison

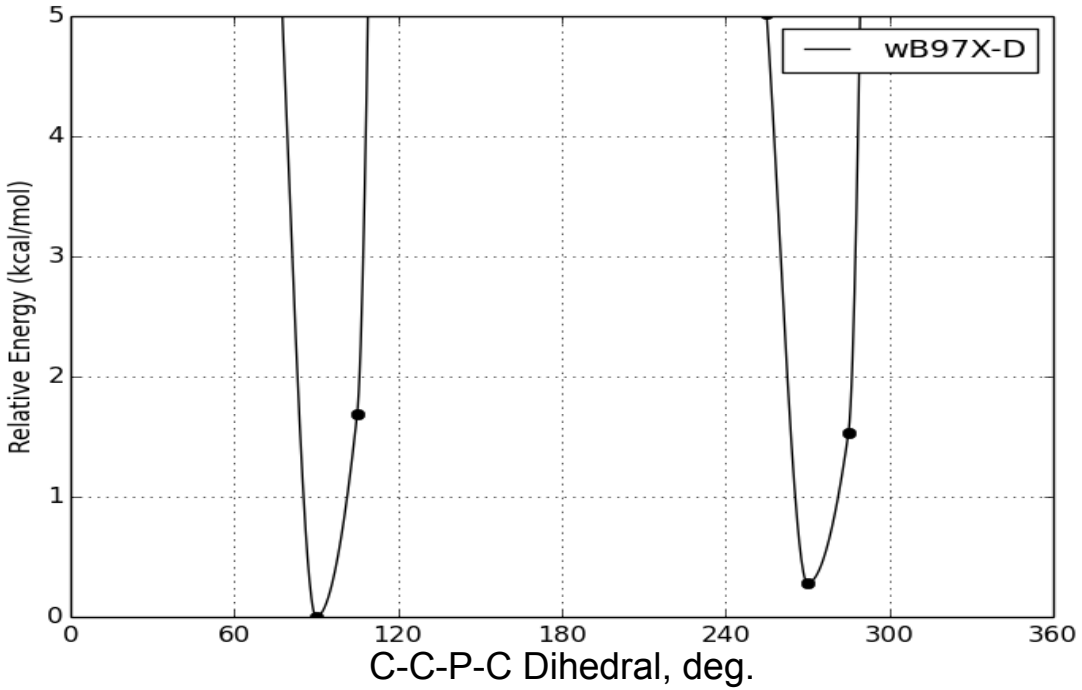


NO CSD HITS

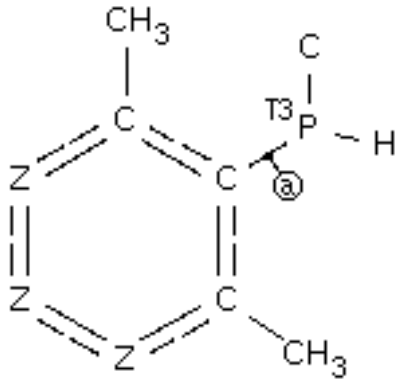
TYPE 3-3:9-2



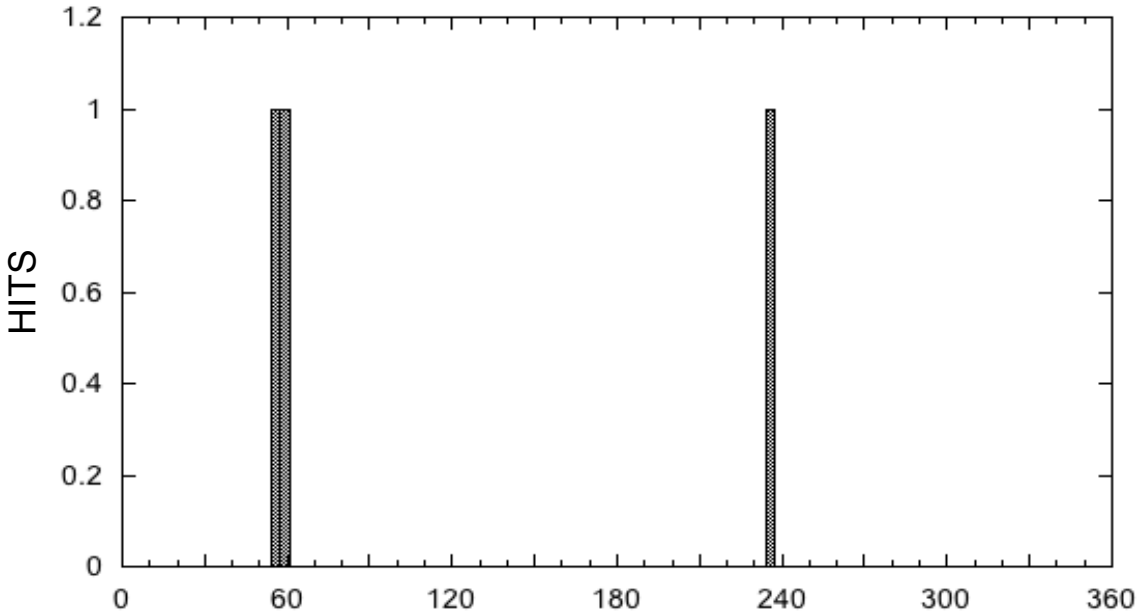
Φ	E
90.0	0.00
270.0	0.28

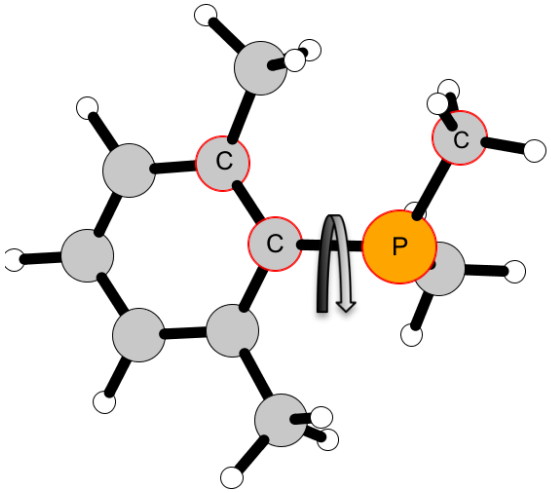


CSD comparison



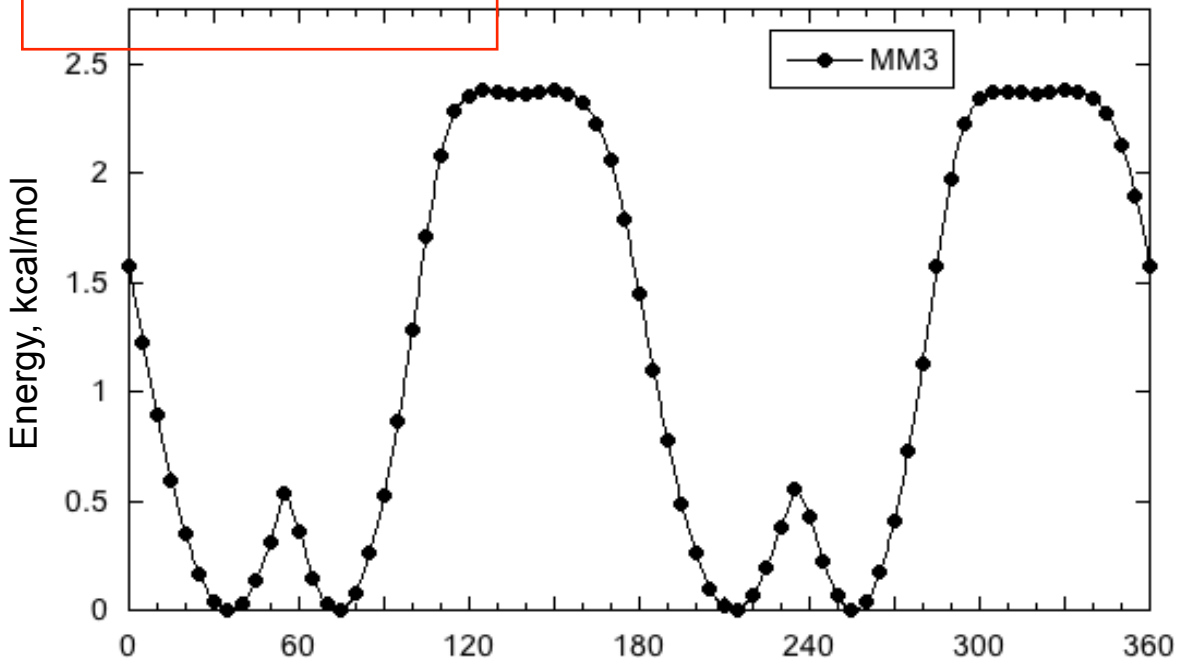
2 hits
C-P = 1.85 ± 0.01 Å



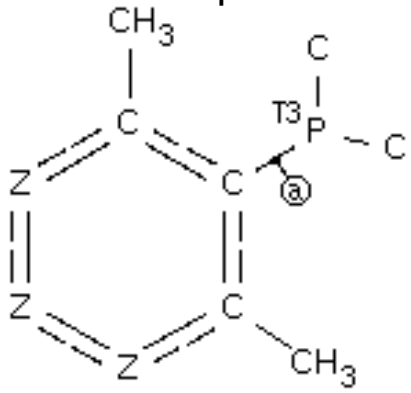


TYPE 3-3:9-4

Φ	E
35.0	0.00
75.0	0.00
215.0	0.00
255.0	0.00

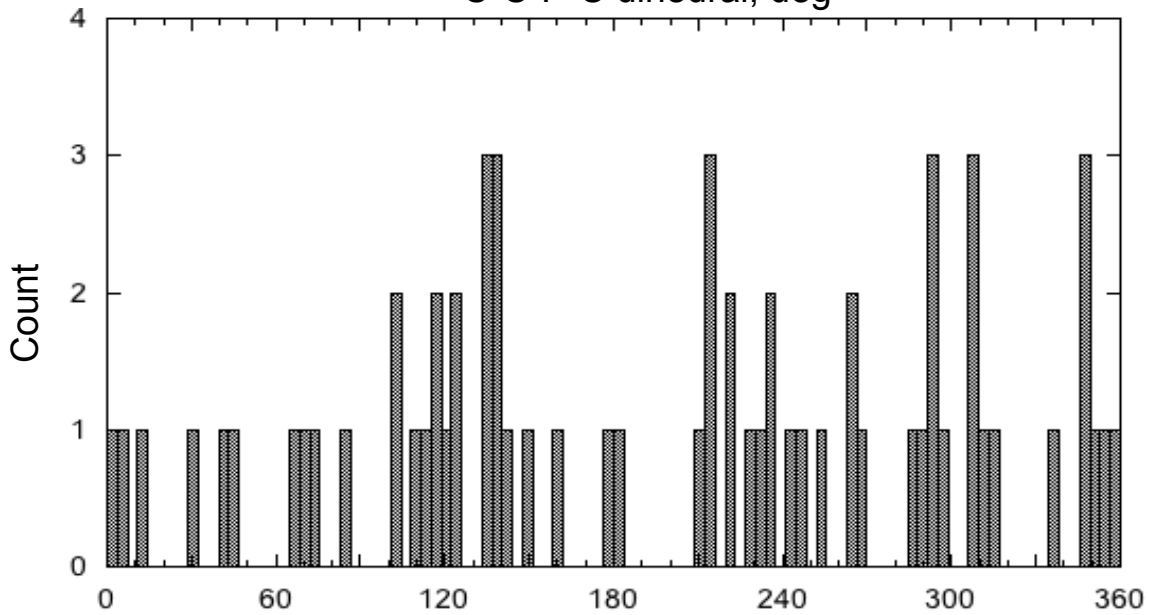


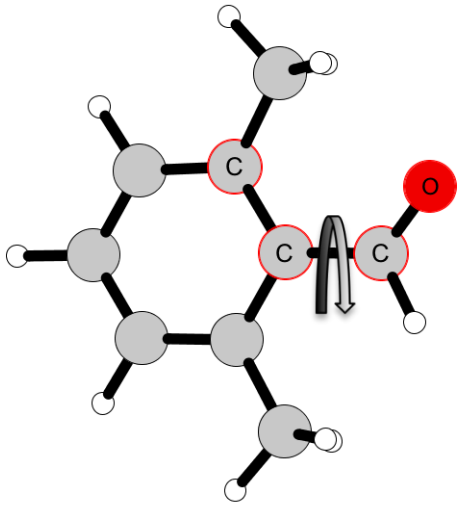
CSD comparison



33 hits

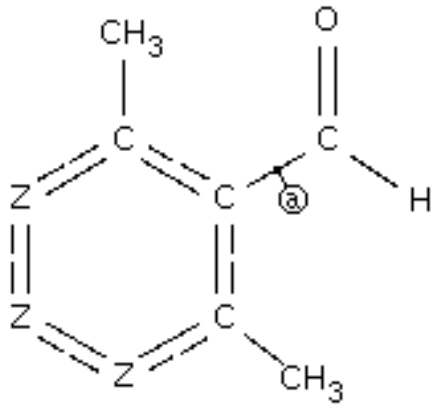
C–P = 1.85 ± 0.01 Å





Φ	E
0.0	0.00
180.0	0.00

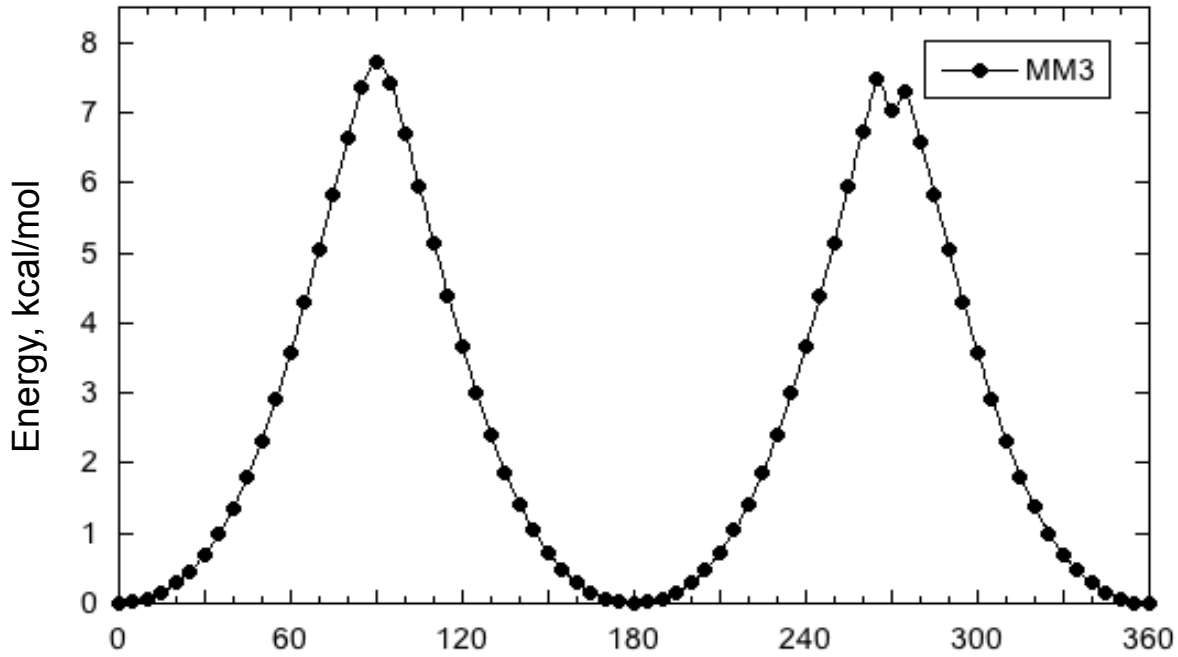
CSD comparison

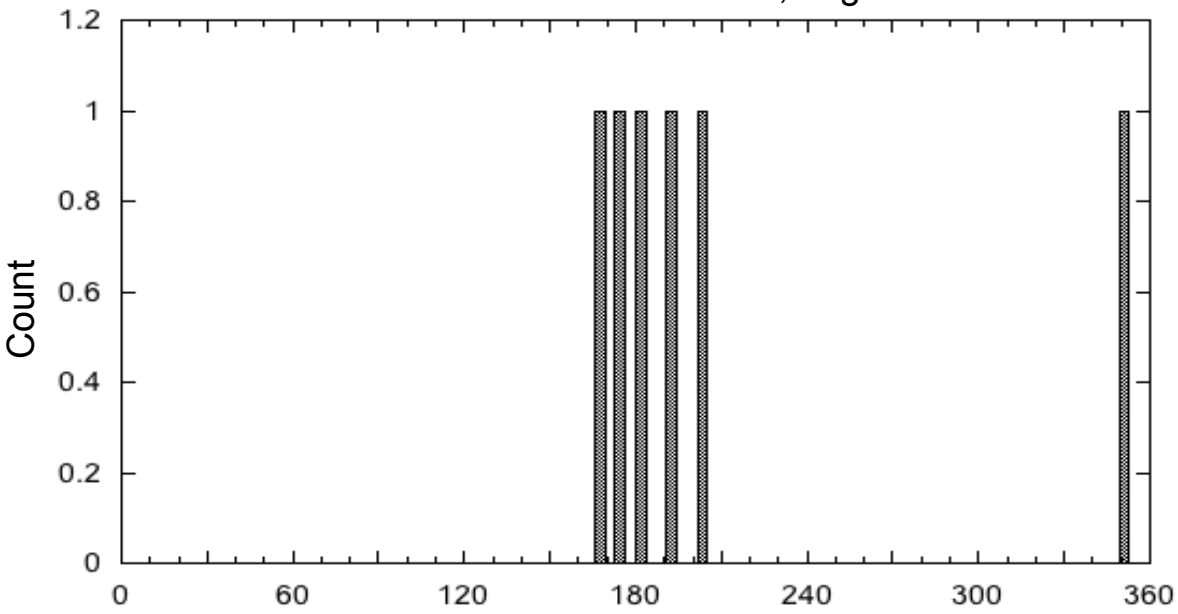


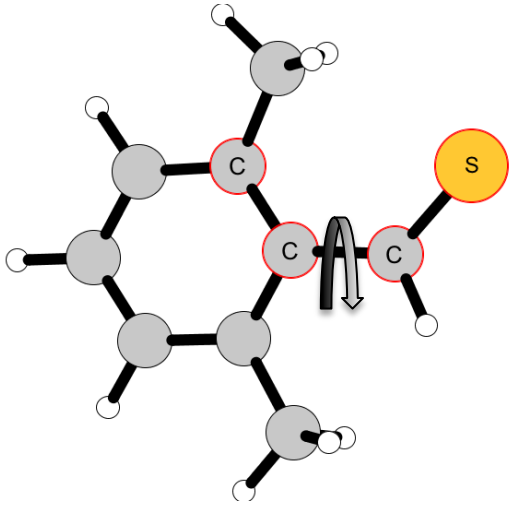
5 hits

C–C = 1.47 ± 0.01 Å

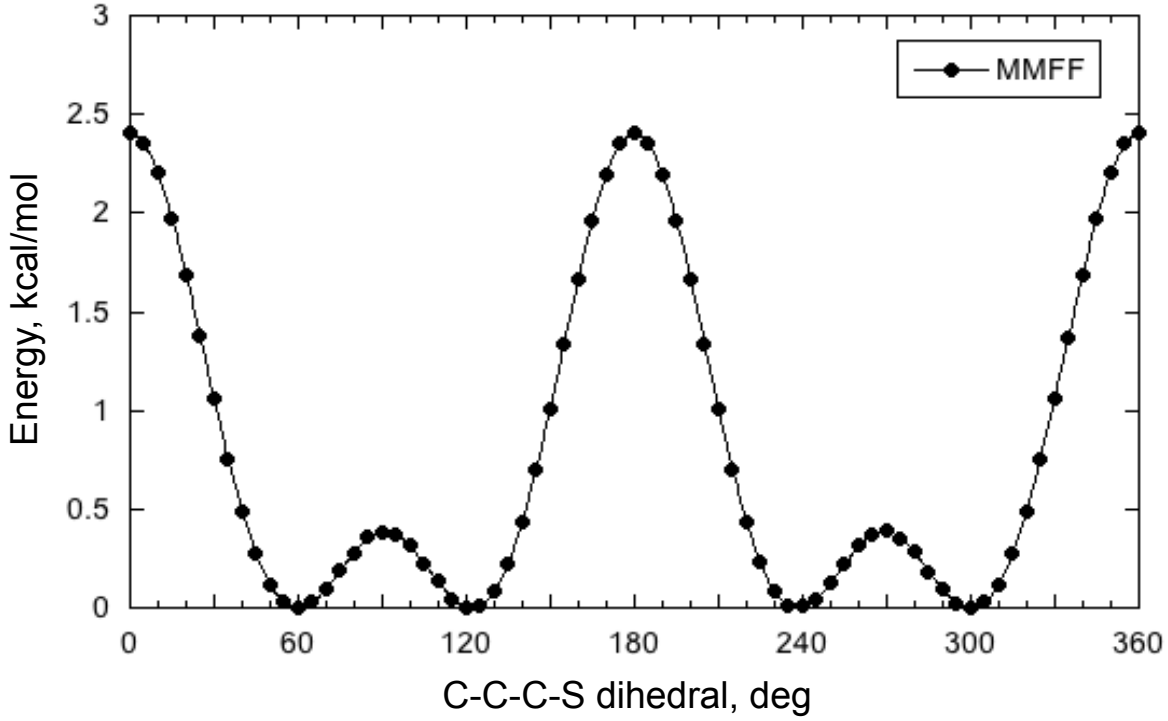
TYPE 3-3:10-1





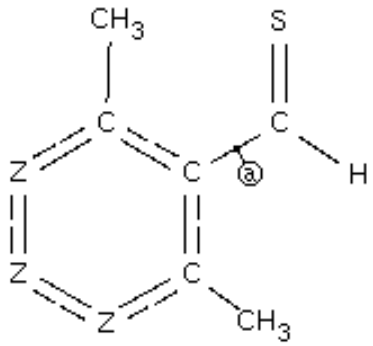


TYPE 3-3:10-2



Φ	E
60.0	0.00
120.0	0.00
240.0	0.01
300.0	0.00

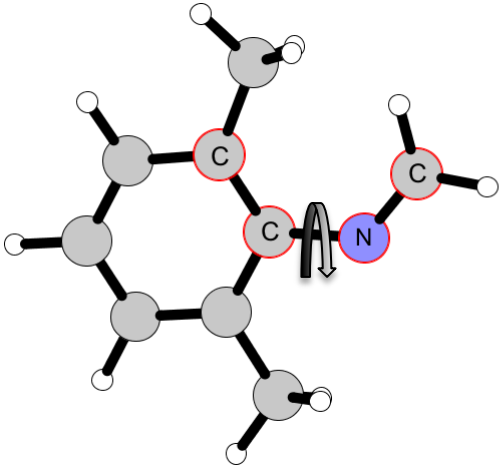
