pywindow: Automated Structural Analysis of Molecular Pores

Marcin Miklitz, Kim Jelfs

Submitted date: 23/07/2018 • Posted date: 23/07/2018
Licence: CC BY-NC 4.0
Citation information: Miklitz, Marcin; Jelfs, Kim (2018): pywindow: Automated Structural Analysis of Molecular Pores. ChemRxiv. Preprint.

Structural analysis of molecular pores can yield important information on their behaviour in solution and in the bulk. We developed pywindow, a python package that allows for the automated analysis of structural features of porous molecular materials, such as molecular cages. Our analysis includes the cavity diameter, number of windows, window diameters and average molecular diameter. Molecular dynamics trajectories of can also be analysed to explore the influence of flexibility. We present the methodology, validation and application of pywindow for the analysis of molecular pores, metal-organic polyhedra and some instances of framework materials. pywindow is freely available from github.com/JelfsMaterialsGroup/pywindow.

File list (2)

- pywindow_MAIN.pdf (2.26 MiB)
- pywindow_ESI.pdf (2.74 MiB)
pywindow: Automated Structural Analysis of Molecular Pores

Marcin Miklitz and Kim E. Jelfs*

Department of Chemistry, Imperial College London, South Kensington, London SW7 2AZ, United Kingdom

E-mail: k.jelfs@imperial.ac.uk
Phone: +44 (0)20 7594 3438

Abstract

Structural analysis of molecular pores can yield important information on their behaviour in solution and in the bulk. We developed pywindow, a python package that allows for the automated analysis of structural features of porous molecular materials, such as molecular cages. Our analysis includes the cavity diameter, number of windows, window diameters and average molecular diameter. Molecular dynamics trajectories of can also be analysed to explore the influence of flexibility. We present the methodology, validation and application of pywindow for the analysis of molecular pores, metal-organic polyhedra and some instances of framework materials. pywindow is freely available from github.com/JelfsMaterialsGroup/pywindow.
Introduction

The bulk properties of porous solids are greatly influenced by structural variations at the atomistic level. The pore network, as defined by the size, shape and interconnectivity of the pores, naturally impact upon the material’s properties. Thus, structural characterisation is often a key part of computer-aided porous materials design. Common structural descriptors for porous networks are the largest cavity diameter (LCD), the largest sphere that can be inserted in the porous material without overlapping with any of the atoms, and the pore limiting diameter (PLD), the largest sphere that can freely diffuse through the porous network without overlapping with any of the atoms. The LCD and PLD, in tandem with the computationally determined pore size distribution (PSD), surface area (SA) and pore volume, are often used to estimate the porous material’s performance in silico.

Various software exists for the structural characterisation of porous materials. An early example, HOLE, determines the ion channel dimensionality with a Monte Carlo (MC) simulated annealing procedure by probing the sphere sizes that can fit along the channel. MC sampling is commonly used to calculate PLD, LCD, PSD and SA in ZEOMICS, MOFomics and Poreblazer. The Zeo++ package determines the pore network interconnectivity with a Voronoi tessellation. The channel interconnectivity allows one to distinguish between accessible and non-accessible SA. Lastly, the pore cavity shape can be analysed in terms of its asphericity, acylindricity, and relative shape anisotropy with pyMolDyn, also equipped with a graphical user interface.

The provided software examples are mostly intended for the analysis of porous materials in the solid-state. However, if one wishes to characterise molecules of intrinsic porosity, also known as “molecular pores”, there are not many software options. Molecular pores are cage- or belt-like molecules, which host an internal cavity that can be accessed by two or more windows. As these molecules are often soluble in common solvents, they can find in-solution applications such as sensing or be incorporated into mixed-matrix membranes for molecular separations. In the bulk, molecular pores can self-assemble and form porous molecular
materials (PMMs). However, the resulting PMM’s structure is often not known until resolved experimentally and is dependent on the solvent and conditions used, and can be polymorphic or amorphous. Simultaneously, computational crystal structure predictions are still too expensive to be used routinely. Thus, the structural descriptors of individual molecular pores, such as the void size and window diameter, and their relationship to properties, might be more informative. This is especially true if the isolated molecules are studied, or the solid state assembly and the void connectivity have a minimal effect on the property of the resulting PMMs. Recently, Garcia et al. reported the analysis of a single molecule’s surroundings and the projection of void regions within the perimeter of a molecule. This method identifies windows, but does not allow calculation of the window diameters.

The void size and window diameters yield the general description of the individual molecular pores, in the same fashion as the LCD and PLD describe a porous network, and in some particular molecular assemblies these can be the same. However, if using the software described in the previous section to characterise PMMs, you do not definitively return the underlying features of the individual molecular pores, but rather of the pore network as a function of the molecular pores’ assembly. Simultaneously, the manual measurement of window diameters using visualisation software is laborious and prone to human error. This approach is also unfeasible for the analysis of thousands of frames of molecular dynamics (MD) trajectories. Here we report the python package “pywindow”, released under an MIT license and available at github.com/JelfsMaterialsGroup/pywindow. The software allows for the rapid analysis of the static, as well as dynamic, features of molecular pores, such as pore breathing from MD trajectories.

Software overview

The pywindow package is intended for the structural analysis of molecular pores, whether in isolation or as PMMs. The software was validated against well established Zeo++, where
the functionality of the two overlaps, and a literature circumcircle method to determine a circular window diameter in porous organic cages.\textsuperscript{24} \texttt{pywindow} has no underlying units and inherits that of the input file, however, all molecular coordinates are assumed to be in Ångstrom. The structural analysis does not depend on the given topology and chemistry of the molecule. Thus, other materials such as coordination polyhedra or representative fragments of metal-organic frameworks can also be analysed.

