

Supporting Information for

Numerical investigation of 2D ordered pillar array columns: an algorithm of unit-cell automatic generation and the corresponding CFD simulation

Qihao Jiang ^{1,†}, Stefano Rocca ², Kareem Shaikhuzzaman ¹ and Simone Dimartino ^{1,*}

¹ Institute for Bioengineering, School of Engineering, The University of Edinburgh, Edinburgh EH9 3BF, UK

² School of Engineering and Architecture, University of Bologna, Via Zamboni, 33, 40126 Bologna, BO, Italy

* Correspondence: simone.dimartino@ed.ac.uk

[†] Current address: Key Laboratory of Information and Structure Efficiency in Extreme Environment, The Ministry of Education of China, The School of Aerospace Science and Technology, Xidian University, Xi'an 710126, China.

Section S1 Column-based approach for matrix generation

Rather than using single elements in X to generate a new matrix, columns $x_{:,j}$ is introduced to represent the basic matrix X . The maximum value of j is the order of X , which is equal to c .

$$X_{c \times c} = (x_{:,1} \dots x_{:,c}) \quad (S1)$$

This approach is recorded in an algorithm named pathway history, which is detailed reported as the following. This requires the details of the possible columns of X . These columns are stored in another matrix called F , which contains c rows and $2^c - 1$ columns in total (one column with all elements equal to 1 is removed from F). Each column in F is recognized as $f_{:,j}$. Therefore, the matrix F can be written as:

$$F_{c \times 2^c} = \begin{pmatrix} f_{1,1} & \dots & f_{1,2^c} \\ \dots & \dots & \dots \\ f_{c,1} & \dots & f_{c,2^c} \end{pmatrix} = (f_{:,1} \dots f_{:,2^c}) \quad (S2)$$

Here is an example of F of the matrix X with order equal to 3:

$$F_{3 \times 2^3} = \begin{pmatrix} 0001110 \\ 0100011 \\ 0010101 \end{pmatrix} \quad (S3)$$

In this way, the generation of new matrix X can be thought of as choosing c columns from the corresponding matrix F . This process allows for a detailed control on the morphology of the generated unit cells. There are two main reasons for this: firstly, the labelling of columns

in F allows exact knowledge of what columns are presented in X . Secondly, when a new column is added, it is possible to check if the constraints are fulfilled from the first column to the new added one.

At the start, a new matrix X is defined as a zero matrix. In each step, a generic column $f_{:,g}$ is inserted into X , starting from $x_{:,1}$ and continuing until $x_{:,c}$. Without considering any constraints, the first c steps can be outlined as follows:

Step 1: $f_{:,1}$ is inserted in X as $x_{:,1}$
Step 2: $f_{:,1}$ is inserted in X as $x_{:,2}$
Step 3: $f_{:,1}$ is inserted in X as $x_{:,3}$
.....
Step c : $f_{:,1}$ is inserted in X as $x_{:,c}$

This new generated X (adding each subscript of the column), can be written as:

$$X1_{c \times c} = (f_{:,1_1} \quad f_{:,1_2} \quad \dots \quad f_{:,1_{c-1}} \quad f_{:,1_c}) \quad (S4)$$

To generate the next new matrix $X2$, one more step is required. As the difference between $X1$ and $X2$ is about $x_{:,c}$, $x_{:,c}$ is substituted by $f_{:,2}$. The new matrix can be generated by inserting all of the columns from F to $x_{:,c}$. This process is shown below.

Step $c+1$: $f_{:,2}$ is inserted in X as $x_{:,c}$
Step $c+2$: $f_{:,3}$ is inserted in X as $x_{:,c}$
Step $c+3$: $f_{:,4}$ is inserted in X as $x_{:,c}$
.....
Step c_F : $f_{:,c_F}$ is inserted in X as $x_{:,c}$

Here, c_F is the number of columns in F . Starting from $X1$, a total of $c_F - 1$ matrices are generated in $c_F - 1$ steps. These matrices can be represented as:

$$X\alpha_{c \times c} = (f_{:,1_1} \quad f_{:,1_2} \quad \dots \quad f_{:,1_{c-1}} \quad f_{:, \alpha_c}) \quad (S5)$$

Here, α is an integer between 1 and c_F . To generate new matrices, the next step is to change $x_{:,c-1}$ from $f_{:,1}$ to $f_{:,2}$, and keep the $x_{:,c}$ as a zero column. Following this way (acting like a mechanical counter), all of the possible matrices can be generated from F . It is important to note that there are still no constraints involved in the generation process, so the matrices generated in this way do not necessarily satisfy them.

