Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

pytrax: A simple and efficient random walk implementation for calculating the directional tortuosity of images

T.G. Tranter [a,b], M.D.R. Kok [b], M. Lam [a], J.T. Gostick [a,*]

[a] Department of Chemical Engineering, University of Waterloo, 200 University Ave. W, Waterloo, ON, Canada
[b] Electrochemical Innovation Lab, Department of Chemical Engineering, University College London, UK

ARTICLE INFO

Article history:
Received 15 January 2018
Received in revised form 4 April 2019
Accepted 28 June 2019

Keywords:
Random walk
Directional tortuosity
Python
Image analysis

ABSTRACT

Given the huge advances in tomographic imaging capability in recent years, image analysis has become a powerful means of measuring transport and structural properties of porous materials. One of the most important material characteristics is the tortuosity, which is difficult to measure experimentally. We present pytrax: (tortuosity from random axial movements) a simple and efficient random walk method implemented in python to calculate the average tortuosity and orthogonal directional tortuosity components of an image. The code works for both two and three-dimensional images and completes a statistically significant number of walks in parallel for large images in a few minutes using a standard desktop computer. By comparison, a Lattice Boltzmann or finite element simulation on similar sized images can take several hours.

Code metadata

Current Code version	*0.1.2*
Permanent link to code / repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-18-00011
Legal Code License	*MIT*
Code Versioning system used	*git*
Software Code Language used	*python*
Compilation requirements, Operating environments & dependencies	*The Anaconda python distribution is recommended which provides all dependencies. In addition porespy is recommended for use of export function and image generation* https://github.com/PMEAL/porespy. Both packages can be found on pip
If available Link to developer documentation / manual	http://pytrax.readthedocs.io/en/latest/
Support email for questions	t.g.tranter@gmail.com

1. Motivation and significance

The study of transport in porous media cuts across a vast range of disciplines: from medicine, biology and the earth sciences to electrochemistry and microfluidics. All of these applications wish to know the transport properties of the porous structures, as these dictate some aspect of the final performance. There have been a wide range of experimental techniques applied to obtain this information, but they are often quite challenging to implement, particularly for thin or nanoscale materials found in many modern devices (i.e. electrodes). The advent of ubiquitous X-ray tomography equipment has changed this dynamic immensely. It is now possible to obtain a tomographic image of a material in a matter of hours (or seconds at a synchrotron), with resolutions as small as 30 nm per voxel. With such images available it becomes possible to probe the transport properties of materials by conducting direct numerical simulations using tools such as the Lattice Boltzmann method (LBM) directly on the images. Alternatively images can be used as the basis for constructing a mesh representation of the material and using a

* Corresponding author.
E-mail address: jgostick@uwaterloo.ca (J.T. Gostick).

https://doi.org/10.1016/j.softx.2019.100277

finite element, or finite volume method. These approaches solve the Navier–Stokes (NS) equations in discretized form, or in the case of LBM Hermitian form, and depending on the size of the domain and resolution of the mesh/image, may take hours or even weeks to solve if the problem is transient. The growing prominence and increasing resolution of tomographic scanning devices now means that with increasing amounts of data, the analysis of the images produced actually presents a significant bottle-neck to the overall scientific process.

Along with porosity, which is trivial to calculate from an image, tortuosity is an important parameter for predicting transport behaviour in porous media and is often used in constitutive relations [1]. Ideally, we like to imagine that tortuosity is a material property that indicates the connectivity, and general shape of the void space. One definition of tortuosity is "the increased path length taken by flow through porous media due to obstacles". However, the observed increase depends on the type of flow. Pressure driven hydraulic flow is more impeded by the presence of small pores compared with diffusive flow due to the no-slip condition at walls, giving rise to different values of tortuosity depending on the transport mechanism at play, thus a general definition is lacking [2]. Efforts to find relations between tortuosity and porosity exist, such as the classic [3]: $\tau^2 = \varepsilon^{-1/2}$ and many other examples [1] have been made, but no generally applicable relation exists due to the wide variety of geometrical configurations found across porous media. The diffusive tortuosity is generally the most widely applicable definition [4] and it has been shown that this property can be calculated simply using random walks [5,6].

