

Computer Programs in Physics

# Fast-QSGS: A GPU accelerated program for structure generation of granular disordered media

Guang Yang, Tong Liu, Xukang Lu, Moran Wang\*

Department of Engineering Mechanics, Tsinghua University, Beijing 100084, China



## ARTICLE INFO

Edited by David W. Walker

Program summary

Program Title: Fast-QSGS

CPC Library link to program files:

Code Ocean capsule:

Licensing provisions: MIT license

Programming language: Python

Nature of Problem: Fast generation/reconstruction of disordered granular media.

Solution method: The QSGS method is vectorized and implemented in Python. Via CuPy, the current implementation of QSGS can be easily accelerated by modern GPUs.

## Keywords:

Disordered media generation

QSGS

Vectorization

GPU computing

## ABSTRACT

We present Fast-QSGS, a GPU-accelerated program for granular disordered media generation. Based on vectorization, Fast-QSGS is accelerated by modern GPU thanks to the NumPy-compatible API provided by CuPy. We also introduce a variable growth probability function and seed spacing control to improve the speed and accuracy of the original QSGS method. Computational performance benchmarks are conducted on both consumer-grade and professional-grade GPUs. Generation of disordered media of size  $400^3$  can be completed in 30 s on A100 and 110 s on RTX4060, achieving a speedup of over 400 compared with the serial version. Physical benchmarks on the reconstruction of Fontainebleau sandstone and hydrated cement are conducted. Our results demonstrate that the permeability of the reconstructed Fontainebleau sandstone falls within the range of experimental values. Additionally, the average relative error of the volume fraction of the unhydrated cement and capillary porosity of hydrated cement is 1.9 % and 3.4 % compared with Powers' law, respectively.

## 1. Introduction

Properties and behaviors of granular disordered media are of great interest in modern physics and related engineering applications. For instance, flow, especially complex flow such as multiphase [1–3], non-Newtonian [4,5], and charged flows [6–8] in disordered porous media is the cutting-edge problem that significantly impacts numerous fields, including petroleum [9], carbon dioxide sequestration [10,11], fuel cells [8,12], and more. Additionally, predicting mechanical [13] and thermal properties [14–16] of multiphase composites with disordered micro-structures is one of the most challenging and important issues in material science. In optical science and engineering, the wave branching of light in disordered media is a highly discussed topic [17–19].

Most traditional macroscopic descriptions and models are based on homogenization assumption which ignores the microscale heterogeneity of disordered media, such as Darcy's equation in porous flow [20], Voigt

model, Reuss model in composite materials [21]. Inevitably, empirical parameters are required in such models and equations. On one hand, macroscopic models exhibit limited accuracy and suitability for complex problems [13,14,22]. On the other hand, due to poor visibility and low reproducibility, these methods are also challenging to reveal the underlying mechanism [20].

With the rapid development of imaging technology, high-performance computing (HPC), and other experimental methods, microscale methods have become a powerful tool for studying disordered media [20,23–25]. Although microstructures of disordered media can be obtained via imaging reconstruction such as Micro-/Nano-CT and SEM/TEM [23,26,27], the imaging methods are not only expensive but also exhibit limited field of view (FOV) and resolutions [24]. Particularly, in recent years, the rapid advancement of high-performance GPU computing [3,28,29] and experimental technologies such as “Lab on a Chip” [30,31] have presented new challenges for swiftly obtaining large, high-quality disordered media structures.

\* Corresponding author.

E-mail address: [mrwang@mail.tsinghua.edu.cn](mailto:mrwang@mail.tsinghua.edu.cn) (M. Wang).

**Algorithm 1**

Serial Growth Process.

---

```

1: while  $\phi_a < \phi_s$ 
2:   for each  $x,y$  in  $domain(N_x, N_y)$ 
3:     if  $struct(x,y)$  is solid
4:       for each  $i$  in directions
5:         if  $(struct(x + ex_i, y + ey_i)$  is fluid) and  $(rand\_num < G_i)$ 
6:            $struct(x + ex_i, y + ey_i) = solid$ 
7:         end if
8:       end for
9:     end if
10:   end for
11:  $\phi_a = \text{sum}(struct)$ 
12: end while

```

---

Alternatively, various generation methods are widely adopted to produce micro-structures of disordered media, including the geological process-based method [32], Multiple-Point Statistics (MPS) [33], Quartet Structure Generation Set (QSGS) [14,15], Simulating Annealing (SA) [34] and Markov Chain Monte Carlo [35]. However, the generation of large, 3D in particular, disordered media can be time-consuming and takes several hours, or even days, to complete [36–38]. For instance, generating a typical  $400^3$  disorder media takes more than 17 h for QSGS and even several days for SA [38]. To address the inherent variability in the generation process, it is typically necessary to produce multiple samples using identical parameters. This, in turn, further extends the time required.

Recently, some researchers accelerated SA [38] and MPS [36,37,39] by parallel computing and GPU computing and achieved a speedup of 7–15. However, these methods require complex C++/CUDA programming and take several hours to generate 3D structures [36–39]. Feng and coworkers [40,41] also developed a deep-learning-based framework for porous media reconstruction. Though the inference process, (i.e., generating a structure) only requires seconds to complete, the training process still requires several hours on a GPU [41]. Additionally, there aren't efficient open-source implementations available. Hence, the fast generation of disordered media remains an open question.

On the other hand, vectorization presents significant potential in parallelizing existing algorithms [42–46]. By converting the element-wise operations into array operations, vectorization enables concurrent processing of elements. This not only leads to programs that are elegant and efficient due to the utilization of array-based operations but also leverages built-in SIMD support in high-level languages like MATLAB, as well as libraries such as NumPy [47], SciPy [48], and TensorFlow [49]. More recently, with the advent of GPU array libraries like CuPy [50] in Python and gpuArray in MATLAB [51], vectorized programs can be seamlessly ported to GPUs, further amplifying the speed of array operations. Consequently, vectorization has found applications in parallelizing various algorithms, including Molecular Dynamics (MD) [45], the Lattice Boltzmann method (LBM) [52,53], and Particle in Cell (PIC) [42].

In this work, we introduce Fast-QSGS, an open-source QSGS implementation based on vectorization in NumPy [47] and CuPy [50]. Originally proposed in 2007, QSGS [14] and its derived Random Generation Growth (RGG) methods [15] have been extensively adopted and highly

acknowledged by fellow researchers. By comprehensively understanding the primary morphology and characteristics of the disordered media, QSGS has been proven to be not only simple and efficient but also robust [13,15,22,25,54–56]. Regarding this study, we further develop QSGS both in computational efficiency and accuracy. Thanks to the NumPy-compatible interface provided by CuPy [50], Fast-QSGS takes advantage of the high computation power and memory bandwidth of modern GPUs. Depending on the specific GPU hardware, the generation of a  $400^3$  or  $4000^2$  disorder media could be achieved within a remarkably brief span of 1–3 min. Meanwhile, the permeability values of the reconstructed Fontainebleau sandstones fall within the experimental data. Additionally, a benchmark on multiphase unhydrated cement is also conducted. Compared with Powers's law [57], our results indicate the average relative error for the volume fraction of unhydrated cement is 1.9%, while for the capillary porosity of hydrated cement, it is 3.4%. Coded in Python, Fast-QSGS could also be easily modified and customized by users and seamlessly integrated with other popular packages within the community, such as OpenPNM [58] and PoreSpy [59].

The rest of this paper is organized as follows. In Section 2, we briefly introduce the vectorized QSGS method and acceleration techniques. Benchmarks on both the computational performance and physical properties of the Fast-QSGS are provided in Section 3. In Section 4, we conclude this paper.

**2. Model and method****2.1. Basic QSGS and its vectorization**

The original QSGS [14] could be roughly divided into the following steps. Firstly, grain seeds are randomly located according to the specified seed distribution  $S_d$  on an orthogonal grid system. Subsequently, in the growth step, each seed element randomly expands into neighboring cells in all directions, based on the provided constant growth probability  $G_i$ , where  $i$  denotes the direction. The newly generated elements are also treated as seeds, enabling them to expand to neighboring cells as well. This growth step continues until the desired volume fraction of the granular phase,  $\phi$ , is attained. If multiple solid phases are desired, the solid phase  $n$  can grow depending on the phase interaction probability  $I_i^{m,n}$  to achieve target volume fraction  $\phi^n$ , where  $m$  denotes the initial

**Algorithm 2**

Vectorized Growth Process.