However, for the vertical symmetry, it is not possible to generate a corresponding algorithm of the matrix construction process. This is because the matrix X is constructed column-by-column, the rows in X cannot be controlled during the matrix generation. An alternative is to examine all generated matrices after the generation procedure, and remove the unit cells presenting the vertical symmetry issue. This is the horizontal symmetry model (VS) in this

work. This approach is not straightforward for X matrices of higher orders due to the large number of matrices generated.

Section S2 Principal and secondary pathways

Every time a new column is inserted into X , it is necessary to check if the constraints are fulfilled. There are five possible cases which need to be examined.

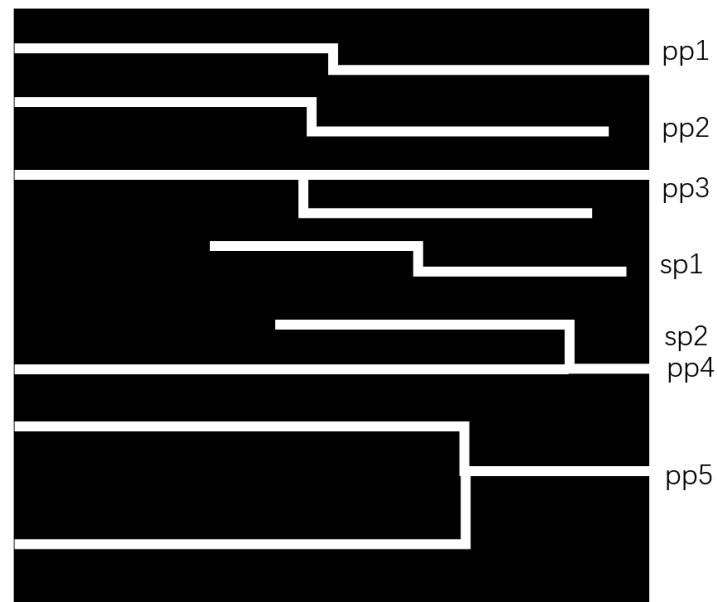


Figure S1. Cases of principal and secondary pathways.

Case 1: The pathway goes through the whole matrix without any ramifications. It can be simply represented by a continuous sequence of adjacent zeros. The fluid channel is near a linear channel (figure S1. pp1).

Case 2: The pathway dies in the matrix, which need to be avoided as it creates dry voids (figure S2. pp2).

Case 3: The pathway branches inside the matrix, continues to exist for a while and then one of the branches dies. This case suggests that a principal pathway is able to create new principal pathways by branching. These new principal pathways do not become a problem even if they die within the matrix since they are not inaccessible voids. However, case 3 highlights the need to discriminate between co-existing principal pathways (figure S1.pp3).

Case 4: The principal pathways can reach secondary pathways at some point. In this case, the secondary pathway is turned into a principal pathway because of the connection between them (figure S1. pp4&sp2).

Case 5: Multiple principal pathways merge together, resulting in the reduction of the number of co-existing principal pathways in a given point of the matrix (figure S1. pp5).

As touched upon before, the first condition for columns compatibility can be expressed as:

Columns compatibility condition 1 (CCC1): two consecutive columns, namely Column 1 and Column 2 (C1 and C2), are compatible if 1) C2 does not cause the end of a non-branched principal pathway and if 2) C2 does not cause the end of a secondary pathway and if 3) at least one principal pathway is preserved from C1 to C2.

The implementation of CCC1 implies the need to distinguish between principal and secondary pathways, by treating them differently according to each specific situation. The most efficient way to discriminate between the two kinds of pathway is to relabel voids from zero to another value.

The distinction of principal pathways and secondary pathways requires a detailed label for the flow pathways inside the unit cell. In this work, this is performed by relabelling the zeros into another values. The relabelling program also allows the implementation of the desired constraints. The detailed implementation process is described below.

Labels are integer numbers equal to or greater than 2, assigned to every elements $x_{i,j} = 0$ in the column, in ascending order. Relabelling starts at the top and ends at the bottom of each column in the matrix. The elements $x_{i,j} = 0$ composing the same pathway should be relabelled with the same label. Moving down the first column from top to bottom, there can be more than one vertical sequence of 0s that can originate a pathway. An example is given below.

$$X = \begin{pmatrix} 0: \\ 0: \\ 0: \\ 1: \\ 0: \\ 0: \\ 1: \\ 1: \end{pmatrix} \quad (S6)$$

These sequences are separated from each other by one or more '1.' The zeros in the first sequence are relabelled as W_1 , the zeros in the second sequence are defined as $W_2 = W_1 + 1$, and the third sequence is $W_3 = W_2 + 1$ and so on. Note that the value of the label does not directly depend on the principal or secondary pathways, so the example matrix can be transferred by labelling.

$$X = \begin{pmatrix} W_1: \\ W_1: \\ W_1: \\ 1: \\ W_2: \\ W_2: \\ 1: \\ 1: \end{pmatrix} = \begin{pmatrix} 2: \\ 2: \\ 2: \\ 1: \\ 3: \\ 3: \\ 1: \\ 1: \end{pmatrix} \quad (S7)$$

The following columns are to be added into X . Labels are assigned for the zeroes by the steps described below.