The present work outlines a framework to address the problem of calculating tortuosity from an image. Comparisons to diffusive flow can be made as the randomness of the walk is intended to simulate Brownian motion and the tortuosity should be thought of as the ratio of diffusivity in free space to the diffusivity in the porous media [5]. For a simple walk with equal and unbiased probability of movement in any direction, the mean square displacement of the walkers follows a Gaussian distribution. It can be shown that the one-dimensional PDF for the location of a walker after time t is given by [7,8]:

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} \exp\left(\frac{-x^2}{4Dt}\right) \tag{1}$$

This is the fundamental solution of the diffusion equation where D is the diffusion coefficient. Furthermore, the mean square displacement (MSD) is shown to be:

$$MSD = E\left(X_t^2\right) = \int_{-\infty}^{\infty} x^2 p\left(x,t\right) dx = 2Dt \tag{2}$$

This generalizes in N dimensions to [8]:

$$\begin{aligned} E\left(R_t^2\right) &= \int_{-\infty}^{\infty} |\boldsymbol{x}|^2 \, p\left(\boldsymbol{x},t\right) d\boldsymbol{x} \\ &= \int_{-\infty}^{\infty} (x_1^2 + \cdots + x_N^2) p\left(x,t\right) dx_1 \ldots dx_N = 2NDt \end{aligned} \tag{3}$$

Therefore, the contributions of movements in each direction sum together and the MSD can be decomposed into N Axial Square Displacements (ASD):

$$ASD_i = E\left(X_{i,t}^2\right) = \int_{-\infty}^{\infty} x_i^2 p\left(x_i,t\right) dx = 2D_i t \tag{4}$$

Therefore, given the general definition of tortuosity as the ratio of MSD in free space and porous media [5] and the fact that MSD increases linearly and directly proportional to time in open space. We define the axial tortuosity component in each orthogonal direction (i) as follows:

$$\tau_i = \frac{t}{N} \frac{1}{E\left(X_{i,t}^2\right)} \tag{5}$$

where the factor t/N comes from the fact that each axial movement is independent and the probability of moving in any direction is equal. In other words, the direction of travel for each walker is restricted to the orthogonal axes and the approximate time spent travelling along each axis is equal.

The main motivation behind this work is to provide a simple and efficient method for calculating the tortuosity of an image using random walks in a highly parallelized way. This is achieved by calculating the reciprocal of the slope of the MSD and ASD vs. time which would be unity in open space.

The reasons behind the emphasis on simplicity and efficiency are two-fold and address the needs of two different target audiences: The first group are experimentalists that are interested in analysis of the image data that they collect and who may not be coding specialists; writing the package in python keeps the code readable and easily distributable. The other target audience are computationally focused scientists who may be looking for a good starting point to adapt the code. As we have already mentioned, random walks have been used for a variety of different purposes. We provide an 'off-the-shelf' solution for quick image analysis on massive datasets that provides the mean tortuosity and axial tortuosity which is very useful for anisotropic media. The *pytrax* package is a small code-base and has much in-line documentation with implementation of vectorized array indexing and parallel processing making it perfect for extensible use.

2. Software description

The *RandomWalk* class contains less than 650 lines of code, over half of which is comprised of comments and methods to export or show the data in various formats. The main python package used is *Numpy* [9], which along with *Scipy* [10] forms the number one tool for scientific computing in python. This is primarily because these packages implement highly efficient C-code behind the scenes, providing that the function calls make use of vectorization. The algorithm parallelizes very well for increasing numbers of walkers as each walk can be entirely computed in a concurrent fashion as the individual walker movements are independent of any other. For analysis of large images, it is better to increase the number of walkers rather than the number of time steps as the simulation will complete more quickly and the results will be similar.

2.1. Random walk rules

A thresholded image of dimension (nd) is required to run the program which should be of integer format, the package assumes that 0 represents the solid space and 1 represents the pore space accessible to the walkers. A designated number of walkers (nw) are given randomly assigned starting coordinates inside the pore space by default but can be set to have the same or different starting positions. It is better to start them in different places to probe the whole image in fewer steps. Then a designated number of steps (nt) are taken by each walker in parallel. It is assumed that all walkers travel at the same speed and that time is discrete, although it would be fairly simple to record continuous time-steps.

Rules govern the completion of each step: It is assumed that walkers may not enter the solid phase, although for certain simulations this could be possible for example intra-diffusion through multiple fluid phases, adsorption, heat transfer, abstract phase