The simplified schematic of the structure of \texttt{pywindow} is presented in Figure 1. The Cartesian coordinates and atom types can be extracted from a commonly used file extension (\textit{e.g.} XYZ, PDB, MOL), ported directly from the RDKit\textsuperscript{25} molecule object, or from the trajectory file loaded with one of the trajectory module classes (XYZ, PDB, DLPOLY). The latter requires pre-processing steps of extracting each individual frame from the trajectory and deciphering forcefield atom keys, features implemented in \texttt{pywindow}. The input source can contain one or more molecular pores that can be individually extracted from the system for the analysis. Molecules split over cell boundaries due to periodic boundary conditions are reconstructed (see Supplementary Information). Lastly, the calculated values can be stored in JSON format.

\textbf{Structural properties of molecular pores}

The maximum diameter of a molecule ($D_{\text{max}}$) is defined as the distance between the edges of the van der Waals spheres of the two atoms at the greatest distance from each other in the molecule. The intrinsic void diameter ($D_{\text{void}}$) is calculated as the distance from the centre of mass (\textit{COM}) of a molecule to the van der Waals edge of the closest atom. This value is then used to calculate the spherical pore volume ($V_{\text{void}}$). However, in some unsymmetric systems, the \textit{COM} might not coincide with the actual pore centre. Thus, we implemented the optimised intrinsic void diameter ($D_{\text{void\_opt}}$) and volume ($V_{\text{void\_opt}}$), which includes \textit{COM} coordinate minimisation to find a better estimate of the pore centre. The average diameter of a molecule ($D_{\text{avg}}$) is determined as a mean distance from the \textit{COM} of a molecule to its van
Figure 1: A schematic of pywindow and some possible workflows. The molecular module contains the MolecularSystem class that is used to load in the input. In the case of trajectories, input is loaded through one of the available classes (DL_POLY, XYZ or PDB) depending on the format. Each frame is then returned as a MolecularSystem object. The MolecularSystem object, if containing multiple molecules, is then separated into individual Molecule objects.

The der Waals surface. The $D_{avg}$ can match experimentally determined solvodynamic diameters, as shown in our recent work.\textsuperscript{26} The methods of calculating these structural properties are detailed in the Supplementary Information.

Finally, the most important feature of pywindow is the window diameter calculations. The schematics of the method is presented in Figure 2. The process is fully automated and does not require prior visualisation of the molecule or assumptions on the number of windows. From a sphere of sampling points distributed evenly around a molecular pore,
vectors connecting these points and the molecule’s COM are analysed for the largest included sphere that can be placed at a given vector point without overlapping with any van der Waals spheres of the host. This allows calculation of the largest sphere diameter that can fit into that window and thus the window diameter (see the Supplementary Information for further details).

![Diagram](image)

**Figure 2:** The steps of determining the window diameter; a) a sphere of evenly distributed sampling points is generated around the host molecule; b) vectors connecting the centre of mass of a molecule and the sampling points are analysed for the overlap with the host’s atoms; c) the vectors that do not overlap with any of the host’s atoms (i.e. vectors passing through windows) are clustered into distinct windows; d) for each window, the vector with the largest included sphere along its path is chosen and the window’s circular diameter calculated.

### Applications of pywindow

#### Computational details

The crystallographic information files (CIFs) were obtained from the Cambridge Structural Database (CSD). The molecular pores are referred to by the CSD references under which the CIFs were found in the CSD. The analysis was then performed using pywindow. For
comparison and validation, the window diameters with the circumcircle method\textsuperscript{21} and the LCD and PLD using the Zeo++0.2.2 package,\textsuperscript{10} were calculated.

MD simulations were performed for the previously obtained CC3 organic cage (code PUDXES) using the DL\_POLY2.20.\textsuperscript{28} The OPLS\_2005 allatom force field\textsuperscript{29} was used. DL\_POLY2.20 software input files were prepared using DL\_FIELD3.3.\textsuperscript{30} A 0.2 ns equilibration followed by a 1 ns production run was performed, with a timestep of 0.5 fs, at 300 K. The Coulomb summation for the electrostatic interactions and the Leapfrog Verlet algorithm\textsuperscript{31} was used. The image conversion was set to 0 in the CONFIG file.

**Computational expense**

The time required to calculate a single window diameter of PUDXES is less than 1s on a single CPU processor (2.4 GHz Quad Core Intel Xeon). However, this time will scale with number of windows, their size and the maximum dimension of a molecule. The process of trajectory analysis is parallelised so that multiple frames can be analysed simultaneously. Thus, the analysis is still cheap and easily achievable over a matter of hours or days at most with the benefit of an automated process for the extraction of the coordinates, reconstruction of the unit cell, and determination of structural properties of individual molecular pores.