---

```

1: while  $\phi_a < \phi_s$ 
2:    $solid\_mask = struct > 0$ 
3:   for each  $i$  in directions
4:      $growth = rand(N_x, N_y) < G_i$ 
5:      $growth = growth * solid\_mask$ 
6:      $struct = struct + roll(roll(growth, ex_i, axis=0), ey_i, axis=1)$ 
7:      $struct = struct > 0$ 
8:   end for
9:    $\phi_a = \text{sum}(struct)$ 
10: end while

```

---

**Algorithm 3**Parallel Seed Generation with Spacing by **convolve**.

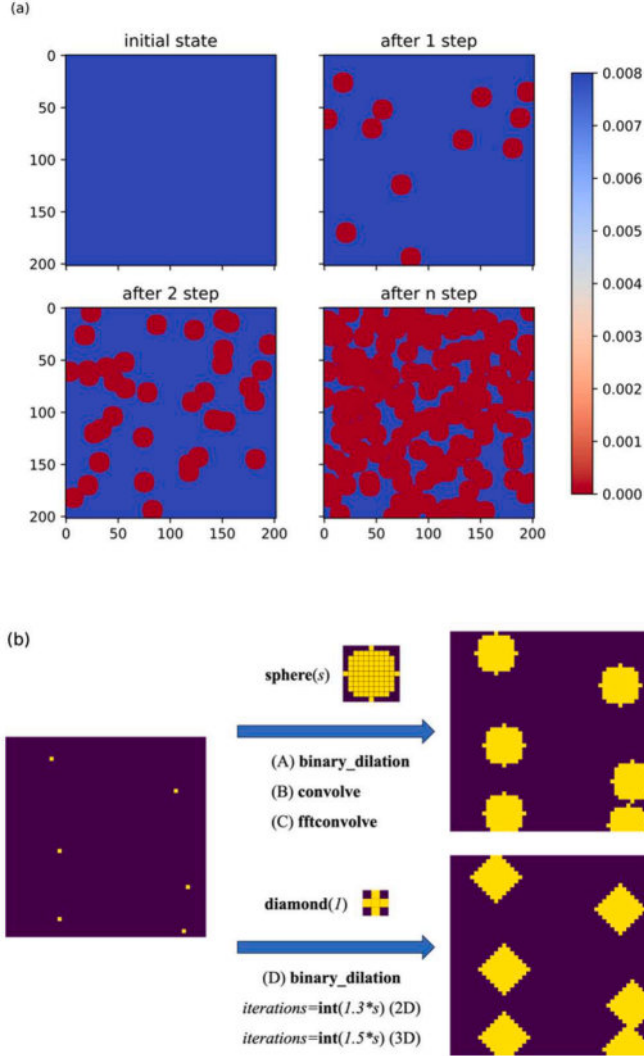
---

```

1:  $sd\_array = S_d * \mathbf{ones}(Nx, Ny)$ 
2:  $struct = \mathbf{zeros}(Nx, Ny)$ 
3: while  $\phi_a < S_d$ 
4:    $struct += (\mathbf{rand}(Nx, Ny) < \frac{S_d}{20})$ 
5:    $b = \mathbf{sphere}(s)$ 
6:    $sd\_array = sd\_array - (\mathbf{convolve}(struct, b, \mathbf{mode}=\mathbf{same}) > 0.5) * S_d$ 
7:    $\phi_a = \mathbf{sum}(struct)$ 
8: end while

```

---



**Fig 1.** (a) Illustration of the variation of the seed probability function during the seed generation process. Each time a seed is generated, the seed probability function is set to 0 within the distance  $s$  of the seed. (b) Illustration of different seed spacing methods: (A) sphere dilation; (B) direct convolve; (C) FFT convolve; (D) diamond dilation. For the first three methods, the dilation is conducted with a large kernel, while diamond dilation is conducted with a small kernel but multiple iterations. The volume (area) of the diamond is equal to that of the sphere.

phase. With these the four parameters  $(S_d, G_i, I_i^{m,n}, \phi^m)$ , a multiphase granular disorder media can be generated. Therefore, it is referred to as the Quartet Structure Generation Set (QSGS).

For the growth processes of different solid phases, apart from the initial differences in the seed distribution, the other steps are identical. Moreover, since the seed generation is conducted only once, whereas the

growth process is repeated until the desired volume fraction, computationally, the growth process is the most time-consuming function. Therefore, we begin our discuss on the vectorization of the grain growth process.

Without loss of generality, we consider generation a single-phase 2D granular disorder media, the solid phase is denoted by 1 while the void phase is denoted by 0. The seeds have been generated by the previous steps. As Algorithm 1 shows, growth steps in original QSGS are conducted in a serial cell-by-cell manner. However, such growth process can be converted into array operations as Algorithm 2 shows. At each growth step when the actual volume fraction  $\phi_a$  is smaller than the desired value  $\phi_s$ , initially, a masking array *solid\_mask* denoting the present seeds is produced. Then, for each direction, a random array is generated and compared with the growth probability which determines all the possible growth. However, such result array contains locations where no seeds exist. Therefore, we employ the masking array to obtain the actual growth position. Then, the growth of the  $i$ th direction is conducted via rolling the array. Finally, the result is normalized since the seeds inside grains may also grow in the vectorized version.

It should be noted that the vectorized version is not exactly identical to the serial version. For the serial version, every newly grown cell is treated as seed immediately, while the newly grown cells are not updated to the *solid\_mask* until the growth of all directions is complete. Since the growth process is highly stochastic, this difference can be neglected.

## 2.2. Improvement and optimization of QSGS

Due to the stochastic growth nature of the QSGS method, it does not account for interactions between grains [14,15]. Consequently, with randomly distributed seeds, the generated structures may exhibit non-physical occurrences such as grain overlap or levitation, resulting in larger pore size and grain size. This is particularly noticeable at the lower grain volume fraction (larger porosity), where non-uniform seeding results in non-uniform distribution of grains.

In this work, we introduce a parameter spacing of seeds  $s$  which denotes the minimum distance of the seeds to control uniformity. To achieve such seed distribution, one possible solution is to generate seeds one by one and determine whether the generating seed meets the spacing requirement. Alternatively, a parallelized version could be employed to improve the performance of the serial version, as Algorithm 3 and Fig. 1(a) show. The main idea is to generate a sphere (circle in 2D) with a radius of  $s$  at the location where a seed is generated. The probability of seed generation within the sphere is set to 0. Generation of spheres can be easily conducted via the **binary\_dilation** or **convolve** function. In this way, generating multiple seeds simultaneously is feasible if we employ a lower probability ( $S_d/20$  for simplicity) to generate seeds to reduce probability of seed overlapping (distance of two seeds smaller than  $s$ ) in a single instance. This procedure could be repeated until the desired number of seeds is achieved. Although this implementation cannot guarantee no seed overlapping in a single generation instance, such overlapping is negligible in practice. On the one hand,  $S_d/20$  is usually very small resulting small overlapping probability. On the other hand, the overlapping of few grains will not affect the

**Table 1**  
Major Specifications of Test GPUs [60].

GPU Model	Architecture	Peak FP64	Peak FP32	Memory Bandwidth	Memory Size	Launch Year
Nvidia RTX 3060	Ampere	199.0 GFLOPs	12.74 TFLOPs	360 GB/s	12 GB	2021
Nvidia RTX 3090	Ampere	556.0 GFLOPs	35.58 TFLOPs	936.2 GB/s	12 GB	2020
Nvidia RTX 4060	Ada Lovelace	236.2 GFLOPs	15.11 TFLOPs	272 GB/s	8 GB	2023
Nvidia Tesla V100	Volta	7.0 TFLOPs	14.0 TFLOPs	900 GB/s	16 GB	2017
Nvidia A100	Ampere	9.7 TFLOPs	19.5 TFLOPs	1555 GB/s	40 GB	2020
Hygon DCU Z100	GCN 5.1	9.5 TFLOPs	9.5 TFLOPs	800 GB/s	32 GB	2022

overall statistical characteristics of the generated structure. Additionally, we suggest further reducing the parallel generation probability if strong seed overlapping is observed.

However, the **binary\_dilation** or **convolve** may also be less efficient, especially in 3D scenarios with larger convolution kernels. While methods like FFT convolution can speed up the operation, they come with a significant memory overhead, particularly for **fftconvolve** in CuPy [50]. To reduce the cost of frequent memory allocation and deallocation, CuPy aggressively occupies resources in the memory pool without releasing them. This can severely limit the maximum size of structures that can be generated on a single GPU via FFT convolution.

One approximate approach is to use regular diamonds (strictly speaking, square in 2D and octahedra in 3D) instead of spheres as Fig. 1 (b) shows, since diamonds can be obtained through the repeated dilation processes based on a simple-connectivity kernel, which is a much lower-cost approach. Since the main concern is to ensure the uniformity of grain seeds, parameters of the dilation (mainly, repeated iteration times) need to be adjusted appropriately to ensure that the volume of the regular diamond matches that of the spheres.