CSD comparison



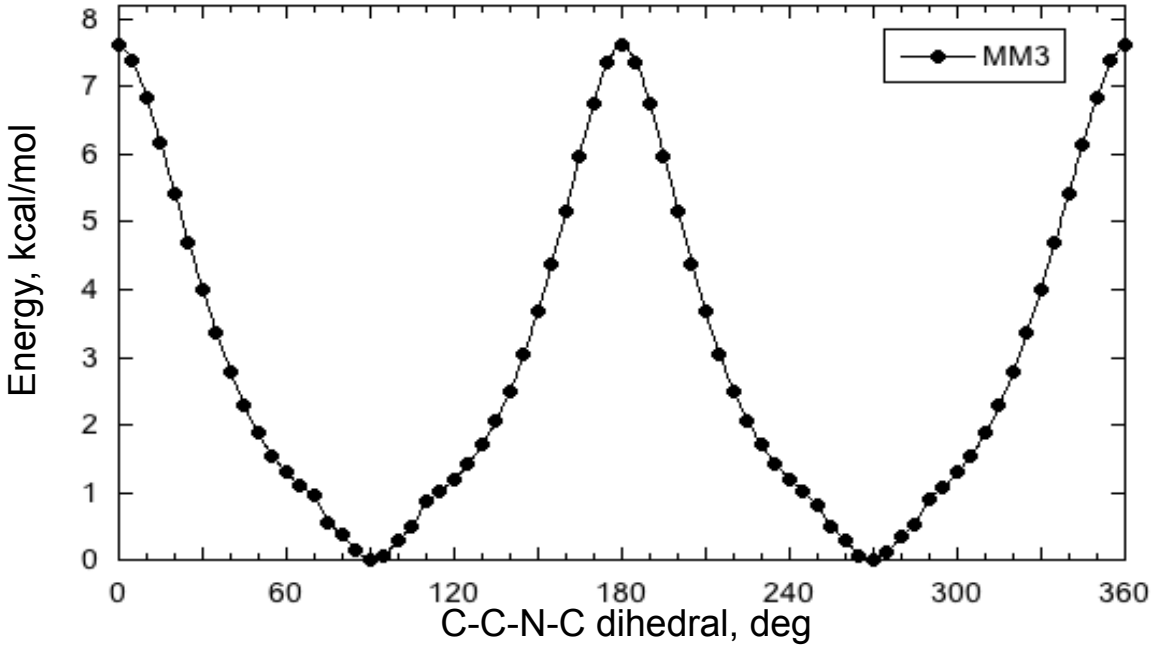
0 hits

NO CSD HITS

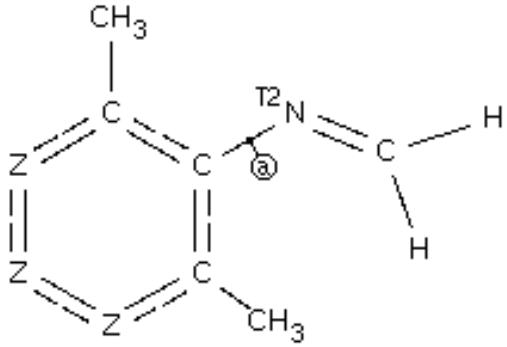
TYPE 3-3:10-3



Φ	E
90.0	0.00
270.0	0.00

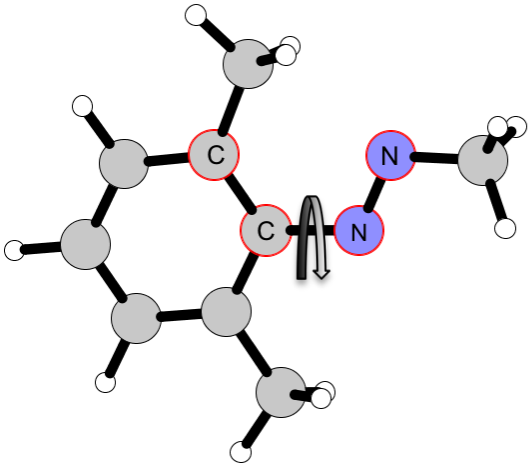


CSD comparison



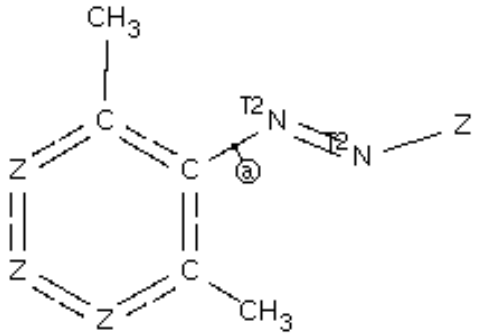
0 hits

NO CSD HITS



Φ	E
25.0	0.00
95.0	0.26
150.0	0.05
205.0	0.01
370.0	0.33
340.0	0.04

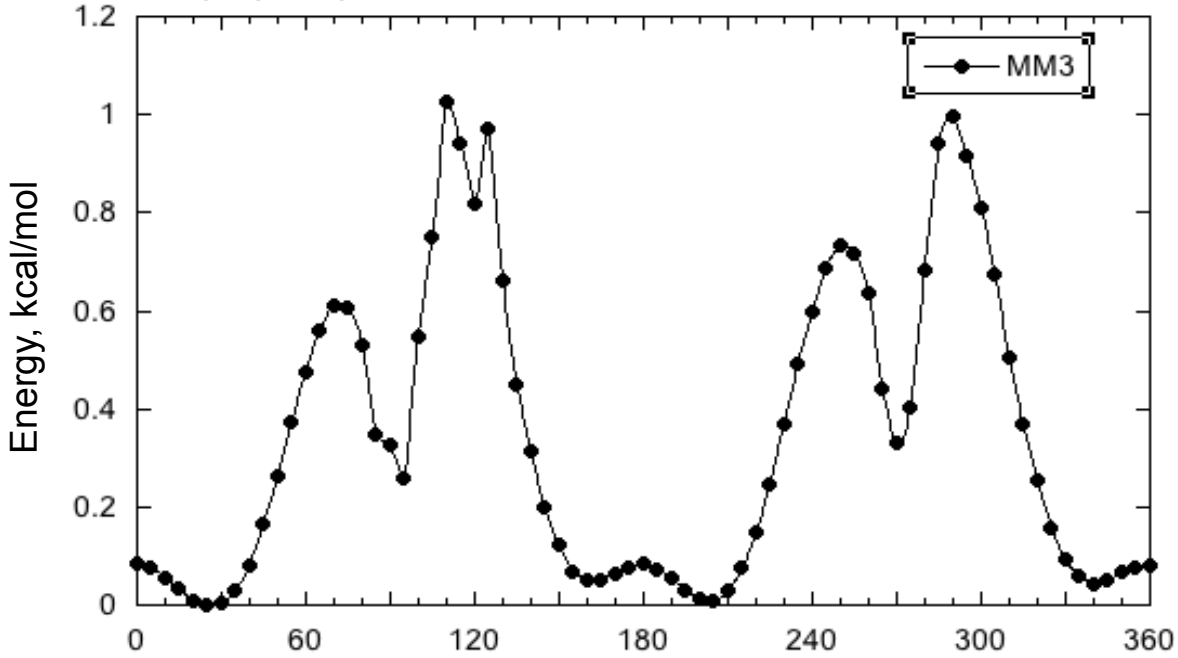
CSD comparison

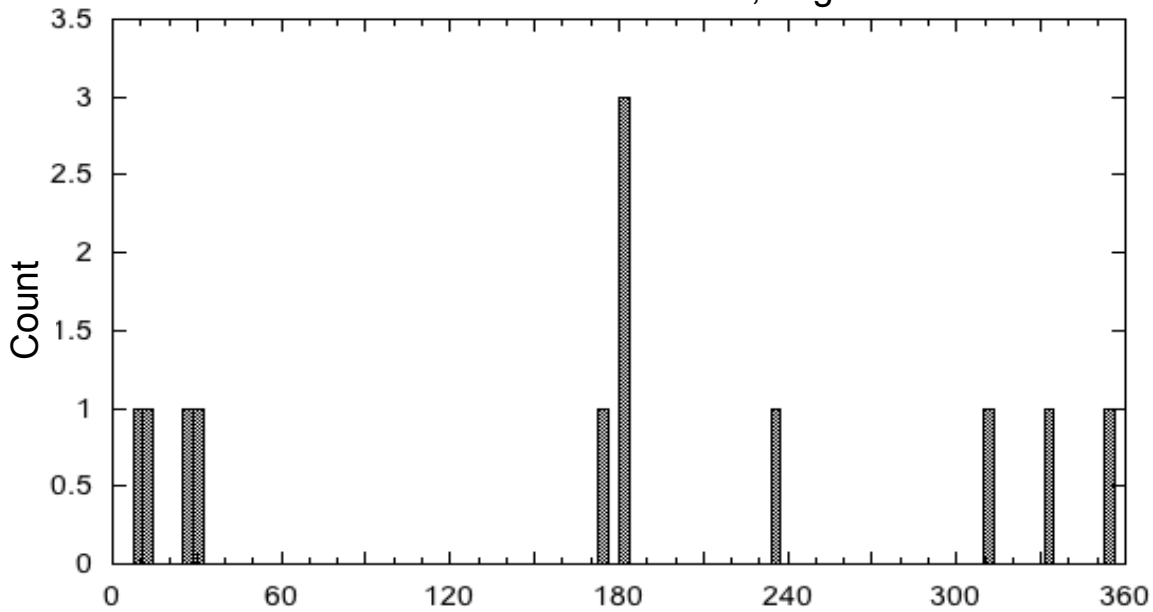


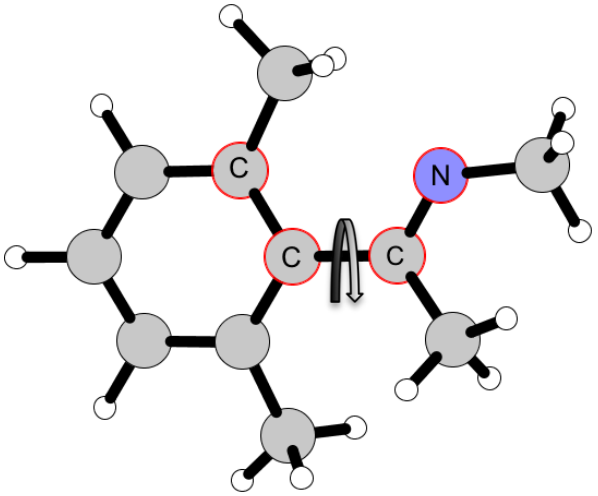
10 hits

C–N = 1.42 ± 0.02 Å

TYPE 3-3:10-4

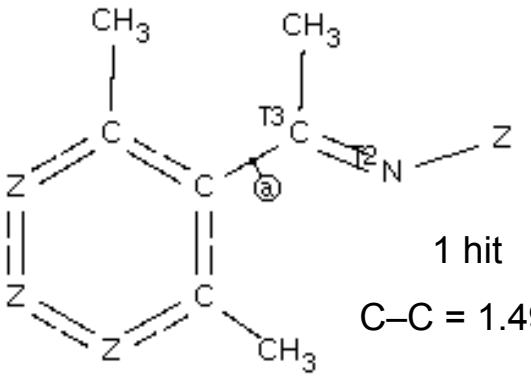






Φ	E
60.0	0.21
65.0	0.21
90.0	0.00
120.0	0.24
240.0	0.21
270.0	0.04
300.0	0.24

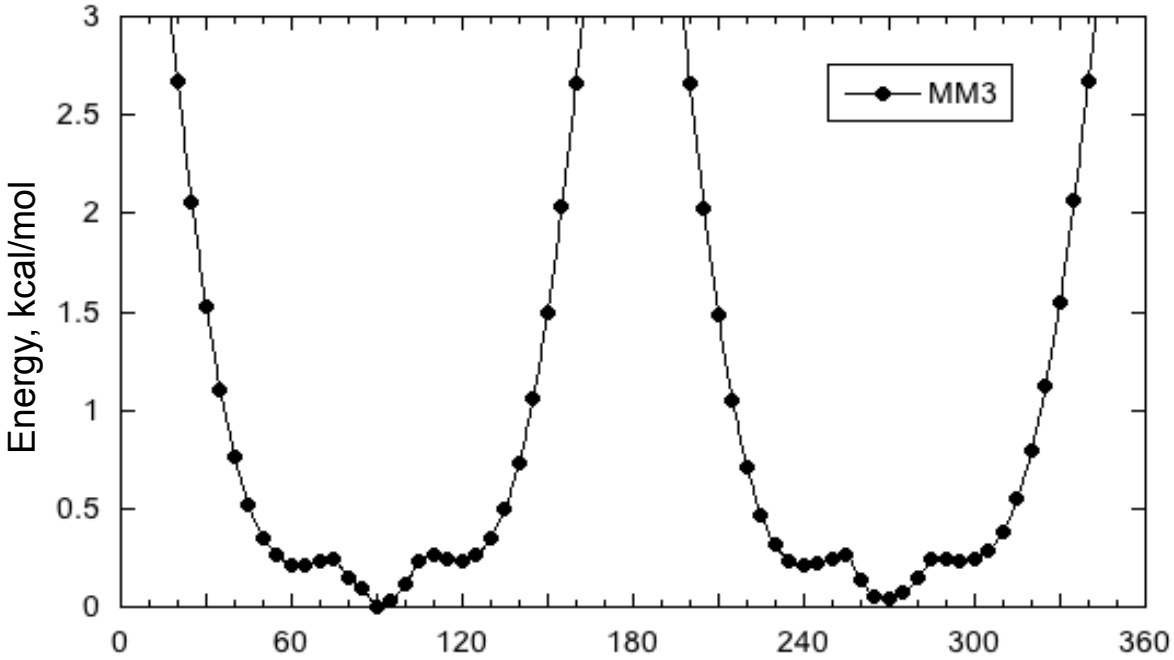
CSD comparison

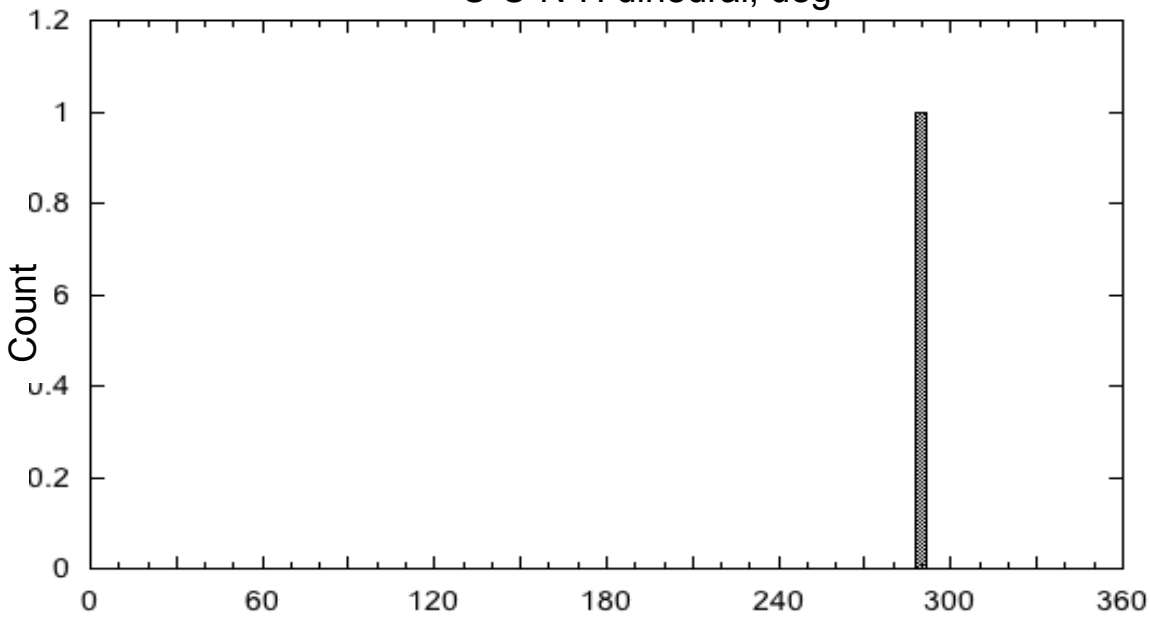


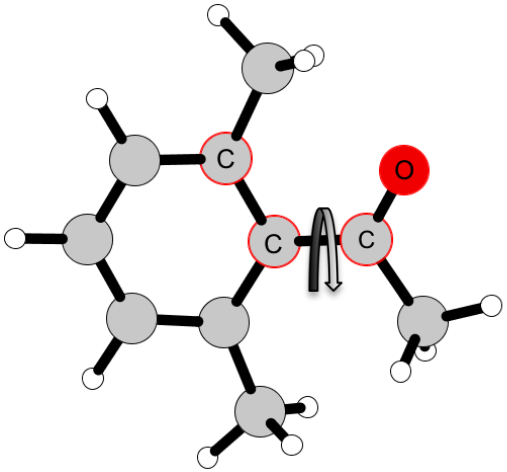
1 hit

C–C = 1.49 Å

TYPE 3-3:11-1



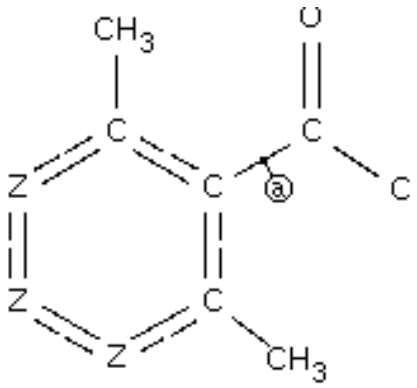




TYPE 3-3:11-2

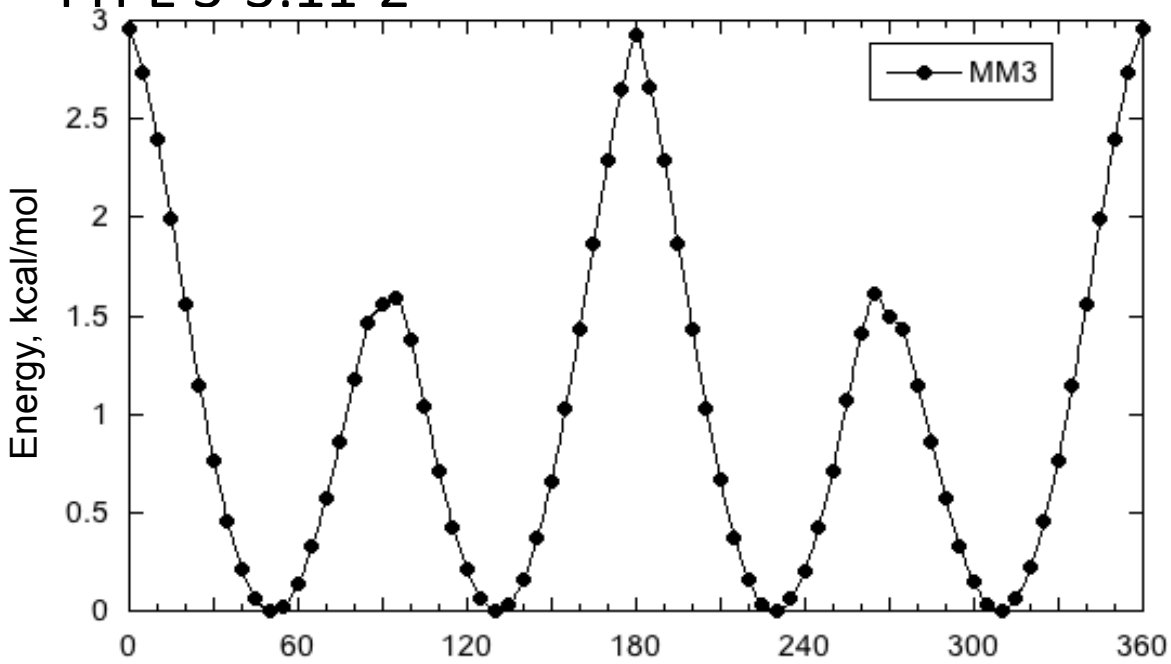
Φ	E
50.0	0.00
130.0	0.00
230.0	0.00
310.0	0.00