Step 1: $x_{:,k}$ is defined as the last relabelled column in the matrix X , and $x_{:,k+1}$, (only containing 0 and 1 elements) is the new column to add into the matrix. The 0 elements in $x_{:,k+1}$, which are adjacent to the relabelled elements within $x_{:,k}$, are labelled as the same label in $x_{:,k}$. The purpose of this step is to ensure horizontal continuity of the labels with the pathways that already exist. The label in the new column is related to the previous column. The following example shows Step 1 of the relabelling process.

$$x_{:,k} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \\ 3 \\ 3 \\ 1 \\ 1 \end{pmatrix}; x_{:,k+1} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}; X = \begin{pmatrix} :21 \\ :20 \\ :20 \\ :10 \\ :31 \\ :30 \\ :11 \\ :10 \end{pmatrix} \gg \begin{pmatrix} :21 \\ :22 \\ :22 \\ :10 \\ :31 \\ :33 \\ :11 \\ :10 \end{pmatrix} \quad (S8)$$

Step 2: After step 1, if there are one or more zero elements are adjacent to W_i in the column $x_{:,k+1}$, they should be relabelled as W_i . An example is shown below.

$$X = \begin{pmatrix} :21 \\ :20 \\ :20 \\ :10 \\ :31 \\ :30 \\ :11 \\ :10 \end{pmatrix} \gg \begin{pmatrix} :21 \\ :22 \\ :22 \\ :10 \\ :31 \\ :33 \\ :11 \\ :10 \end{pmatrix} \gg \begin{pmatrix} :21 \\ :22 \\ :22 \\ :12 \\ :31 \\ :33 \\ :11 \\ :10 \end{pmatrix} \quad (S9)$$

Step 3: After the first 2 steps, if there are still zeroes in the new added column $x_{:,k+1}$, these elements are relabelled in the same way as the $x_{:,1}$, starting from the higher label. This label is equal to the highest label in $x_{:,k+1}$ incremented by one. These elements are the start of a new secondary pathway.

$$X = \begin{pmatrix} :21 \\ :20 \\ :20 \\ :10 \\ :31 \\ :30 \\ :11 \\ :10 \end{pmatrix} \gg \begin{pmatrix} :21 \\ :22 \\ :22 \\ :10 \\ :31 \\ :33 \\ :11 \\ :10 \end{pmatrix} \gg \begin{pmatrix} :21 \\ :22 \\ :22 \\ :12 \\ :31 \\ :33 \\ :11 \\ :10 \end{pmatrix} \gg \begin{pmatrix} :21 \\ :22 \\ :22 \\ :12 \\ :31 \\ :33 \\ :11 \\ :14 \end{pmatrix} \quad (S10)$$

By using the re-labelled matrix, it is possible to distinguish between the principal and secondary pathways. This process is established by the history-recorded vectors.

The information of the contacts between the pathways is defined as the pathway history, and

it is stored as vectors in the program. The vector in the program is named *history_W*, which refers to the pathway with the label *W*. By using the history information, two lists are created to store the principal and secondary pathways separately. These two lists are updated each time a new column is added into the matrix. For a specific pathway to be defined as a principal pathway, it needs to connect to $x_{:,1}$ through a contiguous sequence of elements whose value is not equal to 1. In addition to this, the SPs can be transferred to the PPs when they contact in the matrix. An example describing the distinction process is represented in figure S2.