Moreover, it is worth mentioning that the overall growth rate of grains in each step is highly correlated with their total surface area (i.e., perimeter in 2D), as only interface seeds could expand into neighboring void cells. Therefore, the growth rate is slow at the beginning due to a limited number of seeds. However, as the volume fraction approaches the desired value, typically the surface area increases, resulting in overgrowth and a lower accuracy of volume fraction. In this work, we propose a variable growth probability function  $G_i(\phi_a)$  rather than the constant growth probability in original QSGS [14]. For simplicity, we adopt a linear function,

$$G_i(\phi) = G_{i,ref} \frac{\phi_s - 0.95\phi_a}{0.05\phi_s}. \quad (1)$$

Via this variable growth probability function, the initial growth probability is 20 times of the reference value  $G_{i,ref}$ , while the growth probability decreases linearly to the reference value as the  $\phi_a$  approaches  $\phi_s$ , allowing fine control of the volume fraction.

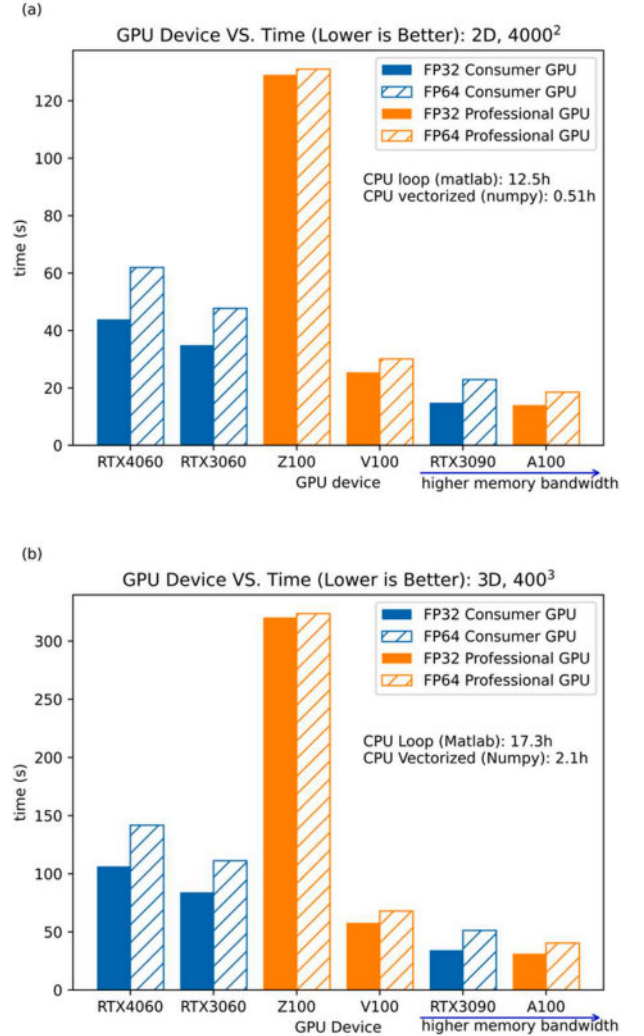
Finally, QSGS-generated structures often exhibit rough grain surfaces [14,15]. There tend to be micro-branching structures on the grain surfaces, which are caused by the diagonal connectivity during the growth procedure. Such micro-branches provide an accurate representation of the morphology of materials such as soil and concrete [15]. However, for materials like sandstone, which are generally considered to have relatively smooth surfaces, we recommend applying a uniform filter at the end of the generation process to smooth the grains.

### 3. Results and discussion

#### 3.1. Benchmark on computing performance

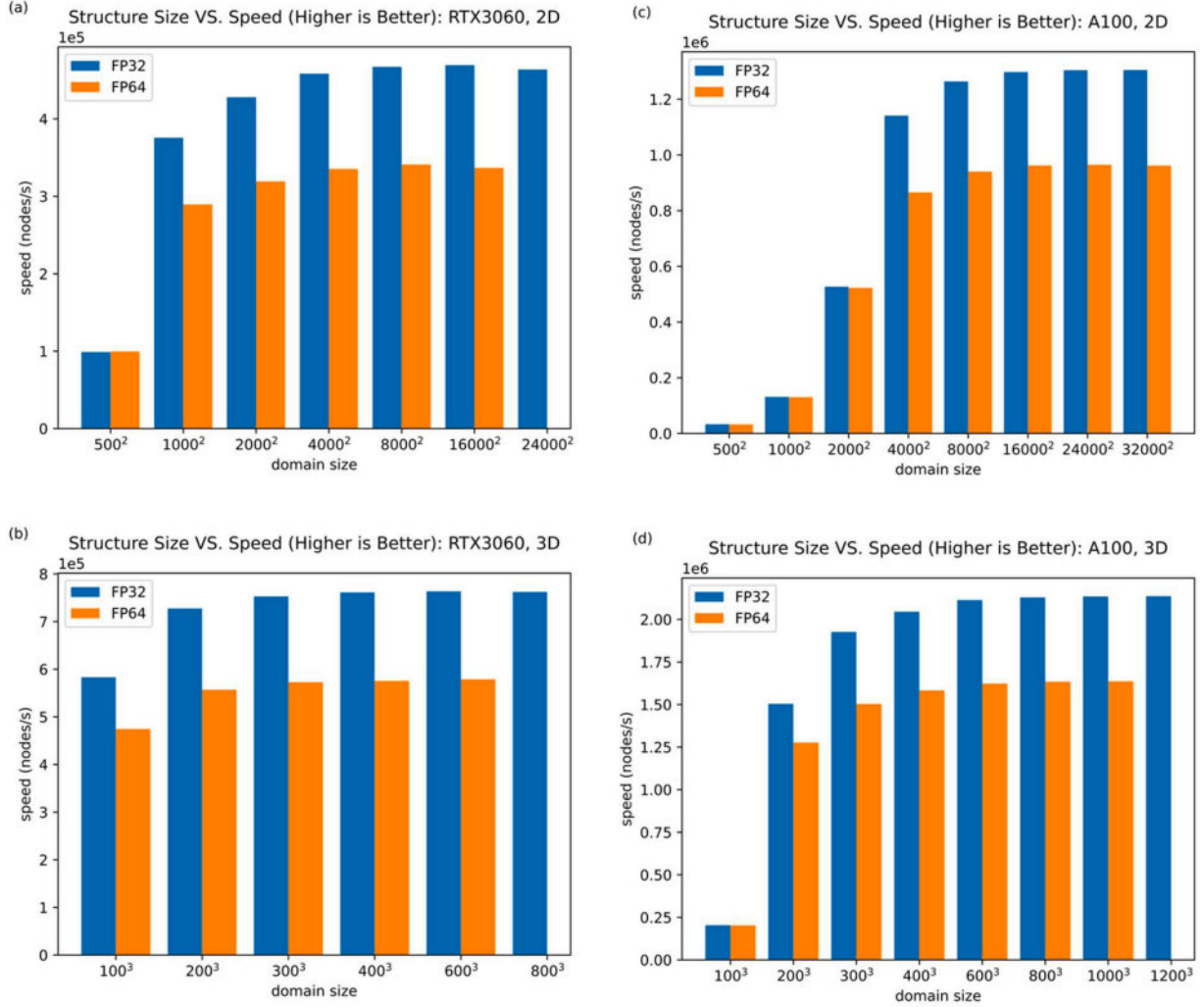
##### 3.1.1. Testbed

To conduct a comprehensive benchmark on the performance of the current implementation, multiple GPUs of different vendors and architectures are adopted as our testbeds. This selection takes into account a wide range of user scenarios, encompassing not only the high-end GPUs, for instance, V100 and A100, typically employed in supercomputers and



**Fig. 2.** Overall performance (time, lower is better) of generating disordered media with typical size on different GPU devices. Each point is the average of 10 independent runs with the same set of QSGS parameters. The parameters in QSGS are set as follows:  $S_d = 2 \times 10^{-4}$ ,  $G_{i,ref} = 8 \times 10^{-4}$ . Diamond-type dilation is employed for all cases with  $s = 15$ . (a) 2D,  $4000^2$ ,  $\phi = 0.5$ ; (b) 3D,  $400^3$ ,  $\phi = 0.2$ .

data centers, but also consumer-grade gaming GPUs (RTX 3060 and RTX 4060) commonly found in everyday PCs. Additionally, we include a Hygon DCU (Deep Computing Unit) Z100 which is based on the AMD HIP platform and can be considered as a GPGPU as well. All of our GPUs are PCIe versions and operated with ECC off if supported. The detailed specifications of GPU testbeds are concluded in Table 1. As for CPU baseline, we employ a typical server-grade CPU, Intel Xeon Silver 4214R with a 2.4 GHz base frequency and 3.5 GHz maximum turbo frequency.