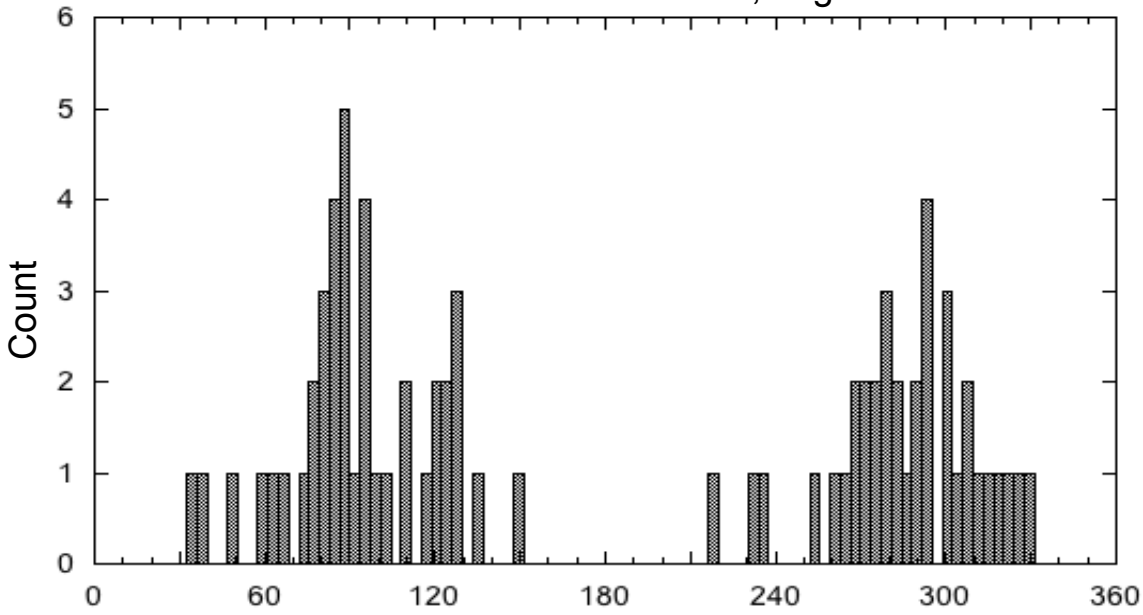
CSD comparison

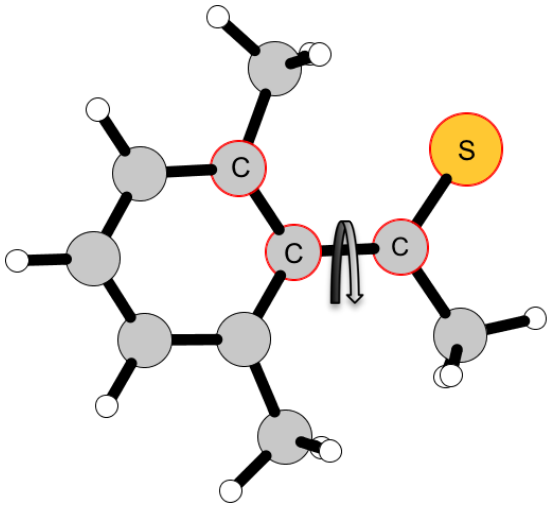


52 hits

C–C = 1.50 ± 0.01 Å

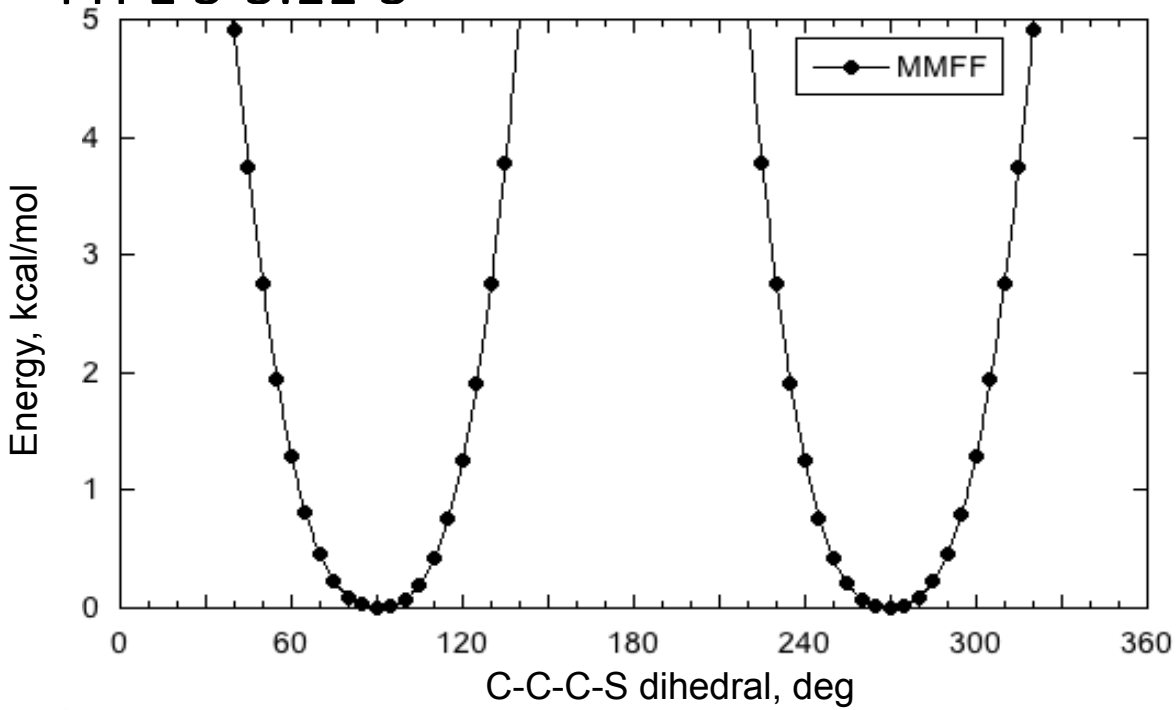




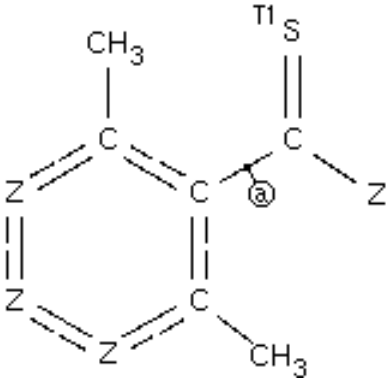


Φ	E
90.0	0.00
270.0	0.00

TYPE 3-3:11-3

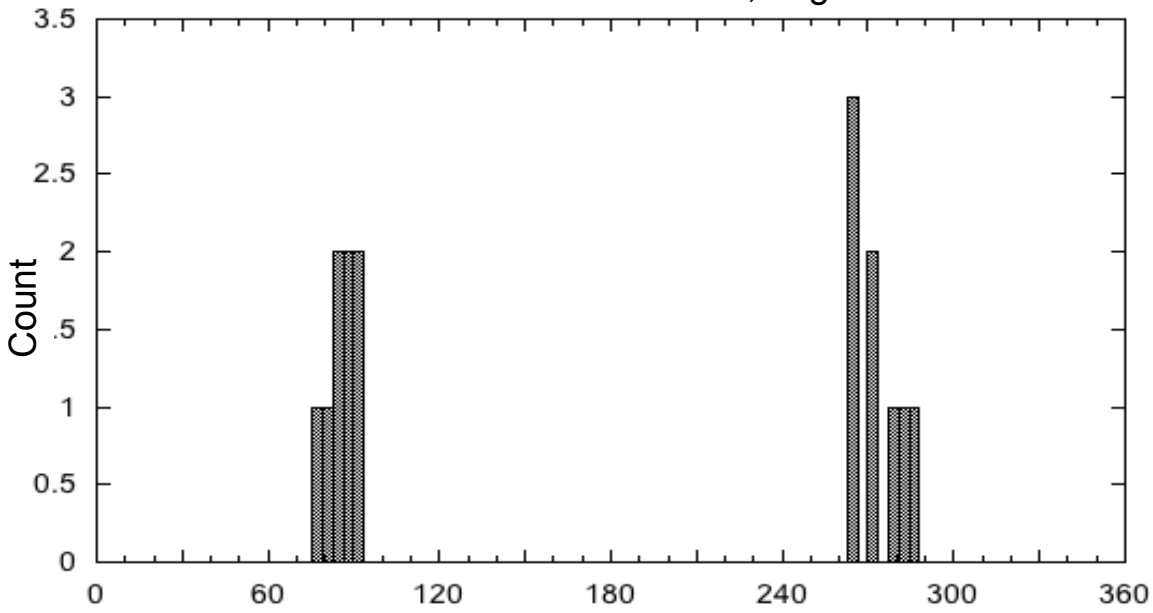


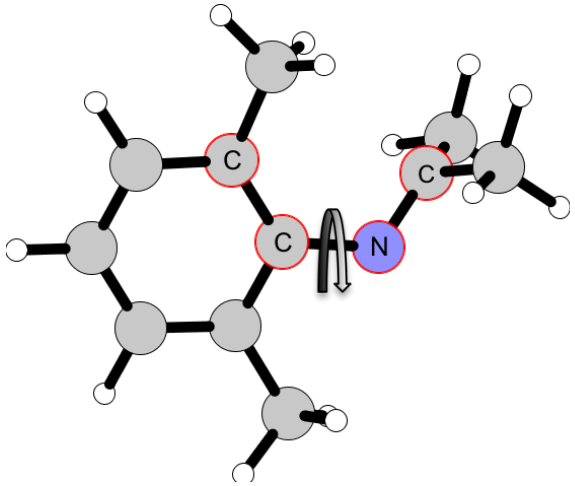
CSD comparison



5 hits

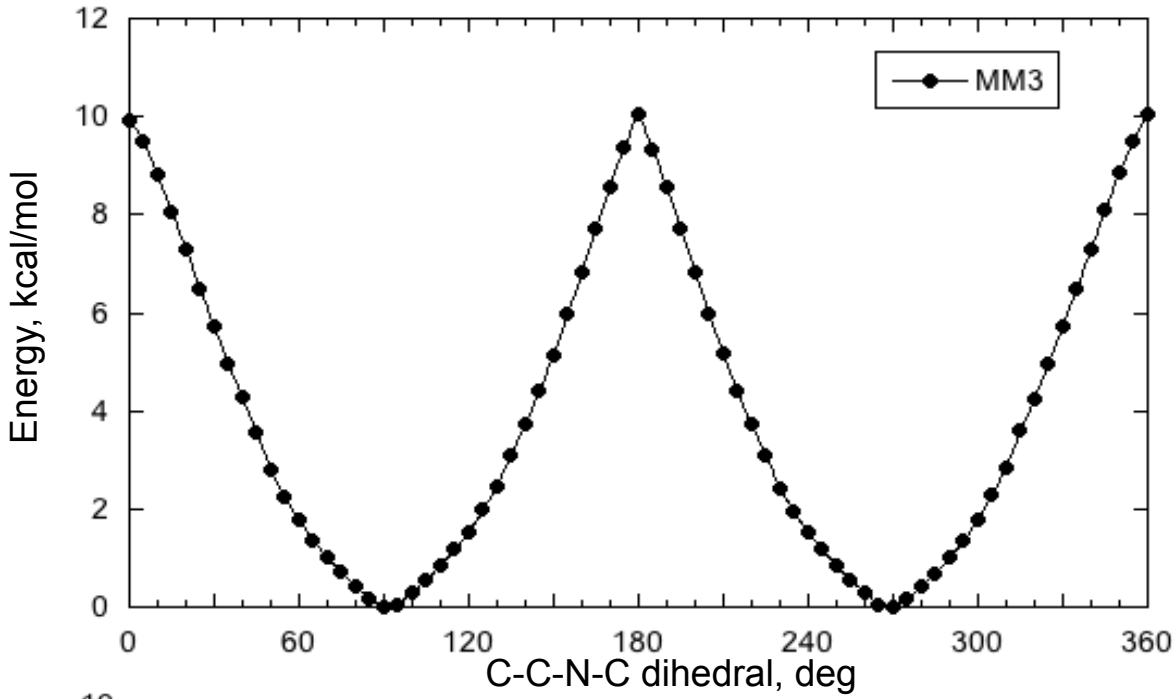
C–C = 1.49 ± 0.01 Å



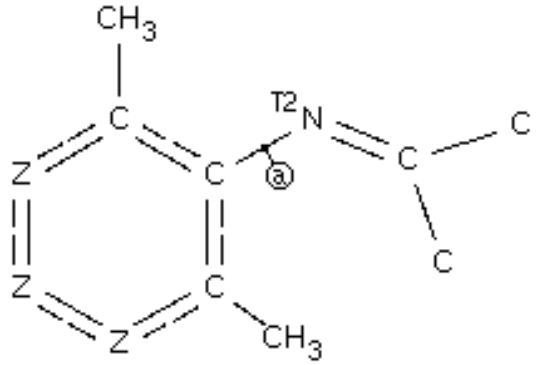


TYPE 3-3:12-1

Φ	E
90.0	0.00
270.0	0.01

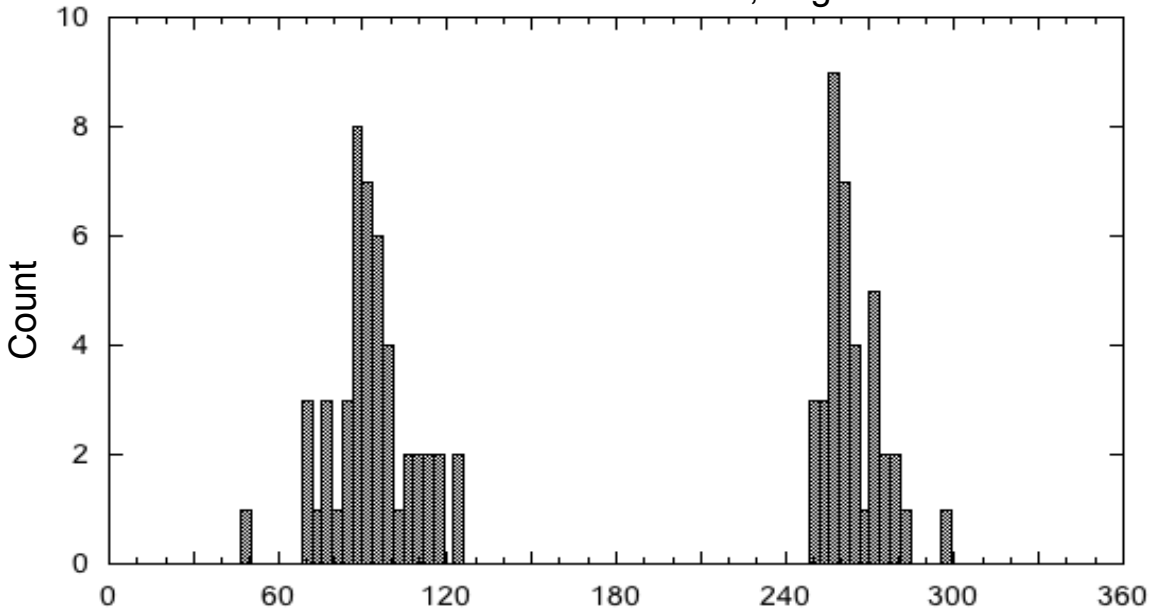


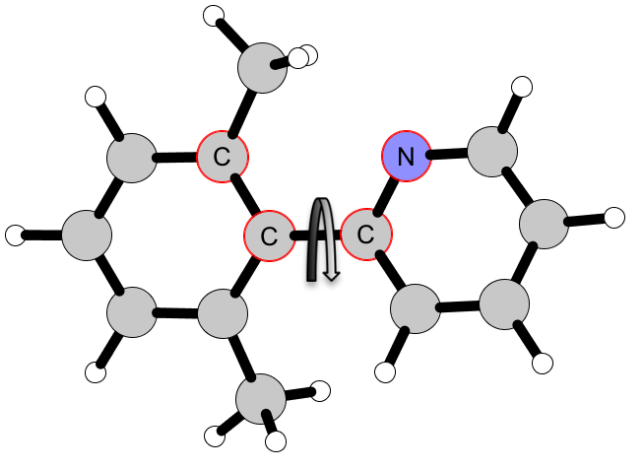
CSD comparison



64 hits

C–N = 1.42 ± 0.01 Å

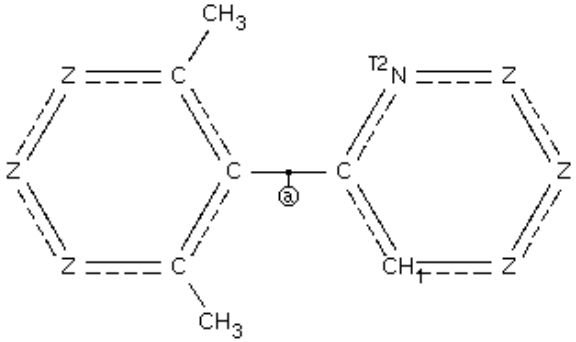




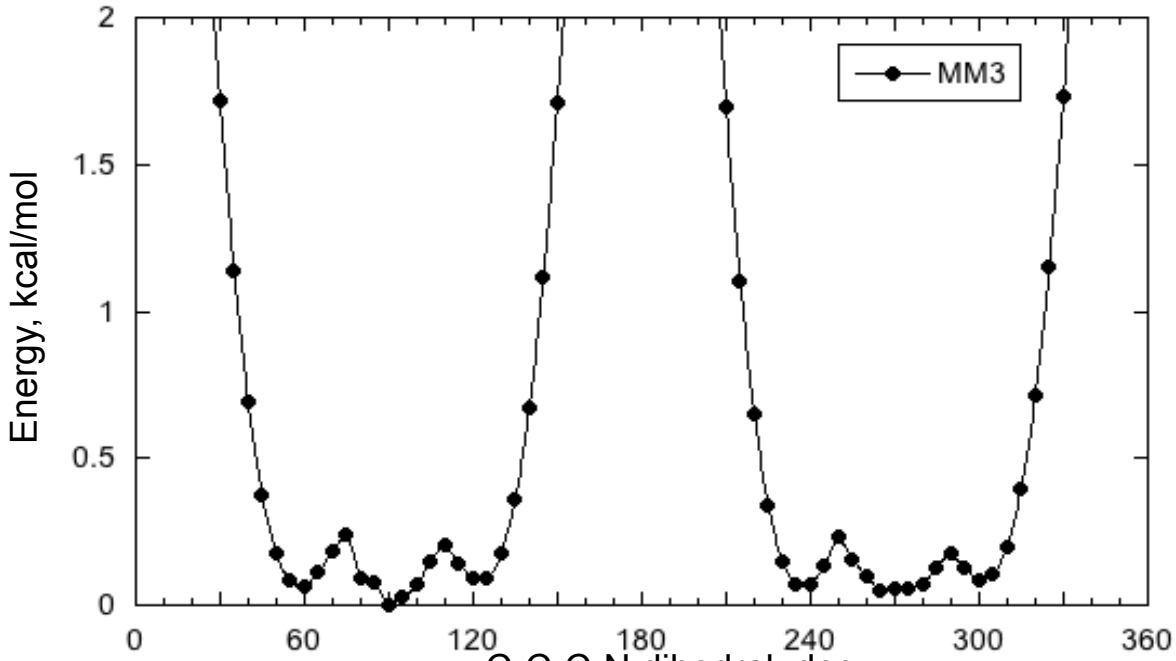
TYPE 3-3:13-1

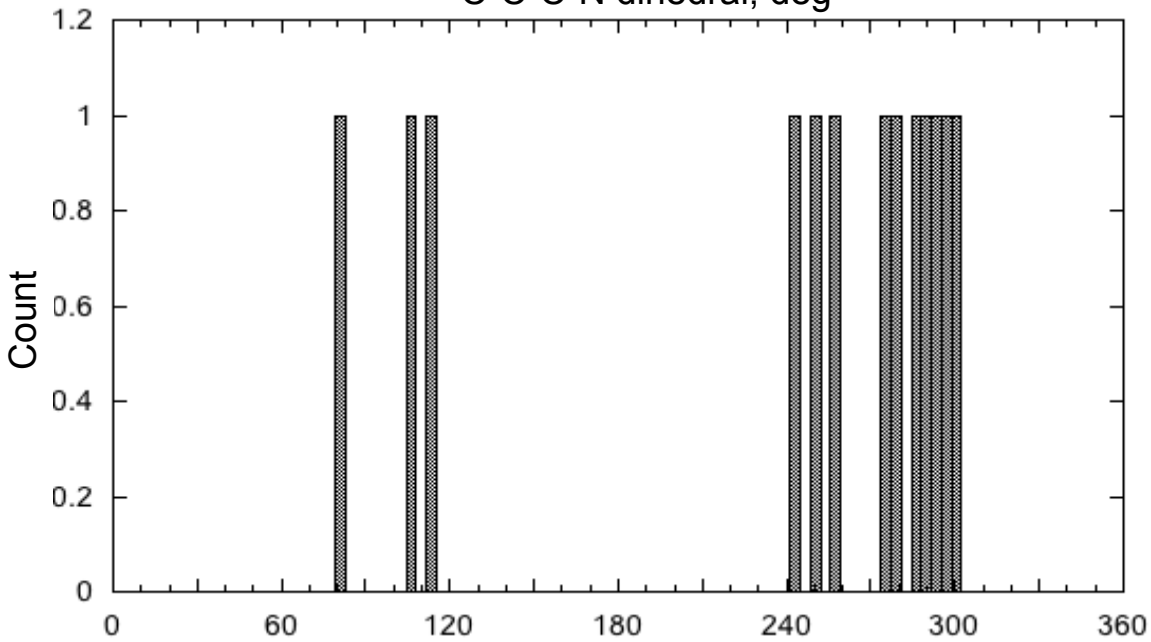
Φ	E
60.0	0.07
90.0	0.00
120.0	0.09
125.0	0.09
235.0	0.07
275.0	0.06
300.0	0.09