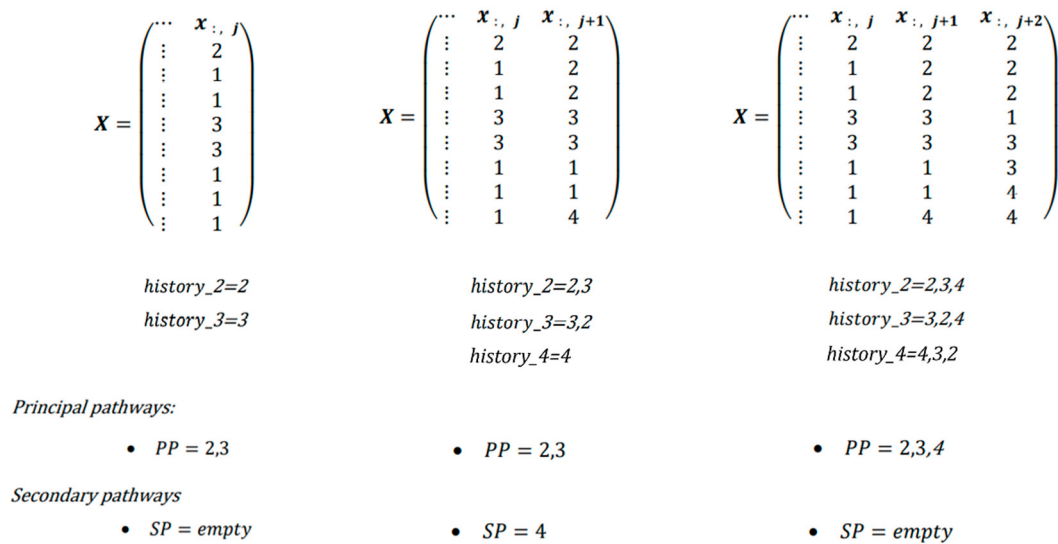


Figure S2. An example to show the distinction between principal and secondary pathways by using $x_{:,j}$. The column $x_{:,j+1}$ and $x_{:,j+2}$ are the two columns added into matrix *X* step by step. The information on the history vectors, principal pathways and secondary pathways is reported below.

The example distinction is given by the column $x_{:,j}$. The history vector of $x_{:,j}$ is 2 and 3, and they both belong to the PPs. As the column $x_{:,j+1}$ is added into the matrix, one secondary pathway 4 is created. The PPs remain the same. When $x_{:,j+2}$ is added, the previous secondary pathway 4 touches the element with a value of 3, and becomes the principal pathway. Another point to mention is that the principal pathway branches in $x_{:,j+2}$, where an element with a value of 1 is created between the elements 2 and 3.

With knowledge of this distinction, it is possible to rework the CCC1 condition into a form that is easy to implement by the program. This condition can be written as:

Columns compatibility condition 2 (CCC2): two consecutive columns, namely column 1 and column 2 (*C1* and *C2*), are compatible if 1) For each pathway *W* present in *C1*, *C2* contains at least one pathway of those present in *history_W*; and if 2) each principal pathway present in *C1* is contained in the *history_{W_p}* of at least one of the principal pathways *W_p* present in *C2*.

The two conditions (CCC1 and CCC2) are equivalent. The difference is that CCC2 can be implemented relatively easily in the program, while CCC1 cannot. The reason for this is that CCC1 contains the wording 'not-branched,' which cannot be immediately translated into code. On the other hand, checking the existence of an element in a list can easily be carried out by the program.

CCC2 represents the control step in the process for generating X . Based on this, the generation process (without considering any constraints) can be improved by applying the control step. After the implementation of the relabelling procedure and CCC2, the resulting algorithm for each single step is a set of nested instructions. A new nested loop is required each time a new column is added into X . If the new column is compatible, a further column is added, but if it is incompatible then it is discarded and the following one from F is added to X .

Section S3 Horizontal and vertical symmetry constraints

Horizontal and vertical symmetry are the phenomena that occur when two different X matrices are replicated to create an ordered array of M matrices, and the final two matrices follow the same pattern. These matching patterns can be converted to each other in one direction by a distance equivalent to the side of X .

This phenomenon happens when two starting matrices X and X' are symmetric to one another with respect to the x or y axis. When the symmetry is with respect to the x axis it is called vertical symmetry; otherwise it is called horizontal symmetry. An example of the vertical symmetry is reported in main text figure 8. The effect of vertical and horizontal symmetry on this work is that since they are equivalent, a double pattern is produced which requires more time for matrix generation and investigation, in order to provide new useful information about the fluid behaviour in different morphologies. On the other hand, this phenomenon can be considered as an opportunity to dramatically decrease the number of matrices. To do this, it is worth developing an algorithm which does not generate the second matrix of the pair in case the first one has been already generated.

Section S3.1 Horizontal symmetry (HS) model

Horizontal symmetry can be defined more accurately by exploiting the columns-oriented description of X . One example of the horizontal symmetry condition is provided in figure S3. Since it is possible to define X as a sequence of known columns of F , it is possible to find a link between the order (whereby columns are combined in sequence to form X) and horizontal symmetry.