**Fig. 3.** Performance (speed =  $\frac{\text{domain size}}{\text{total time}}$ , higher is better) of generating disordered media of various sizes on RTX3060 and A100. (a) RTX3060, 2D; (b) RTX3060, 3D; (c) A100, 2D; (d) A100, 3D. Each point is the average of 10 independent runs with the same set of QSGS parameters. The parameters in QSGS are set as follows:  $S_d = 2 \times 10^{-4}$ ,  $G_{i,ref} = 8 \times 10^{-4}$ ,  $\phi = 0.5$  (2D) and  $0.2$  (3D). Diamond-type dilation is employed for all cases with  $s = 15$ .

### 3.1.2. Overall performance

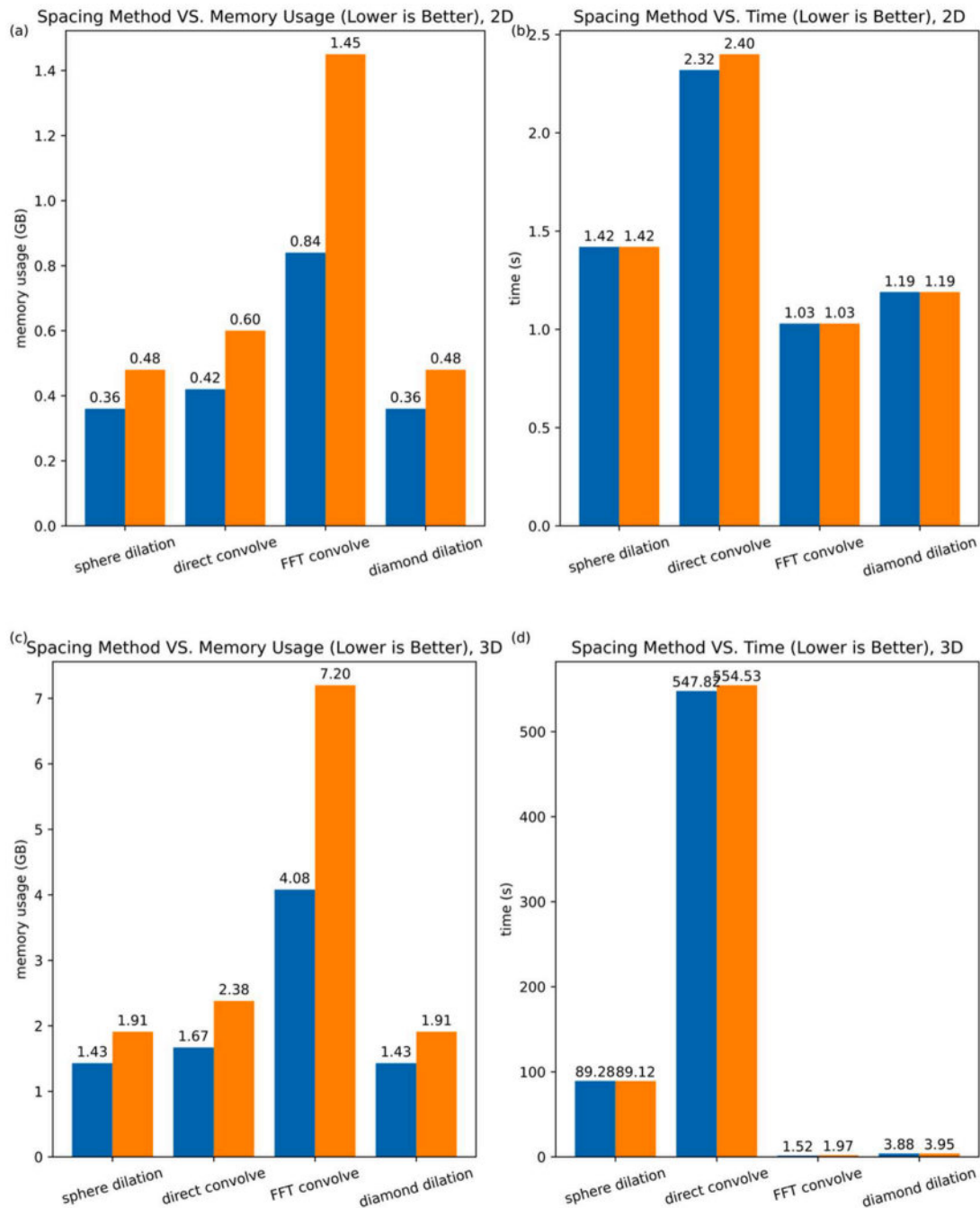
Firstly, we present the overall performance of generating typical disordered porous media on various devices. Porous media of size  $4000^2$  and  $400^3$  are generated, respectively for 2D and 3D, to represent typical scenarios. The parameters in QSGS are set as follows:  $S_d = 2 \times 10^{-4}$ ,  $G_{i,ref} = 8 \times 10^{-4}$ ,  $\phi = 0.5$  (2D) and  $0.2$  (3D) respectively, considering the connectivity distinct between 2D and 3D. Diamond-type dilation is employed for all cases with  $s = 15$ . The results are concluded in Fig. 2. Here, two baselines on CPU are provided both use FP64, the first is a loop-based version implemented in MATLAB which represents the original QSGS in [14] without seed spacing control and smoothing procedure, and the second is a vectorized version in NumPy. In other words, the current NumPy version involves a few additional procedures. As explained earlier, thanks to the compatible interface provided by CuPy, Fast-QSGS seamlessly switches between CPU and GPU with minimal modifications required. Comparing the two baselines, our results demonstrate that vectorization, leveraging the SIMD feature offered by modern CPUs, achieved speedups of 24.5 and 8.2 for 2D and 3D respectively.

Regarding the GPU acceleration, it can be observed that all GPUs exhibit significant acceleration, particularly Nvidia GPUs. All two-dimensional results are generated in approximately 60 s, while the three-dimensional tasks take a maximum of 150 s, achieving minimum

speedup (RTX4060, FP64) of 42.5 (2D) and 53.8 (3D) compared with the vectorized CPU version. The maximum speedup achieved on A100 with FP32 is 129.4 and 242.3 for 2D and 3D, respectively. In contrast, the performance of Z100 is comparatively low, about 14.1 (2D) and 23.4 (3D) times speedup, which may be attributed to the early-stage support of CuPy support on HIP and GPU architecture differences [50].

It can also be observed that GPUs with large memory bandwidth exhibit better performance except for the Hygon DCU Z100, indicating that Fast-QSGS may be primarily memory bandwidth bound. Meanwhile, comparing the consumer-grade GPUs and professional GPUs, the performance decreases due to FP64 being smaller on professional GPUs, thanks to their higher FP64 computational power. For the Hygon DCU Z100, the performance difference between FP32 and FP64 can be neglected as the computational power for FP64 matches that of FP32 at a 1:1 ratio. It should be noted that Fast-QSGS is not a typical floating-point intensive application, where only smoothing and random number generation are conducted via the floating-point operations. Conversely, most operations are conducted on an *uint8* array representing the binary image. Therefore, the floating-point precision does not significantly impact the overall performance.

To further demonstrate the performance of Fast-QSGS in generating disorder media of various sizes, we conducted tests using RTX3060 and A100 as representatives of consumer-grade and professional-grade



**Fig. 4.** Comparison of different seed spacing methods: memory usage (lower the better) and time (lower the better). (a) memory usage, 2D; (b) time, 2D; (c) memory usage, 3D; (d) time, 3D. The memory usage denotes the total size of the default memory pool in CuPy.

GPUs, respectively. The results are concluded in Fig. 3. Our findings suggest that as the domain size expands, the computational speed, measured by the domain size divided by the total time, initially increases before stabilizing around the peak value. For professional-grade GPU A100, due to larger computational power and memory bandwidth, such phenomenon is more pronounced. In contrast, for consumer-grade GPUs, even a relatively modest domain size can well saturate the GPU's capability. Notably, the typical domain sizes we selected ( $4000^2$  for 2D,  $400^3$  for 3D) attain over 95 % of the peak performance on both RTX3060 and A100, effectively showcasing the optimal performance of Fast-QSGS on each respective GPU.

Additionally, the current implementation is capable of generating structures with a maximum size of  $24,000^2$  within 9.1 min for 2D and

$800^3$  within 11.2 min for 3D on RTX3060. On A100, it can handle even larger sizes, with a maximum of  $32,000^2$  within 13.1 min for 2D and  $1200^3$  within 13.5 min for 3D. This level of capacity is adequate for direct numerical simulations, which often require substantial computational resources.