CSD comparison

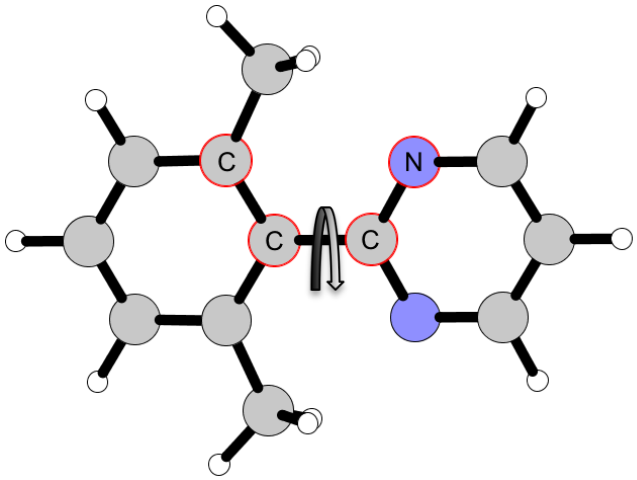


10 hits
C–C = 1.50 ± 0.01 Å

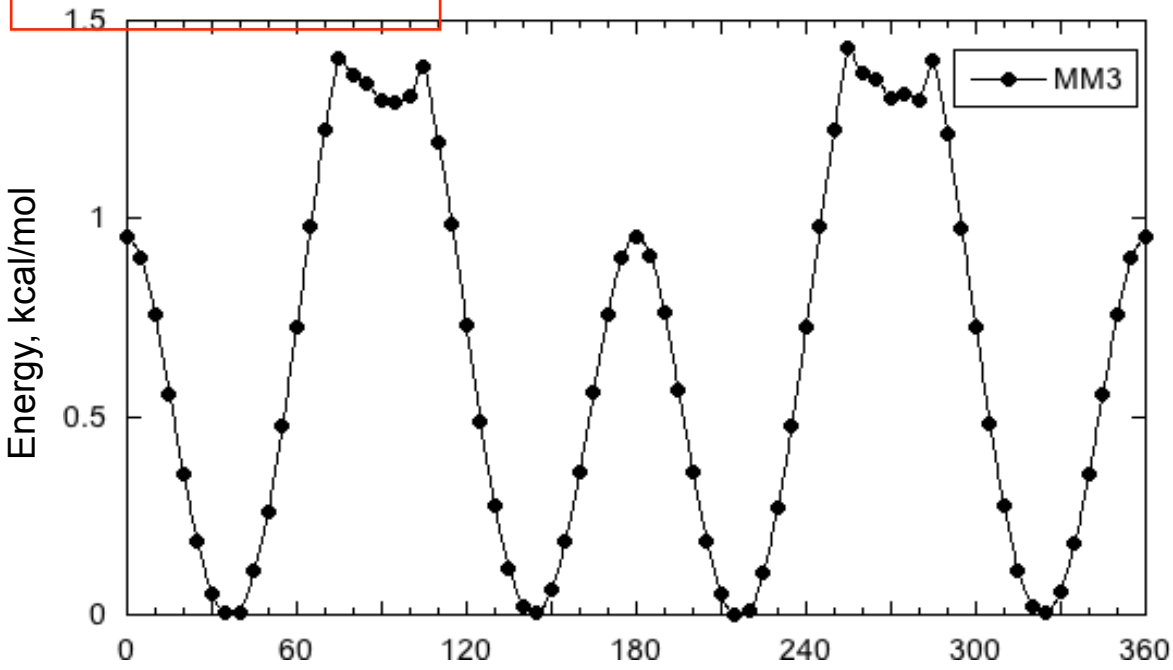




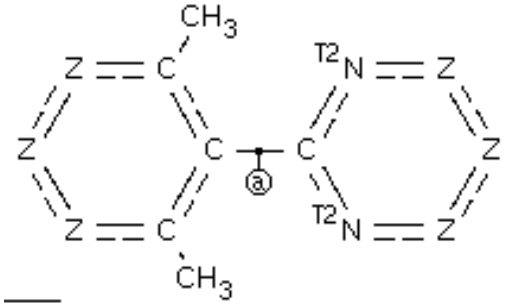
TYPE 3-3:13-2



Φ	E
35.0	0.00
95.0	1.30
145.0	0.01
215.0	0.00
270.0	1.30
325.0	0.01

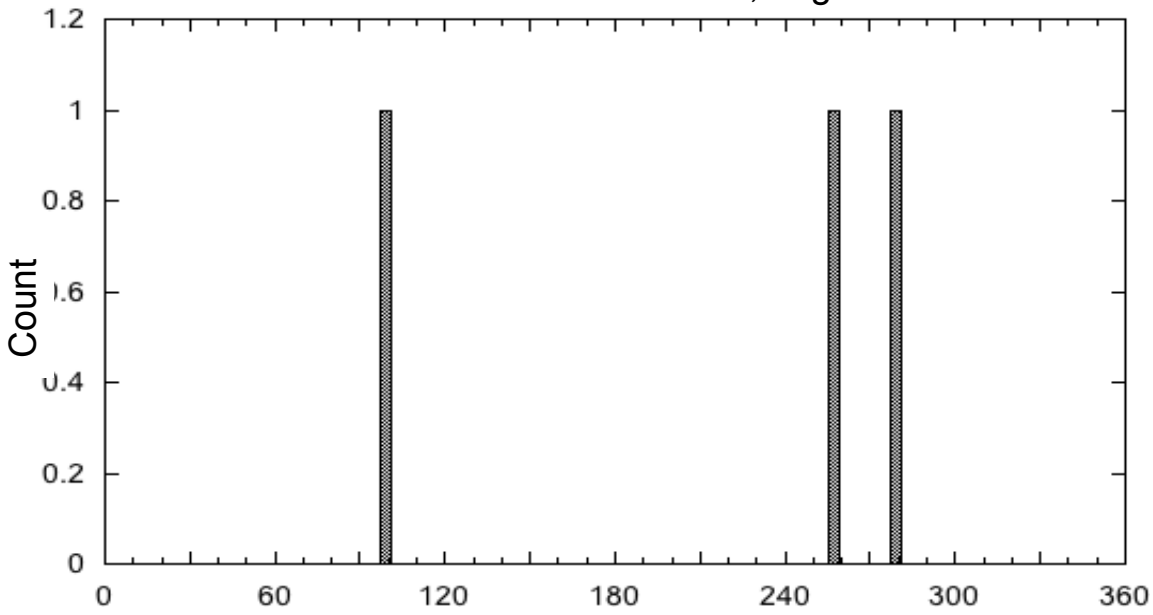


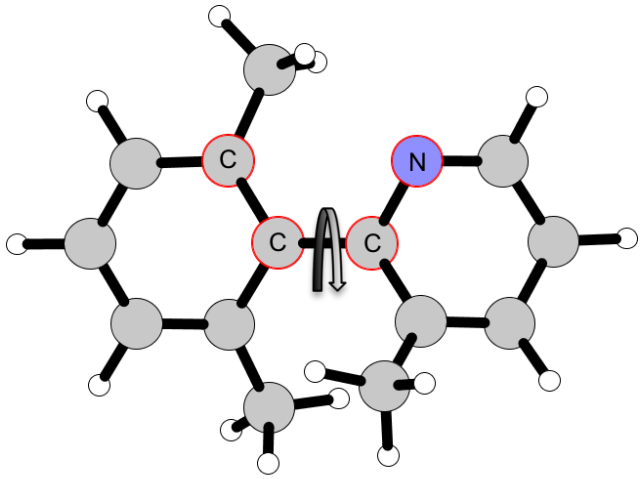
CSD comparison



1 hit

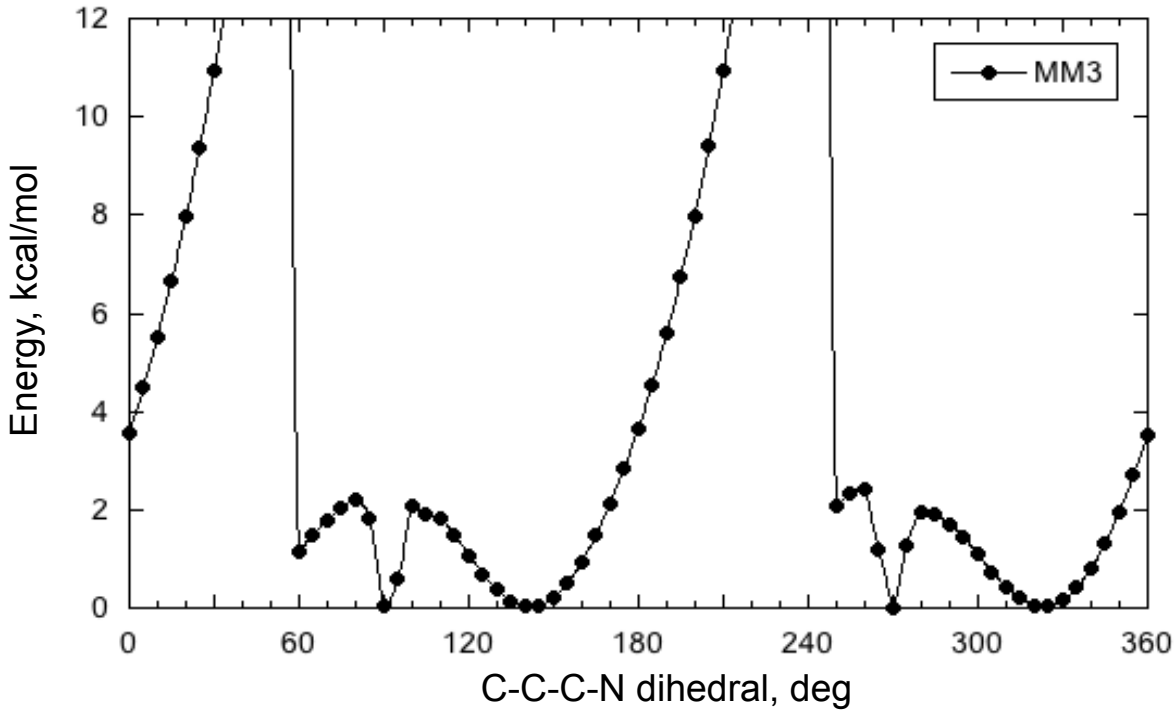
C–C = 1.50 ± 0.01 Å



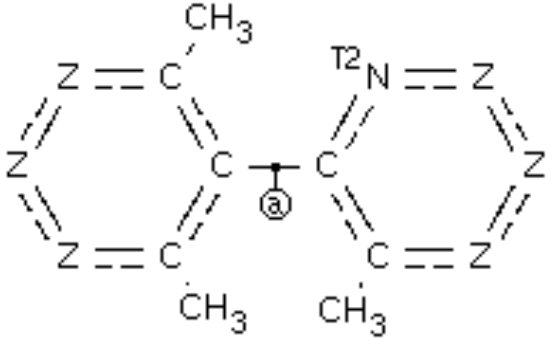


Φ	E
40.0	*0.04*
90.0	0.05
140.0	0.04
220.0	*0.04*
270.0	0.00
325.0	0.04

TYPE 3-3:13-3



CSD comparison



0 hits

NO CSD HITS

assignclass Subroutine

```
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c           Copyright (c) 2014 by Billy Wayne McCann, Ben Hay
c                     Chemical Sciences Division
c                     Chemical Separations Group
c                       Oak Ridge National Lab
c                         All rights reserved
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c
c                               ASSIGNCLASS
c
c
c     This subroutine determines the class and subclass of a dihedral
c     as well as the driver atom of a fragment. Classes and subclasses
c     are determined by the fragment's bond topology, e.g. trigonal
c     pyramidal, trigonal planar, tetrahedral, etc. While each subclass
c     is meant to be as inclusive as possible, sometimes distinctions
c     must be made between two members of the same class. For example,
c     the XPX angle of a phosphine is much small than the XNX angle of an
c     amine. This leads to a qualitatively different rotational
c     potential energy surface. Therefore the identity of the link atom
c     must be tested.
c
c     The driver atom refers to the atom which was used in constructing
c     the rotational potential energy surface. This atom must be
c     correctly identified in this subroutine.
c
c     A bond is formed by connecting a member of one class with
c     a member of another class.  The actual values of the dihedral
c     angles were determined by forming test molecules and doing
c     rotational PES's using Allinger's (MM3) force field or density
c     functional theory (wB97XD/6-31+G**).  These values are stored
c     in the CONSTANTS file, referenced by class and subclass.
c
c-----------------------------------------------------------------------
c
c                  LIST OF CLASSES AND SUBCLASSES
c
c     Note that unless otherwise specified R is any non-H atom and X is
c     any atom. Refer to user manual for three dimensional depicitons.
c
c     Class 1 - Tetrahedral - B, C, Si, N, P, As
c       subclass 1:  -RH2X
c       subclass 2:  -RHX2
c       subclass 3:  -RX3 or -RH3
c     Class 2 - Trigonal Planar (alkene-like)
c       subclass 1:  -R(H)=R(H)(X) where X is trans
c       subclass 2:  -R(H)=R(H)(X) where X is cis,  or C(H)=CX2
c       subclass 3:  -R(X)=R(H)(X) where X is trans, or C(X)=CH2
c       subclass 4:  -R(X)=R(H)(X) where X is cis,   or C(X)=CX2
c     Class 3 - Trigonal Planar - Ring
c       subclass 1:  -(arene) two ortho H
c       subclass 2:  -(arene) one ortho H and one ortho X
c       subclass 3:  -(arene) two ortho X
c     Class 4 - Amide-like - Beta Carbonyl
c       subclass 1:  -N(H)C(=O)X attaching cis to carbonyl
c       subclass 2:  -N(H)C(=O)X attaching trans to carbonyl
c       subclass 3:  -N(R)C(=O)X attaching cis to carbonyl
c       subclass 4:  -N(R)C(=O)X attaching trans to carbonyl
c     Class 5 - Ether-like
c       subclass 1:  -OR
c       subclass 2:  -OH
c       subclass 3:  -XR    -X not Oxygen
c       subclass 4:  -XH    -X not Oxygen
c     Class 6 - Pyramidal Amine
c       subclass 1:  -NH2
c       subclass 2:  -NHR, attachment down
c       subclass 3:  -NHR, attachment up
c       subclass 4:  -NR2
c     Class 7 - Amide-like
c       subclass 1:  -C(=O)N(H)(R), where R is cis to carbonyl
c       subclass 2:  -C(=O)N(R)(X), where R is trans to carbonyl
```

```
c       subclass 3:  -C(=N-O)NH2, amidoxime
c     Class 8 - Oxides
c       subclass 1:  Phosphine Oxide : -P(=O)X2
c       subclass 2:  Sulfoxide : -S(=O)X -- S (driver on +y axis)
c       subclass 3:  Sulfoxide : -S(=O)X -- R (driver on -y axis)
c       subclass 4:  Sulfone   : -S(=O)2X
c     Class 9 - Phosphines
c       subclass 1:  -PH2
c       subclass 2:  -PHX -- S (X on -y axis)
c       subclass 3:  -PHX -- R (X on +y axis)
c       subclass 4:  -PX2
c     Class 10 - Aldehyde-like
c       subclass 1:  -X(=O)H
c       subclass 2:  -X(=S)H
c       subclass 3:  -X=XHX   (H cis to loser atom)
c       subclass 4:  -X=XX    (second X trans to loser atom)
c     Class 11 - Ketone-like
c       subclass 1:  -X(=X-X)R  (second X trans to loser atom)
c       subclass 2:  -X(=O)R
c       subclass 3:  -X(=X)R
c     Class 12 - Ketimine-like
c       subclass 1:  -X=XR2
c     Class 13 - Planar Ring - unsubstituted ortho positions
c       subclass 1:  1 ortho unsubstituted ; one H substituted
c       subclass 2:  2 ortho unsubstituted
c       subclass 3:  1 ortho unsubstituted ; one substituted - not H
c     Class 14 - Borane-like (trigonal planar)
c       subclass 1:  -XH2
c       subclass 2:  -XRH
c       subclass 3:  -XR2
c     Class 15 - No torsional preference or Unknown
c       subclass 1:  Linear, Metallic, Hypervalent, or unknown
c     Class 16 - Terminal case
c       subclass 1:  Coordinating atom attached to only a loser atom
c
c-----------------------------------------------------------------------
c
c                                NOTE
c
c     If you want to add another type of bond, then you will have
c     to (1) define a new class/subclass, (2) run it against all of
c     the existing classes to determine dihedral assignments, and
c     (3) edit the routine to add the new stuff.
c
c     CAUTION:  Three arrays in constants.i must be updated if you
c     add more classes.  These are as follows:
c
c       nrotval(#classes,#subclasses,#classes,#subclasses)
c       rotval(#classes,#subclasses,#classes,#subclasses,#minima)
c       roten(#classes,#subclasses,#classes,#subclasses,#minima)
c
c       At this time #classes=15, #subclasses=4, and #minima=7
c
c       The number of minima is also defined in params.i as MAXTORVAL
c       Currently set at 7.
c
c-----------------------------------------------------------------------
c
c                            META
c
c     This subroutine is called by readhosta.F (line 563) and readhostb.F
c     (line 396) as
c
c     call assignclass(xa(1,1,i),ya(1,1,i),za(1,1,i),n12a,i12a,
c    &                 atlaba,iguesta,torrefa(i),ibondera(i),loosera(i),
c    &                 classa(i),sclassa(i))
c
c     from readhosta.F. From readhostb.F, change the 'a' suffixes to b.
c     Compare this to the dummy variable which are used in this subroutine
c
c     subroutine assignclass(x, y, z, n12, i12, itype, atlab, guest,
c    &                       ii, jj, kk, class, sub)
c
```

```
c     Therefore the actual --> dummy variable mappings are:
c       xa(1,1,i)   --> x
c       ya(1,1,i)   --> y
c       za(1,1,i)   --> z
c       n12a        --> n12
c       i12a        --> i12
c       itypea      --> itype
c       atlaba      --> atlab
c       iguesta     --> guest
c       torrefa(i)  --> ii
c       ibondera(j) --> jj
c       loosera(i)  --> kk
c       classa(i)   --> class
c       sclassa(i)  --> sub
c
c-----------------------------------------------------------------------
c
c     P A S S E D   V A R I A B L E S:
c
c     real      x, y, z     coordinates of the atoms
c     integer   n12         array of number attached atoms
c     integer   i12         array of attached atoms
c     char      atlab       atom labels
c     integer   guest       guest flags
c     integer   ii          serial number of driver atom i
c     integer   jj          serial number of attach atom j
c     integer   kk          serial number of the dummy atom to be lost
c     integer   class       the class of the rotor
c     integer   sub         the subclass of the rotor
c     integer   alt_ii      alternative driver atom to be used when
c                           a host is mirrored. only necessary for
c                           asymmetric trigonal pyramidal hosts.
c
c-----------------------------------------------------------------------
c
c                         B E G I N
c
c-----------------------------------------------------------------------
      subroutine assignclass(n_atoms, x, y, z, n12, i12, atlab,
     &                       guest, ii, jj, kk, class, sub, alt_ii)

      implicit none
      include 'params.i'

      integer n_atoms               ! number of atoms in structure
      integer r_atoms               ! number of real atoms
      integer i, j                  ! indexing variables
      integer ii                    ! serial # of driver atom
      integer alt_ii                ! serial # of alternative driver atom
      integer jj                    ! serial # of attaching atom
      integer kk                    ! serial # of loser atom
      integer numhyd                ! # of H's connected to an atom
      integer numox                 ! # of O's connected to an atom
      integer numdubox              ! # of O's double bonded to an atom
      integer numoth                ! # of "other atoms" connected ''
      integer hatt(n_atoms)         ! serial #'s of numhyd's
      integer oxatt(n_atoms)        ! serial #'s of numox's
      integer duboxatt(n_atoms)     ! serial #'s of numdubox's
      integer othatt(n_atoms)       ! serial #'s of numoth's
      integer guest(MAXATOMD)       ! status of whether an atom is a
                                    ! guest: 1 = yes, it is a guest.
                                    !        0 = no, it is not a guest
      integer guest_arr(n_atoms)    ! array of serial # of guest atom
      integer dummy_arr(n_atoms)    ! array of serial # of dummy atoms
      integer class, sub            ! class and subclass of rotor
      integer n12(MAXATOMD), i12(MAXVALA,n_atoms) ! connectivity matrices
      integer n12r(n_atoms), i12r(MAXVALA,n_atoms) ! real connectivity
                                                   ! matrices
      integer valency               ! coordination # of jj

      real x(MAXATOMD), y(MAXATOMD), z(MAXATOMD) ! atomic coordinates
      real dx(5),dy(5),dz(5)        ! "temp" atomic coordinates
      real dihed, ang, phi          ! dihedral angle, angle, dihedral angle
```

```
      character*2 atlab(n_atoms)    ! periodic table atom labels
      character*2 linkatom          ! atom label of jj


      ! variables concerning rings
      integer rings                     ! number of rings in molecule
      integer nring(MAXRINGS)           ! number of atoms in ring
      integer iring(n_atoms,MAXRINGS)   ! serial numbers in ring

      real xlist(n_atoms)      ! used for constructing a list of xyz
      real ylist(n_atoms)      ! coordinates
      real zlist(n_atoms)
      integer atlist(n_atoms)  ! used in constructing array of
                               ! atoms attached to any other atom

c     variables concerning alpha positions
      integer numalpha                  ! number of alplha atoms
      integer ialpha(n_atoms)           ! serial numbers of alpha atoms
      integer ialpha1                   ! serial number of 1st alpha
      character*2 alphalab(n_atoms)     ! label of alphas

c     variables concerning beta positions
      integer numbeta                   ! indexer for counting betas
      integer nbeta(n_atoms)            ! number of beta atoms connected
                                        ! to a certain alpha
      integer ibeta(n_atoms,n_atoms)    ! serial numbers of beta atoms
      integer ibeta1
                                        ! connected to a certain alpha
      logical has_cisbeta               ! true if alpha has a cis beta
      logical cisbeta_isH               ! true if cis beta is Hydrogen
      integer ibetacis                  ! serial # of cis beta
      character*2  betacis              ! atom label of cis beta

c     True if ring is flat
      logical flat              ! true if ring is flat
      logical ismetal           ! True if link atom is a meta
c     Debugging
c     debug1 -> debugging the passed variables
c     debug2 -> print out class, subclass, and driver
c     debug3 -> verbose printing of many steps of assigning class etc.
      logical debug1, debug2, debug3

      integer numguests         ! number of guests in the host complex
      integer numdummies        ! number of dummies in atom arrays


c-----------------------------------------------------------------------
c               END OF VARIABLE DECLARATIONS
c-----------------------------------------------------------------------


c-----------------------------------------------------------------------
c     Initialize some variables.
c-----------------------------------------------------------------------
      debug1 = .false.
      debug2 = .false.
      debug3 = .false.

      ismetal = .false.

      dummy_arr  = 0
      guest_arr  = 0
      numguests  = 0
      numdummies = 0
      alt_ii     = 0


c-----------------------------------------------------------------------
c
c     When debugging, print a line of hashes to separate assignments
c     which are written to the screen.
c
```

```
      if (debug1.or.debug3) then
         write(*,5)
      end if
    5 format(/,"#######################################################",/)

c-----------------------------------------------------------------------
c       create connectivity arrays without dummies or guests
c-----------------------------------------------------------------------
c
c     Count the number of guest and dummy atoms.
c     Put their serial numbers into arrays for some testing below.
c
      do i = 1, n_atoms

          if (guest(i).eq.1) then
              numguests = numguests + 1
              guest_arr(numguests) = i
          elseif (atlab(i).eq.'Du') then
              numdummies = numdummies + 1
              dummy_arr(numdummies) = i
          end if

      end do
c
c     For debugging purposes, list the guests and dummies.
c
      if (debug3) then
          write(*,*) "Numguests is ", numguests
          write(*,*) "Guests are ", guest_arr(:)
          write(*,*)
          write(*,*) "Numdummies is ", numdummies
          write(*,*) "Dummies are ", dummy_arr(:)
      end if
c
c     Remove any line corresponding to a guest atom or dummy from the
c     connectivity matrix. Because guests are the final entries in
c     the z-matrix, simply remove the final (numguests) from n_atoms.
c     This create a "real" connectivity matrix (i12r).
c
      r_atoms = n_atoms - numguests

      i12r = 0
      n12r = 0

      do i = 1, r_atoms
          i12r(:,i) = i12(:,i)
      end do
c
c     Now, scan through the new connectivity matrix to find entries
c     which are either guests or dummy atoms. If one is found, zero out
c     its entry in the "real" connectivity matrix (i12r) and decrement
c     the coordination number by one to create a "real" coordination
c     number array (n12r).
c
      do i = 1, r_atoms
          do j = 1, n12(i)

              if (guest(i12r(j,i)).eq.1          ! if it's a guest
     &        .or. atlab(i12r(j,i)).eq.'Du') then ! or if it's a dummy
                  i12r(j,i) = 0                  ! zero the entry
                  n12r(i) = n12(i) - 1           ! decrement valency
              else
                  n12r(i) = n12(i)      ! move entry to "real" matrix
              end if

          end do
      end do
c
c     Display the old and new connectivity arrays for debugging purposes
c
      if (debug3) then
          write(*,*) "n12,n12r,i12,i12r after removing guests/dummies:"
```

```
          do i = 1, n_atoms
               write(*,'(1X,A11,I3,A11)') "========== ",i," ==========="
               write(*,*)
               write(*,*) "n12  = ",n12(i)
               write(*,*)
               write(*,*) "n12r = ",n12r(i)
               write(*,*)
               write(*,10) "i12(:,",i,")  = ", i12(:,i)
               write(*,*)
               write(*,10) "i12r(:,",i,") = ", i12r(:,i)
               write(*,*)
          end do

      end if

10    format (1X,A7,I1,A4,10I4)

c---------------------------------------------------------------------
c                 DISPLAY NEW FRAGMENT INFORMATION (DEBUG)
c     Display fragment information to verify that the host file has
c     been read correctly and that the new connectivity arrays are
c     correct.
c---------------------------------------------------------------------

      if (debug1) then
          write(*,*) '========= FRAGMENT INFO ================='
          write(*,*)
          write(*,*) 'Label | Serial# | Guest? | Valency | Connected To'

          do i=1, r_atoms
              write(*,'(5X,A2)',advance='no') atlab(i)
              write(*,'(4X,I2)',advance='no') i
              write(*,'(8X,I2)',advance='no') guest(i)
              write(*,'(8X,I2)',advance='no') n12r(i)
              do j=1, n12r(i)
                  write(*,20,advance='no') atlab(i12r(j,i))
              end do
              write(*,*)
          end do
          write(*,30) 'Total number of atoms is     = ', r_atoms
          write(*,30) 'Loser atom is serial number  = ', kk
          write(*,40) '             which has label  = ', atlab(kk)
          write(*,40) 'Attach atom (jj)              = ', atlab(jj)
          write(*,30) 'jj serial number              = ', jj
          write(*,30) 'jj coordination number        = ', n12r(jj)
          write(*,50) 'jj connectivity', atlab(i12r(1:n12r(jj),jj))
          write(*,*) '========================================='
          write(*,*)
          write(*,*) "Coordinate matrix"
          write(*,60) 'X','Y','Z'
          do i=1, n_atoms
              write(*,*) i, atlab(i), x(i), y(i), z(i)
          end do
          write(*,*)
          write(*,*) '========================================='
      end if

20        format (8X,A2,1X,I1)
30        format (1X,A31,1X,I3)
40        format (1X,A31,2X,A3)
50        format (1X,A15,18X,4(A3,1X))
60        format (22X,A1,15X,A1,14X,A1)

c---------------------------------------------------------------------
c                          END PRIMARY DEBUG
c---------------------------------------------------------------------

c---------------------------------------------------------------------
c                          BEGIN MAIN BODY
c     The primary test is the valency and geometry of the link
c     atom (jj). If jj is a metal or is hypervalent, it is considered
c     to be the "general" case, 15-1. Subsequent tests include the
```

```
c     identity of the element name of jj, as well as the nature of the
c     atoms alpha to it and sometimes its betas (the atoms connected to
c     the alphas). Generally, the cases fall into tetrahedral, trigonal
c     pyramidal, acyclic trigonal planar, cyclic trigonal planar
c     (arene-like), bent, linear, and terminal.
c-----------------------------------------------------------------------

c
c     Valency is the coordination number of the link atom (jj).
c
      valency = n12r(jj)
c
c     A new variable, linkatom, containing the whitespace-trimmed name
c     of jj.
c
      linkatom = trim(atlab(jj))

c-----------------------------------------------------------------------
c                          METAL OR NONMETAL
c-----------------------------------------------------------------------
c
c     determine if linkatom is a metal. Aluminum is kept due to its
c     resemblance to boron and carbon for trivalent and tetravalent
c     geometries, respectively.
c
      select case (linkatom)
      case ('B','Al','C','N','O','F','Si','P','S','Cl','As','Se', 'Br',
     &      'Te','I','At')
          ismetal = .false.
      case default
          ismetal = .true.
      end select
c
c     if metal, place into 'general' class and make driver atom
c     the first atom with angle .lt. 160. This choice of angle is
