NAME

 Psi4CalculatePartialCharges.py - Calculate partial atomic charges

SYNOPSIS

 Psi4CalculatePartialCharges.py [--basisSet <text>] [--chargesType <Mulliken or Lowdin>] [
 --chargesRespParams <Name,Value,...>] [--chargesSDFormat <AtomAliases or DataField>] [
 --dataFieldLabel <text>] [--infileParams <Name,Value,...>] [--methodName <text>] [--mp <yes or no>] [
 --mpParams <Name, Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--precision <number>]
 [--psi4OptionsParams <Name,Value,...>] [--psi4RunParams <Name,Value,...>] [--quiet <yes or no>] [
 --reference <text>] [-w <dir>] -i <infile> -o <outfile>

 Psi4CalculatePartialCharges.py -h | --help | -e | --examples

DESCRIPTION

 Calculate partial atomic charges for molecules using a specified method name and basis set. The molecules must
 have 3D coordinates in input file. The molecular geometry is not optimized before the calculation. In addition,
 hydrogens must be present for all molecules in input file. A single point energy calculation is performed before
 calculating the partial atomic charges. The 3D coordinates are not modified during the calculation.

 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)

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

OPTIONS

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

 Basis set to use for calculating single point energy before partial atomic charges. 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
 ```

 -c, --chargesType <Mulliken, Lowdin, or RESP> [default: Mulliken]

 Type of partial atomic charges to calculate. Possible values: Mulliken, Lowdin, or RESP [Ref 158].
 Multiprocessing is not supported during the calculation of RSEP charges. In addition, the RSEP calculation
 relies on the presence of the RESP Psi4 Plugin in your environment.

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

 A comma delimited list of parameter name and value pairs for calculating RESP [Ref 158] charges. A
 space is used as a delimiter for multiple values in a name and value pair. The supported parameter names,
 along with their default values, are shown below:

 ```
 maxIter, 25
 restrainHydrogens, no
 removeGridFiles, yes
 respA, 0.0005
 respB, 0.1
 tolerance, 1e-5
 vdwRadii, auto
 vdwScaleFactors, 1.4 1.6 1.8 2.0
 vdwPointDensity, 1.0
 ```

 maxIter: Maximum number of iterations to perform during charge fitting.

restrainHydrogens: Restrain hydrogens during charge fitting.

removeGridFiles: Keep or remove the following ESP grid and output files: 1_default_grid.dat, 1_default_grid_esp.dat, results.out. The output files are removed by default. You may optionally keep the output files. The output files are automatically renamed to the following file for 'No' value of 'removeGridFiles': Mol<MolNum>_grid.dat, Mol<MolNum>_grid_esp.dat, Mol<MolNum>_resp_results.out.

respA: Scale factor defining the asymptotic limits of the strength of the restraint.

respB: The 'tightness' of the hyperbola around its minimum for the restraint.

tolerance: Tolerance for charges during charge fitting to the ESP.

vdwRadii: vdw radii for elements in angstroms. It's a space delimited list of element symbol and radius value pairs. The default list is shown below:

```
H 1.20 He 1.20 Li 1.37 Be 1.45 B 1.45 C 1.50 N 1.50 O 1.40 F 1.35
Ne 1.30 Na 1.57 Mg 1.36 Al 1.24 Si 1.17P 1.80 S 1.75 Cl 1.7
```

You may specify all or a subset of element symbol and vdw radius pairs to update the default values.

vdwScaleFactors: The vdw radii are scaled by the scale factors to set the grid points at the shells for calculating the ESP using quantum methodology. The default number of shells is 4 and corresponds to the number of vdw scale factors.The 'shell' points are written to a grid file for calculating the ESP.

vdwPointDensity: Approximate number of points to generate per square angstrom surface area.

--chargesSDFormat <AtomAliases or DataField> [default: AtomAliases]

Format for writing out partial atomic charges to SD file. Possible values: AtomAliases or DataField.

The charges are stored as atom property named 'molFileAlias' for 'AtomAliases' format and may be retrieved using the RDKit function 'GetProp' for atoms: Aotm.GetProp('molFileAliases').

The charges are stored under a data field label specified using '-d, --dataFieldLabel' for 'DataField' format and may be retrieved using the RDKit function 'GetProp' for molecules.

-d, --dataFieldLabel <text> [default: auto]

Data field label to use for storing charged in SD file during 'DataField' value of '-c, --chargesSDFormat'. Default: Psi4_<ChargesType>_Charges (a.u.)

-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
```

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

Method to use for calculating single point energy before partial atomic charges. 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
```

--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'.

--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
```

--overwrite

Overwrite existing files.

--precision <number> [default: 4]

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 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, --reference <text> [default: auto]

Reference wave function to use for calculating single point energy before partial atomic charges. 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.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To calculate Mulliken partial atomic charges using B3LYP/6-31G** and B3LYP/6-31+G** for non-sulfur and sulfur containing molecules in a SD file with 3D structures, use RHF and UHF for closed-shell and open-shell molecules, and write a new SD file, type:

```
% Psi4CalculatePartialCharges.py  -i Psi4Sample3D.sdf
  -o Psi4Sample3DOut.sdf
```

To run the first example for calculating RESP charges using a default set of parameters for the RESP calculation and write out a SD file, type:

```
% Psi4CalculatePartialCharges.py  --chargesType RESP
   -i Psi4Sample3D.sdf -o Psi4Sample3DOut.sdf
```

To run the first example for calculating RESP charges using an explicit set of specific parameters for the RESP calculation and write out a SD file, type:

```
% Psi4CalculatePartialCharges.py  --chargesType RESP
```

```
        --chargesRespParams "respA, 0.0005, respB, 0.1, vdwScaleFactors,
        1.4 1.6 1.8 2.0" -i Psi4Sample3D.sdf -o Psi4Sample3DOut.sdf
```

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

```
% Psi4CalculatePartialCharges.py --mp yes -i Psi4Sample3D.sdf
  -o Psi4Sample3DOut.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:

```
% Psi4CalculatePartialCharges.py  --mp yes --mpParams "inputDataMode,
  InMemory" -i Psi4Sample3D.sdf  -o Psi4Sample3DOut.sdf
```

To run the first example in multiprocessing mode on all available CPUs without loading all data into memory along with multiple threads for each Psi4 run and write out a SD file, type:

```
% Psi4CalculatePartialCharges.py --mp yes --psi4RunParams "NumThreads,2"
   -i Psi4Sample3D.sdf -o Psi4Sample3DOut.sdf
```

To run the first example for writing out charges to a new SD file under a datafield instead of storing them as atom property, type:

```
% Psi4CalculatePartialCharges.py  --chargesSDFormat DataField
  -i Psi4Sample3D.sdf  -o Psi4Sample3DOut.sdf
```

To calculate specific partial atomic charges using a specific method and basis set for molecules in a SD ontaining 3D structures and write them out to a specific datafield in a new SD file, type:

```
% Psi4CalculatePartialCharges.py  -c Lowdin -m SCF -b aug-cc-pVDZ
  --chargesSDFormat DataField --dataFieldLabel "Lowdin_Charges"
  -i Psi4Sample3D.sdf  -o Psi4Sample3DOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

Psi4CalculateEnergy.py, Psi4PerformMinimization.py, Psi4GenerateConformers.py

COPYRIGHT

Copyright (C) 2022 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.