### 3.1.3. Seed generation

In this section, we discuss the performance issues of different methods in the seed generation steps. As Fig. 1(b) shows we compare four different methods as Fig. 1(b) shows: (A) sphere dilation, (B) direct convolve, (C) FFT convolve, and (D) diamond dilation. We generate a porous structure of  $4000^2$  and  $400^3$  on A100. The results of elapsed time and memory usage are displayed in Fig. 4. The memory usage is

measured via `total_bytes` of the `default_memory_pool` in CuPy because of the memory pool mechanism [50].

It can be observed that for the generation of 2D structures, all methods complete the seed generation within 3 s. However, in the case of 3D scenarios, it is evident that the direct convolve and sphere dilation approaches exhibit poorer performance, taking over 90 s and 540 s, respectively. As mentioned in the previous section, on an A100 GPU, the end-to-end time using diamond dilation is approximately 50 s, which is unacceptable in terms of performance. Meanwhile, the FFT convolve and diamond dilation demonstrates faster completion, generating the seed within 4 s. Nevertheless, it should be noted that FFT convolve takes 2.38, 3.77 times memory consumption compared with the diamond dilation method. The substantial memory requirements will significantly restrict the generation size on GPUs, given that GPU memory is typically constrained, particularly on consumer-grade GPUs. To illustrate, when utilizing FFT convolve with FP64 on our RTX4060, the generation size is capped at  $400^3$ . Even on our A100 with 40GB of memory, the limit is set at  $700^3$ . Whereas the diamond dilation has the lowest memory consumption and second highest speed, within 4 s. Therefore, we recommend adopting diamond dilation for generating large structures.

### 3.2. Benchmark on physical properties

QSGS has been widely applied in many applications, demonstrating its accuracy and robustness [13,15,22,25,30,54–56]. To validate the parallel implementation presented in this study, we present two benchmarks assessing the physical property of the disordered media generated by Fast-QSGS. The first one is the permeability of the Fontainebleau sandstone while the second one is the generation of hydrated cement.

#### 3.2.1. Fontainebleau sandstone

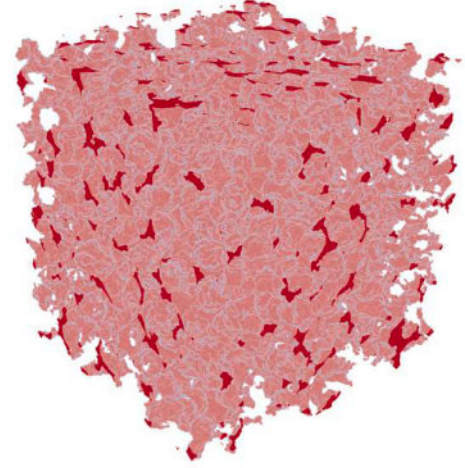
Fontainebleau sandstone is a typical granular, disordered porous media, consisting entirely of well-sorted quartz grains with an average diameter of  $200 - 250 \mu\text{m}$  [61–66]. There is an absence of clay or intergranular porosity. The porosity of Fontainebleau sandstone ranges from 0.03 to 0.3 without noticeable grain size modification. These characteristics make it a commonly adopted reference porous medium in geological applications [61–66]. In this work, we will employ the Fast-QSGS to generate porous structures similar to Fontainebleau sandstone.

The first thing to determine is the seed distribution  $S_d$ , in other words, grain size  $d$ . Although the reported grain size for Fontainebleau sandstone in experimental literature is often cited as  $250 \mu\text{m}$ , this value represents the equivalent diameter denoted by  $d = \frac{4A}{P}$ , where  $A$  and  $P$  are the area and perimeter of grain measured on a thin section of rock samples [65]. However, considering the 3D nature of grains, local thickness, which measures the diameter of the maximum sphere that can fit within the grain, is also widely employed. According to Latief [65], the local thickness of the Fontainebleau sandstone is around  $180 \mu\text{m}$  while the equivalent diameter measures  $250 \mu\text{m}$ .

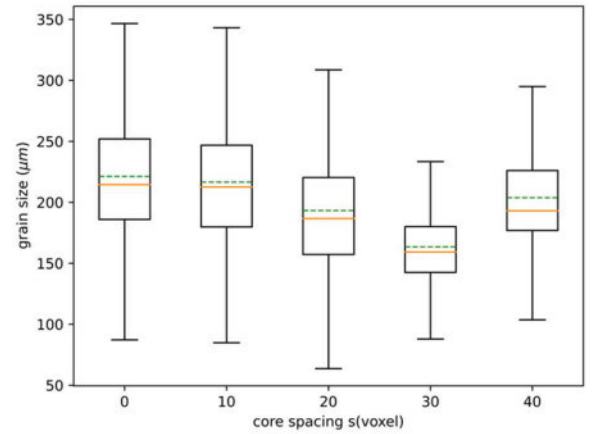
Additionally, due to the tendency of QSGS-generated particles to approach a more spherical shape, whereas real quartz grains are more irregular and often exhibit overgrowth [66], this leads to a lower permeability for grains of the same size. Therefore, we have opted to generate QSGS sandstone with a grain size of  $160 \mu\text{m}$ . The seed distribution could be computed via  $S_d = \frac{6\phi_s \Delta^3}{\pi d^3}$ , where  $\phi_s$  is the solid fraction,  $\Delta$  is the resolution. Considering the requirement of REV and critical resolution of sandstone, we select  $\Delta = 4 \frac{\mu\text{m}}{\text{voxel}}$  and a domain size of  $500^3$ .

Firstly, we present the effect of the seed spacing  $s$  on the grain size distribution and pore size distribution. Five sets of porous structures, each consisting of 10 structures with the same set of parameters are generated. The porosity is 15 % and the initial grow probability  $G_{i,ref} = 0.0008$ , while  $s$  ranges from 0 to 40 (i.e., 0 to  $160 \mu\text{m}$ ) via the spherical

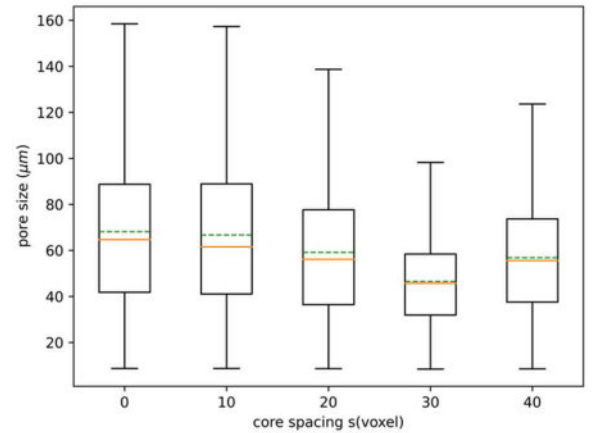
(a)



(b)



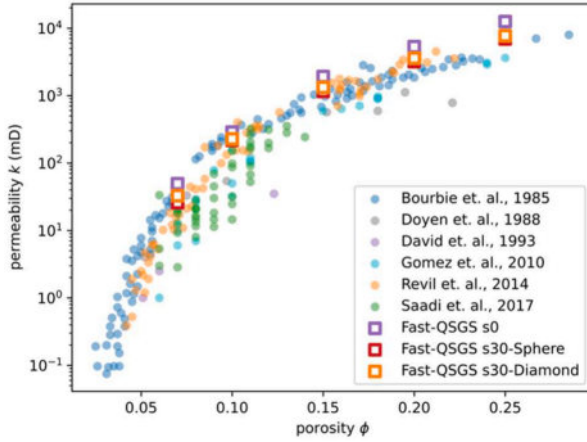
(c)



**Fig. 5.** (a) Illustration of the reconstructed Fontainebleau sandstone by Fast-QSGS.  $\phi = 0.15$ , size  $500^3$  and  $s = 30$ . (b) Box plot of the grain size distribution with different seed spacing  $s$ . (The orange line denotes the median value while the green dashed line denotes the mean value, hereafter). (c) Box plot of the pore size distribution with different seed spacing  $s$ .

spacing method. Subsequently, the grain size distribution and pore size distribution are determined via the local thickness filter provided in PoreSpy. The results are concluded in Fig. 5.

As the seed spacing increases, the grain size and pore size of the generated structures exhibit a trend of initially increasing and then decreasing. At  $s = 30$ , both the grain size and pore size are the smallest,



**Fig. 6.** Comparison of the permeability of the reconstructed Fontainebleau sandstone by Fast-QSGS and experimental values [61,64, 66–69]. The reconstructed samples with  $s = 30$  agree well with the experimental results. The deviation between diamond spacing and sphere spacing is neglectable.

with the lowest deviation as well. This suggests that increasing seed spacing  $s$  leads to a more uniformly spread seed distribution, reducing unwanted grain overlapping. However, if the seed spacing becomes too large, the seed distribution decreases resulting in larger grain size. Additionally, only when  $s = 30$ , the mean grain size approaches the desired  $160 \mu\text{m}$ . Therefore, we set  $s = 30$  as the optimal seed spacing for the Fontainebleau sandstone.