**Window diameters and number of windows**

The first step in validation of the pywindow package was the comparison of the calculated window diameters to other methods. The porous organic cage (CC3) PUDXES system was chosen as a test case for two reasons: (i) PUDXES was previously studied using the circumcircle method to calculate window diameters, which will allow for a direct comparison with that method; ii) The window-to-window assembly of molecules in the crystal structure result in the window diameter coinciding with the narrowest point in the porous network and therefore it is possible to compare the PLD output by Zeo++ to the window diameter. PUDXES is shown in Figure 3.
The result of a comparison of the \texttt{pywindow} window analysis ($D_{\text{window}}$) to the window diameters determined with the circumcircle method and the PLD output by Zeo++ software is given in Table 1. The circumcircle method is also equivalent to a manual measurement in any software with a graphical user interface. The results for all three methods are comparable. The Zeo++ results give the largest window diameters of 3.66 Å. The circumcircle method gives a value of 3.63 Å and \texttt{pywindow}, 3.64 Å.

Next, a set of molecular pores that are intrinsically porous and possess a variety of number
Table 1: The comparison of the four window diameters determined for PUDXES using pywindow, the circumcircle method and Zeo++. There is only one PLD generated for the Zeo++, however, from the symmetry considerations we report the value for each window separately.

| Method        | $D_{\text{win.1}}$ [Å] | $D_{\text{win.2}}$ [Å] | $D_{\text{win.3}}$ [Å] | $D_{\text{win.4}}$ [Å] |
|---------------|-------------------------|-------------------------|-------------------------|-------------------------|
| pywindow      | 3.64                    | 3.64                    | 3.64                    | 3.63                    |
| circumcircle  | 3.63                    | 3.63                    | 3.63                    | 3.63                    |
| Zeo++ 0.2.2   | 3.66                    | 3.66                    | 3.66                    | 3.66                    |

of windows were analysed with pywindow. These are shown in Figure 3. The BATVUP, NUXHIZ, PUDXES, and REYMAL molecules contain 2, 3, 4 and 6 windows, respectively. YAQHOQ ($C_{60}$) has no windows and was used as a control. The results are shown in Table 2. The total number of windows identified for each molecule equals the number of windows identified from visual inspection.

Table 2: The structural parameters calculated with pywindow for different molecular pores.

| System     | $D_{\text{max}}$ [Å] | $D_{\text{void opt}}$ [Å] | $V_{\text{void opt}}$ [Å$^3$] | $n(\text{windows})$ | mean($D_{\text{win.x}}$) [Å] |
|------------|-----------------------|----------------------------|-----------------------------|---------------------|-----------------------------|
| YAQHOQ     | 10.5                  | 3.6                        | 25.0                        | 0                   | -                           |
| BATVUP     | 14.8                  | 5.0                        | 63.6                        | 2                   | 3.5                         |
| NUXHIZ     | 18.6                  | 9.0                        | 377.0                       | 3                   | 7.2                         |
| PUDXES     | 22.2                  | 5.4                        | 82.3                        | 4                   | 3.6                         |
| REYMAL     | 34.0                  | 13.8                       | 1363.1                      | 6                   | 9.1                         |

Analysis of molecular dynamics trajectories

The MD trajectories of molecular pores can be analysed to study breathing effects by determining the pore limiting envelopes (PLE), a distribution of window diameters over the trajectory, the fluctuations of $D_{\text{void opt}}$ and the fluctuations of $D_{\text{max}}$. Results for PUDXES are presented in Figure 4. We recently published a study where the PLEs was used to characterise possible application of molecular pores in xenon/krypton separation.22
Other material classes

As `pywindow` depends only on Cartesian coordinates and atom types, host molecules need not be fully organic. For example, we analysed a metal-organic cage (CSD reference: SAYGOR). Systems that are not inherently discrete, such as framework materials can also be analysed if the user provides the representative fragment, for example of a metal-organic framework (CSD reference: MIBQAR), for void and window diameters to be calculated. The analysed materials are shown in Figure 5 and the `pywindow` output in Table 3.
Table 3: The structural features calculated for a metal-organic cage (SAYGOR) and a representative fragment of metal-organic framework (MIBQAR) with pywindow.

| System   | $D_{\text{max}}$ [Å] | $D_{\text{void opt}}$ [Å] | $V_{\text{void opt}}$ [Å$^3$] | $n(\text{windows})$ | mean($D_{\text{win.x}}$) [Å] |
|----------|-----------------------|-----------------------------|-------------------------------|----------------------|-------------------------------|
| MIBQAR   | -                     | 12.3                        | 968.9                         | 6                    | 7.9                           |
| SAYGOR   | 32.5                  | 9.4                         | 435.6                         | 4                    | 7.2                           |

Conclusions

pywindow allows for the structural characterisation of molecular pores. The possibility to use pywindow to determine the number of windows and their diameters in molecules of various shapes and chemistry has been demonstrated, as has the applicability to metal-organic cages and, in certain circumstances, to framework materials. The object-oriented character of the code and modularity makes it easily extendable, and the code is freely available. pywindow can be used to determine whether a material has a predisposition to be porous and whether the windows are big enough to allow for guest diffusion. We have presented the functionality to analyse molecular dynamics trajectories, in an automated and straightforward fashion. The pore limiting envelope allows one to relate the dynamic change of window diameter to the diffusion of guests in flexible materials. The possibility of guest molecules to bind within the pores can be assessed with the calculation of the cavity size. There is no requirement to visually assess the molecules prior to the analysis and no predefined information about the molecule is required, thus the software is easily applied for high-throughput simulation. The planned future extension of pywindow is to include methods for further structural properties, such as the host molecule’s shape, in terms of asphericity and acylindricity, shape of the cavities and windows.
Acknowledgement

We acknowledge a Royal Society University Research Fellowship (K.E.J.) and the EPSRC (EP/M017257/1 and EP/N004884/1) and ERC through grant agreement number 758370 (ERC-StG-PE5-CoMMaD) for funding. We thank Dr. Enrico Berardo and Lukas Turcani for useful discussions.

