NAME

Psi4GenerateConstrainedConformers.py - Generate constrained molecular conformations

SYNOPSIS

Psi4GenerateConstrainedConformers.py [--basisSet <text>] [--confParams <Name,Value,...>] [--energyOut <yes or no>] [--energyDataFieldLabel <text>] [--energyUnits <text>] [--energyRMSDCalcMode <RMSD or BestRMSD>] [--energyRMSDCutoff <number>] [--energyRMSDCutoffMode <All or Lowest>] [--energyWindow <number>] [--infileParams <Name,Value,...>] [--maxIters <number>] [--methodName <text>] [--mcsParams <Name,Value,...>] [--mp <yes or no>] [--mpLevel <Molecules or Conformers>] [--mpParams <Name, Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--precision <number>] [--psi4OptionsParams <Name,Value,...>] [--psi4RunParams <Name,Value,...>] [--quiet <yes or no>] [--reference <text>] [--scaffold <auto or SMARTS>] [--scaffoldRMSDOut <yes or no>] [-w <dir>] -i <infile> -r <reffile> -o <outfile>

Psi4GenerateConstrainedConformers.py -h | --help | -e | --examples

DESCRIPTION

Generate molecular conformations by performing a constrained energy minimization against a reference molecule. The molecular conformations are generated using a combination of distance geometry and forcefield minimization followed by geometry optimization using a quantum chemistry method.

An initial set of 3D conformers are generated for input molecules using distance geometry. A common core scaffold, corresponding to a Maximum Common Substructure (MCS) or an explicit SMARTS pattern, is identified between a pair of input and reference molecules. The core scaffold atoms in input molecules are aligned against the same atoms in the reference molecule. The energy of aligned structures are sequentially minimized using the forcefield and a quantum chemistry method to generate the final 3D structures.

A Psi4 XYZ format geometry string is automatically generated for each molecule in input file. It contains atom symbols and 3D coordinates for each atom in a molecule. In addition, the formal charge and spin multiplicity are present in the the geometry string. These values are either retrieved from molecule properties named 'FormalCharge' and 'SpinMultiplicty' or dynamically calculated for a molecule.

The supported input file formats are: Mol (.mol), SD (.sdf, .sd), SMILES (.smi, .csv, .tsv, .txt)

The supported output file formats are: SD (.sdf, .sd)

OPTIONS

-b, --basisSet <text> [default: auto]

Basis set to use for constrained energy minimization. Default: 6-31+G** for sulfur containing molecules; Otherwise, 6-31G** [Ref 150]. The specified value must be a valid Psi4 basis set. No validation is performed.

The following list shows a representative sample of basis sets available in Psi4:

```
STO-3G, 6-31G, 6-31+G, 6-31++G, 6-31G*, 6-31+G*,  6-31++G*,
6-31G**, 6-31+G**, 6-31++G**, 6-311G, 6-311+G, 6-311++G,
6-311G*, 6-311+G*, 6-311++G*, 6-311G**, 6-311+G**, 6-311++G**,
cc-pVDZ, cc-pCVDZ, aug-cc-pVDZ, cc-pVDZ-DK, cc-pCVDZ-DK, def2-SVP,
def2-SVPD, def2-TZVP, def2-TZVPD, def2-TZVPP, def2-TZVPPD
```

--confParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for generating initial 3D coordinates for molecules in input file. A common core scaffold is identified between a pair of input and reference molecules. The atoms in common core scaffold of input molecules are aligned against the reference molecule followed by constrained energy minimization using forcefield available in RDKit. The 3D structures are subsequently constrained and minimized by a quantum chemistry method available in Psi4.

The supported parameter names along with their default values are shown below:

```
confMethod,ETKDGv2,
forceField,MMFF, forceFieldMMFFVariant,MMFF94,
enforceChirality,yes,embedRMSDCutoff,0.5,maxConfs,50,
useTethers,yes
```

```
confMethod,ETKDGv2   [ Possible values: SDG, KDG, ETDG,
    ETKDG , or ETKDGv2]
forceField, MMFF   [ Possible values: UFF or MMFF ]
forceFieldMMFFVariant,MMFF94   [ Possible values: MMFF94 or MMFF94s ]
enforceChirality,yes   [ Possible values: yes or no ]
useTethers,yes   [ Possible values: yes or no ]
```

confMethod: Conformation generation methodology for generating initial 3D coordinates. Possible values: Standard Distance Geometry (SDG), Experimental Torsion-angle preference with Distance Geometry (ETDG), basic Knowledge-terms with Distance Geometry (KDG) and Experimental Torsion-angle preference along with basic Knowledge-terms and Distance Geometry (ETKDG or ETKDGv2) [Ref 129, 167] .