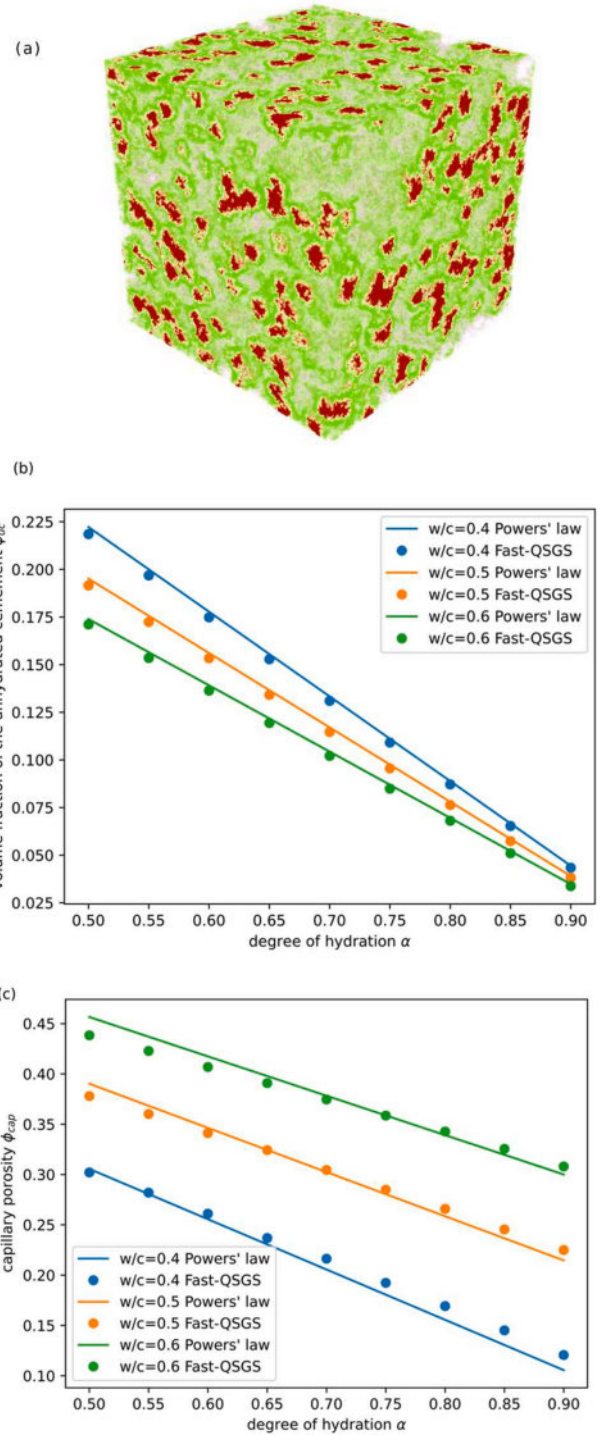
The permeabilities of the generated samples calculated by an in-house GPU-LBM [3] (lattice Boltzmann method) are compared with the experimental results in literatures [61,64,66–69] as Fig. 6 displays. With both the sphere and diamond spacing method, the permeabilities of the generated samples agree well with the literature values and the deviation between the two spacing methods is neglectable. However, the permeabilities of those samples without spacing ( $s0$ ) are larger than the experimental values by about 60 %–80 %, which demonstrates the necessity of uniformly distributing seeds when reconstructing sandstones. Through this benchmark, we validate that Fast-QSGS can be used to reconstruct naturally disordered granular media.

### 3.2.2. Hydrated cement

To further demonstrate the capability of Fast-QSGS, we provide a benchmark for generating a multiphase disordered media, the hydrated cement. Yang and Wang [55] introduced a QSGS-based method to generate the hydrated cement. Initially, cement particles are formed via the standard QSGS. This is followed by a hydration process, encompassing three key stages: dissolution, diffusion/reaction, and invasion. Detailed information on the algorithm can be found in [55]. In this work, the dissolution, diffusion/reaction, and invasion are also vectorized and implemented via CuPy [50].

For the hydrated cement, the seed distribution of the cement particles is determined via  $S_d = \frac{6\Delta^3}{\pi d^3(3.2w/c+1)}$ , where  $d$  is the mean diameter of the cement particles,  $w/c$  is the water-cement ratio and  $\Delta$  denotes the resolution. Then the hydration steps are conducted till the desired hydration degree  $\alpha$  is achieved. Therefore, the hydrated cement generation is controlled by three parameters ( $w/c$ ,  $\alpha$ ,  $d$ ). In this work, we select  $d = 17 \mu\text{m}$  and  $\Delta = 1 \frac{\mu\text{m}}{\text{voxel}}$  for all our cases to generate cement of  $200^3$ , while the hydration degree and water-cement ratio vary. An illustration of the generated hydrated cement structure is displayed in Fig. 7(a).

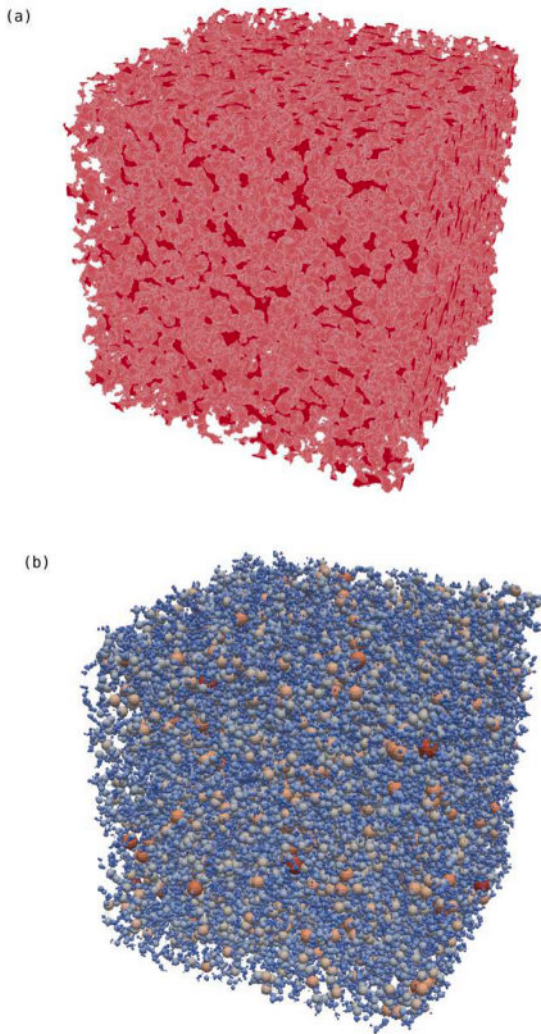
The capillary porosity and the volume fraction of the un-hydrated cement by Fast-QSGS are compared with the Powers' law [57] in Fig. 7. The average relative error is 1.9 % for the volume fraction of the unhydrated cement phase  $\phi_{uc}$  and 3.4 % for the capillary porosity  $\phi_{cap}$ . The maximum relative error is 15 % for the case ( $\alpha = 0.9$ ,  $w/c = 0.6$ ) on



**Fig. 7.** (a) Illustration of the reconstructed cement by Fast-QSGS: red (unhydrated cement phase), green (production of hydration phase). The pores are displayed in transparent. The sample displayed is  $200^3$ ,  $w/c=0.4$  and  $\alpha = 0.6$ . (b) Comparison between the volume fraction of the unhydrated cement phase of the reconstructed cement and Powers' law [57]. (c) Comparison between the capillary porosity of the reconstructed cement and Powers' law [57].

capillary porosity  $\phi_{cap}$ . Our results indicate good agreement between the generated structures and the empirical results on a wide range of parameters. Therefore, we validate the present Fast-QSGS on generating multiphase structures.