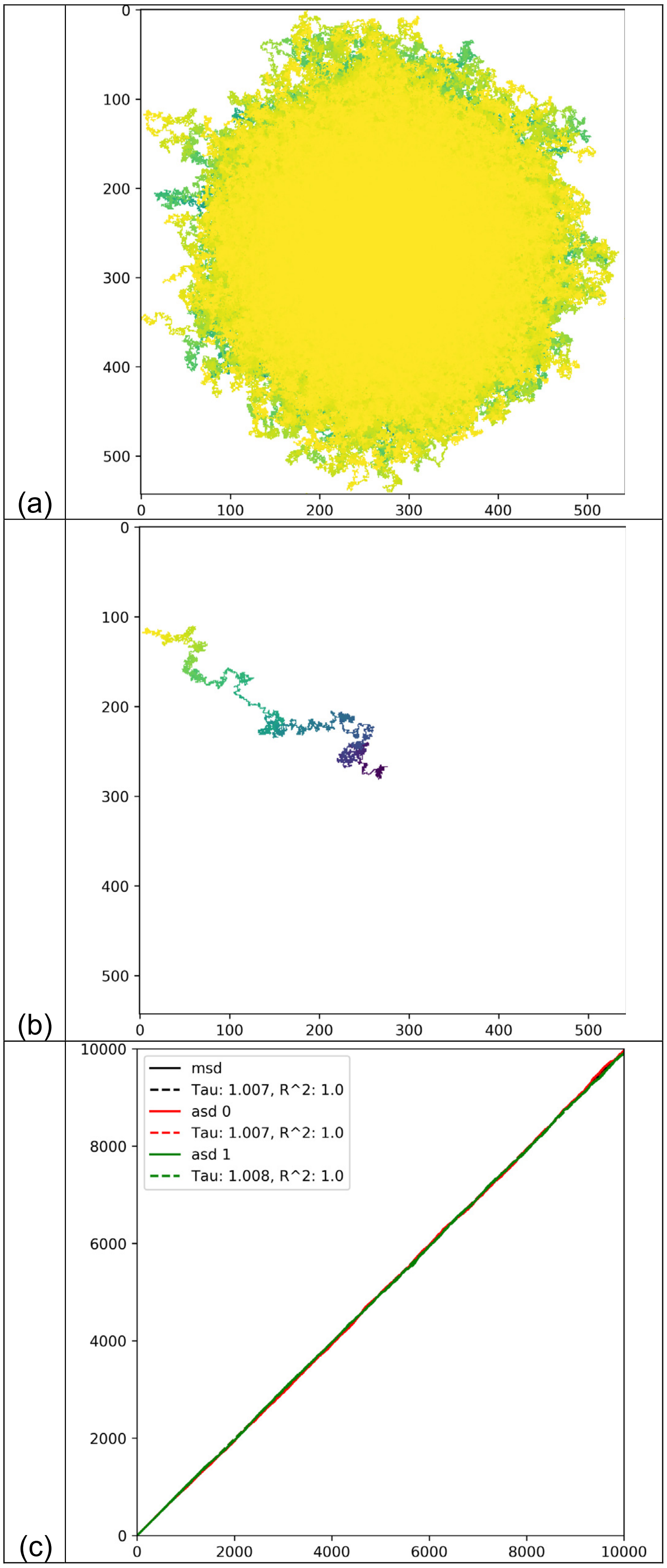

Fig. 1. Open Space (a) all walkers, (b) longest walk, (c) mean and axial square displacement vs time with movement in each axis equal and tau = 1.

Table 1
Simulation parameters for each example.

Example	Number of walkers	Number of time steps
Open space	10,000	10,000
Sierpinski carpet	1,000,000	2500
Tau	1000	20,000
Cylinder packs	10,000	10,000

space exploration to name a few. Walkers are considered to move from unit cell (pixel/voxel) centre to adjacent cell centre in one time step and may only move in orthogonal directions. Diagonal movement involving more than one axis would complicate the axial tortuosity calculation and studies using the Lattice Boltzmann method to simulate diffusion have found that orthogonal movement is sufficient to produce results with good accuracy [11]. A walker that at time t is adjacent to a solid unit and whose random movement would take it into the solid phase will be returned to its original position at the next step. A walker effectively travels half a unit cell, is reflected, and travels back again when hitting a solid.

Special consideration must be given to walkers that reach the edge or boundary of an image. It is assumed that images are subsections of some larger space, so upon reaching the image boundary, the walker must not be confined within the image boundaries or else the walks would become artificially confined and tortuosity would be overestimated as the MSD would have an upper limit. Confining walkers is not a faithful representation of the transport dynamics through the material as a whole, upon which constitutive relations are based. To address this issue, the trajectories of the walkers passing through the boundaries are considered to take place in reality and such walkers have their coordinates projected into a reflected image about the boundary along the axis of travel that produced the reflection. This is achieved quite simply by storing a *reality* array which is $[nw \times nd]$ long and is initially filled with value 1. When a walker passes through a boundary the value in the reality array corresponding to that walker and that axis of travel is switched to -1 and all updates to the coordinate from further random movements are multiplied by the *reality* array. The next time the walker passes through a boundary along this particular axis, whether it is the same or opposite side, the sign of the vector component is switched again. The walkers can be thought of as travelling through a tessellation of real and mirrored images and the *plot_walk_2d* function in *pytrax* enables visualization of this for 2D images (all examples pictured in 2D are combinations of real and mirrored images). The same scenario happens in 3D and each axis can be treated completely independently as movements are restricted to happen along one axis at a time during each step. However, for visualization in 3D the '*export_walk*' function is provided and the generated files '.vti' for the image and a series of '.vtu' for the coordinates of each walker at each time-step, may be visualized with *Paraview* using particle trace filters. To save on memory, only the original image is exported in 3D. Should a user prefer a random walk without reflection, they can simply pad the image with solid unit cells to prevent the walker from ever entering the reflected domain.

