How to Use MOF Pore Size Calculator

By Samuel Darer

The purpose of this algorithm is to calculate the largest solid sphere that can pass through a given pore for each frame of a molecular dynamics simulation (MDS) of the pore. Below is the principle command and the arguments it takes, which is contained in the PoreCalculator.py file.

**PoreCalculator**(file)

* file: (string, required) Path to .pdb file containing the positions information for each frame of the MDS of the pore being analyzed

Additional Arguments

* i=1: (integer) First frame number of the MDS being analyzed, used to index output information in produced excel file. For example, if you were analyzing frames 1001-4000 of a simulation, i=1001. Default i=1.
* n=150: (integer) Number of atoms closet to the center of the pore used to create a sub-list from which candidate spheres are generated. Center of the pore currently found by finding the max and min x,y,z values present in the .pdb frame and averaging them. Atoms’ closeness found by calculating Euclid distance from the atom to the center of pore. Increasing n will increase calculation times. Default n=150.
* name=”PoreCalculator”: (string) Name of excel file produced algorithm. .xlsx file extension added by algorithm. Default name=”PoreCalculator”.
* noReturn=False: (Boolean) If False, algorithm produced an excel file that contains the data about the calculated pore size. If True, does not produce a file instead printing the calculated pore size radii into the Python Console. Default noReturn=False.
* start=1: Frame of .pdb file to start on. Takes into account i, and skips all frames until current frame number greater than start-1 (will start calculating on start frame). Ex. If calculating frames 1001-4000, start=2001 would make sure that only the pore sizes for frames 2001-4000 would be calculated. Default start=1.
* kwargs\*\*
  + working\_frame=(integer): Only runs algorithm on given frame of the .pdb, taking into account i. Useful for debugging.
  + testing=(float): working\_frame must have a value. Returns the path the of the largest candidate sphere to pass through the pore took to the top and bottom of the pore and all atoms within (float) Euclid distance of the candidate sphere’s starting position in GeoGebra notation. Useful for debugging.

Input

This function takes a .pdb file as an input. Below is an example of a .pdb file truncated to show the file end. Only carbon atoms are present in the truncated example.

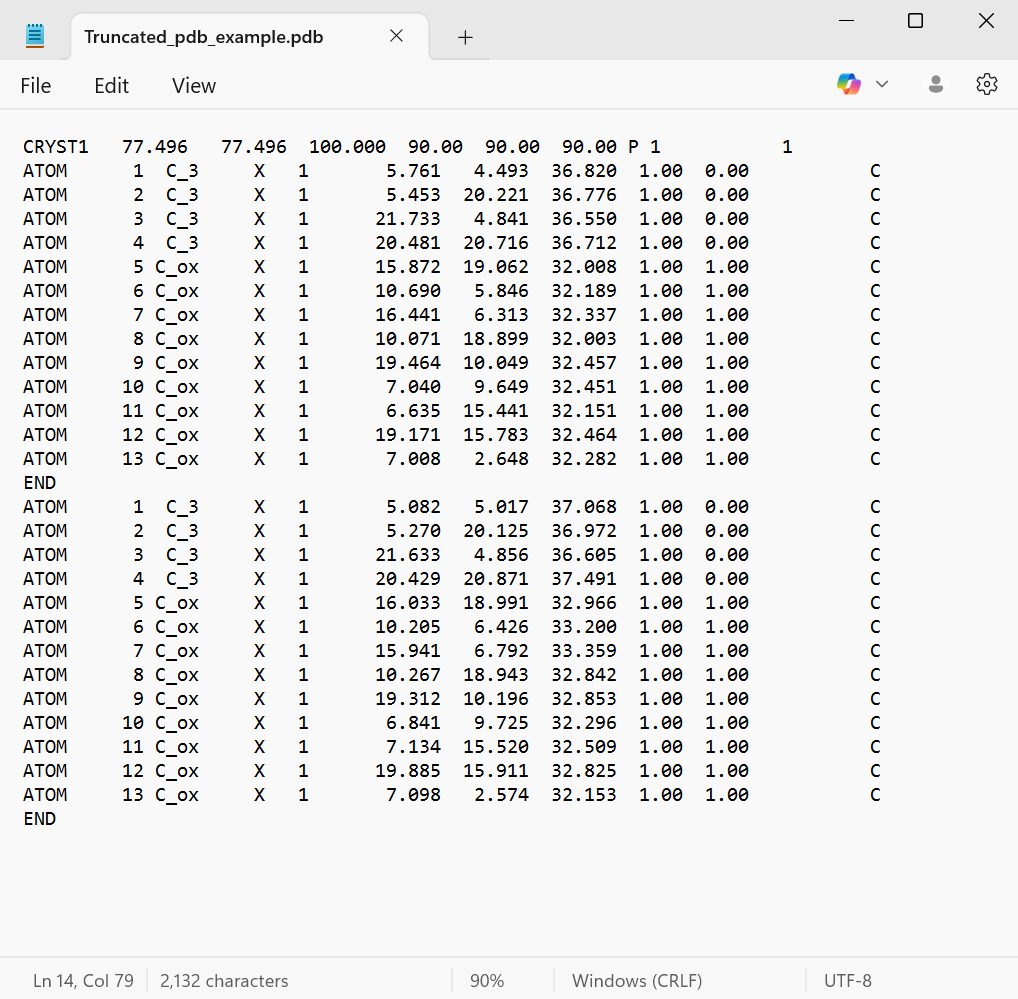
A screenshot of a computer

AI-generated content may be incorrect.