A Fontainebleau sandstone of size  $1200^3$  reconstructed by Fast-QSGS is displayed in Fig. 8(a). This reconstruction is conducted on A100 in

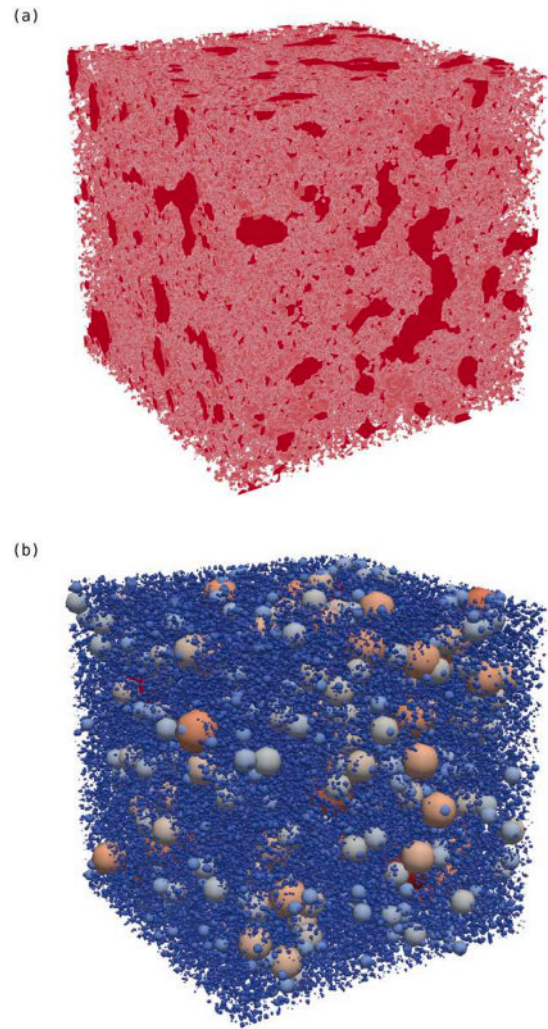


**Fig. 8.** (a) Illustration of a large Fontainebleau sandstone reconstructed by Fast-QSGS.  $\phi = 0.15$ , size  $1200^3$  and  $s = 30$ . This sample is generated on A100 in 25.8 min. (b) Pore network model of the reconstructed sample. Extraction of this sample takes more than 6 h on an Intel Xeon Silver 4214R CPU.

merely 25.8 min, which demonstrates the capability of generating large structures via Fast-QSGS. The resolution of this sample is  $4 \mu\text{m}$ , and the physical size is 4.8 mm. The extracted network by PoreSpy [59] and OpenPNM [58] is displayed in Fig. 8(b), containing 68,101 pores and 118,276 throats. The extraction takes over 6 h on our Intel Xeon Silver 4214R CPU, indicating that even for the pore network method, structures of this size pose certain computational challenges. Fig. 9(a) also demonstrated a hierarchical disordered media of size  $1000^3$  generated by Fast-QSGS, which is completed on A100 within 7.9 min. The detailed generation method of hierarchical structures can be referred to [25]. Two different scales of pores are contained in this structure, with larger-scale pores having an average diameter of 56 voxels, and the smaller-scale pores having an average diameter of 8 voxels. The extracted network, which takes 5.18 h to extract, is displayed in Fig 9(b).

#### 4. Conclusion

In this work, we present Fast-QSGS, an open-source program designed for granular disordered media generation. The classic QSGS method is vectorized and implemented using NumPy and CuPy. We also improve the original QSGS by introducing variable growth probability function and seed spacing. Depending on GPU devices, granular disor-



**Fig. 9.** (a) Illustration of a hierarchical sandstone reconstructed by Fast-QSGS.  $\phi = 0.24$ , size  $1000^3$ . Two different scales of pores are contained in this structure, with larger-scale pores having an average diameter of 56 voxels, and the smaller-scale pores having an average diameter of 8 voxels. This sample is generated on A100 in 7.9 min. (b) Pore network model of the reconstructed sample. Extraction of this sample takes more than 5.18 h on an Intel Xeon Silver 4214R CPU.

dered media of size  $400^3$  can be generated in just 50–150 s by Fast-QSGS, compared to the original version which takes 17.3h, resulting in a speedup of over 400. Furthermore, the physical properties of the disordered structures are also verified through the reconstruction of Fontainebleau sandstone and hydrated cement. Coded in Python, Fast-QSGS is both simple and efficient, allowing easy customization and seamless integration with other popular open-source packages in the community. Additionally, multi-GPU and CPU core-level parallelism features may be introduced via mpi4py [70] and halo layers in future developments. We believe Fast-QSGS will be a valuable tool for researchers studying disordered media.

#### CRediT authorship contribution statement

**Guang Yang:** Investigation, Software, Validation, Writing – original draft. **Tong Liu:** Software. **Xukang Lu:** Software, Validation, Writing – review & editing. **Moran Wang:** Conceptualization, Supervision, Validation, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.

## Acknowledgments

This work is financially supported by the NSF grant of China (No. 12272207) and National Key R&D Program of China (No. 2019YFA0708704).

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at [doi:10.1016/j.cpc.2024.109241](https://doi.org/10.1016/j.cpc.2024.109241).