Supporting Information Available

The supporting information contains additional tables, figures and methodological details.

References

(1) Martin, R. L.; Smit, B.; Haranczyk, M. Addressing challenges of identifying geometrically diverse sets of crystalline porous materials. J. Chem. Inf. Model. 2012, 52, 308–18.

(2) Jelfs, K. E.; Cooper, A. I. Molecular Simulations to Understand and to Design Porous Organic Molecules. Curr. Opin. Solid State Mater. Sci. 2013, 17, 19–30.

(3) Thornton, A. W.; Winkler, D. A.; Liu, M. S.; Haranczyk, M.; Kennedy, D. F. Towards computational design of zeolite catalysts for CO2 reduction. RSC Adv. 2015, 5, 44361–44370.

(4) Haldoupis, E.; Nair, S.; Sholl, D. S. Pore Size Analysis of >250,000 Hypothetical Zeolites. Phys. Chem. Chem. Phys. 2011, 13, 5053–5060.

(5) Simon, C. M.; Kim, J.; Gomez-Gualdron, D. A.; Camp, J. S.; Chung, Y. G.; Martin, R. L.; Mercado, R.; Deem, M. W.; Gunter, D.; Haranczyk, M.; Sholl, D. S.; Snurr, R. Q.; Smit, B. The materials genome in action: identifying the performance limits for methane storage. Energy Environ. Sci. 2015, 8, 1190–1199.
(6) Smart, O. S.; Neduvelil, J. G.; Wang, X.; Wallace, B. A.; Sansomt, M. S. P. HOLE: A program for the analysis of the pore dimensions of ion channel structural models. *J. Mol. Graph.* **1996**, *7855*, 354–360.

(7) First, E. L.; Gounaris, C. E.; Wei, J.; Floudas, C. A. Computational characterization of zeolite porous networks: an automated approach. *Phys. Chem. Chem. Phys.* **2011**, *13*, 17339.

(8) First, E. L.; Floudas, C. A. MOFomics: Computational pore characterization of metal-organic frameworks. *Micropor. Mesopor. Mat.* **2013**, *165*, 32–39.

(9) Sarkisov, L.; Harrison, A. Computational structure characterisation tools in application to ordered and disordered porous materials. *Mol. Simulat.* **2011**, *37*, 1248–1257.

(10) Willems, T. F.; Rycroft, C. H.; Kazi, M.; Meza, J. C.; Haranczyk, M. Algorithms and tools for high-throughput geometry-based analysis of crystalline porous materials. *Micropor. Mesopor. Mat.* **2012**, *149*, 134–141.

(11) Pinheiro, M.; Martin, R. L.; Rycroft, C. H.; Jones, A.; Iglesia, E.; Haranczyk, M. Characterization and comparison of pore landscapes in crystalline porous materials. *J. Mol. Graph. Model.* **2013**, *44*, 208–19.

(12) Heimbach, I.; Rhiem, F.; Beule, F.; Knodt, D.; Heinen, J.; Jones, R. O. pyMolDyn: Identification, structure, and properties of cavities/vacancies in condensed matter and molecules. *J. Comput. Chem.* **2017**, *38*, 389–394.

(13) Hasell, T.; Cooper, A. I. Porous Organic Cages: Soluble, Modular and Molecular Pores. *Nat. Rev. Mater.* **2016**, *1*, 16053.

(14) Zwijnenburg, M. A.; Berardo, E.; Peveler, W. J.; Jelfs, K. E. Amine Molecular Cages as Supramolecular Fluorescent Explosive Sensors: A Computational Perspective. *J. Phys. Chem. B* **2016**, *120*, 5063–5072.
(15) Evans, J. D.; Huang, D. M.; Hill, M. R.; Sumby, C. J.; Thornton, A. W.; Doonan, C. J. Feasibility of Mixed Matrix Membrane Gas Separations Employing Porous Organic Cages. *J. Phys. Chem. C* 2014, 118, 1523–1529.

(16) Mao, H.; Zhang, S. Mixed-matrix membranes incorporated with porous shape-persistent organic cages for gas separation. *J. Colloid Interface Sci.* 2017, 490, 29–36.

(17) Mastalerz, M. Permanent Porous Materials from Discrete Organic Molecules—towards Ultra-High Surface Areas. *Chem. Eur. J.* 2012, 18, 10082–10091.

(18) Hasell, T.; Culshaw, J. L.; Chong, S. Y.; Schmidtmann, M.; Little, M. A.; Jelfs, K. E.; Pyzer-knapp, E. O.; Shepherd, H.; Adams, D. J.; Day, G. M.; Cooper, A. I. Controlling the Crystallization of Porous Organic Cages: Molecular Analogs of Isoreticular Frameworks Using Shape-Specific Directing Solvents. *J. Am. Chem. Soc.* 2014, 136, 1438–1448.

(19) Little, M. A.; Chong, S. Y.; Schmidtmann, M.; Hasell, T.; Cooper, A. I. Guest control of structure in porous organic cages. *Chem. Commun.* 2014, 50, 9465–9468.

(20) Jiang, S.; Jelfs, K. E.; Holden, D.; Hasell, T.; Chong, S. Y.; Haranczyk, M.; Trewin, A.; Cooper, A. I. Molecular Dynamics Simulations of Gas Selectivity in Amorphous Porous Molecular Solids. *J. Am. Chem. Soc.* 2013, 135, 17818–17830.