c     somewhat arbitrary, being midway between bent 120 and linear 180.
c
      if (ismetal.or.(valency.ge.5)) then
          do i=1, valency
              if (i12r(i,jj).eq.kk) cycle  ! ignore loser atom (kk)
              call findangle(x(kk),y(kk),z(kk),
     &                       x(jj),y(jj),z(jj),
     &                       x(i12r(i,jj)),y(i12r(i,jj)),z(i12r(i,jj)),
     &                       ang)

              if (ang.lt.160.0) then
                  ii = i12r(i,jj)
                  exit
              end if

          end do

          class = 15
          sub   = 1
          if (debug2) write(*,*) '==>', class, sub, ii
          return
      end if

c-----------------------------------------------------------------------
c                        VALENCY OF 4
c-----------------------------------------------------------------------

      if (valency.eq.4) then

          if (debug3) write(*,*) "=== Valency of 4 detected ==="
c
c     Initialize variables used for counting atom types connected to
c     the attach atom (jj).
c
          numhyd   = 0
          numdubox = 0
          numox    = 0
          numoth   = 0
```

```
c
c     Inspect all atoms connected to the link atom (jj).
c     The identity and/or the topology of these atoms determines the
c     dihedral class and subclass.
c

          do i=1, valency
c
c     Be sure to keep the loser atom, kk, among the substituents. It's
c     identity doesn't determine the rotor type.
c
                if (i12r(i,jj).eq.kk) cycle
c
c     First, look for monovalent atoms. Double bonded oxygens, sulfurs,
c     phosphoruses, etc. will be found here. While the variable names
c     are 'numdubox', 'duboxatt', etc., all signifying oxygen atoms,
c     this naming is for convenience only.
c
                if (n12r(i12r(i,jj)).eq.1) then
c
c     Don't count hydrogens with among other monovalent atoms. They are
c     handled differently, below.
c
                    if (atlab(i12r(i,jj)).ne.'H ') then
                        numdubox = numdubox + 1
                        duboxatt(numdubox) = i12r(i,jj)
                    else
c
c     Hydrogens
c
                        numhyd = numhyd + 1
                        hatt(numhyd) = i12r(i,jj)
                    end if
c
c     Divalent atoms possess similar dihedral profiles to monovalent
c     ones *if* its second substituent (not jj) is trans to the loser
c     atom (jj).
c
                else if (n12r(i12r(i,jj)).eq.2) then
c
c     Find the beta atom
c
                    do j=1, 2
                        if (i12r(j,i12r(i,jj)).eq.jj) cycle
                        ibeta1 = i12r(j,i12r(i,jj))
                    end do
c
c     Determine it's torsion angle with respect to the loser atom (jj)
c
                    call torsionval(x,y,z,kk,jj,i12r(i,jj), ibeta1, dihed)

                    if (dihed.ge.180.0e0) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (debug3) write(*,*) "Dihedral is ", dihed
c
c     If it's trans, count this as an 'oxygen'. named ox only for
c     convenience
c
                    if (dihed.ge.90.0e0) then
                        numox = numox + 1
                        oxatt(numox) = i12r(i,jj)
c
c     If not, make it an "other atom" (othatt).
c
                    else
                        if (debug3) write(*,*) "Numoth = ", numoth
                        numoth = numoth + 1
                        othatt(numoth) = i12r(i,jj)
                    end if
c
c     Everything else
```

```
c
              else
                  numoth = numoth + 1
                  othatt(numoth) = i12r(i,jj)
              end if

          end do

          if (debug3) then
              write(*,*) "numhyd   = ", numhyd
              write(*,*) "numoth   = ", numoth
              write(*,*) "numox    = ", numox
              write(*,*) "numbudox = ", numdubox
          end if
c
c     Determine linker atom atomic type. Determining the atom type is
c     important to distinguish between e.g. sulfones and phosphine
c     oxides, which have different dihedral potential energy surfaces
c     for some substituents.
c
          select case (linkatom)
c
c     Determine case for boron, carbon, nitrogen.
c     These tetrahedral elements are simpler to classify than those
c     for which double bonds must be tested, e.g. phosphorus and sulfur,
c     which are handled below.
c
          case ('B','C','N')
              if (debug3) then
                  write(*,*) "Tetravalent Boron, Carbon, or Nitrogen"
              end if
c
c     Determine class, subclass, and driver based upon number of H's
c
                  select case (numhyd)
                  case (3) ! 3 hydrogens
                      class = 1
                      sub   = 3
                      ii    = hatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (2) ! 2 hydrogens
                      class = 1
                      sub   = 1

                      if (numoth.eq.1) then
                          ii    = othatt(1)
                      else if (numox.eq.1) then
                          ii    = oxatt(1)
                      end if

                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (1) ! 1 hydrogen
                      class = 1
                      sub   = 2
                      ii    = hatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (0) ! 0 hydrogens
                      class = 1 ! 3 hydrogens and 0 hydrogens have the same
                      sub   = 3 ! dihedral potential energy surface

                      if (numox.ge.1) then
                          ii = oxatt(1)
                      else
                          ii = othatt(1)
                      end if

                      if (debug2) write(*,*) class, sub, ii
                      return
                  end select
c
```

```
c     Determine case for phosphorus or arsenic.
c     Find the oxygen atom, as it is not found,
c     then cases for 3, 2, or 1 hydrogens will be same as above.
c
          case ('P','As','Si')
c
c     If there's a monovalent atom or a divent atom with a trans beta,
c     (oxatt) make it the driver. In both cases, it's a class 8-1
c
                if (numdubox.ge.1) then
                    class = 8
                    sub   = 1
                    ii    = duboxatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
                else if (numox.gt.0) then
                    class=  8
                    sub   = 1
                    ii    = oxatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
c
c     If none are found, classify it as a generic tetrahedral
c
                else if (numhyd.eq.3) then
                    class = 1
                    sub   = 3
                    ii    = hatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
                else if (numhyd.eq.2) then
                    class = 1
                    sub   = 1
                    ii    = othatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
                else if (numhyd.eq.1) then
                    class = 1
                    sub   = 2
                    ii    = hatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
                else if (numhyd.eq.0) then
                    class = 1
                    sub   = 3
                    ii    = othatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
                else
                    class = 15
                    sub   = 1
                    ii    = othatt(1)
                end if
c
c     Cases for Sulfur, Selenium, and Tellurium. Again, the most
c     important case is the case in which there is oxygen attached.
c
          case ('S', 'Se', 'Te')
c
c     Monovalent non-hydrogens.
c
                if (numdubox.gt.0) then
                    class = 8
                    sub   = 4
                    ii    =  othatt(1)
                    if (debug2) write(*,*) class, sub, ii
                    return
c
c     Divalent atoms with second connection trans.
c
                else if (numox.gt.0) then
                    class = 8
                    sub   = 4
```

```
c
c     Determine which is driver.
c
                  select case (numox)
                  case (3)      ! 3 oxygens
                      ii    = oxatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (2)      ! 2 oxygens
c
c     Make the non-oxygen the driver to achieve symmetric surface.
c
                      if (numhyd.ge.1) then
                          ii = hatt(1)
                      else
                          ii = othatt(1)
                      end if

                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (1)      ! 1 oxygen
                      ii = oxatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  end select
              else
c
c     If there were no oxygens, then assign sulfur, selenium, and
c     tellurium as one would the other tetravalent organic species.
c
                  select case (numhyd)
                  case (3) ! 3 hydrogens
                      class = 1
                      sub   = 3
                      ii    = hatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (2) ! 2 hydrogens
                      class = 1
                      sub   = 1
                      ii    = othatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (1) ! 1 hydrogen
                      class = 1
                      sub   = 2
                      ii    = hatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  case (0) ! 0 hydrogens
                      class = 1
                      sub   = 3
                      ii    = othatt(1)
                      if (debug2) write(*,*) class, sub, ii
                      return
                  end select
              end if
          end select
c
c     If the dihedral type for a tetravalent linker atom hasn't yet
c     been found, default to class 15-1 and make the driver atom the
c     first atom which isn't the loser atom.
c
          numoth = 0

          do i=1,valency
              if (i12r(i,jj).eq.kk) cycle
              numoth = numoth + 1
              othatt(numoth) = i12r(i,jj)
          end do

          class = 15
          sub   = 1
```

```
          ii    = othatt(1)
          if (debug2) write(*,*) class, sub, ii
          return

      end if    ! end of if valency .eq. 4 block

c-----------------------------------------------------------------------
c                       COORDINATION OF 3
c-----------------------------------------------------------------------

      if (valency.eq.3) then
c
c     A valency of three can represent many different rotor groups. The
c     three main categories are trigonal planar cyclic, trigonal planar
c     non-cyclic, and trigonal pyramidal (trigonal pyramidal rings have
c     no special treatment; instead they are treated as a simple
c     trigonal pyramidal.
c
c     Pyramidal cases are straightforward to test. Planar cases require
c     extensive testing of the ortho positions and their substituents,
c     which are tested for
c       1. no substitution (lone pair)
c       2. hydrogen substituted
c       3. monovalent non-hydrogen substituted (ketone, etc.)
c       4. other (rotational minima acquired using a methyl group)
c
c     Because of the amount of testing involved with planar cases,
c     planar cyclics and planar non-cyclics are given their own
c     subroutines.
c
          if (debug3) write(*,*) "=== Valency of 3 detected ==="
c
c     Create an array of alpha atom serial numbers, without the
c     loser atom or dummies
c
          numalpha = 0

          do i = 1, n12r(jj)
              if (i12r(i,jj).eq.kk) cycle
              numalpha = numalpha + 1
              ialpha(numalpha) = i12r(i,jj)
          end do

          if (debug3) then
c
c     For debugging purposes, create an atom list of jj and its alphas.
c
              do i=1, 3
                  atlist(i) = i12r(i,jj)
              end do

              atlist(4) = jj
              write(*,*) "Atom list used to determine if trigonal ",
     &                   "host is planar is ", atlab(atlist(1:4))
          end if
c
c     Create an xyz list from the attach atom and it's alpha atoms.
c     This xyz list will be used to determine the planarity of the
c     attach atom (jj).
c
          do i=1, 3
              xlist(i) = x(i12r(i,jj))
              ylist(i) = y(i12r(i,jj))
              zlist(i) = z(i12r(i,jj))
          end do

          xlist(4) = x(jj)
          ylist(4) = y(jj)
          zlist(4) = z(jj)
c
c     Determine if attach atom is planar. A tolerance of 0.2 Angstrom
c     is employed. Below this threshold is planar. This value was
c     determined using the Cambridge Crystal Structure Database using
```

```
c     trivalent nitrogen as a test.
c
          call distance2plane(xlist, ylist, zlist, flat, debug3)

          if (debug3) write(*,*) "Results of planarity test = ", flat
c
c     Are there rings in the system?
c
          call findrings(r_atoms,n12r,i12r,jj,r_atoms,rings,
     &                   nring, iring)
c
c     Ensure that no alpha atom has four substituents.
c     In that case, it is a (partially) saturated ring which will be
c     treated as an alkene.
c
          if (flat.and.(rings.gt.0)
     &   .and.(n12r(ialpha(1)).le.3)
     &   .and.(n12r(ialpha(2)).le.3)) then

               if (debug3) write(*,*) "Host has a planar ring"
c
c     Perform assignments for flat rings.
c
               call assign_arene(r_atoms, jj, valency, kk, n12r,
     &                           i12r, atlab, class, sub, ii, debug3)
c
c     assign_arene subroutine returns class, sub, and ii therefore ...
c
               if (debug2) write(*,*) class, sub, ii
               return
          end if

c-------------------------------------------------------------------------
c                       ACYCLIC TRIGONAL PLANAR
c-------------------------------------------------------------------------

c
c     Perform tests in subroutine.
c
          if(flat.and.(rings.eq.0)) then
               call assign_trigplanar(r_atoms,x,y,z,jj,kk,n12r,i12r,
     &              atlab,class,sub,ii,ialpha,debug2)
               return
          end if

c-------------------------------------------------------------------------
c                        TRIGONAL PYRAMIDAL CASE
c-------------------------------------------------------------------------

c
c     For trigonal pyramidal cases, atom types must be tested because
c     e.g. P and N have very different bond angles, yield distinct
c     rotational potential energy surfaces.
c     The general case simply looks for Hydrogens and non-Hydrogens.
c     This is acceptable for all trigonal pyramidal cases except Sulfur,
c     Selenium, and Tellurium, many of which are double bonded to
c     e.g. Oxygen. There are no examples of trivalent, pyramidal Silicon
c     bonded to a monovalent atom.
c
          if (.not.flat) then
c
c     Count number of hydrogens and non-hydrogens
c
               numhyd = 0
               numoth = 0

               do i=1, valency

                    if (i12r(i,jj).eq.kk) cycle

                    if (atlab(i12r(i,jj)).eq.'H ') then
                         numhyd = numhyd + 1
                         hatt(numhyd) = i12r(i,jj)
```

```
                  else
                      numoth = numoth + 1
                      othatt(numoth) = i12r(i,jj)
                  end if

               end do
c
c     In the pyramidal case, we have nonsymmetric rotational
c     potential surfaces.  This means that we have to do some extra work
c     to determine which group is the driver when the R groups are the
c     same and which subclass to assign when they are different.
c
c     Orient the group such that the two R groups are on the z-axis,
c     one at the origin and one on the positive z axis.
c
c     If R groups are different, one C and one H, then the H goes
c     to the origin and the C goes to the positive z axis.
c
c     Atom numbering used in orientation is as follows:
c
c     1 = dummy
c     2 = R at origin
c     3 = R at + z-axis
c     4 = central pyramidal atom
c     5 = loser atom
c


c
c     Group 16 elements need to be tested for double bonded oxygens,
c     which affects their assignments differently than others.
c     Do not include them in this sectin of assignments.
c
            if(linkatom.ne.'S'
     &      .and.linkatom.ne.'Te'
     &      .and.linkatom.ne.'Se') then
c
c     Please dummy at origin
c
                  dx(1) = 0.0
                  dy(1) = 0.0
                  dz(1) = 0.0
c
c     Assign jj and kk coordinates
c
                  dx(4) = x(jj)
                  dy(4) = y(jj)
                  dz(4) = z(jj)

                  dx(5) = x(kk)
                  dy(5) = y(kk)
                  dz(5) = z(kk)


c
c     Determine symmetric and assymetric cases, determined by the
c     number of hydrogens
c
                  select case (numhyd)
                  case (2)       ! 2 hydrogens
                     dx(2) = x(hatt(1))
                     dy(2) = y(hatt(1))
                     dz(2) = z(hatt(1))

                     dx(3) = x(hatt(2))
                     dy(3) = y(hatt(2))
                     dz(3) = z(hatt(2))

                  case (1)       ! 1 hydrogen
                     dx(2) = x(hatt(1))
                     dy(2) = y(hatt(1))
                     dz(2) = z(hatt(1))

                     dx(3) = x(othatt(1))
                     dy(3) = y(othatt(1))
```

```
                  dz(3) = z(othatt(1))

               case (0)       ! 0 hydrogens
                  dx(2) = x(othatt(1))
                  dy(2) = y(othatt(1))
                  dz(2) = z(othatt(1))

                  dx(3) = x(othatt(2))
                  dy(3) = y(othatt(2))
                  dz(3) = z(othatt(2))

               end select

c
c     Align the loser atom (kk) along the z-axis.
c
               call alignz(5,dx,dy,dz,2,3,1)
c
c     Now we rotate on the z-axis to place the central atom, d4,
c     in the xz plane.  Requires the use of a dummy atom, d1.
c
               dx(1) = 1.0
               dy(1) = 0.0
               dz(1) = 0.0

               call finddihedral(dx,dy,dz,1,2,3,4,phi)

               ang = 0.0

               call setdihedral(1,5,dx,dy,ang,phi)
            end if
c
c     The sign of the y coordinate of the loser atom, the fifth atom in
c     the dx,dy,dz arrays, provides the criterion for assignments
c
c     Symmetric:
c     y value positive, ii = hatt(2) or catt(2)  (this is the down form)
c     y value negative, ii = hatt(1) or catt(1)  (this is the up form)
c        two hydrogens, subclass = 1
c        two carbons, subclass = 4
c
c    Asymmetric:
c        y value positive, subclass = 2   (this is down form)
c        y value negative, subclass = 3   (this is up form)
c        in either event, ii = catt(1)
c

c
c     Perform assignments based upon link atom element identity
c
            select case (linkatom)
c
c     Nitrogen
c
            case ('N')
                class = 6     ! pyramidal nitrogen is always class 6
c
c     Assign based upon the number of hydrogens.
c
                select case (numhyd)
                case (0)      ! 0 hydrogens
                    sub = 4

                    if (dy(5) .gt. 0.00e0) then
                        ii     = othatt(2)
                        alt_ii = othatt(1)
                    else
                        ii     = othatt(1)
                        alt_ii = othatt(2)
                    end if

                    if (debug2) write(*,*) class, sub, ii, alt_ii
                    return
```

```
                    case (1)      ! 1 hydrogen
                        ii = othatt(1)

                        if (dy(5) .gt. 0.00e0) then
                            sub = 2
                        else
                            sub = 3
                        end if

                        if (debug2) write(*,*) class, sub, ii
                        return

                    case (2)      ! 2 hydrogens
                        sub   = 1