forceField: Forcefield method to use for constrained energy minimization. Possible values: Universal Force Field (UFF) [Ref 81] or Merck Molecular Mechanics Force Field [Ref 83-87] .

enforceChirality: Enforce chirality for defined chiral centers during forcefield minimization.

maxConfs: Maximum number of conformations to generate for each molecule during the generation of an initial 3D conformation ensemble using conformation generation methodology. The conformations are constrained and minimized using the specified forcefield and a quantum chemistry method. The lowest energy conformation is written to the output file.

embedRMSDCutoff: RMSD cutoff for retaining initial set of conformers embedded using distance geometry and forcefield minimization. All embedded conformers are kept for 'None' value. Otherwise, only those conformers which are different from each other by the specified RMSD cutoff, 0.5 by default, are kept. The first embedded conformer is always retained.

useTethers: Use tethers to optimize the final embedded conformation by applying a series of extra forces to align matching atoms to the positions of the core atoms. Otherwise, use simple distance constraints during the optimization.

--energyOut <yes or no> [default: yes]

Write out energy values.

--energyDataFieldLabel <text> [default: auto]

Energy data field label for writing energy values. Default: Psi4_Energy (<Units>).

--energyUnits <text> [default: kcal/mol]

Energy units. Possible values: Hartrees, kcal/mol, kJ/mol, or eV.

--energyRMSDCalcMode <RMSD or BestRMSD> [default: RMSD]

Methodology for calculating RMSD values during the application of RMSD cutoff for retaining conformations after the final energy minimization. Possible values: RMSD or BestRMSD. This option is ignore during 'None' value of '--energyRMSDCutoff' option.

During BestRMSMode mode, the RDKit 'function AllChem.GetBestRMS' is used to align and calculate RMSD. This function calculates optimal RMSD for aligning two molecules, taking symmetry into account. Otherwise, the RMSD value is calculated using 'AllChem.GetConformerRMS' without changing the atom order. A word to the wise from RDKit documentation: The AllChem.GetBestRMS function will attempt to align all permutations of matching atom orders in both molecules, for some molecules it will lead to 'combinatorial explosion'.

--energyRMSDCutoff <number> [default: 0.5]

RMSD cutoff for retaining conformations after the final energy minimization. By default, only those conformations which are different from other low energy conformation by the specified RMSD cutoff and are with in the specified energy window are kept. The lowest energy conformation is always retained. A value of zero keeps all minimized conformations with in the specified energy window from the lowest energy.

--energyRMSDCutoffMode <All or Lowest> [default: All]

RMSD cutoff mode for retaining conformations after the final energy minimization. Possible values: All or Lowest. The RMSD values are compared against all the selected conformations or the lowest energy conformation during 'All' and 'Lowest' value of '--energyRMSDCutoffMode'. This option is ignored during zero value of '--energyRMSDCutoff'.

By default, only those conformations which all different from all selected low energy conformations by the specified RMSD cutoff and are with in the specified energy window are kept.

--energyWindow <number> [default: auto]

Psi4 Energy window for selecting conformers after the final energy minimization. The default value is dependent on '--energyUnits': 20 kcal/mol, 83.68 kJ/mol, 0.8673 ev, or 0.03188 Hartrees. The specified value must be in '--energyUnits'.

-e, --examples

Print examples.

-h, --help

Print this help message.

-i, --infile <infile>

Input file name.

--infileParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for reading molecules from files. The supported parameter names for different file formats, along with their default values, are shown below:

```
SD, MOL: removeHydrogens,no,sanitize,yes,strictParsing,yes

SMILES: smilesColumn,1,smilesNameColumn,2,smilesDelimiter,space,
    smilesTitleLine,auto,sanitize,yes
```

Possible values for smilesDelimiter: space, comma or tab.

--maxIters <number> [default: 50]

Maximum number of iterations to perform for each molecule or conformer during energy minimization by a quantum chemistry method.