(21) Pyzer-Knapp, E. O.; Thompson, H. P. G.; Schiffmann, F.; Jelfs, K. E.; Chong, S. Y.; Little, M. A.; Cooper, A. I.; Day, G. M. Predicted crystal energy landscapes of porous organic cages. *Chem. Sci.* 2014, 5, 2235–2245.

(22) Miklitz, M.; Jiang, S.; Clowes, R.; Briggs, M. E.; Cooper, A. I.; Jelfs, K. E. Computational Screening of Porous Organic Molecules for Xenon/Krypton Separation. *J. Phys. Chem. C* 2017, 121, 15211–15222.
(23) Garcia, I. G.; Bernabei, M.; Soto, R. P.; Haranczyk, M. Out-of-Oblivion Cage Molecules and Their Porous Crystalline Phases. *Cryst. Growth Des.* **2017**, *17*, 5614–5619.

(24) Holden, D.; Jelfs, K. E.; Cooper, A. I.; Trewin, A.; Willock, D. J. Bespoke Force Field for Simulating the Molecular Dynamics of Porous Organic Cages. *J. Phys. Chem. C* **2012**, *116*, 16639–16651.

(25) RDKit: Open-source cheminformatics. [http://www.rdkit.org](http://www.rdkit.org).

(26) Greenaway, R. L.; Santolini, V.; Bennison, M. J.; Alston, B. M.; Pugh, C. J.; Little, M. A.; Miklitz, M.; Eden-Rump, E. G. B.; Clowes, R.; Shakil, A.; Cuthbertson, H. J.; Armstrong, H.; Briggs, M. E.; Jelfs, K. E.; Cooper, A. I. High-throughput discovery of organic cages and catenanes using computational screening fused with robotic synthesis. *Nat. Commun.* **2018**, *9*, 2849.

(27) Groom, C. R.; Bruno, I. J.; Lightfoot, M. P.; Ward, S. C. The Cambridge Structural Database. *Acta Crystallogr. Sect. B* **2016**, *72*, 171–179.

(28) Smith, W.; Yong, C. W.; Rodger, P. M. DL_POLY: Application to Molecular Simulation. *Mol. Simulat.* **2002**, *28*, 385–471.

(29) Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. Development and Testing of the OLPS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *J. Am. Chem. Soc.* **1996**, *118*, 11225–11236.

(30) Yong, C. W. Descriptions and Implementations of DL_F Notation: A Natural Chemical Expression System of Atom Types for Molecular Simulations. *J. Chem. Inf. Model.* **2016**, *56*, 1405–1409.

(31) Gunsteren, W. F. V.; Berendsen, H. J. C. A Leap-frog Algorithm for Stochastic Dynamics. *Mol. Simulat.* **1988**, *1*, 173–185.
Graphical TOC Entry

Porous Molecular Materials:
✓ maximum dimension
✓ average diameter
✓ pore diameter
✓ number of windows
✓ window diameter
Supporting information for:

pywindow: Automated Structural Analysis of Molecular Pores

Marcin Miklitz and Kim E. Jelfs*

Department of Chemistry, Imperial College London, South Kensington, London SW7 2AZ,
United Kingdom

E-mail: k.jelfs@imperial.ac.uk
Phone: +44 (0)20 7594 3438
Reconstruction of the periodic unit cell

As a result of periodic boundary conductions, the atom positions for a molecule enclosed in a unit cell will often cross through periodic boundaries. The analysis requires the reconstruction of the molecules based on the atomic positions and the connectivity between atoms. Therefore, a simple algorithm was implemented in pywindow that allows one to rebuild the systems (the `rebuild_system()` method of the `MolecularSystem` class). First, a $3 \times 3 \times 3$ supercell is constructed resulting in 27 symmetry representations of the system, with the initial unit cell in the centre of such supercell. For all atomic positions (excluding some elements\(^1\)) the one closest to the origin is picked as the seed for the individual molecule reconstruction. With a formula to determine if a bond is present between two atoms A and B, we iterate through all atoms in the system. The atom pair A and B is connected if the distance ($d_{A,B}$) meets the follow criterion:

$$d_{A,B} < 2 \times r_{\text{cov,max}} + t$$

(1)

where $r_{\text{cov,max}}$ is the largest covalent radius of all the atom types in the molecular system and $t$ is a user defined cut-off (default = 0.4 Å). In each iteration, the atoms that are determined as bonded to any of the atoms in the previous set are sampled for their atomic neighbours. This is performed for the initial unit cell only, the role of the atom representations from the supercell is to allow cross boundary connections. The atoms from the supercell area can populate the search set. However, none of the atoms from the supercell area can act as a starting seed. Only one occurrence of an atom, out of the 27 symmetry equivalences, can be assigned to a molecule.

For each molecule found, the centre of mass (COM) is determined and if it is in the initial unit cell, this molecule is then returned as part of the rebuilt molecular system. The performance of the algorithm is presented in Figure S1 with an example of a cubic, monoclinic

\(^1\)H, Cl, Br, F, He, Ar, Ne, Kr, Xe, Rn
and triclinic symmetry periodic unit cell. The individual molecules for the analysis are then extracted using the `make_modular()` method, of the `MolecularSystem()` class, that returns each molecule as the `Molecule` object. This is done on the connectivity criteria based on the individual atom pair distances and the covalent radii.