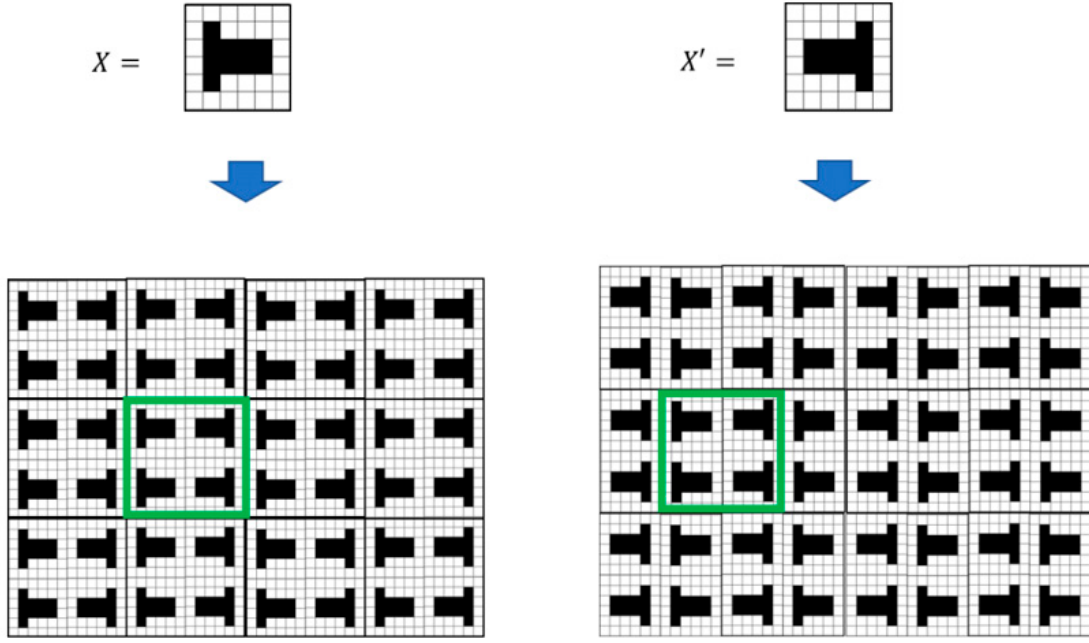


Figure S3. An example of the horizontal symmetry condition. The same patterns of two generated matrices are bordered by the green squares.

X is defined as a square matrix with order c . These c columns are stored in the corresponding matrix F . The matrix X may be represented as follows:

$$X_{c \times c} = (f_{:,g_1} \quad f_{:,g_2} \quad \cdots \quad f_{:,g_{c-1}} \quad f_{:,g_c}) \quad (S11)$$

Here, the $f_{:,g}$ elements are the generic columns in F . To simplify the representation process, the generated matrix X can be shown in terms of the indexes of the matrix F .

$$X = (g_1 \quad g_2 \quad \cdots \quad g_{c-1} \quad g_c) \quad (S12)$$

Therefore, the reverse matrix corresponding to X is given by:

$$X_h = (g_c \quad g_{c-1} \quad \cdots \quad g_2 \quad g_1) \quad (S13)$$

To satisfy the horizontal symmetric condition, X_h must be equal to X . Given that the number of columns in F is equal to $2^c - 1$, the total number of the possible combinations of matrix X is $(2^c - 1)^c$. In this total number $(2^c - 1)^c$, the self-symmetric condition ($X_h = X$) is counted once, and the non self-symmetric condition ($X_h \neq X$) is counted twice. The equation can be written as:

$$(2^c - 1)^c = |A| + 2|B| \quad (S14)$$

Here, A is the self-symmetric condition, and B is the non-self-symmetric condition.

$$N_{\text{symmetry}} = |A| + |B| = \frac{(2^c - 1)^c + |A|}{2} \quad (S15)$$

In equation 16, N_{symmetry} is the number of matrices which remain after the removal of the undesired symmetrical matrices. $|A|$ is defined by the first $\left\lceil \frac{c}{2} \right\rceil$ terms, which means the types of A can be written as $(2^c - 1)^{\left\lceil \frac{c}{2} \right\rceil}$. Hence N_{symmetry} can be written as:

$$N_{\text{symmetry}} = \frac{(2^c - 1)^c + (2^c - 1)^{\left\lceil \frac{c}{2} \right\rceil}}{2} \quad (S16)$$

Here, $\left\lceil \frac{c}{2} \right\rceil$ is the ceiling function applied to the first $\frac{c}{2}$ terms. This number is used to check the algorithm's performance in the results section later on, but it does not provide any insight into how to generate only the desired matrices. An empirical approach has been used in order to determine how to prevent the generation of horizontally symmetric matrices.

The first check is for a 3rd order X matrix. It is possible to generate the N_{symmetry} matrices by imposing the following condition:

For a 3rd order matrix:

Given that $X_h = (g_i \ g_j \ g_k)$ is the sequence representing X , it must be true that:

$$1) k \geq i.$$

Similar conditions have been found also for a 4th order X matrix.