## References

- [1] S. Schmieschek, et al., LB3D: A parallel implementation of the Lattice-Boltzmann method for simulation of interacting amphiphilic fluids, *Comput. Phys. Commun.* 217 (2017) 149–161.
- [2] K. Kono, et al., Application of lattice Boltzmann model to multiphase flows with phase transition, *Comput. Phys. Commun.* 129 (1-3) (2000) 110–120.
- [3] G. Yang, et al., Implementation of a direct-addressing based lattice Boltzmann GPU solver for multiphase flow in porous media, *Comput. Phys. Commun.* (2023) 291.
- [4] C. Xie, et al., Lattice Boltzmann modeling for multiphase viscoplastic fluid flow, *J. Non-Newton Fluid* 234 (2016) 118–128.
- [5] C. Xie, M.T. Balhoff, Lattice Boltzmann Modeling of the Apparent Viscosity of Thinning–Elastic Fluids in Porous Media, *Transport Porous Med.* 137 (1) (2021) 63–86.
- [6] M. Wang, Q. Kang, E. Ben-Naim, Modeling of electrokinetic transport in silica nanofluidic channels, *Anal. Chim. Acta* 664 (2) (2010) 158–164.
- [7] A. Alizadeh, X. Jin, M. Wang, Pore-scale Study of Ion Transport Mechanisms in Inhomogeneously Charged Nanoporous Rocks: Impacts of Interface Properties on Macroscopic Transport, *J. Geophys. Res.* 124 (6) (2019) 5387–5407.
- [8] R. Vetter, J.O. Schumacher, Free open reference implementation of a two-phase PEM fuel cell model, *Comput. Phys. Commun.* 234 (2019) 223–234.
- [9] S. Thomas, Enhanced Oil Recovery - An Overview, *Oil & Gas Science and Technology - Revue de l'IFP* 63 (1) (2007) 9–19.
- [10] L.K. Abidoye, K.J. Khudaida, D.B. Das, Geological Carbon Sequestration in the Context of Two-Phase Flow in Porous Media: A Review, *Crit. Rev. Env. Sci. Tec.* 45 (11) (2015) 1105–1147.
- [11] F. Municchi, et al., Heterogeneous Multi-Rate mass transfer models in OpenFOAM®, *Comput. Phys. Commun.* (2021) 261.
- [12] N.P. Brandon, D.J. Brett, Engineering porous materials for fuel cell applications, *Philos. Trans. A Math. Phys. Eng. Sci.* 364 (1838) (2006) 147–159.
- [13] M. Wang, N. Pan, Elastic property of multiphase composites with random microstructures, *J. Comput. Phys.* 228 (16) (2009) 5978–5988.
- [14] M. Wang, et al., Mesoscopic predictions of the effective thermal conductivity for microscale random porous media, *Phys. Rev. E* 75 (3 Pt 2) (2007) 036702.
- [15] M. Wang, N. Pan, Predictions of effective physical properties of complex multiphase materials, *Mat. Sci. Eng. R* 63 (1) (2008) 1–30.
- [16] E.C. Stern, et al., Nonequilibrium flow through porous thermal protection materials, Part I: Numerical methods, *J. Comput. Phys.* 380 (2019) 408–426.
- [17] A. Brandstotter, et al., Shaping the branched flow of light through disordered media, *Proc. Natl. Acad. Sci. U. S. A.* 116 (27) (2019) 13260–13265.
- [18] A. Patsyk, et al., Observation of branched flow of light, *Nature* 583 (7814) (2020) 60–65.
- [19] J.A. Rebollo López, Numerical simulation of propagation of light through random disordered media to model branched flow phenomena, Instituto Tecnológico y de Estudios Superiores de Monterrey, 2022.
- [20] M.J. Blunt, *Multiphase Flow in Permeable Media A Pore-Scale*, Cambridge University Press, Cambridge, UK, 2017.
- [21] W. Kreher, W. Pompe, *Internal Stresses in Heterogeneous Solids*, Akademie-Verlag, Berlin, Germany, 1989.
- [22] M. Wang, J. Wang, S. Chen, Roughness and cavitations effects on electro-osmotic flows in rough microchannels using the lattice Poisson–Boltzmann methods, *J. Comput. Phys.* 226 (1) (2007) 836–851.
- [23] M.J. Blunt, et al., Pore-scale imaging and modelling, *Adv. Water Resour.* 51 (2013) 197–216.
- [24] T. Liu, X. Jin, M. Wang, Critical Resolution and Sample Size of Digital Rock Analysis for Unconventional Reservoirs, *Energies* 11 (7) (2018).
- [25] Z. Wang, et al., Pore-scale geometry effects on gas permeability in shale, *J. Nat. Gas Sci. Eng.* 34 (2016) 948–957.
- [26] S. Kelly, et al., Assessing the utility of FIB-SEM images for shale digital rock physics, *Adv. Water Resour.* 95 (2016) 302–316.
- [27] A.Q. Raeini, M.J. Blunt, B. Bijeljic, Direct simulations of two-phase flow on micro-CT images of porous media and upscaling of pore-scale forces, *Adv. Water Resour.* 74 (2014) 116–126.
- [28] J. Tölke, M. Krafczyk, TeraFLOP computing on a desktop PC with GPUs for 3D CFD, *Int. J. Comput. Fluid D* 22 (7) (2008) 443–456.
- [29] F. Jiang, et al., A GPU-accelerated fluid–structure-interaction solver developed by coupling finite element and lattice Boltzmann methods, *Comput. Phys. Commun.* (2021) 259.
- [30] W. Lei, et al., Enhanced oil recovery mechanism and recovery performance of micro-gel particle suspensions by microfluidic experiments, *Energy Sci. Eng.* 8 (4) (2019) 986–998.
- [31] C. Xie, et al., Self-adaptive preferential flow control using displacing fluid with dispersed polymers in heterogeneous porous media, *J. Fluid Mech.* 906 (2020) A10.
- [32] P.-E. Øren, S. Bakke, Process Based Reconstruction of Sandstones and Prediction of Transport Properties, *Transport Porous Med.* 46 (2/3) (2002) 311–343.
- [33] H. Okabe, M.J. Blunt, Pore space reconstruction using multiple-point statistics, *J. Petrol. Sci. Eng.* 46 (1-2) (2005) 121–137.
- [34] R.D. Hazlett, Simulation of capillary-dominated displacements in microtomographic images of reservoir rocks, *Transport Porous Med.* 20 (1-2) (1995) 21–35.
- [35] K. Wu, et al., An Efficient Markov Chain Model for the Simulation of Heterogeneous Soil Structure, *Soil Sci. Soc. Am. J.* 68 (2) (2004) 346–351.
- [36] T. Huang, et al., GPU-based SNESIM implementation for multiple-point statistical simulation, *Comput. Geosci.* 54 (2013) 75–87.
- [37] T. Zhang, et al., GPU-accelerated 3D reconstruction of porous media using multiple-point statistics, *Comput. Geosci.* 19 (1) (2014) 79–98.
- [38] Zhou, X.-P. and N. Xiao, 3D Numerical Reconstruction of Porous Sandstone Using Improved Simulated Annealing Algorithms. *Rock Mech. Rock Eng.*, 2018. 51(7): p. 2135-2151.
- [39] P. Tahmasebi, et al., Accelerating geostatistical simulations using graphics processing units (GPU), *Comput. Geosci.* 46 (2012) 51–59.
- [40] J. Feng, et al., Reconstruction of porous media from extremely limited information using conditional generative adversarial networks, *Phys. Rev. E* 100 (3-1) (2019) 033308.
- [41] J. Feng, et al., An end-to-end three-dimensional reconstruction framework of porous media from a single two-dimensional image based on deep learning, *Comput. Method Appl. M.* (2020) 368.
- [42] H. Vincenti, et al., An efficient and portable SIMD algorithm for charge/current deposition in Particle-In-Cell codes, *Comput. Phys. Commun.* 210 (2017) 145–154.
- [43] F. Robertsén, K. Mattila, J. Westerholm, High-performance SIMD implementation of the lattice-Boltzmann method on the Xeon Phi processor, *Concurr. Comp. Pract. E* (13) (2018) 31.
- [44] G.S. Grest, B. Dünweg, K. Kremer, Vectorized link cell Fortran code for molecular dynamics simulations for a large number of particles, *Comput. Phys. Commun.* 3 (55) (1989) 269–285.
- [45] H. Watanabe, K.M. Nakagawa, SIMD vectorization for the Lennard-Jones potential with AVX2 and AVX-512 instructions, *Comput. Phys. Commun.* 237 (2019) 1–7.
- [46] C.P. Stone, A.T. Alferman, K.E. Niemeyer, Accelerating finite-rate chemical kinetics with coprocessors: Comparing vectorization methods on GPUs, MICs, and CPUs, *Comput. Phys. Commun.* 226 (2018) 18–29.
- [47] C.R. Harris, et al., Array programming with NumPy, *Nature* 585 (7825) (2020) 357–362.
- [48] P. Virtanen, et al., SciPy 1.0: fundamental algorithms for scientific computing in Python, *Nat. Methods* 17 (3) (2020) 261–272.
- [49] Abadi, M., et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*. arXiv, 2016.
- [50] R. Okuta, et al., Cupy: A numpy-compatible library for nvidia gpu calculations, in: *Proceedings of workshop on machine learning systems (LearningSys) in the thirty-first annual conference on neural information processing systems (NIPS)*, 2017.
- [51] The MathWorks Inc, *gpuArray: Array stored on GPU*, 2023. Available from: <https://uk.mathworks.com/help/parallel-computing/gpuarray.html#d126e37835>.
- [52] M. Mohrhard, et al., Auto-vectorization friendly parallel lattice Boltzmann streaming scheme for direct addressing, *Comput. Fluids* 181 (2019) 1–7.
- [53] V.L. A. Perepelkina, Heterogeneous LBM Simulation Code with LrNLA Algorithms *Commun. Comput. Phys.* 33 (1) (2023) 214–244.
- [54] M. Wang, Q. Kang, Electrokinetic transport in microchannels with random roughness, *Anal. Chem.* 81 (8) (2009) 2953–2961.
- [55] Y. Yang, M. Wang, Pore-scale modeling of chloride ion diffusion in cement microstructures, *Cement Concrete Comp.* 85 (2018) 92–104.
- [56] G. Yang, M. Wang, Surface roughness effect on dynamic wettability in imbibition process, *Comput. Fluids* (2023) 263.
- [57] T.C. Powers, T.L. Brownyard, Studies of the physical properties of hardened Portland cement paste, in: *Journal Proceedings* (1946).
- [58] J. Gostick, et al., OpenPNM: A Pore Network Modeling Package, *Comput. Sci. Eng.* 18 (4) (2016) 60–74.
- [59] J.T. Gostick, et al., PoreSpy: A python toolkit for quantitative analysis of porous media images, *Journal of Open Source Software* 4 (37) (2019) 1296.
- [60] TechPowerUp, GPU Specs Database, TechPowerUp, 2023.
- [61] T. Bourbie, B. Zinszner, Hydraulic and acoustic properties as a function of porosity in Fontainebleau Sandstone, *J. Geophys. Res.* 90 (B13) (2012) 11524–11532.

- [62] J.T. Fredrich, K.H. Greaves, J.W. Martin, Pore geometry and transport properties of Fontainebleau sandstone, *Int. J. Rock Mech. Min.* 30 (7) (1993) 691–697.
- [63] B. Ferreol, D.H. Rothman, Lattice-Boltzmann simulations of flow through Fontainebleau sandstone. *Multiphase Flow in Porous Media*, Springer, 1995, pp. 3–20.
- [64] D.A. Coker, S. Torquato, J.H. Dunsmuir, Morphology and physical properties of Fontainebleau sandstone via a tomographic analysis, *J. Geophys. Res.* 101 (B8) (1996) 17497–17506.
- [65] F.D.E. Latief, Analysis and Visualization of 2D and 3D Grain and Pore Size of Fontainebleau Sandstone Using Digital Rock Physics, *J. Phys. Conf. Ser.* (2016) 739.
- [66] F.A. Saadi, K.-H. Wolf, C.v. Kruijsdijk, Characterization of Fontainebleau Sandstone: Quartz Overgrowth and its Impact on Pore-Throat Framework, *Journal of Petroleum & Environmental Biotechnology* 08 (03) (2017).
- [67] P.M. Doyen, Permeability, conductivity, and pore geometry of sandstone, *J. Geophys. Res.* 93 (B7) (2012) 7729–7740.
- [68] C.T. Gomez, J. Dvorkin, T. Vanorio, Laboratory measurements of porosity, permeability, resistivity, and velocity on Fontainebleau sandstones, *Geophysics* 75 (6) (2010) E191–E204.
- [69] A. Revil, P. Kessouri, C. Torres-Verdín, Electrical conductivity, induced polarization, and permeability of the Fontainebleau sandstone, *Geophysics* 79 (5) (2014) D301–D318.
- [70] L. Dalcin, Y.-L.L. Fang, mpi4py: Status Update After 12 Years of Development, *Comput. Sci. Eng.* 23 (4) (2021) 47–54.