Once the walks are complete, an ensemble of coordinates is available to calculate the mean square displacement vs. time. The tortuosity is extracted by the inverse of the slope of the mean square displacement which is calculated with linear regression forcing the intercept through zero. The axial tortuosity can be calculated similarly along each axis and then by multiplying the result by the number of axes (Tau in the figure legends refers to this mean or axial component of tortuosity). The chosen axis for movement is uniformly random and so over the period of thousands to millions of steps the number of steps taken along each axis is approximately equal. The displacement along each axis will depend on the configuration of obstacles and for paths that are more tortuous along a particular axis the walkers will hit more walls and travel a shorter distance. It would be entirely possible to bias the selection of the axis and also direction of movement

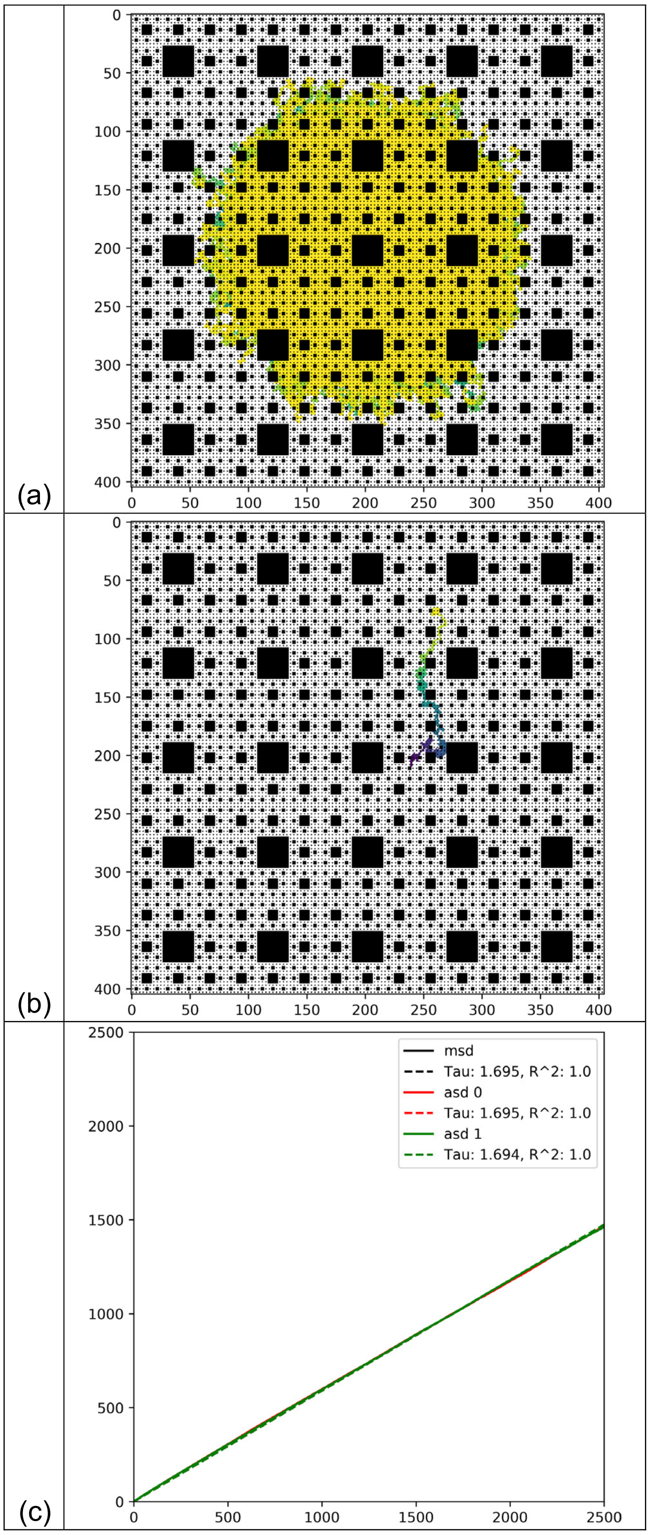

Fig. 2. Sierpinski Carpet (a) all walkers, (b) longest walk, (c) mean and axial square displacement vs time, with all axial movements equal and tau greater than 1.