For a 4th order matrix:

Given that $X_h = (g_i \ g_j \ g_k \ g_l)$ is the sequence representing X , it must be true that:

$$1) l \geq i.$$

$$\text{and } 2) \text{ if } j > k, \ l > i.$$

It is thus clear that the complexity of the above conditions increases as the order of X increases. This is the reason why it has not been possible to determine similar conditions for X matrices of higher orders. However, a simplified condition can be suggested based on the previous conditions, as follows:

The index of last column in $X_h \geq$ The index of first column in X

A rigorous proof of this condition is not provided in this work. However, as the last element must be equal to or greater than the first element, a large number of matrices which have a

corresponding reverse matrix are excluded from the generation process. Numerical simulations using row vectors were performed to test the performance of the simplified condition. These simulations were limited by the computational power of the computer available. The accuracy was calculated as the percentage difference between the number of combinations left after the implementation of the simplified condition and the theoretical value $N_{symmetry}$.

Simulations were performed on sets composed by the vectors obtained by the combination of w elements taken from m , where m is the equivalent of the number of columns of F . In these simulations, the vectors of w elements correspond to the X of order w . In this way, each vector generated is the equivalent to X_h .

Figure S4 shows the how results of the percentage difference between the simplified condition and $N_{symmetry}$ varies with the theoretical number of the HS. The dashed lines in the figure represent the trends fitted to the data points. For an actual square matrix X , w orders of 4, 6 and 8 correspond to F orders of 15, 63 and 255, respectively. The data points obtained suggest that the percentage difference between the two methods decreases as m increases. Considering the actual values used for generating the matrices, it is possible that for the sets of X matrices obtained, the percentage difference between the simplified and rigorous approaches in the more optimistic situation tends to zero as $m \rightarrow \infty$. Alternatively, it is possible that the percentage difference asymptotically reaches a finite value lower than 15%, which is the last value obtained from simulation for $w = 8$. This gives support to the argument that the simplified condition is accurate enough to be used during the morphology generation, as it tends to show only a small difference with the theoretical value when the order of X is high.

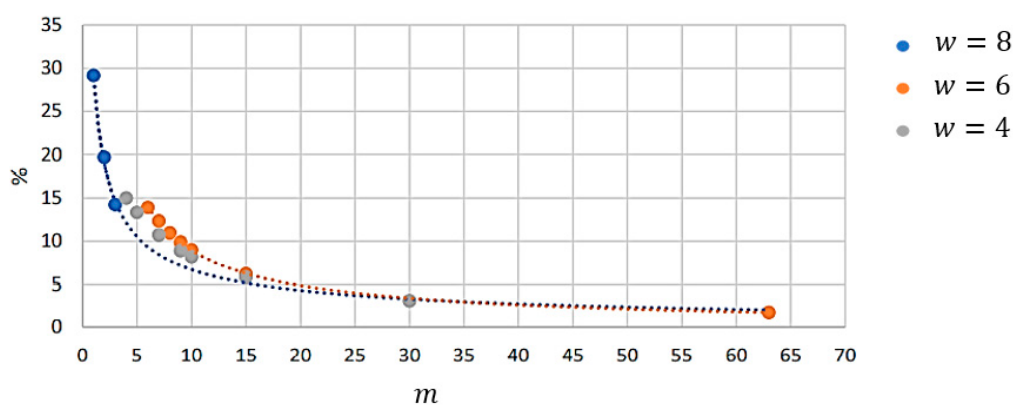


Figure S4. The percentage error between the simplified condition of horizontal symmetry and $N_{symmetry}$. The three tested orders are w equal to 4, 6, and 8.

Section S3.2 Vertical symmetry (VS) model

In contrast to horizontal symmetry, a satisfactory theoretical treatment has not been developed for vertical symmetry due to the difficulty involved in the definition of the generation process. The matrix X is constructed column-by-column. In other words, while

for horizontal symmetry it is possible to reduce the number of combinations X matrices for the morphology generation, this cannot be achieved for vertical symmetry as the rows composing X cannot be controlled. However, since the CFD simulations performed for this study deal with unit cells based on a X matrix of order 3, the number of the unit cells in the set is limited. An alternative is to examine all generated matrices after the generation procedure, and remove the unit cells presenting the vertical symmetry issue. This approach is not straightforward for X matrices of higher orders due to the large number of matrices generated.

Section S4 Porosity model

A minimum porosity value is required to guarantee that there is at least one straight fluid channel in the matrix. For a square matrix X of order 3, three white pixels are needed for a straight fluid channel, which means the minimum porosity should be 0.33. Furthermore, since there must be at least 2 shaded pixels in X , the value of the maximum porosity is 0.77. These maximum and minimum values are the two thresholds of the porosity for the morphology generation process.

The limitation performance of PO is evaluated for different orders of X , numerical simulations of several sets of binary vectors constructed by d_x elements. The tested range of d_x values is from 2 to 27. In the vector of w elements, the value of each element can be 0 or 1, where 0 is a fluid channel and 1 is a portion of the solid phase. The set is composed by all the possible different combinations of vectors. Similar to the construction of X , the minimum and maximum thresholds of the porosity for each vector are 0.33 and 0.77 respectively.