-m, --methodName <text> [default: auto]

Method to use for constrained energy minimization. Default: B3LYP [Ref 150]. The specified value must be a valid Psi4 method name. No validation is performed.

The following list shows a representative sample of methods available in Psi4:

```
B1LYP, B2PLYP, B2PLYP-D3BJ, B2PLYP-D3MBJ, B3LYP, B3LYP-D3BJ,
B3LYP-D3MBJ, CAM-B3LYP, CAM-B3LYP-D3BJ, HF, HF-D3BJ,  HF3c, M05,
M06, M06-2x, M06-HF, M06-L, MN12-L, MN15, MN15-D3BJ,PBE, PBE0,
PBEH3c, PW6B95, PW6B95-D3BJ, WB97, WB97X, WB97X-D, WB97X-D3BJ
```

--mcsParams <Name,Value,...> [default: auto]

Parameter values to use for identifying a maximum common substructure (MCS) in between a pair of reference and input molecules.In general, it is a comma delimited list of parameter name and value pairs. The supported parameter names along with their default values are shown below:

```
atomCompare,CompareElements,bondCompare,CompareOrder,
maximizeBonds,yes,matchValences,yes,matchChiralTag,no,
minNumAtoms,1,minNumBonds,0,ringMatchesRingOnly,yes,
completeRingsOnly,yes,threshold,1.0,timeOut,3600,seedSMARTS,none
```

Possible values for atomCompare: CompareAny, CompareElements, CompareIsotopes. Possible values for bondCompare: CompareAny, CompareOrder, CompareOrderExact.

A brief description of MCS parameters taken from RDKit documentation is as follows:

```
atomCompare - Controls match between two atoms
bondCompare - Controls match between two bonds
maximizeBonds - Maximize number of bonds instead of atoms
matchValences - Include atom valences in the MCS match
matchChiralTag - Include atom chirality in the MCS match
minNumAtoms - Minimum number of atoms in the MCS match
minNumBonds - Minimum number of bonds in the MCS match
ringMatchesRingOnly - Ring bonds only match other ring bonds
completeRingsOnly - Partial rings not allowed during the match
threshold - Fraction of the dataset that must contain the MCS
```

```
seedSMARTS - SMARTS string as the seed of the MCS
timeout - Timeout for the MCS calculation in seconds
```

--mp <yes or no> [default: no]

Use multiprocessing.

By default, input data is retrieved in a lazy manner via mp.Pool.imap() function employing lazy RDKit data iterable. This allows processing of arbitrary large data sets without any additional requirements memory.

All input data may be optionally loaded into memory by mp.Pool.map() before starting worker processes in a process pool by setting the value of 'inputDataMode' to 'InMemory' in '--mpParams' option.

A word to the wise: The default 'chunkSize' value of 1 during 'Lazy' input data mode may adversely impact the performance. The '--mpParams' section provides additional information to tune the value of 'chunkSize'.

--mpLevel <Molecules or Conformers> [default: Molecules]

Perform multiprocessing at molecules or conformers level. Possible values: Molecules or Conformers. The 'Molecules' value starts a process pool at the molecules level. All conformers of a molecule are processed in a single process. The 'Conformers' value, however, starts a process pool at the conformers level. Each conformer of a molecule is processed in an individual process in the process pool. The default Psi4 'OutputFile' is set to 'quiet' using '--psi4RunParams' for 'Conformers' level. Otherwise, it may generate a large number of Psi4 output files.

--mpParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs to configure multiprocessing.

The supported parameter names along with their default and possible values are shown below:

```
chunkSize, auto
inputDataMode, Lazy   [ Possible values: InMemory or Lazy ]
numProcesses, auto   [ Default: mp.cpu_count() ]
```

These parameters are used by the following functions to configure and control the behavior of multiprocessing: mp.Pool(), mp.Pool.map(), and mp.Pool.imap().

The chunkSize determines chunks of input data passed to each worker process in a process pool by mp.Pool.map() and mp.Pool.imap() functions. The default value of chunkSize is dependent on the value of 'inputDataMode'.

The mp.Pool.map() function, invoked during 'InMemory' input data mode, automatically converts RDKit data iterable into a list, loads all data into memory, and calculates the default chunkSize using the following method as shown in its code:

```
chunkSize, extra = divmod(len(dataIterable), len(numProcesses) * 4)
if extra: chunkSize += 1
```

For example, the default chunkSize will be 7 for a pool of 4 worker processes and 100 data items.