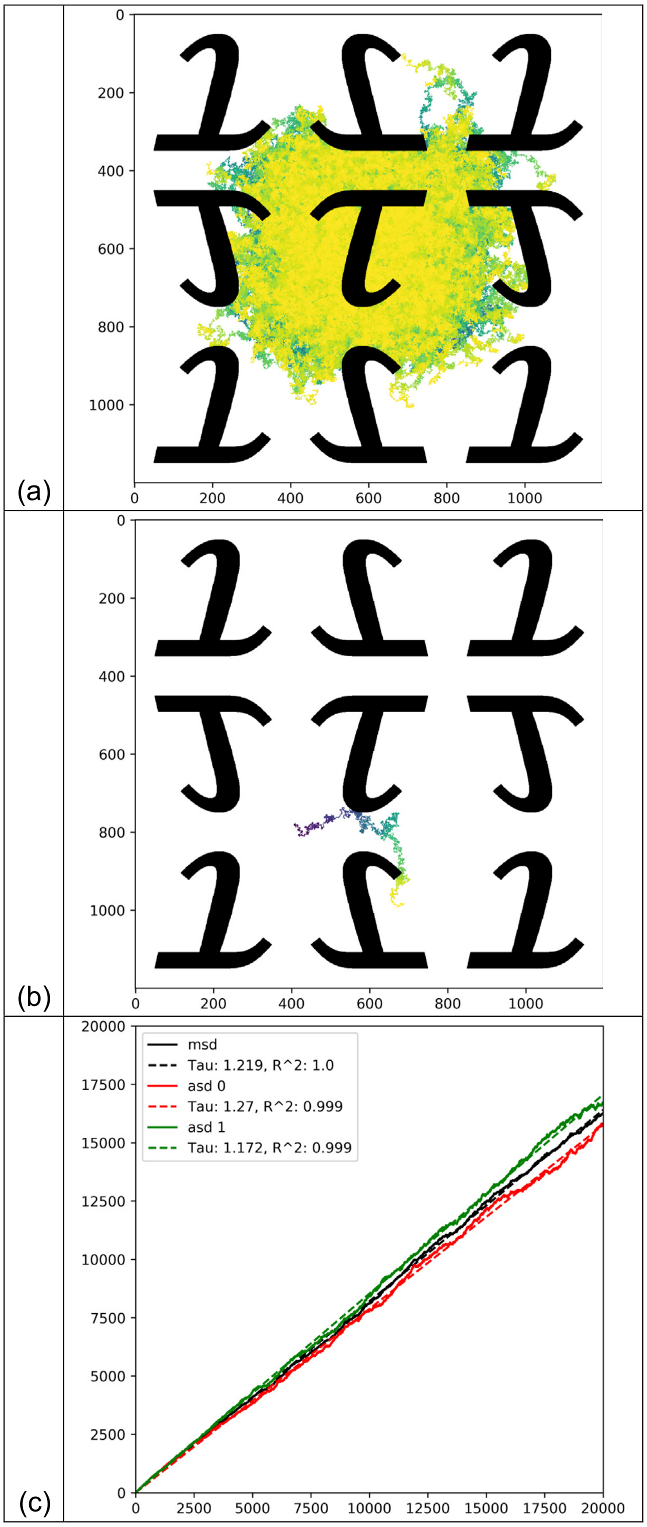

Fig. 3. Space with tau-shaped obstacles: (a) all walkers, (b) longest walk, (c) mean and axial square displacement vs time showing some anisotropy, finite size effects and variation from insufficient number of walkers.

along a particular axis non-uniformly to simulate effects such as gravity or magnetic fields if the walkers are to represent charged particles. However, this extension is left for future work.

3. Provided examples

The following section provides four examples of the random walk to determine the components of the tortuosity, the details of which are provided in Table 1.

3.1. Open space

The simplest example is a random walk in open space. Without the presence of obstacles the walkers are free to move in any direction unimpeded and their mean square displacement increases linearly with time as per Equation (2). The image passed to the simulation is an integer array of ones with shape [3, 3]. Fig. 1a shows the position of all the walkers at the final time step with a circular shape simulating radial diffusion away from a point source. Fig. 1b shows the longest walk where the mean square displacement is the largest coloured by time, and Fig. 1c shows the mean square displacement and axial displacement in both principal directions. The displacement along each axis is equal as one would expect from an isotropic image and the slope of the curves are all equal to unity meaning that tortuosity (1/slope) is also unity, in agreement with theory. Note that the image size displayed is much larger than the original because the real and reflected domains are visualized in 2D by default.

3.2. Sierpinski carpet

The second example is a fractal image known as the Sierpinski Carpet which was generated by calling the following function recursively:

```
def tileandblank(image, n):
  if n > 0:
    n -= 1
    shape = np.asarray(np.shape(image))
    image = np.tile(image, (3, 3))
    image[shape[0]:2*shape[0], shape[1]:2*shape[1]] = 0
    image = tileandblank(image, n)
  return image

im = numpy.ones([1, 1], dtype=int)
im = tileandblank(im, 4)
```

The resulting image is isotropic and square with edge length $3^4 = 81$ with a repeating pattern at four length scales where central sections are designated as solid space. The results are similar to the open space example and shown in Fig. 2. Here the full image with real and reflected parts is shown with the original image at the centre. The walkers travel far enough to escape the original image and completely traverse the neighbouring reflected images again so a second reflection occurs along each axis at the outer boundaries. The presence of obstacles means that the MSD increases less than in the open space over time and as the paths are more tortuous, yet still equal in each direction due to the isotropy of the image and uniformly random starting positions.