Figure S1: Three examples of the periodic unit cells before (top) and after (bottom) the rebuild process. The example on the left is of the cubic symmetry unit cell containing eight discrete molecules, the middle example of a monoclinic unit cell containing three discrete molecules and the right example of a triclinic unit cell containing two discrete molecules. Any solvent molecules have been omitted for clarity.
Structural properties of molecular pores

Centre of mass

The centre of mass (COM) of a molecule is calculated according to the equation:

\[ \text{COM} = \frac{1}{MW} \sum_{i=1}^{n} m_i v_i \]  

(2)

where \( MW \) is the molecular weight of the molecule, \( v \) is a vector containing the Cartesian coordinates of an atom and \( m \) is the atomic mass for the element. The formula yields \( \text{COM} \), which is a vector that corresponds to the \( x \), \( y \) and \( z \) coordinates of the centre of mass of the molecule. The centre of mass is calculated with the \text{calculate}_\text{centre}_\text{of}_\text{mass}() \ method of \text{Molecule} \ class.

Maximum dimension

The molecule’s maximum diameter (\( D_{\text{max}} \)) is determined as the distance between two furthest atoms in the molecule. It is obtained using an Euclidean distance matrix between all atoms in the molecule. From the obtained matrix of distances, the largest value found is between the coordinates of the two furthest atoms. The distance value is then corrected for the corresponding \( r_{\text{vdw}} \) for the pair of elements, by adding it to the calculated distance yielding the \( D_{\text{max}} \). The maximum dimension of a molecule is calculated with the \text{calculate}_\text{maximum}_\text{diameter}() \ method of \text{Molecule} \ class.

Intrinsic void diameter

The intrinsic void diameter (\( D_{\text{void}} \)) is determined as the distance between the \( \text{COM} \) of a molecule and the closest atom. This is calculated using the Euclidean distance matrix between the \( \text{COM} \) vector and the positions of all elements in the molecule. The smallest value in the set of distances is then corrected by subtracting the appropriate \( r_{\text{vdw}} \) of the determined closest element and multiplying it by 2 to yield the \( D_{\text{void}} \). The intrinsic void
diameter is calculated with the calculate_pore_diameter() method of Molecule class.

**Intrinsic void volume**

The intrinsic void volume ($V_{\text{void}}$) of a molecule is calculated with a formula for a sphere volume, where the void radius $R_{\text{void}}$ equals to $\frac{1}{2}D_{\text{void}}$. The intrinsic void volume is calculated with the calculate_pore_volume() method of the Molecule class.

**Optimised intrinsic void diameter and volume**

The function for the optimised intrinsic void diameter $D_{\text{void\_opt}}$ facilitates the possibility where the assumed overlap of the COM with the pore centre, as in $D_{\text{void}}$ and $V_{\text{void}}$, does not occur in the non-symmetrical molecule. This was addressed by introducing an optimisation step in the void centre determination before calculating the void diameter. The COM of the molecule is used as a starting guess for the coordinates of the pore centre in the minimisation function that calculates the negative value of the the $D_{\text{void}}$ as a function of coordinates that are being minimised. This allows one to find the optimal coordinates of the intrinsic void and the $D_{\text{void\_opt}}$ is yielded in the same fashion as the $D_{\text{void}}$. The $D_{\text{void\_opt}}$ is calculated with the calculate_pore_diameter_opt() method of the Molecule class. The $V_{\text{void\_opt}}$ is equivalent to $V_{\text{void}}$ with the difference of using $R_{\text{void\_opt}}$ as half of the obtained $D_{\text{void\_opt}}$. The optimised intrinsic void volume is calculated with the calculate_pore_volume_opt() method of the Molecule class.

**Methodology of calculating window diameters**

The process of calculating the window diameter ($D_{\text{window}}$) for each of the windows in the molecule, determined as the channel necking that connects the intrinsic void with molecule’s surrounding, presented in Figure 2 in the main manuscript, is performed as following:

1. From the Cartesian coordinates of a single molecule, the COM is calculated and the molecule is shifted to the origin ($O$) of the Cartesian system by subtracting the COM
vector from the molecule’s coordinates.

2. On a sphere, with radius equal to $D_{\text{max}}$ and the centre of the sphere located at $O$, a set of sampling points is generated with the Vogel’s method\textsuperscript{S1} for a spiral distribution of points on a disc, adopted for a sphere using cylindrical coordinates. The approach allows one to obtain a uniform distribution of points on a sphere surface (see Figure 2a in the main manuscript).

3. The number of sampling points is system dependent and calculated according to the formula:

$$n = a \times \log_{10}(A) \times 250$$ \hspace{1cm} (3)

where $A$ is the surface area of the sphere in Å$^2$. The logarithmic scaling is used that prevents oversampling of larger molecules, which is the bottleneck for the algorithm performance, and the 250 factor proved in the design and validation processes to give a sufficient sampling level. Further, the scaling factor $a$ allows further control on the sampling level.

4. For each sampling point a sampling vector is defined that starts at the $O$ and the end point are the coordinates of a given sampling point. Using the line-sphere intersection geometrical formula, each vector is analysed in respect to all atomic positions and the corresponding van der Waals spheres. If the vector is found to intersect with any of the van der Waals spheres that compose the molecule, it is discarded from further analysis. For the remaining vectors, these that are assumed to pass through the windows (see Figure 2b in the main manuscript), a set of coordinates along the vector in increments is calculated and defined as the ‘vector path’ (see Figure S2). At each point along the vector’s path, the distance of the point to the position of the closest atom is calculated and corrected for the appropriate $r_{\text{vdw}}$ of an element by subtracting it from the distance.
5. Vectors are then clustered using the density-based spatial clustering algorithm (DBSCAN) from the \texttt{scikit-learn} package (see Figure 2c in the main manuscript).\textsuperscript{S2} The number of returned clusters defines the number of windows found in the molecule.