                        if (dy(5) .gt. 0.00e0) then
                            ii     = hatt(2)
                            alt_ii = hatt(1)
                        else
                            ii     = hatt(1)
                            alt_ii = hatt(2)
                        end if

                        if (debug2) write(*,*) class, sub, ii, alt_ii
                        return

                    end select    ! end of nitrogen tests

c
c     Phosphorus or Arsenic
c
                case ('P', 'As')
                    class = 9       ! always class 9
                    select case (numhyd)
                    case (0)      ! 0 hydrogens
                        sub = 4

                        if (dy(5) .gt. 0.00e0) then
                            ii     = othatt(1)
                            alt_ii = othatt(2)
                        else
                            ii     = othatt(2)
                            alt_ii = othatt(1)
                        end if

                        if (debug2) write(*,*) class, sub, ii, alt_ii
                        return

                    case (1)      ! 1 hydrogen
                        ii = othatt(1)

                        if (dy(5) .gt. 0.00e0) then
                            sub = 3
                        else
                            sub = 2
                        end if

                        if (debug2) write(*,*) class, sub, ii
                        return

                    case (2)      ! 2 hydrogens
                        sub   = 1

                        if (dy(5) .gt. 0.00e0) then
                            ii     = hatt(1)
                            alt_ii = hatt(2)
                        else
                            ii     = hatt(2)
                            alt_ii = hatt(1)
                        end if

                        if (debug2) write(*,*) class, sub, ii, alt_ii
```

```
                        return

                    end select    ! end of P or At tests

c
c     Sulfur, Selenium, Tellurium
c
            case ('S', 'Se', 'Te')
                class = 8
c
c     Search for monovalent non-hydrogens (dubox) and their protonated
c     forms (numox, must be trans to loser). All others go into numoth.
c
                    numdubox = 0
                    numox    = 0
                    numoth   = 0

                    do i=1, valency
                        if (i12r(i,jj).eq.kk) cycle
c
c     Monovalent nonhydrogens
c
                        if (n12r(i12r(i,jj)).eq.1
     &                  .and.atlab(i12r(i,jj)).ne.'H ') then
                            numdubox = numdubox + 1
                            duboxatt(numdubox) = i12r(i,jj)
c
c     Divalent atoms possess similar dihedral profiles to monovalent
c     ones *if* its second substituent (not jj) is trans to the loser
c     atom (jj).
c
                        else if (n12r(i12r(i,jj)).eq.2) then
c
c     Find the beta atom
c
                            do j=1, 2
                                if (i12r(j,i12r(i,jj)).eq.jj) cycle
                                ibeta1 = i12r(j,i12r(i,jj))
                            end do
c
c     Determine it's torsion
c
                            call torsionval(x,y,z,kk,jj,i12r(i,jj),
     &                           ibeta1, dihed)

                            if (dihed.ge.180.0e0) then
                                dihed = 360.0e0 - dihed
                            end if
c
c     If it's trans, count this as an 'oxygen'. named ox only for
c     convenience
c
                            if (dihed.ge.90.0e0) then
                                numox = numox + 1
                                oxatt(numox) = i12r(i,jj)
                            else
                                numoth = numoth + 1
                                othatt(numoth) = i12r(i,jj)
                            end if

                        else
                            numoth = numoth + 1
                            othatt(numoth) = i12r(i,jj)
                        end if

                    end do
c
c     Chirality test. Atom numbering used in orientation is as follows:
c
c     1 = dummy
c     2 = double bonded oxygen
c     3 = R at + z-axis
c     4 = central pyramidal atom
```

```
c     5 = loser atom
c
                dx(1) = 0.0
                dy(1) = 0.0
                dz(1) = 0.0
c
c     Because the link atom (jj) has two substituents (not counting kk)
c     they will either be two dubox (not likely), one dubox + one oxatt,
c     or a combination of dubox, oxatt, or othatt.
c     In the following if block, if there is no duboxatt's and no
c     oxatt's, then there must be two othatt's, and so the following
c     tests hold in all cases.
c
c     For assigning the driver atom (ii), asymmetric cases are assigned
c     immediately. Symmetric cases must undergo the chirality test.
c     Since in some cases drivers aren't assigned, set ii = -1 and then
c     test for it below.
c
                ii = -1

                if (numdubox.eq.1) then
                    dx(2) = x(duboxatt(1))
                    dy(2) = y(duboxatt(1))
                    dz(2) = z(duboxatt(1))

                    dx(3) = x(othatt(1))
                    dy(3) = y(othatt(1))
                    dz(3) = z(othatt(1))

                    ii = duboxatt(1)

                else if ((numdubox.eq.1).and.(numox.eq.1)) then
                    dx(2) = x(duboxatt(1))
                    dy(2) = y(duboxatt(1))
                    dz(2) = z(duboxatt(1))

                    dx(3) = x(oxatt(1))
                    dy(3) = y(oxatt(1))
                    dz(3) = z(oxatt(1))

                    ii = duboxatt(1)

                else if (numdubox.eq.2) then
                    dx(2) = x(duboxatt(1))
                    dy(2) = y(duboxatt(1))
                    dz(2) = z(duboxatt(1))

                    dx(3) = x(duboxatt(2))
                    dy(3) = y(duboxatt(2))
                    dz(3) = z(duboxatt(2))

                else if (numox.eq.2) then
                    dx(2) = x(oxatt(1))
                    dy(2) = y(oxatt(1))
                    dz(2) = z(oxatt(1))

                    dx(3) = x(oxatt(2))
                    dy(3) = y(oxatt(2))
                    dz(3) = z(oxatt(2))

                else if (numox.eq.1) then
                    dx(2) = x(oxatt(1))
                    dy(2) = y(oxatt(1))
                    dz(2) = z(oxatt(1))

                    dx(3) = x(othatt(1))
                    dy(3) = y(othatt(1))
                    dz(3) = z(othatt(1))

                    ii = oxatt(1)

                else
                    dx(2) = x(othatt(1))
```

```
                    dy(2) = y(othatt(1))
                    dz(2) = z(othatt(1))

                    dx(3) = x(othatt(2))
                    dy(3) = y(othatt(2))
                    dz(3) = z(othatt(2))

                end if

                dx(4) = x(jj)
                dy(4) = y(jj)
                dz(4) = z(jj)

                dx(5) = x(kk)
                dy(5) = y(kk)
                dz(5) = z(kk)

                call alignz(5,dx,dy,dz,2,3,1)
c
c     Now we rotate on the z-axis to place the central atom, d4,
c     in the xz plane.  Requires the use of a dummy atom, d1.
c
                dx(1) = 1.0
                dy(1) = 0.0
                dz(1) = 0.0

                call finddihedral(dx,dy,dz,1,2,3,4,phi)

                ang = 0.0

                call setdihedral(1,5,dx,dy,ang,phi)
c
c     The class has already been assigned (all of these cases are for
c     class 8, specifically S, Se, Te). Determine the subclass based
c     upon the loser atom's y value. If ii.eq.-1, then it was never
c     assigned in the above if clause. Test for ii.eq.-1 and assign the
c     driver atom (ii) if needed.
c
                if (dy(5).gt.0.00e0) then
                    sub = 2

                    if (ii.eq.-1) then
                        if (numdubox.eq.2) then
                            ii = duboxatt(2)
                        else if (numox.eq.2) then
                            ii = oxatt(2)
                        else if (numoth.eq.2) then
                            ii = othatt(2)
                        end if
                    end if

                else
                    sub = 3

                    if (ii.eq.-1) then
                        if (numdubox.eq.2) then
                            ii = duboxatt(1)
                        else if (numox.eq.2) then
                            ii = oxatt(1)
                        else if (numoth.eq.2) then
                            ii = othatt(1)
                        end if
                    end if

                end if

                if (debug2) write(*,*) class, sub, ii
                return
c
c     The class and subclass haven't been found. Return the general case.
c
                class = 15
                sub   = 1
```

```
                if (numoth.ge.1) then
                    ii = othatt(1)
                else
                    ii = hatt(1)
                end if

                if (debug2) write(*,*) class, sub, ii
                return

            end select     ! end of selecting based upon element
        end if    ! end of trigonal pyramidal block
      end if    ! end of valency .eq. 3 block

c-----------------------------------------------------------------------
c                     COORDINATION OF 2.
c-----------------------------------------------------------------------
c
c     Divalent atoms may be linear or bent. Therefore a measurement of
c     the angle between the loser, the attach atom, and it's sole
c     substituent must be tested. Angles less than 150 degrees will be
c     considered bent. Otherwise, it is considered linear.
c
      if (valency .eq. 2) then

          if (debug3) write(*,*) "=== Valency of 2 detected ==="
c
c     Find alpha atom serial number (not counting loser kk)
c
          do i=1,2
              if (i12r(i,jj).eq.kk) cycle
              ialpha1 = i12r(i,jj)
          end do

          if (debug3) then
              write(*,*) 'ialpha1 is ', ialpha1
              write(*,*) 'which is a         ', atlab(ialpha1)
          end if
c
c     Find the angle between the loser atom, the attach atom, and the
c     substituent
c
          call findangle(x(jj),y(jj),z(jj),x(kk),y(kk),z(kk),
     &                   x(ialpha1),y(ialpha1),z(ialpha1),ang)
          if (debug3) then
              write(*,*) 'Angle for valency 2 was found to be ', ang
          end if
c
c     Consider a bond angle >= 150 linear.
c
          if (ang.ge.150.0e0) then  ! is linear
c
c     If angle is equal to 180.0, slightly adjust the position of
c     the link atom (jj), so that the dihedral isn't undefined.
c
              if (ang.eq.180.0e0) then
                  x(ialpha1) = x(ialpha1) + 0.02
                  y(ialpha1) = y(ialpha1) + 0.02
              end if

              class = 15
              sub   = 1
              ii    = ialpha1
              if (debug2) write(*,*) class, sub, ii
              return
          end if
c
c     The angle isn't "linear", therefore further characterization is
c     needed. Find nature of atoms attached to attach atom (jj).
c
c     Count hyrogens and not-hydrogens and create arrays of their
c     serial numbers.
c
```

```
          numhyd = 0
          numoth = 0

          do i=1, 2
              if (i12r(i,jj).eq.kk) cycle

              if (atlab(i12r(i,jj)).eq.'H ') then
                  numhyd = numhyd + 1
                  hatt(numhyd) = i12r(i,jj)
              else
                  numoth = numoth + 1
                  othatt(numoth) = i12r(i,jj)
              end if

          end do
c
c     Oxygens have a much larger angle than other divalent atoms. Test
c     for it.
c
          select case (linkatom)
          case('O')

              if (numhyd.gt.0) then
                  class = 5
                  sub   = 2
                  ii    = hatt(1)
                  if (debug2) write(*,*) class, sub, ii
                  return
              else
                  class = 5
                  sub   = 1
                  ii    = othatt(1)
                  if (debug2) write(*,*) class, sub, ii
                  return
              end if

          case('S','Se','Te')

              if (numhyd.gt.0) then
                  class = 5
                  sub   = 4
                  ii    = hatt(1)
                  if (debug2) write(*,*) class, sub, ii
                  return
              else
                  class = 5
                  sub   = 3
                  ii    = othatt(1)
                  if (debug2) write(*,*) class, sub, ii
                  return
              end if
c
c     Nitrogen, Phosphorus, Arsenic are assumed to be double bonded
c     to alpha atom, which will also have beta atom(s). Therefore,
c     the dihedral angle of kk, jj, alpha1, beta1 need to be determined.
c
          case('N','P','As')
c
c     Determine number of betas on alpha atom and their serial numbers
c
              numbeta = 0

              do i=1, n12r(ialpha1)
                  if (i12r(i,ialpha1).eq.jj) cycle
                  numbeta = numbeta + 1
                  ibeta(numbeta,1) = i12r(i,ialpha1)
              end do

              if (debug3) then
                  write(*,*) "Found ",numbeta, " beta atoms"
                  write(*,*) "which are ", ibeta(1:numbeta,1)
              end if
c
```

```
c     Determine if any beta is cis.
c
              has_cisbeta = .false.

              do i=1, numbeta
                  call torsionval(x, y, z, kk, jj, ialpha1,
     &                            ibeta(i,1), dihed)

                  if (dihed.gt.180.0e0) then
                      dihed = 360.0e0 - dihed
                  end if

                  if (debug3) then
                      write(*,*) 'Dihedral angle of beta ', ibeta(i,1),
     &                           ' is ', dihed
                  end if

                  if (dihed.ge.0.0e0.and.dihed.le.90.e0) then ! cis

                      if (debug3) then
                          write(*,*) "Beta atom", ibeta(i,1),"is cis"
                      end if

                      has_cisbeta = .true.
                      ibetacis   = ibeta(i,1)

                      if (atlab(ibetacis).eq.'H ') then
                          cisbeta_isH = .true.
                      end if

                      if (debug3) then
                          write(*,*) "ibetacis is ", ibetacis
                          write(*,*) "betacis is  ", betacis
                      end if

                      exit ! found a cis, break from do loop

                  end if
              end do

c
c     if there's a beta atom which is cis to the loser atom, then it's
c     either a 10-3 rotor or a 12-1 rotor.
c             #   cisbeta == H --> 10-3  else  --> 12-1
c     If there isn't a cis beta atom, it's a 10-4 rotor.
c
              if (has_cisbeta) then ! there is a cis sub

                  if (cisbeta_isH) then  ! the cis sub' is H
                      class = 10
                      sub   = 3
                      ii    = ialpha1
                      if (debug2) write(*,*) class, sub, ii
                      return
                  else                        ! the cis sub' isn't H
                      class = 12
                      sub   = 1
                      ii    = ialpha1
                      if (debug2) write(*,*) class, sub, ii
                      return
                  end if

              else                            ! there is no cis sub'
                  class = 10
                  sub   = 4
                  ii    = ialpha1
                  if (debug2) write(*,*) class, sub, ii
                  return
              end if

          case default
              class = 15
              sub   = 1
```

```
              ii    = ialpha1
              if (debug2) write(*,*) class, sub, ii
              return
          end select
      end if  ! end of if valency = 2

c-----------------------------------------------------------------------
c                 Valency of 1. Terminal case.
c-----------------------------------------------------------------------

      if (valency.eq.1) then
          class = 16
          sub   = 1
          ii    = 1
          if (debug2) write(*,*) class, sub, ii
          return
      end if

c-----------------------------------------------------------------------
c               Valency of 0. Mission Abort.
c-----------------------------------------------------------------------
      if (valency .eq. 0) then
          write(*,*) "*** ABORTING FROM ASSIGNCLASS.F. ***"
          write(*,*) "*** Linker atom has valency of zero!! ***"
          stop
      end if

c-----------------------------------------------------------------------
c          Made it to the end without assigning a rotor type.
c-----------------------------------------------------------------------
      stop "Could not determine rotor type"

      end subroutine

c-----------------------------------------------------------------------
c          SUBROUTINE TO ASSIGN FLAT RING CLASS AND SUBCLASS
c
c     The rotational potential energy surface of flat rings is
c     determined by the nature of the ortho atoms. In this subroutine,
c     the ortho positions are characterized as:
c       1. unsubstituted: is divalent
c       2. hydrogen substituted
c       3. non-hydrogen substituted
c
c     Both ortho positions are analyzed and the result may be any
c     combination of the above.
c-----------------------------------------------------------------------

      subroutine assign_arene(numatoms, attachatom, atomval, loseratom,
     &                        valence_arr, serial_arr, atlabels,
     &                        dihed_class, subclass, driver, debug)

      implicit none

      integer numatoms                          !  = r_atoms
      integer i                                 !  indexing variable
      integer attachatom                        !  = jj
      integer atomval                           !  = valency (n12r(jj))
      integer loseratom                         !  = kk
      integer valence_arr(numatoms)             ! = n12r
      integer serial_arr(10, numatoms)          ! = i12r
      character*2 atlabels(numatoms)            ! = atlab
      integer dihed_class                       ! = class
      integer subclass                          ! = sub
      integer driver                            ! = ii
c
c     Variables concerning alpha positions
c
      integer numortho                     ! number of alplha atoms
      integer iortho(3)                    ! ortho atom serial numbers
      integer iortho1                      ! serial number of 1st ortho
      integer iortho2                      ! serial number of 2nd ortho
c
```

```
c     Logical variables used to characterize ortho positions
c
      logical ortho1hyd, ortho2hyd      ! true if H attached at ortho
      logical ortho1unsub, ortho2unsub  ! true if ortho unsubstituted
      logical ortho1sub, ortho2sub      ! true if ortho substituted

      logical debug

c
c     Find serial numbers of ortho positions, their coordination numbers
c     , and the number of H's attached to them.
c
      numortho = 0

      do i=1, atomval
          if (serial_arr(i,attachatom).eq.loseratom) cycle
          numortho = numortho + 1
          iortho(numortho) = serial_arr(i,attachatom)
      end do

      iortho1 = iortho(1)
      iortho2 = iortho(2)

      if (debug) then
          write(*,*) 'ortho subsitutents         = ', numortho
          write(*,*) 'Serial number of them are  = ', iortho(1:numortho)
          write(*,*) 'ortho1 is                  = ', iortho1
          write(*,*) 'ortho2 is                  = ', iortho2
          write(*,*) 'coordination of ortho1     = ',
     &                                        valence_arr(iortho1)
          write(*,*) 'connection of ortho1       = ',
     &                atlabels(serial_arr(1:valence_arr(iortho1),