3.3. Space with tau-shaped obstacles

The third example uses an image of the Greek symbol tau (commonly used to denote tortuosity) which is of size [400, 400]. This example illustrates the need to pick a significant number of walkers and time-steps. The image is also somewhat anisotropic

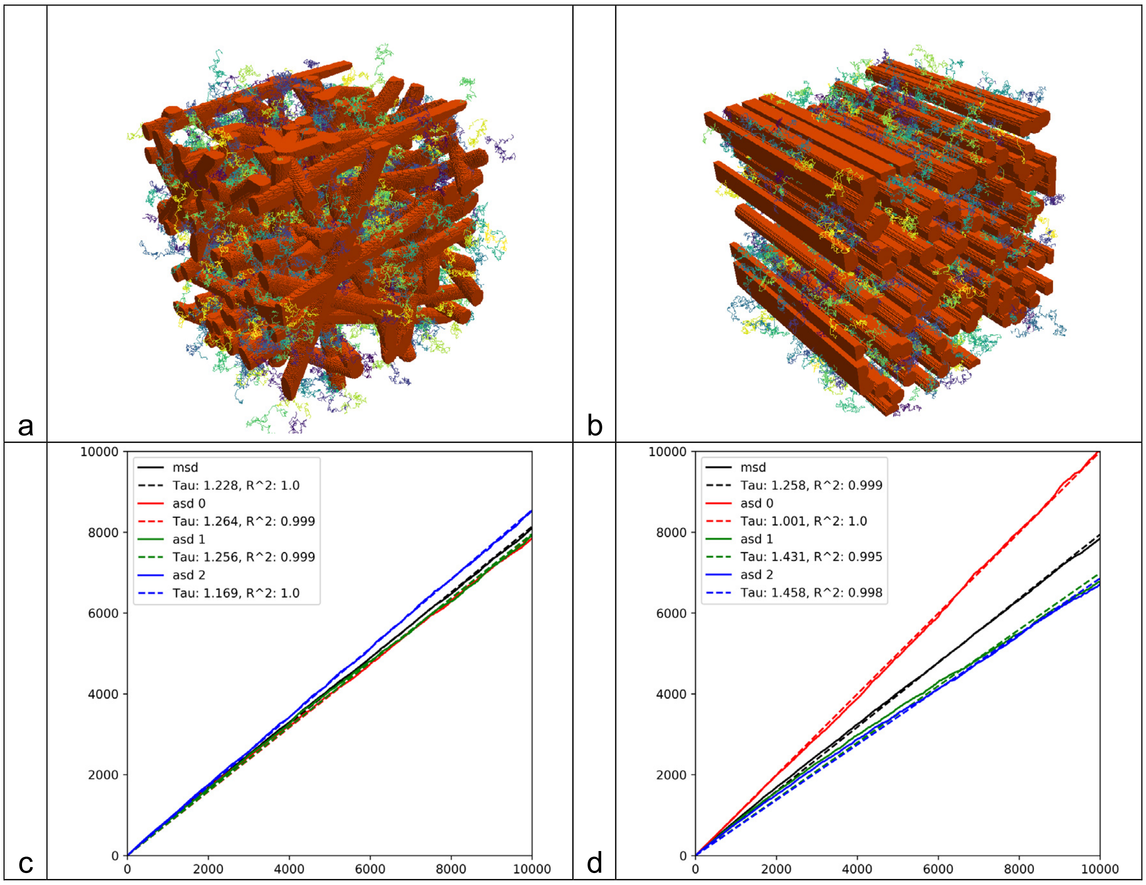

Fig. 4. A selection of walkers in a random cylinder pack (a) and aligned cylinder pack (b). MSD and ASD for random alignment (c) and axial alignment (d). Random alignment is not perfectly isotropic and so some differences are observed in ASD. Clear differences are observable for the axis with aligned cylinders and higher tortuosity for the other two axes.

with movement left and right being slightly easier than up and down which manifests in differing axial displacements. The fitted linear slopes are not such a good fit to the actual displacements which follow a non-linear curve. This is because the image has one large pattern giving a pore size comparable to the image size and also comparable to the length of the walks. Fig. 3(a) and (b) show that the walkers manage to escape the original image as the domain is reflected about each axis. However, the MSD shown in Fig. 3(c) is not straight or smooth. Increasing the number of walkers would smoothen the displacement curves but the non-linearity is a feature of the walkers not fully probing all areas of the image a significant number of times. The image is not a typical example of a porous material but demonstrates the need for walks to be much larger than the average pore size.