6. The found clusters are analysed separately. For each such cluster, the vector’s paths sets are compared for its minimum, therefore the smallest distance in the set, and the vector with highest value for that minimum is determined as the rough estimate of the window centre and the coordinates of this minimum and the distance calculated is used to determine the window circular plane perpendicular to the vector (see Figure 2d in the main manuscript).

7. The diameter of a window ($D_{\text{window}}$) is determined by the largest circle that can be circumscribed into the window. The radius of that circle is defined by the distance between the centre of the window and the closest atom. From that radius, the appropriate $r_{\text{vdw}}$ of that element is subtracted and it finally is multiplied by 2 to yield $D_{\text{window}}$. First the rotation of the molecule’s coordinates is performed to align the sampling vector with the Z axis with the COM of the molecule kept at \(O\) (see Figure S3). In such an arrangement, the plane of the window is perpendicular to the \(XY\) plane of the coordinate system. The two step minimisation, where the coordinates of the window centre are used as variables and $D_{\text{window}}$ as the output is performed. The $x$ and $y$ components are minimised by using them as the variables in the \texttt{minimize} function from \texttt{SciPy} package\textsuperscript{S3} and the negative of $D_{\text{window}}$ is returned. This ensures that the correct centre is found, \textit{i.e.} one that has the largest distance to the closest atom in the window plane. Next, the optimisation of the $z$ coordinate is performed in the $Z$-axis direction, also using the \texttt{minimize} function of \texttt{SciPy} package, with the $D_{\text{window}}$ value as output. This ensures the correct distance of the window centre from the cage COM at which the window is the narrowest. For this optimised centre of the window, the $D_{\text{window}}$ value is recalculated. At this point, the reverse rotation and ini-
tial molecule’s COM translation to the O is applied to the window centre coordinates by applying the appropriate rotation matrix and adding the initial COM vector.

8. The $D_{\text{window}}$ for each window, and the central coordinates for each window with respect to the original molecule’s input coordinates are returned as output.

Figure S2: Process of finding windows in molecular pores. From the centre of mass of a molecule (COM) the projected vectors are sampled in increments of 1 Å along the vectors path (black dots). At each such point the distance to the closest atom (corrected for the appropriate van der Waals radius) is calculated. If vector passes through an atom, such as the $p_{n+2}$ vector, it is discarded. From the resulting vectors, the one that passes closest to the centre of the window marked with a green dot (vector in yellow) is then analysed as shown in Figure S3.
Figure S3: In this schematic the process of finding the window centre is presented. The coordinates of the molecule are translated and rotated so that the vector that passes closest to the window centre (in yellow in Figure S2) starts at the origin of the Cartesian system and is aligned to the Z axis. The window plane is then shifted to the origin. The $x$ and $y$ coordinates of the window centre are optimised to find the largest window diameter. The $z$ coordinate is also minimised to find the necking of the window channel and true window centre.
Method of calculating the average molecular diameter

The molecule’s average diameter ($D_{\text{avg}}$), calculated from the lowest energy modelled structures, can allow for comparison to the experimentally measured solvodynamic diameters, to help identify the reaction product when other experimental methods such as crystallisation or mass spectra cannot determine which stoichiometry product was synthesised. We used this approach successfully in recent work of ours.\textsuperscript{S4} The average diameter is calculated as follows:

1. A molecule is taken and the $D_{\text{max}}$ is determined.

2. A set of sampling points is distributed evenly on a sphere with radius equal to the $D_{\text{max}}$, using Vogel’s golden vector approach for an even points distribution on a disc, modified for a sphere (see Figure S4a).\textsuperscript{S1}

3. Each sampling point is connected to the COM of the molecule by a vector. The overlap of the vector paths and the van der Waals spheres of the molecule’s atoms is then analysed using the line-sphere intersection geometrical formula.

4. The molecule’s outline, as a set of points determined by the vectors crossing through the van der Waals spheres, is created (see Figure S4b).

5. The average of the distances of the molecular outline points yields the $D_{\text{avg}}$ and is given by the equation:

$$D_{\text{avg}} = \frac{2}{n} \sum_{i=1}^{n} x_i$$  \hspace{1cm} (4)

where $x_i$, is equal to the distance of the $i^{th}$ outline point from the centre of mass (see Figure S4c).
Figure S4: The simplified 2D model of the method for the calculation of the molecule’s average diameter: (a) a sphere of evenly distributed sampling points is generated around a molecule and the vectors connecting the sampling points and the centre of mass of the molecule (green dot) are analysed using the line-sphere intersection geometrical formula; (b) the intersection points define an outer outline of the molecule; (c) the average distance to the outer outline from the centre of mass yields the average diameter of a molecule.
**pywindow results for the studied examples**