     &                                    iortho1))
          write(*,*) 'coordination of ortho2     = ',
     &                 valence_arr(iortho2)
          write(*,*) 'connection of ortho2       = ',
     &                 atlabels(serial_arr(1:valence_arr(iortho2),
     &                          iortho2))
          write(*,*)
       end if
c
c     Initialize all ortho related variables to false.
c
      ortho1hyd   = .false.
      ortho2hyd   = .false.
c
c     Ortho is considered unsubstitued if it has a connectivity of
c     two or if its 3rd substituent is hydrogen
c
      ortho1unsub = .false.
      ortho2unsub = .false.
c
c     Ortho positions considered substituted only if its substituent
c     isn't hydrogen.
c
      ortho1sub   = .false.
      ortho2sub   = .false.
c
c     Test the valency of the ortho positions. Valencies of two means
c     that it's unsubstituted. Valencies of three will be tested for an
c     Hydrogen. If there's no hydrogen, it's considered substituted,
c     i.e. orthoXunsub = .false. and orthoXsub = .true.
c
      select case (valence_arr(iortho1))
      case(2)
          ortho1unsub = .true.
      case(3)
c
c     Loop through the atom labels of the serial number array, looking
c     for 'H ' for Hydrogen.
c
          do i=1,3
```

```
                if (atlabels(serial_arr(i,iortho1)) .eq. 'H ') then
                    ortho1hyd   = .true.
                    ortho1unsub = .true.
                end if

            end do
        end select

        select case (valence_arr(iortho2))
        case(2)
            ortho2unsub = .true.
        case(3)
            do i=1,3

                if (atlabels(serial_arr(i,iortho2)).eq.'H ') then
                    ortho2hyd   = .true.
                    ortho2unsub = .true.
                end if

            end do
        end select
c
c       Invert the logical value of orthoXsub for easier testing below.
c
        ortho1sub = .not.ortho1unsub
        ortho2sub = .not.ortho2unsub

        if (debug) then
            write(*,*) 'Is ortho1 unsubd? ', ortho1unsub
            write(*,*) 'Is ortho1 subd?   ', ortho1sub
            write(*,*) 'H on ortho1?      ', ortho1hyd
            write(*,*) 'Is ortho2 unsubd? ', ortho2unsub
            write(*,*) 'Is ortho2 subd?   ', ortho2sub
            write(*,*) 'H on ortho2?      ', ortho2hyd
        end if
c
c               Begin Assignment of Flat Rings
c
        if (ortho1hyd.and.ortho2hyd) then
c
c       Hydrogens on both ortho positions
c
             dihed_class = 3
             subclass    = 1
             driver      = iortho1
             if (debug) write(*,*) dihed_class, subclass, driver
             return
c
c       Because the assignment of which ortho is ortho1 and which is
c       ortho2 is arbitrary, asymmetric cases must be tested twice
c
        elseif (ortho1hyd.and.ortho2sub) then
c
c       Ortho1 is hydrogen substituted; ortho2 is substituted
c
            dihed_class = 3
            subclass    = 2
            driver      = iortho1
            if (debug) write(*,*) dihed_class, subclass, driver
            return
c
c       Opposite of previous
c
        elseif (ortho2hyd.and.ortho1sub) then
            dihed_class = 3
            subclass    = 2
            driver      = iortho1
            if (debug) write(*,*) dihed_class, subclass, driver
            return
c
c       Both ortho's substituted.
c
        elseif (ortho1sub.and.ortho2sub) then
```

```
          dihed_class = 3
          subclass    = 3
          driver      = iortho1
          if (debug) write(*,*) dihed_class, subclass, driver
          return
c
c     Ortho1 has is not substituted at all; Ortho2 has a Hydrogen
c
      elseif (ortho1unsub.and.ortho2hyd) then
          dihed_class = 13
          subclass    = 1
          driver      = iortho1
          if (debug) write(*,*) dihed_class, subclass, driver
          return
c
c     Reverse of previous.
c
      elseif (ortho2unsub.and.ortho1hyd) then
          dihed_class = 13
          subclass    = 1
          driver      = iortho2
          if (debug) write(*,*) dihed_class, subclass, driver
          return
c
c     Both ortho's unsubstituted.
c
      elseif (ortho1unsub .and. ortho2unsub) then
          dihed_class = 13
          subclass    = 2
          driver      = iortho1
          if (debug) write(*,*) dihed_class, subclass, driver
          return
c
c     Both ortho's unsubstituted.
c
      elseif (ortho1unsub .and. ortho2sub) then
          dihed_class = 13
          subclass    = 3
          driver      = iortho1
          if (debug) write(*,*) dihed_class, subclass, driver
          return
c
c     Otho2 is unsubstituted; ortho1 is substituted.
c
      elseif (ortho2unsub .and. ortho1sub) then
          dihed_class = 13
          subclass    = 3
          driver      = iortho1
          if (debug) write(*,*) dihed_class, subclass, driver
          return
c
c     Tests have been exhausted, yet the rotor hasn't been assigned.
c     Default to general class, 15-1.
c
      else
          dihed_class = 15
          subclass    = 1
          driver      = iortho1
          if (debug) write(*,*) dihed_class, subclass, driver
          return
      end if

      end subroutine assign_arene

c-----------------------------------------------------------------------
c        SUBROUTINE TO ASSIGN TRIGONAL PLANAR CLASS AND SUBCLASS
c
c     The rotational potential energy surface of flat rings is
c     determined by the nature of the alpha atoms. In this subroutine,
c     the alpha positions are characterized primarily by their valency
c     minus one (this accounts for the alpha atom being bonded back to
c     the attach atom (jj)).
c       1. Unsubstituted:
```

```
c          a) Hydrogen attached directly to attach atom (jj), such as a
c             aldehyde.
c          b) Monovalent atom, such as a double bonded oxygen or a
c             halogen.
c       2. hydrogen substituted
c       3. non-hydrogen substituted
c
c     Both ortho positions are analyzed and the result may be any
c     combination of the above.
c-----------------------------------------------------------------------

      subroutine assign_trigplanar(r_atoms,x,y,z,jj,kk,n12r,i12r,
     &                             atlab, class, sub, ii, ialpha, debug)

      implicit none
      integer r_atoms
      real x(r_atoms)
      real y(r_atoms)
      real z(r_atoms)
      integer ii                      ! driver atom
      integer jj                      ! attach atom
      integer kk                      ! loser atom
      integer n12r(r_atoms)
      integer i12r(10, r_atoms)

      character*2 atlab(r_atoms)
      integer class
      integer sub
      integer ialpha(2)
      logical debug

      integer ialpha1                 ! serial # of alpha1 atom
      integer ialpha2                 ! serial # of alpha2 atom
      integer numbeta                 ! dumb indexer of beta atoms
      integer nbeta(2)                ! array containing number of betas
                                      ! on each alpha
      integer nbeta1                  ! number of betas on alpha1
      integer nbeta2                  ! number of betas on alpha2
      integer ibeta(r_atoms,r_atoms)  ! array containing serial #s of
                                      ! of beta atoms on each alpha
      integer ibeta1(r_atoms)         ! serial #s of beta atoms on alpha1
      integer ibeta2(r_atoms)         ! serial #s of beta atoms on alpha2
      logical has_cisbeta1            ! true if a beta1 is cis to kk
      logical has_cisbeta2            ! true if a beta2 is cis to kk
      logical beta1_isH               ! true if there is a beta2 H atom
      logical beta2_isH               ! true if there is a beta2 H atom
      logical cisbeta1_isH            ! true if beta1 H atom is cis
      logical cisbeta2_isH            ! true if beta2 H atom is cis
      integer icisbeta1               ! serial number of the cis beta1
                                      ! atom
      integer icisbeta2               ! serial number of the cis beta2
                                      ! atom
      integer itransbeta1             ! serial # of trans beta on alpha1
      integer itransbeta2             ! serial # of trans beta on alpha2
      real dihed                      ! used in dihedral angle calc's

      integer i, j                    ! dumb indexing variables

      has_cisbeta1 = .false.
      has_cisbeta2 = .false.
      beta1_isH    = .false.
      beta2_isH    = .false.
      cisbeta1_isH = .false.
      cisbeta2_isH = .false.
      icisbeta1 = 0
      icisbeta2 = 0
      itransbeta1 = 0
      itransbeta2 = 0
c
c     Because a certain alpha atom may only be connected back to jj, it
c     may have 0 beta atoms. In that case, the 2 dimensional ibeta array
c     will not be initialized. Therefore, initialize it manually and if
c     the alpha atom(s) have betas then they will be placed in the
```

```
c     array.
c
      ibeta   = 0
c
c     now, create arrays containing the number of beta atoms on the two
c     alpha atoms, not counting the attach atom jj. Also, create an
c     array of the serial numbers of the beta atoms on the alphas.
c
      do i = 1, 2
          nbeta(i)  = n12r(ialpha(i)) - 1   ! subtract one to account
                                            ! for bonding to jj
          if (nbeta(i).eq.0) cycle  ! beta is only connected to jj
                                    ! done't create beta arrays
          numbeta = 0

          do j=1, n12r(ialpha(i)) ! elsewise, not all betas are sampled
              if (i12r(j,ialpha(i)).eq.jj) cycle ! keep jj from array
              numbeta = numbeta + 1
              ibeta(numbeta,i)  = i12r(j,ialpha(i))
          end do

      end do
c
c     if the link atom is attached to a hydrogen, make it alpha1
c     for easier testing.  otherwise, make alpha1 the alpha with
c     the lowest number of substituents. if number of substituents
c     is equal for both alphas, make alpha1 simply ialpha(1).
c
      if (atlab(ialpha(1)).eq.'H ') then
          ialpha1 = ialpha(1)
          nbeta1  = nbeta(1)
          ibeta1  = ibeta(:,1)
          ialpha2 = ialpha(2)
          nbeta2  = nbeta(2)
          ibeta2  = ibeta(:,2)
      elseif (atlab(ialpha(2)).eq.'H ') then
          ialpha1 = ialpha(2)
          nbeta1  = nbeta(2)
          ibeta1  = ibeta(:,2)
          ialpha2 = ialpha(1)
          nbeta2  = nbeta(1)
          ibeta2  = ibeta(:,1)
      elseif (n12r(ialpha(1)).le.n12r(ialpha(2))) then
          ialpha1 = ialpha(1)
          nbeta1  = nbeta(1)
          ibeta1   = ibeta(:,1)
          ialpha2 = ialpha(2)
          nbeta2  = nbeta(2)
          ibeta2  = ibeta(:,2)
      else
          ialpha1 = ialpha(2)
          nbeta1  = nbeta(2)
          ibeta1  = ibeta(:,2)
          ialpha2 = ialpha(1)
          nbeta2  = nbeta(1)
          ibeta2  = ibeta(:,1)
      end if
c
c     The end result is that fewer cases need to be tested. For example,
c     if there are two betas on alpha 1, then there is no need to test
c     for 1 beta on alpha 2.
c
      if (debug) then
          write(*,*)
          write(*,*)'========== ALPHA & BETA INFO ============='
          write(*,*)
          write(*,'(A21,A2)') 'alpha1  is           ', atlab(ialpha1)
          write(*,'(A21,I2)') '# betas on alpha1  = ', nbeta1
          write(*,*)
          write(*,'(A21,A2)') 'alpha2  is           ', atlab(ialpha2)
          write(*,'(A21,I2)') '# betas on alpha2  = ', nbeta2
          write(*,*)
          write(*,*) '========================================='
```

```
            write(*,*)
        end if
c
c       Begin logic tree by determining the coordination numbers of the
c       alpha atom with the fewest substituents
c
        select case (nbeta1)  ! how many betas on alpha1 ?
        case (0)              ! zero betas on alpha1

            if (debug) write(*,*) "0 betas on ialpha1"

            if (atlab(ialpha1).eq.'H ') then ! if alpha1 beta is an H

                if (debug) write(*,*) "ialpha1 is an H."

                select case (nbeta2) ! how many betas on alpha2?
                case(0)              ! zero betas on alpha2

                    if (debug) write(*,*) "0 betas on ialpha2."

                    select case (atlab(ialpha2)) ! atom label of ibeta2
                    case ('H ')

                        if (debug) write(*,*) "ialpha2 is an H."
c
c       Both alpha atoms are hydrogen.
c
                        class = 14
                        sub   = 1
                        ii    = ialpha2
                        if (debug) write(*,*) class, sub, ii
                        return

                    case ('O ')

                        if (debug) write(*,*) "ialpha2 is an O."
c
c       One alpha is an H and the other is an oxygen. ==> Aldehyde
c
                        class = 10
                        sub   = 1
                        ii    = ialpha2
                        if (debug) write(*,*) class, sub, ii
                        return

                    case default
c
c       One alpha is an H, and the other is only connected to the attach
c       atom jj, but it isn't another H and it isn't an aldehyde. Assign
c       it to be a "sulfaldehyde".
c
                        class = 10
                        sub   = 2
                        ii    = ialpha2
                        if (debug) write(*,*) class, sub, ii
                        return

                    end select ! end of ibeta2 atom label testing

                case (1)  ! 1 beta on alpha2

                    if (debug) write(*,*) "Alpha2 has 1 beta."
c
c       alpha2 is connected to 1 other atom. test if it other atom
c       is cis to the loser atom.
c
                    call torsionval(x,y,z,kk,jj,ialpha2,ibeta2(1),dihed)

                    if (dihed.gt.180.0e0) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
```

```
                    if (debug) write (*,*) "Beta atom is cis"

                    has_cisbeta2 = .true.
                    icisbeta2    = ibeta2(1)

                    if (atlab(ibeta2(1)).eq.'H ') then
                        cisbeta2_isH = .true.
                    end if

                end if

                if (has_cisbeta2) then

                    if (cisbeta2_isH) then
                        class = 2
                        sub   = 1
                        ii    = ialpha2
                        if (debug) write(*,*) class, sub, ii
                        return
                    else
                        class = 2
                        sub   = 2
                        ii    = ialpha2
                        if (debug) write(*,*) class, sub, ii
                        return
                    end if

                else      ! beta 2 is trans
                    class = 10
                    sub   = 1
                    ii    = ialpha2
                    if (debug) write(*,*) class, sub, ii
                    return
                end if
                ! end of case 1, where alpha2 has 1 beta

            case (2)      ! alpha 2 has 2 betas

                if (debug) write(*,*) "YESSS! Alpha 2 has 2 betas."
c
c     Inspect the first beta atom connected to alpha 2. It is either
c     cis or trans.
c
                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                              ibeta2(i), dihed)
                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if(atlab(icisbeta2).eq.'H ') then
                            cisbeta2_isH = .true.
                            if (debug) write(*,*) 'Yeehaw! It be H'
                        end if

                    else
                        itransbeta2 = ibeta2(i)
                    end if

                end do

                if (cisbeta2_isH) then
                    class = 2
                    sub   = 1
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                elseif (n12r(icisbeta2).eq.1) then
                    class = 4
```

```
                    sub   = 1
                    ii    = ialpha2
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                elseif (n12r(itransbeta2).eq.1 .and.
     &                  atlab(itransbeta2).ne.'H ') then
                    class = 4
                    sub   = 2
                    ii    = ialpha2
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                else
                    class = 2
                    sub   = 2
                    ii    = ialpha2
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (3)       ! alpha 2 has 3 betas
c
c     With a hydrogen for alpha 1 and a tertiary alpha 2, this is
c     similar to methyl borane.
c
                if (debug) write(*,*) "alpha 2 has 3 betas."
                class = 14
                sub   = 2
                ii    = ialpha2
                if (debug) write(*,*) "===>",class, sub, ii
                return

            case default
                class = 15
                sub   = 1
                ii    = ialpha2
                if (debug) write(*,*) "===>",class, sub, ii
                return

            end select
c
c     End of if alpha1 beta is an H
c
c     There are zero betas on alpha 1, but it isn't an H. Could be a
c     carbonyl, halide, etc.
c
        else
            select case (nbeta2)
c
c     Begin intial assessment based upon the number of betas on alpha2
c
            case (0)   ! 0 betas on alpha 2

                 if (debug) write(*,*) "Alpha 2 has 0 betas."
                 class = 13
                 sub   = 2
                 ii    = ialpha2
                 if (debug) write(*,*) "===>",class, sub, ii
                 return

            case (1)   ! 1 beta on alpha 2

                 if (debug) write(*,*) "Alpha 2 has 1 beta."
c
c     alpha2 is connected to 1 other atom. test if it is cis
c     to the loser atom
c
                 call torsionval(x,y,z,kk,jj,ialpha2,ibeta2(1),dihed)

                 if (dihed.gt.180.0e0) then
                     dihed = 360.0e0 - dihed
                 end if

                 if (dihed.ge.0.0e0.and.dihed.le.90.0e0) then
```

```
                    if (debug) write(*,*) "Beta 2 atom is cis"
                    has_cisbeta2 = .true.

                    if (atlab(ibeta2(1)).eq.'H ') then
                        cisbeta2_isH  = .true.
                    end if

                end if

                if (has_cisbeta2) then

                    if (cisbeta2_isH) then
                        class = 13
                        sub   = 1
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    else
                        class = 13
                        sub   = 3
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    end if

                else
                    class = 13
                    sub   = 2
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (2)  ! alpha 2 has two betas

                if (debug) write(*,*) "Krikey! Alpha2 has 2 betas."

                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                              ibeta2(i), dihed)
                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if (atlab(icisbeta2).eq.'H ') then
                            cisbeta2_isH = .true.
                        end if

                    end if

                end do

                if (cisbeta2_isH) then
                    class = 7
                    sub   = 1
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                else
                    class = 7
                    sub   = 2
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (3) ! alpha 2 has three betas

                if (debug) write(*,*) "Alpha2 has 3 betas."
```

```
                if (atlab(ialpha1).eq.'O ') then
                    class = 11
                    sub   = 2
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                else
                    class = 11
                    sub   = 3
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