3.4. *Random and aligned cylinder packs*

The final example compares two images of cylinder packs with the same porosity but different alignment to show the effect of anisotropy on the ASD. Both images are 300^3 voxels in size and contains 100 cylinders with a radius of 10 voxels that are oriented in an isotropic and anisotropic fashion. Fig. 4a and b show the walker positions with tracers coloured by time step at the end of each simulation and were visualized with *Paraview* [12] with data using the *export_walk* function. Fig. 4c and d shows the isotropic and anisotropic axial displacements. The cylinders in the second case are aligned along the first principal axis and consequently the walkers have fewer obstacles to flow and the tortuosity is lowest along this axis. In fact there are effectively no obstacles to flow along this axis as the cylinders are perfectly aligned with the walker direction giving an axial tortuosity very close to one. This demonstrates that the method does in-fact probe the porosity independent tortuosity and not the effective tortuosity obtained by other methods which has porosity included.

4. Performance

To demonstrate the performance of the package a *run_analytics* function is provided which takes lists of the number of walkers and time steps then iterates through all combinations in turn, recording the simulation time. A number of parallel processes to compute over is also accepted as an argument and walkers are divided into batches using python's multiprocessing module. A 3D isotropic image of segmented random noise with side length 100 is used to demonstrate the performance of the package and the results are shown in Fig. 5 for a single process and 5 and 10 parallel processors. The simulations were performed on a desktop computer with an Intel® Xeon® CPU with two 2.76 GHz processors with a total of 24 cores and 72.0 GB of RAM. The results show that the single process outperforms the parallel processors for simulation times around 10 s and less with few time steps as there is some overhead for setting up the process pool. However, for longer simulations with larger numbers of walkers typical of the simulations needed to investigate modern tomography images, the speed up in parallel processing is worth the overhead as simulation times are reduced by a factor roughly proportional to the number of processors. The simulation time also scales linearly with the size of the image as shown in Fig. 6 with an image comprising over 100 million voxels completing 10,000 steps for 10,000 walkers in less than a minute using 10 processors.

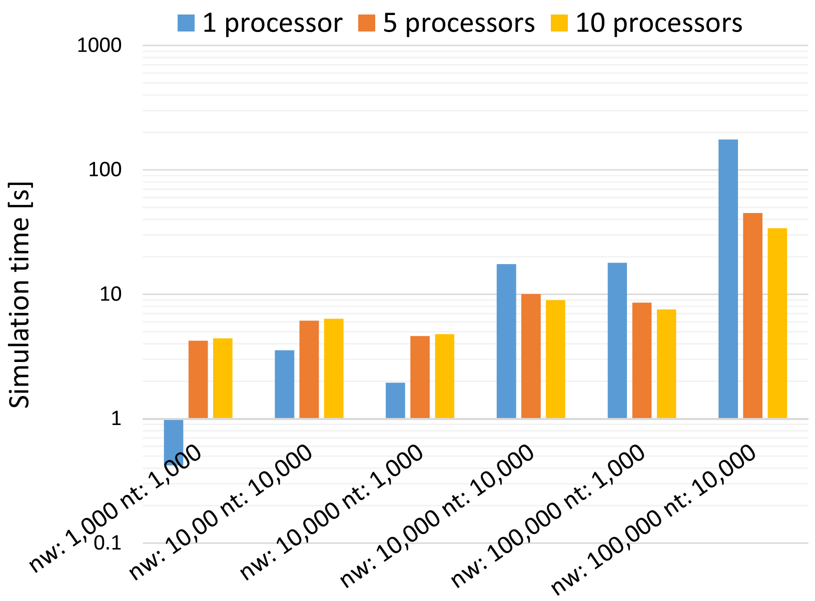

Fig. 5. Simulation times for 100^3 voxel noisy image with various numbers of walkers (*nw*), time steps (*nt*) and parallel processes. Some overhead is introduced by running parallel processing but the benefits pay-off significantly for larger images that require longer simulations and more walkers.