This algorithm is only capable of analyzing 1 pore at a time. Below is an example of a large metal organic framework surface and what a single pore means in this context.

|  |  |
| --- | --- |
|  |  |
| Full TPAA capped MOF-5 Surface | Single pore of TPAA capped MOF-5 |

When multiple frames are analyzed, the .pdb files will have multiple sections, each frame delineated by an END line. A simple example is shown below.



Output

A screenshot of a computer

AI-generated content may be incorrect.

Here is an example of the excel output of a .pdb file. Frame is the frame the candidate sphere was calculated for (starting from i). Radius is the radius of the candidate sphere in angstroms. x, y, z are the respective coordinates for the center of the candidate sphere calculated. The candidate sphere represents the largest solid sphere that was calculated as able to pass through the given pore at the specified frame.

A screenshot of a computer

AI-generated content may be incorrect.

This is an example of the output printed to the Python Console. The order of the information is the same as the excel file within the brackets, being [frame, radius, x, y, z]. Providing the answer in the form of the largest candidate sphere radii below the bracketed line is intended to speed up testing and debugging.

Debugging

Being newly developed and having only been applied to one MOF, being MOF-5 with TPAA caps, this code may run into issues that cause the program to terminate or produce erroneous answers. Beyond print statements, a number of functions are present to help with debugging.

One is .printCoords() present in the AtomicPoint.py file. This prints the coordinates of the AtomicPoint it is applied to. Ex.

A = AtomicPoint(0,1,2)

A.printCoords()

-----------

(0,1,2) with vdwr = 0

-----------

Using A.printCoords(GeoGebraOffset=B) prints the results in a format easy to copy to GeoGebra, the printed location being relative to an origin centered on AtomicPoint B.

In SphericalPath.py there are two more commands that may be helpful.

**atomCollide**(a, pointList, \*\*kwargs)

*Function checks if given AtomicPoint intersects with any other atoms in the pointList. Returns True if it does so, false otherwise.  
  
a : AtomicPoint  
 AtomicPoint being checked  
pointList : List of AtomicPoints  
 List of AtomicPoints that p is being checked against  
--- \*\*kwargs ---  
 pv : List of Planar Values [origin, surface normal, ab, in plane normal] from*

*functions in NumericalMethods. Prints the colliding point p with coordinates*

*converted into the plane given by the planar values  
 pd : Flag  
 If pd value present, also prints the distance between a and the colliding*

*AtomicPoint (accounting for van der Waals radii)*

*Returns  
-------  
True if p intersects with any of the atoms in the pointList, returns false otherwise.*

**geoGebraOut**(a, cs, epl, pv, r, \*\*kwargs):

*Function prints list of atoms within a certain distance of the orbit in GeoGebra Sphere notation  
  
Parameters  
----------  
a : AtomicPoint  
 Orbit orgin.  
cs : AtomicPoint  
 Candidate sphere.  
epl : List of AtomicPoints  
 Environmental point list.  
pv : List  
 Principle planar values.  
r : Float  
 Orbit radius.  
kwargs  
 offset : Float  
 Modifies collision calculation distance*

GeoGebra

GeoGebra notation is ‘x,y,z,vdwr’. This can be copied directly into a GeoGebra sheet to visualize the position of the atoms with the Sphere command. Below is an example as well instructions to access the GeoGebra 3D view and spreadsheet from GeoGebra Classic.

A screenshot of a computer

AI-generated content may be incorrect.

What GeoGebra Classi looks like upon being opened.

|  |  |
| --- | --- |
| A graph with a grid line  AI-generated content may be incorrect. | A screenshot of a graph  AI-generated content may be incorrect. |
| Clicking the bars and shapes in the righthand corner opens the Style Bar. | Clicking the 3 dots in the Style Bar opens a dropdown menu. From here a 3D Graphics Window and Spreadsheet window can be opened. |

A screenshot of a computer

AI-generated content may be incorrect.

Example of opened 3D Graphics and Spreadsheet windows.

To see how GeoGebra can be used to help visualize the atoms, the following location data is going to be copied into the GeoGebra spreadsheet.

A screenshot of a computer

AI-generated content may be incorrect.

Data in the console

A screenshot of a computer

AI-generated content may be incorrect.

Data in GeoGebra

A screenshot of a computer

AI-generated content may be incorrect.

After creating points using X,Y,Z locations.

A screenshot of a computer

AI-generated content may be incorrect.

After creating spheres with points and radii.