             case default
                    class = 15
                    sub   = 1
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return

             end select

         end if  ! end of case 0

      case (1)  ! alpha 1 has 1 beta atom

      if (debug) write(*,*) "Jeepers! Alpha 1 has 1 beta"
c
c     Having only one beta, the first test is whether or not it is cis.
c
         call torsionval(x,y,z,kk,jj,ialpha1,ibeta1(1),dihed)

         if (dihed.gt.180.0e0) then
             dihed = 360.0e0 - dihed
         end if

         if (dihed.ge.0.0e0.and.dihed.le.90.0e0) then
             has_cisbeta1 = .true.

             if (atlab(ibeta1(1)).eq.'H ') then
                 cisbeta1_isH = .true.
             end if

         end if

         if (cisbeta1_isH) then     ! beta on alpha 1 is a cis H

             if (debug) write(*,*) "OMG! Alpha 1 has a cis hydrogen"

             select case (nbeta2)   ! how many betas on alpha 2?
             case (1)  ! 1 beta on alpha 2 (no need to test for case 0)

                 if (debug) write(*,*) "Dude like, Alpha 2 has 1 beta."

                 call torsionval(x,y,z,kk,jj,ialpha2,ibeta2(1),dihed)

                 if (dihed.gt.180.0e0) then
                     dihed = 360.0e0 - dihed
                 end if

                 if (dihed.ge.0.0e0.and.dihed.le.90.0e0) then
                     has_cisbeta2 = .true.

                     if (atlab(ibeta2(1)).eq.'H ') then
                         cisbeta2_isH = .true.
                         if (debug) write(*,*) 'Whoah. Cis beta2 H'
                     end if

                 end if

                 if (has_cisbeta2) then
```

```
                    if (cisbeta2_isH) then
                        class = 3
                        sub   = 1
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    else
                        class = 3
                        sub   = 2
                        ii    = ialpha2
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    end if

                else       ! beta on alpha2 is trans.
                    class = 13
                    sub   = 1
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (2)  ! 2 betas on alpha 2

                if (debug) write(*,*) "Wow, man. Alpha 2 has 2 betas."

                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                              ibeta2(i), dihed)
                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if (atlab(icisbeta2).eq.'H ') then
                            cisbeta2_isH = .true.
                        end if

                    end if

                end do

                if (cisbeta2_isH) then
                    class = 3
                    sub   = 1
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                else
                    class = 3
                    sub   = 2
                    ii    = ialpha2
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (3)   ! 3 betas on alpha 2

                if (debug) write(*,*) "Alpha2 has 3 betas."

                class = 2
                sub   = 3
                ii    = ialpha1
                if (debug) write(*,*) "===>",class, sub, ii
                return

            case default    ! alpha 2 has 4 or more betas

                class = 15
                sub   = 1
```

```
                ii    = ialpha1
                if (debug) write(*,*) "===>",class, sub, ii
                return

            end select     ! end of selecting case for beta on alpha
                           ! one is a cis hydrogen

        elseif (has_cisbeta1) then ! alpha 1 has a cis beta but
                                   ! it's not a hydrogen
            if (debug) write(*,*) "Um, cisbeta 1 not Hydro, bro..."

            select case (nbeta2)
            case (1)  ! 1 beta on alpha 2 (no need to test for case 0)

                if (debug) write(*,*) "Batman: Alpha 2 has 1 beta."

                call torsionval(x,y,z,kk,jj,ialpha2,ibeta2(1),dihed)

                if (dihed.gt.180.0e0) then
                    dihed = 360.0e0 - dihed
                end if

                if (dihed.ge.0.0e0.and.dihed.le.90.0e0) then
                    has_cisbeta2 = .true.

                    if (atlab(ibeta2(1)).eq.'H ') then
                        cisbeta2_isH = .true.
                    end if

                end if

                if (has_cisbeta2) then

                    if (cisbeta2_isH) then
                        class = 3
                        sub   = 2
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    else
                        class = 3
                        sub   = 3
                        ii    = ialpha2
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    end if

                else       ! beta on alpha2 is trans.
                    class = 13
                    sub   = 3
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (2)  ! 2 betas on alpha 2

                if (debug) write(*,*) "Alpha2 double up dem betas."

                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                                  ibeta2(i), dihed)
                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if (atlab(ibeta2(i)).eq.'H ') then
                            cisbeta2_isH = .true.
                        end if
```

```
                    end if

                end do

                if (cisbeta2_isH) then
                    class = 3
                    sub   = 2
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                else
                    class = 3
                    sub   = 3
                    ii    = ialpha2
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (3)   ! 3 betas on alpha 2

               if (debug) write(*,*) "alpha2 has 3 betas."

               class = 2
               sub   = 4
               ii    = ialpha1
               if (debug) write(*,*) "===>",class, sub, ii
               return

            case default    ! alpha 2 has 4 or more betas

               class = 15
               sub   = 1
               ii    = ialpha1
               if (debug) write(*,*) "===>",class, sub, ii
               return

            end select     ! end of selecting case for beta on alpha
                           ! one is a cis hydrogen

         else       ! beta 1 is trans to the leaving atom
c
c     further classification based on the number of beta atoms
c     connected to alpha 2.
c
            select case (nbeta2)
            case (1)   ! 1 beta on alpha 2

                if (debug) write(*,*) "Alpha 2 has 1 betas."

                call torsionval(x,y,z,kk,jj,ialpha2,ibeta2(1),dihed)

                if (dihed.gt.180.0e0) then
                    dihed = 360.0e0 - dihed
                end if

                if (dihed.ge.0.0e0.and.dihed.le.90.0e0) then
                    has_cisbeta2 = .true.

                    if (atlab(ibeta2(1)).eq.'H ') then
                        cisbeta2_isH = .true.
                    end if

                end if

                if (has_cisbeta2) then  ! beta 2 is cis

                    if (cisbeta2_isH) then  ! cis beta 2 is hydrogen
                        class = 13
                        sub   = 1
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    else
```

```
                        class = 13
                        sub   = 3
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    end if

                else        ! beta on alpha 2 is trans.
                    class = 13
                    sub   = 2
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

            case (2)    ! alpha 2 has 2 betas, one of which is cis
                        ! find the cis beta

                if (debug) write(*,*) "Ooooo. Alpha 2 has 2 betas."

                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                                  ibeta2(i), dihed)
                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if (atlab(icisbeta2).eq.'H ') then
                            cisbeta2_isH = .true.
                        end if

                    end if

                end do


                if (cisbeta2_isH) then

                    if (atlab(ibeta1(1)).eq.'O '.and.
     &                  atlab(ialpha1).eq.'N ') then
                        class = 7
                        sub   = 3
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    else
                        class = 13
                        sub   = 1
                        ii    = ialpha1
                        if (debug) write(*,*) "===>",class, sub, ii
                        return
                    end if

                else
                    class = 13
                    sub   = 3
                    ii    = ialpha1
                    if (debug) write(*,*) "===>",class, sub, ii
                    return
                end if

                ! end of case in which alpha 2 has 2 betas

            case (3)        ! alpha 2 has 3 betas

                if (debug) write(*,*) "Derp. Alpha 2 has 3 betas."
                class = 11
                sub   = 1
                ii    = ialpha1
                if (debug) write(*,*) "===>",class, sub, ii
```

```
                return

            case default
                class = 15
                sub   = 1
                ii    = ialpha1
                if (debug) write(*,*) '===>',class, sub, ii
                return

            end select
c
c     End of assignment based upon substitution of number of betas on
c     alpha 2.
c
         end if
c
c     End of assignment based upon torsion value of betas on alpha 1.
c
      case (2)  ! alpha 1 has 2 beta atoms

         if (debug) write(*,*) "BOOM! Alpha 1 has 2 betas."

         do i = 1, 2
            call torsionval(x,y,z,kk,jj,ialpha1,
     &                      ibeta1(i), dihed)
            if (dihed.gt.180e00) then
                dihed = 360.0e0 - dihed
            end if

            if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                icisbeta1 = ibeta1(i)

                if (atlab(icisbeta1).eq.'H ') then
                    cisbeta1_isH = .true.
                end if

            else
                itransbeta1 = ibeta1(1)
            end if

         end do

         if (cisbeta1_isH) then

            select case (nbeta2) ! how many betas on alpha 2 ?
            case (2)   ! 2 betas on alpha 2

                if (debug) write(*,*) "Fooey. Alpha 2 has 2 betas."

                ! find out which one is cis
                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                                  ibeta2(i), dihed)

                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if (atlab(icisbeta2).eq.'H ') then
                            cisbeta2_isH = .true.
                        end if

                    end if

                end do

                if (cisbeta2_isH) then
                    class = 3
                    sub   = 1
                    ii    = ialpha1
```

```
                    if (debug) write(*,*) '===>',class, sub, ii
                    return
                else
                    class = 3
                    sub   = 2
                    ii    = ialpha2
                    if (debug) write(*,*) '===>',class, sub, ii
                    return
                end if

            case (3)   ! 3 betas on alpha 2

                if (debug) write(*,*) "ialpha2 has 3 betas."

                class = 2
                sub   = 3
                ii    = ialpha1
                if (debug) write(*,*) '===>',class, sub, ii
                return

            case default
                class = 15
                sub   = 1
                ii    = ialpha1
                if (debug) write(*,*) '===>',class, sub, ii
                return

            end select
            ! end of assignments based upon alpha 1 being an H

        else  ! cis beta on alpha 1 isn't H
            select case (nbeta2)
            case (2)  ! 2 betas on alpha 2. find out which one is cis

                if (debug) write(*,*) "ialpha2 has 2 betas."

                do i = 1, 2
                    call torsionval(x,y,z,kk,jj,ialpha2,
     &                                  ibeta2(i), dihed)
                    if (dihed.gt.180e00) then
                        dihed = 360.0e0 - dihed
                    end if

                    if (dihed.ge.0.0e0.and.dihed.le.90.e0) then
                        icisbeta2 = ibeta2(i)

                        if (atlab(icisbeta2).eq.'H ') then
                            cisbeta2_isH = .true.
                        end if

                    end if

                end do

                if (cisbeta2_isH) then
                    class = 3
                    sub   = 2
                    ii    = ialpha1
                    if (debug) write(*,*) '===>',class, sub, ii
                    return
                else
                    class = 3
                    sub   = 3
                    ii    = ialpha1
                    if (debug) write(*,*) '===>',class, sub, ii
                    return
                end if

            case (3)                                ! 3 betas on alpha 2

                if (debug) write(*,*) "alpha2 has 3 betas."

                if (n12r(icisbeta1).eq.1) then
```

```
                    class = 4
                    sub   = 3
                    ii    = ialpha1
                    write(*,*) '==>', class, sub, ii
                    return
                else if (n12r(itransbeta1).eq.1) then
                    class = 4
                    sub   = 4
                    ii    = ialpha1
                    write(*,*) '==>', class, sub, ii
                    return
                else
                    class = 2
                    sub   = 4
                    ii    = ialpha1
                    if (debug) write(*,*) '===>',class, sub, ii
                    return
                end if

            case default
                class = 15
                sub   = 1
                ii    = ialpha1
                if (debug) write(*,*) '===>',class, sub, ii
                return

            end select
            ! end of selecting based on betas on alpha 2

         end if ! end of testing nature of cis beta on alpha 2
         ! end of case in which alpha 2 has two betas

      case (3) ! 3 betas on alpha 1

          if (debug) write(*,*) "Alpha1 has 3 betas."

          class = 14
          sub   = 3
          ii    = ialpha1
          if (debug) write(*,*) '===>',class, sub, ii
          return
c
c     There are no classes/subclasses with alpha atoms having more than
c     three substituents. If the subroutine has gotten to this point,
c     then this is the case. Assign to the "general" class/subclass.
c     Arbitrarily make alpha1 the driver.
c
      case default
          class = 15
          sub   = 1
          ii    = ialpha1
          if (debug) write(*,*) '===>',class, sub, ii
          return

      end select

      end subroutine assign_trigplanar

c-----------------------------------------------------------------------
c                           FINDANGLE
c-----------------------------------------------------------------------
c
c     Subroutine to find the angle between three points. Nine reals are
c     given to the subroutine, there being three reals for every point
c     corresponding to its x, y, and z coordinates. The angle is found
c     and returned as 'ang'.
c
      subroutine findangle(x1, y1, z1, x2, y2, z2, x3, y3, z3, ang)

          real x1,y1,z1,x2,y2,z2,x3,y3,z3,ang  ! coordinates and angle
          real vect1(3), vect2(3)  ! vectors creates by coordinates
          real magvect1, magvect2  ! magnitude of vectors
          real pi
```

```
          pi = acos(-1.0e0)
c
c     Vector created by points 1 and 2.
c
          vect1 = [x2-x1, y2-y1, z2-z1]
c
c     Vector created by points 1 and 3.
c
          vect2 = [x3-x1, y3-y1, z3-z1]

c
c     Magnitudes of the two vectors.
c
          magvect1 = sqrt(vect1(1)*vect1(1) + vect1(2)*vect1(2) +
     &                    vect1(3)*vect1(3))
          magvect2 = sqrt(vect2(1)*vect2(1) + vect2(2)*vect2(2) +
     &                    vect2(3)*vect2(3))

c
c     The angle is the arc-cosine of the dot product of the two
c     (normalized) vectors.
c
          ang = acos((dot_product(vect1, vect2))/(magvect1*magvect2))
c
c     Convert to dregrees
c
          ang = (180.0/pi)*ang   ! convert to degrees

          return

      end subroutine findangle

c
c-----------------------------------------------------------------------
c                           DISTANCE2PLANE
c-----------------------------------------------------------------------
c
c     This subroutine finds the distance between a point and a plane. It
c     is used to measure the planarity of trigonal moeities. Returns
c     either .true. or .false., stored in the variable 'flat', which is
c     determined by the distance. If the distance is below 'threshold',
c     it is considered to be flat
c     The plane is created by the first three coordinates. The point of
c     interest is the fourth.
c     Algebra adapted from
c          http://mathworld.wolfram.com/Point-PlaneDistance.html
c
      subroutine distance2plane(x, y, z, flat, debug)

          real x(4), y(4), z(4)
          real distance          ! distance of jj from plane
          real threshhold        ! threshhold for testing planarity
          real vect1(3), vect2(3), normvect(3)  ! vectors
          real magnormvect       ! magnitude of normal vector
          real unitvect(3)       ! unit vector of normal vector
          logical flat           ! true if below threshhold
          logical debug

          threshhold = 0.20e0 ! below this threshhold is considered flat

          if (debug) then
              write(*,*)
              write(*,*) "X array --> ", x
              write(*,*) "Y array --> ", y
              write(*,*) "Z array --> ", z
              write(*,*)
          end if
c
c     Create the two vectors which will be used to find the normal to
c     the plane.
c
          vect1 = [x(3)-x(1), y(3)-y(1), z(3)-z(1)]
```

```
          vect2 = [x(2)-x(1), y(2)-y(1), z(2)-z(1)]

          if (debug) then
              write(*,*)
              write(*,*) "Vector 1 is ", vect1
              write(*,*) "Vector 2 is ", vect2
              write(*,*)
          end if
c
c     Compute their normal (cross product).
c
          normvect = [vect1(2)*vect2(3) - vect1(3)*vect2(2),
     &                vect1(3)*vect2(1) - vect1(1)*vect2(3),
     &                vect1(1)*vect2(2) - vect1(2)*vect2(1)]
          if (debug) write(*,*) "The normal vector is", normvect
c
c     Compute the magnitude of the normal.
c
          magnormvect = sqrt(dot_product(normvect, normvect))

          if (debug) write(*,*) "The magnitude of the normal vector is",
     &                           magnormvect
c
c     Turn it into a unit vector.
c
          unitvect = normvect / magnormvect

          if (debug) write(*,*) "The unit vector is", unitvect
c
c     The distance from the plane to the point is the projection of the
c     vector created between the point of interest off of the plane and
c     and any point on the plane onto the unit normal vector.
c     The first point in the plane [x(1), y(1), z(1)] is used.
          distance = dot_product(unitvect,
     &                           [x(4)-x(1),y(4)-y(1),z(4)-z(1)])
c
c     Don't want negative distances as an artifact of the directionality
c     of the unit vector
c
          distance = abs(distance)

          if (debug) write(*,*) "The distance from the point to ",
     &                          "the plane is ", distance

          if (distance.le.threshhold) then
              flat = .true.
          else
              flat = .false.
          end if

          return

      end subroutine distance2plane

c-----------------------------------------------------------------------
```