The mp.Pool.imap() function, invoked during 'Lazy' input data mode, employs 'lazy' RDKit data iterable to retrieve data as needed, without loading all the data into memory. Consequently, the size of input data is not known a priori. It's not possible to estimate an optimal value for the chunkSize. The default chunkSize is set to 1.

The default value for the chunkSize during 'Lazy' data mode may adversely impact the performance due to the overhead associated with exchanging small chunks of data. It is generally a good idea to explicitly set chunkSize to a larger value during 'Lazy' input data mode, based on the size of your input data and number of processes in the process pool.

The mp.Pool.map() function waits for all worker processes to process all the data and return the results. The mp.Pool.imap() function, however, returns the the results obtained from worker processes as soon as the results become available for specified chunks of data.

The order of data in the results returned by both mp.Pool.map() and mp.Pool.imap() functions always corresponds to the input data.

-o, --outfile <outfile>

Output file name.

--outfileParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for writing molecules to files. The supported parameter names for different file formats, along with their default values, are shown below:

```
SD: kekulize,yes,forceV3000,no
```

--overwrite

Overwrite existing files.

--precision <number> [default: 6]

Floating point precision for writing energy values.

--psi4OptionsParams <Name,Value,...> [default: none]

A comma delimited list of Psi4 option name and value pairs for setting global and module options. The names are 'option_name' for global options and 'module_name__option_name' for options local to a module. The specified option names must be valid Psi4 names. No validation is performed.

The specified option name and value pairs are processed and passed to psi4.set_options() as a dictionary. The supported value types are float, integer, boolean, or string. The float value string is converted into a float. The valid values for a boolean string are yes, no, true, false, on, or off.

--psi4RunParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for configuring Psi4 jobs.

The supported parameter names along with their default and possible values are shown below:

```
MemoryInGB, 1
NumThreads, 1
OutputFile, auto   [ Possible  values: stdout, quiet, or FileName ]
ScratchDir, auto   [ Possivle values: DirName]
RemoveOutputFile, yes   [ Possible values: yes, no, true, or false]
```

These parameters control the runtime behavior of Psi4.

The default file name for 'OutputFile' is <InFileRoot>_Psi4.out. The PID is appended to output file name during multiprocessing as shown: <InFileRoot>_Psi4_<PIDNum>.out. The 'stdout' value for 'OutputType' sends Psi4 output to stdout. The 'quiet' or 'devnull' value suppresses all Psi4 output. The 'OutputFile' is set to 'quiet' for 'auto' value during 'Conformers' of '--mpLevel' option.

The default 'Yes' value of 'RemoveOutputFile' option forces the removal of any existing Psi4 before creating new files to append output from multiple Psi4 runs.

The option 'ScratchDir' is a directory path to the location of scratch files. The default value corresponds to Psi4 default. It may be used to override the deafult path.

-q, --quiet <yes or no> [default: no]

Use quiet mode. The warning and information messages will not be printed.

-r, --reffile <reffile>

Reference input file name containing a 3D reference molecule. A common core scaffold must be present in a pair of an input and reference molecules. Otherwise, no constrained minimization is performed on the input molecule.

--reference <text> [default: auto]

Reference wave function to use for energy calculation. Default: RHF or UHF. The default values are Restricted Hartree-Fock (RHF) for closed-shell molecules with all electrons paired and Unrestricted Hartree-Fock (UHF) for open-shell molecules with unpaired electrons.

The specified value must be a valid Psi4 reference wave function. No validation is performed. For example: ROHF, CUHF, RKS, etc.

The spin multiplicity determines the default value of reference wave function for input molecules. It is calculated from number of free radical electrons using Hund's rule of maximum multiplicity defined as 2S + 1 where S is the total electron spin. The total spin is 1/2 the number of free radical electrons in a molecule. The value of 'SpinMultiplicity' molecule property takes precedence over the calculated value of spin multiplicity.

-s, --scaffold <auto or SMARTS> [default: auto]

Common core scaffold between a pair of input and reference molecules used for constrained minimization of molecules in input file. Possible values: Auto or a valid SMARTS pattern. The common core scaffold is automatically detected corresponding to the Maximum Common Substructure (MCS) between a pair of reference and input molecules. A valid SMARTS pattern may be optionally specified for the common core scaffold.