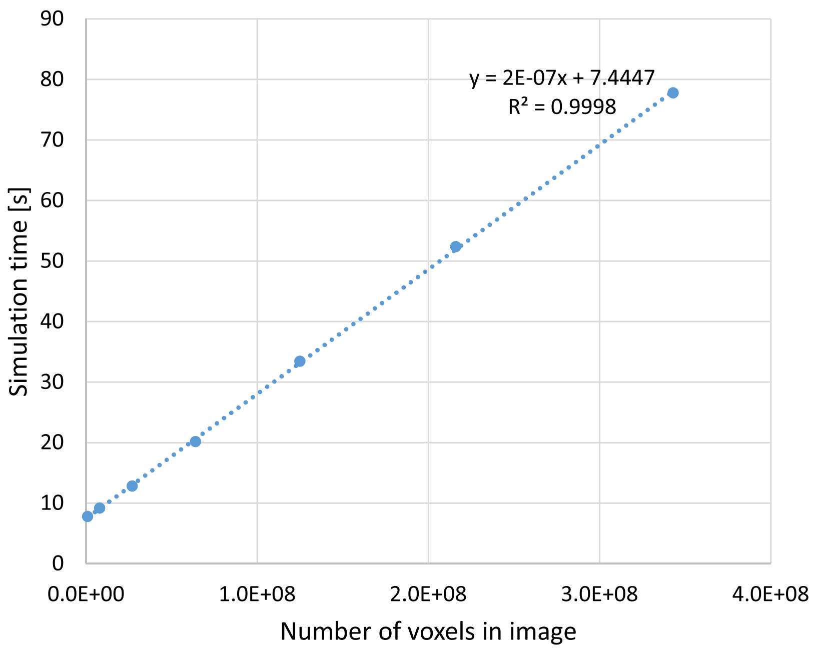

Fig. 6. Simulation times for 10,000 walkers and 10,000 steps for noisy image of 0.75 porosity applied to a series of domains of increasing size (largest 700^3), computed using 10 parallel processes.

5. Conclusion

We first present *pytrax* the open source python package for calculating tortuosity of images by random axial movement. The package is light-weight, simple and efficient and will allow the fast calculation of the tortuosity in all principle directions of large images in just a few minutes. The method employs a discrete time unbiased random walk which is the most simple implementation but is written in a modular fashion to allow easy extension. The package should allow for fast analysis of large microstructural datasets applicable to all sciences concerned with the study of porous media.

6. Data disclosure statement

All data used for the study is contained and presented within the paper.

Acknowledgments

Funding for this work was provided by the Natural Science and Engineering Research Council of Canada through the Discovery grant and Undergraduate Student Research Award program, as well as a Collaborative Research and Development grant in partnership with the Automotive Fuel Cell Cooperation.

Declaration of competing interest

The authors declared that they had no conflicts of interest with respect to their authorship or the publication of this article.

References

[1] Shen L, Chen Z. Critical review of the impact of tortuosity on diffusion. Chem. Eng. Sci. 2007;62(14):3748–55.

[2] Clennell M Ben. Tortuosity: a guide through the maze. Geol. Soc. London Spec. Publ. 1997;122(1):299–344, Available at: http://sp.lyellcollection.org/lookup/doi/10.1144/GSL.SP.1997.122.01.18.

[3] Bruggeman DAG. Berechnung verschiedener physikalischer Konstanten von heterogenen Substanzen. III. Die elastischen Konstanten der quasiisotropen Mischkörper aus isotropen Substanzen. Ann. Phys. 1937;421(2):160–78.

[4] Cooper SJ, et al. TauFactor: An open-source application for calculating tortuosity factors from tomographic data. SoftwareX 2016;5:203–10, Available at: http://dx.doi.org/10.1016/j.softx.2016.09.002.

[5] Watanabe Y, Nakashima Y. Two-dimensional random walk program for the calculation of the tortuosity of porous media. J. Groundw. Hydrol. 2001;43(1):13–22, Available at: https://www.jstage.jst.go.jp/article/jagh1987/43/1/43_13/_article.

[6] Nakashima Y, Watanabe Y. Estimate of transport properties of porous media by microfocus X-ray computed tomography and random walk simulation. Water Resour. Res. 2002;38(12):1271, Available at: http://www.agu.org/pubs/crossref/2002/2001WR000937.shtml.

[7] Chandrasekhar S. Stochastic problems in physics and astronomy. Rev. Modern Phys. 1943;15(1):1–89.

[8] Codling EA, Plank MJ, Benhamou S. Random walk models in biology. J. R. Soc. Interface 2008;5(25):813–34, Available at: http://rsif.royalsocietypublishing.org/cgi/doi/10.1098/rsif.2008.0014.

[9] Van Der Walt S, Colbert SC, Varoquaux G. The NumPy array: A structure for efficient numerical computation. Comput. Sci. Eng. 2011;13(2):22–30.

[10] Jones E, Oliphant T, Peterson P. SciPy: Open source scientific tools for Python. 2001, Available at: http://www.scipy.org/.

[11] Huang HB, Lu XY, Sukop MC. Numerical study of lattice Boltzmann methods for a convection–diffusion equation coupled with Navier–Stokes equations. J. Phys. A 2011;44(5).

[12] Henderson A. Paraview Guide, A Parallel Visualization Application. Kitware Inc., 2007, Available at: http://www.kitware.com/products/paraview.html.