The results show a decreasing trend, in line with the one of X matrices after the imposition of the PP constraint (figure 8 in the main text). The blue circles highlight the data of the vectors whose size is equal to the generated X matrices of order 3, 4, and 5 (the results are summarised in figure. S5). This phenomenon represents that the proportion of morphologies that meet the porosity requirement will be greater for the matrices with higher order. The agreement between the numerical simulations and the X generation results suggests that for X matrices with a high order, porosity thresholds do not noticeably affect the number of generated matrices.

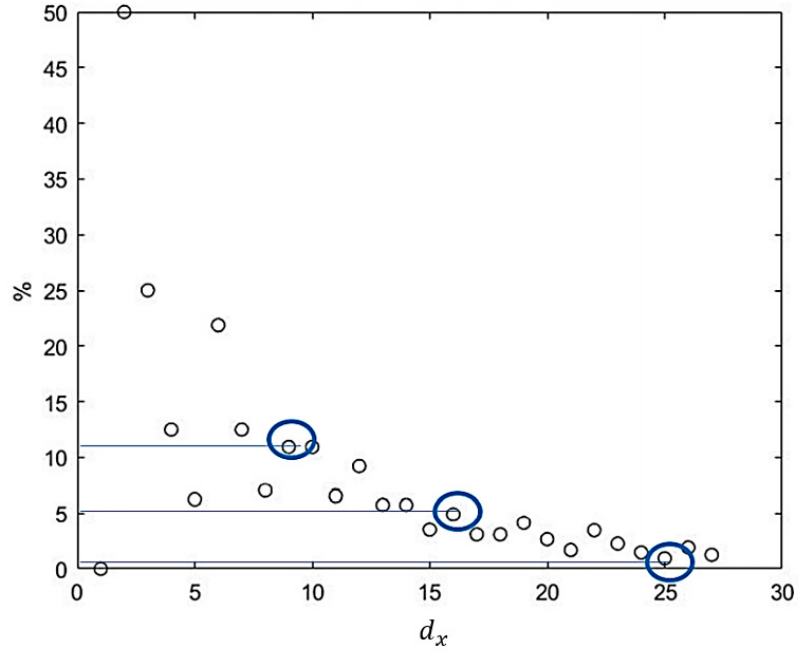


Figure S5. The percentage of vectors whose porosity exceeds the porosity thresholds. The three circled data points correspond to the X matrices of order 3, 4, and 5.

Section S5 Model validation

Model validation is performed before applying models for simulations. As there are little literatures discussed about the dispersion in 2D non-porous PACs, validation was performed by using data available for 3D triply periodic minimal surface (TPMS) [1–3].

TPMS are attractive candidate for 3D porous monoliths [2,3]. Minimal surfaces are defined as structures where every point is a saddle point with equal bending in the two opposite directions of the normal vector to the surface. These structures minimize the specific area of the solid-liquid interface, reducing the pressure drop over the column. The structure evaluated in this work was the single gyroid, which was discovered by the NASA scientist Alan Schoen [4]. Its fluid properties were well established by many researchers [5], and separation efficiency evaluated by Fabian Dolamore [6] on test column made out of a combination of six unit cells.

Figure. S6 shows the morphology evaluated in this work and its performance. The green line is the simulation established by Fabian Dolamore using the lattice Boltzmann method (LBM), whose difference is no more than 5% compared to this simulation.