Presented here are code snippets and outputs of the analysis, performed with pywindow, of individual molecular pores (CSD reference: YAQHOQ, BATVUP, NUXHIZ, PUDXES, REYMAL), the metal-organic cage (CSD reference: SAYGOR), and the representative fragment of a metal-organic framework (CSD reference: MIBQAR), presented in the main manuscript. All the input files and the code examples (including the analysis of molecular dynamics trajectories), can be also found in the pywindow GitHub repository: github.com/JelfsMaterialsGroup/pywindow in the Examples directory.

```
In [1]: import pywindow as pw

In [2]: molsys = pw.MolecularSystem.load_file("./YAQHOQ.xyz")

In [3]: mol = molsys.system_to_molecule()

In [4]: mol.full_analysis()

Out[4]: {
    'centre_of_mass': array([-1.66666667e-05, -4.92980557e-18, 3.33333333e-05]),
    'maximum_diameter': {'atom_1': 43,
                        'atom_2': 54,
                        'diameter': 10.495187523948891},
    'no_of_atoms': 60,
    'pore_diameter': {'atom': 11, 'diameter': 3.6101494139251806},
    'pore_diameter_opt': {'atom_1': 42,
                          'centre_of_mass': array([ 0.00332103,  0.01618183, -0.00053063]),
                          'diameter': 3.628842522285096},
    'pore_volume': 24.636224433953796,
    'pore_volume_opt': 25.023835308827408,
    'windows': {'centre_of_mass': None, 'diameters': None}
}
```

Figure S5: Results for YAQHOQ
Figure S6: Results for BATVUP

Figure S7: Results for NUXHIZ
Figure S8: Results for PUDXES
Figure S9: Results for REYMAL
Figure S10: Results for SAYGOR

```
In [1]: import pywindow as pw

In [2]: molsys = pw.MolecularSystem.load_file("./SAYGOR.pdb")

In [3]: mol = molsys.system_to_molecule()

In [4]: mol.full_analysis()

Out[4]: {'centre_of_mass': array([[ 21.05422233, 10.13114265, 0.90872813]],
                             'maximum_diameter': {'atom_1': 316,
                                                  'atom_2': 345,
                                                  'diameter': 32.514780269519015},
                             'no_of_atoms': 440,
                             'pore_diameter': {'atom': 411, 'diameter': 8.8504405705868212},
                             'pore_diameter_opt': {'atom_1': 411,
                                                   'centre_of_mass': array([ 20.89674406, 10.30869434, 1.05311357]),
                                                   'diameter': 9.4049849477381215},
                             'pore_volume': 362.98885667376584,
                             'pore_volume_opt': 435.58502104435024,
                             'windows': {'centre_of_mass': array([[ 23.15736804, 12.82039376, 4.78076577],
                                                         [ 20.78772048, 3.84868555, 1.7882738 ],
                                                         [ 16.21087654, 12.37202167, -0.03612376],
                                                         [ 22.48958164, 11.49730933, -2.50068636]),
                                                         'diameters': array([ 7.89184444, 8.29658872, 5.95684323, 6.80863739])})}}
```
Figure S11: Results for MIBQAR
Circumcircle atom sets

Figure S12: The atoms chosen in the PUDXES molecules to represent the window diameters in the circumcircle method. For each window, three carbon atoms that lie in the same plane and describe the narrowest point of the window channel necking were chosen. These are labelled with numbers here.
Figure S13: Window diameters and window centres calculated using the circumcirle method in pywindow. Implemented as described in Holden et al. work. The input atoms sets indexes are reduced by 1 to facilitate the numbering format in python that starts with 0.
Limitations of *pywindow*

The *pywindow* package can be unfit for the analysis of some molecular pores. The definition of the window that has been described earlier states that it is the necking in the molecule’s structure that connects the intrinsic void and the molecule’s exterior. However, the example of a molecular pore can be found where such a necking is not present, or to be more accurate, it is the intrinsic void that is the narrowest point, as shown in Figure S14.

![Figure S14: The example of a molecular pore where the intrinsic void (green solid sphere) is simultaneously the channel necking - the window.](image)

Additionally, the assumption that the window is of a circular shape will result in variety of range of underestimations of the actual window’s area. For the PUDXES molecule, in the middle of Figure S15, the window is actually of a triangular shape. However, the assumption of the circular window shape is intentional for its simplicity. In some extreme cases, such as cryptophane-A (far right in Figure S15), very narrow windows close to the shape of number 8 can result in each of the circles to being identified as a separate window and the wrong number of windows output by *pywindow*. 
Figure S15: Examples of molecular pores with various windows shapes and the possible fit of the spherical window approximation (green dashed circle) as assumed in the pywindow package window diameter calculation. Molecules not shown to scale.
References

(S1) Vogel, H. A Better Way to Construct the Sunflower Head. *Math. Biosci.* 1979, 44, 179–189.

(S2) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 2011, 12, 2825–2830.

(S3) Jones, E.; Oliphant, T.; Peterson, P. SciPy: Open Source Scientific Tools for Python. 2001--; [http://www.scipy.org/](http://www.scipy.org/).

(S4) Greenaway, R. L.; Santolini, V.; Bennison, M. J.; Alston, B. M.; Pugh, C. J.; Little, M. A.; Miklitz, M.; Eden-Rump, E. G. B.; Clowes, R.; Shakil, A.; Cuthbertson, H. J.; Armstrong, H.; Briggs, M. E.; Jelfs, K. E.; Cooper, A. I. High-throughput discovery of organic cages and catenanes using computational screening fused with robotic synthesis. *Nat. Commun.* 2018, 9, 2849.

(S5) Holden, D.; Jelfs, K. E.; Cooper, A. I.; Trewin, A.; Willock, D. J. Bespoke Force Field for Simulating the Molecular Dynamics of Porous Organic Cages. *J. Phys. Chem. C* 2012, 116, 16639–16651.