--scaffoldRMSDOut <yes or no> [default: No]

Write out RMSD value for common core alignment between a pair of input and reference molecules.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To generate conformers by performing constrained energy minimization for molecules in a SMILES file against a reference 3D molecule in a SD file using a common core scaffold between pairs of input and reference molecules identified using MCS, generating up to 50 conformations using ETKDGv2 methodology followed by initial MMFF forcefield minimization and final energy minimization using B3LYP/6-31G** and B3LYP/6-31+G** for non-sulfur and sulfur containing molecules, applying energy RMSD cutoff of 0.5 and energy window value value of 20 kcal/mol, and write out a SD file:

```
% Psi4GenerateConstrainedConformers.py -i Psi4SampleAlkanes.smi
  -r Psi4SampleEthane3D.sdf -o Psi4SampleAlkanesOut.sdf
```

To run the first example in a quiet mode and write out a SD file, type:

```
% Psi4GenerateConstrainedConformers.py -q yes -i Psi4SampleAlkanes.smi
  -r Psi4SampleEthane3D.sdf -o Psi4SampleAlkanesOut.sdf
```

To rerun the first example in multiprocessing mode on all available CPUs without loading all data into memory and write out a SD file, type:

```
% Psi4GenerateConstrainedConformers.py --mp yes
  -i Psi4SampleAlkanes.smi -r Psi4SampleEthane3D.sdf
  -o Psi4SampleAlkanesOut.sdf
```

To run the first example in multiprocessing mode on all available CPUs by loading all data into memory and write out a SD file, type:

```
% Psi4GenerateConstrainedConformers.py --mp yes --mpParams
  "inputDataMode,InMemory"-i Psi4SampleAlkanes.smi
  -r Psi4SampleEthane3D.sdf -o Psi4SampleAlkanesOut.sdf
```

To rerun the first example in multiprocessing mode on specific number of CPUs and chunk size without loading all data into memory and write out a SD file, type:

```
% Psi4GenerateConstrainedConformers.py --mp yes --mpParams
  "inputDataMode,Lazy,numProcesses,4,chunkSize,8"
  -i Psi4SampleAlkanes.smi -r Psi4SampleEthane3D.sdf
  -o Psi4SampleAlkanesOut.sdf
```

To rerun the first example using an explicit SMARTS string for a common core scaffold and write out a SD file, type:

```
% Psi4GenerateConstrainedConformers.py --scaffold "CC"
  -i Psi4SampleAlkanes.smi -r Psi4SampleEthane3D.sdf
  -o Psi4SampleAlkanesOut.sdf
```

To run the first example using a specific set of parameters for generating an initial set of conformers followed by energy minimization using forcefield and a quantum chemistry method and write out a SD file type:

```
% Psi4GenerateConstrainedConformers.py --confParams "confMethod,
  ETKDGv2, forceField,MMFF, forceFieldMMFFVariant,MMFF94s, maxConfs,20,
```

```
embedRMSDCutoff,0.5" --energyUnits "kJ/mol" -m B3LYP
-b "6-31+G**" --maxIters 20 --energyRMSDCutoff 0.5
--energyRMSDCutoffMode All -i Psi4SampleAlkanes.sdf
-r Psi4SampleEthane3D.sdf -o Psi4SampleAlkanesOut.sdf
```

To rerun the first example using molecules in a CSV SMILES file, SMILES strings in column 1, name in column2, and write out a SD file, type:

```
% Psi4GenerateConstrainedConformers.py  --infileParams
   "smilesDelimiter,comma,smilesTitleLine,yes,smilesColumn,1,
   smilesNameColumn,2" -i Psi4SampleAlkanes.csv
   -r Psi4SampleEthane3D.sdf -o Psi4SampleAlkanesOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

Psi4CalculateEnergy.py, Psi4CalculatePartialCharges.py, Psi4GenerateConformers.py, Psi4PerformConstrainedMinimization.py, Psi4PerformMinimization.py

COPYRIGHT

Copyright (C) 2026 Manish Sud. All rights reserved.

The functionality available in this script is implemented using Psi4, an open source quantum chemistry software package, and RDKit, an open source toolkit for cheminformatics developed by Greg Landrum.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.