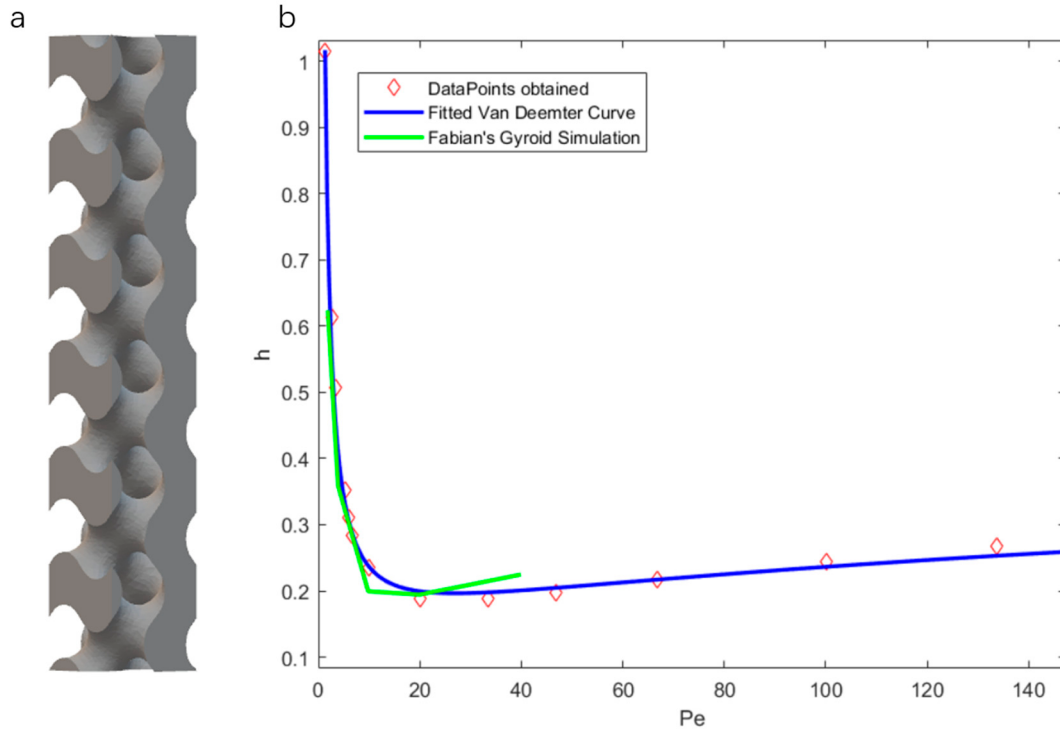


Figure S6. a) The morphology constructed by six repeating single gyroids; b) data points obtained by the simulations and plotted together with best fit of van Deemter curve. The green line is the dispersion curve of the same morphology reported by Fabian Dolamore [6].

The tortuosity and permeability of this model against literature are reported in Table. S1, where the errors are again smaller than 5%. The low errors in both level 2 and level 3 parameters suggest that the model compiled on COMSOL is appropriate to describe the chromatography column in the chromatographic process.

T	1.23 (this work)	1.25 [6]	
k	2.36×10^{-3} (this work)	2.21×10^{-3} [6]	2.29×10^{-3} [5]

Table S1. Flow properties of the single gyroid obtained from the simulation. Literature permeability and tortuosity determined by Dolamore is based on lattice Boltzmann method. And literature permeability is calculated by NS equation [5].

Section S6 Examples of the generated morphologies

Based on the morphology generation program and the models applied, various matrices M can be generated in different orders. An inverse discretization process is applied to the generated matrices to convert them into 2D morphologies. Figure S7 gives an example of this process.

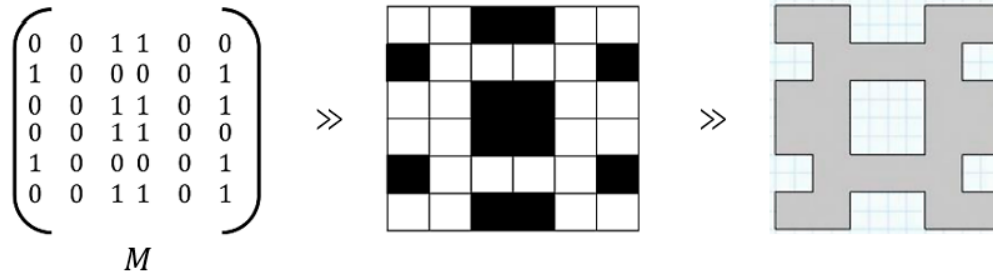


Figure S7. An example of the inverse discretization process. The generated matrix M is converted into a combination of white and black pixels. Then the pixels are transformed to the shaded fluid channel.

Although the impact (in terms of reducing the number of matrices) of the constraints is significant, an enormous number of matrices can still be generated, especially for matrix orders higher than 3. Furthermore, investigating the fluid behaviour within these morphologies (via CFD simulations) is time-consuming, and requires significant computational power. In figure S7, the morphology generation started from the matrix X (with an order equal to 3), which corresponds to the matrix M with order equal to 6. Table S2 shows the number of matrices generated for a range of external porosity values.

External Porosity	Number of generated matrices
0.33	2
0.44	6
0.55	16
0.66	15
0.77	13

Table S2. The number of generated matrices corresponding to different porosity values.

References:

- [1] H. Karcher, The triply periodic minimal surfaces of Alan Schoen and their constant mean curvature companions, *Manuscripta Mathematica* 64 (1989) 291–357. <https://doi.org/10.1007/BF01165824>.
- [2] H. Karcher, K. Polthier, Construction of triply periodic minimal surfaces, *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 354 (1996) 2077–2104.
- [3] W.H. Meeks III, The theory of triply periodic minimal surfaces, *Indiana University Mathematics Journal* (1990) 877–936.
- [4] A. Schoen, Infinite periodic minimal surfaces without self-intersections, n.d.
- [5] Y. Jung, S. Torquato, Fluid permeabilities of triply periodic minimal surfaces, *Physical Review E* 72 (2005) 56319.
- [6] F. Dolamore, In Silico Analysis of Flow and Dispersion in Ordered Porous Media, (2017) 196.