SoftwareX 19 (2022) 101151

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

juSPH: A Julia-based open-source package of parallel Smoothed Particle Hydrodynamics (SPH) for dam break problems

Mimi Luo, Jiayu Qin [*], Gang Mei [*]

School of Engineering and Technology, China University of Geosciences (Beijing), Beijing 100083, China

A R T I C L E   I N F O

Article history:
Received 18 May 2022
Received in revised form 3 July 2022
Accepted 11 July 2022

Keywords:
Smoothed Particle Hydrodynamics
CPU parallelization
Julia language
Dam break problems

A B S T R A C T

Numerous software packages have emerged to implement the SPH method and are being used for the analysis of various scientific and engineering problems. Nevertheless, few efforts have been made to develop packages of SPH by considering both the efficiency and readability in balance. Considering efficiency and readability, we designed and developed a Julia-based open-source package of parallel SPH called juSPH. We described juSPH regarding the SPH methodology, software architecture, and software functionalities, and verified the accuracy and evaluated the efficiency by modeling a dam-break experiment. Furthermore, the modular structure of juSPH makes the code easily readable and facilitates further development.

Code metadata

Current code version	1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00107
Permanent link to Reproducible Capsule	–
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Julia
Compilation requirements, operating environments & dependencies	Windows, Visual Studio Code
If available Link to developer documentation/manual	https://github.com/GangMei-CUGB/juSPH
Support email for questions	gang.mei@cugb.edu.cn

1. Motivation and significance

The Smoothed Particle Hydrodynamics (SPH) method is a meshless particle method based on Lagrangian mechanics, which has unique advantages in the field of free surface flows due to its meshless nature [1]. It was first proposed in solving astrophysical problems [2,3], and after a series of research developments, its research field has been constantly expanded and later applied to the field of free-surface flow such as dam-breaking flow or wave-breaking flow [4,5], as well as to the simulation of wave flooding and collision with obstacles [6] and multiphase flow in porous media [7]. In addition, a lot of improvements have been made to the classical SPH method, and thus many SPH variants such as ISPH [8], δ-SPH [9], δ^{+}-SPH [10] and δ-ALE-SPH [11] have been proposed, which are now used to simulate complex free surface flow and fluid–solid interaction problems.

With the increasing maturity of the SPH method, numerous software packages have emerged in several years to implement the SPH method and are being used for the analysis of various scientific and engineering problems [12–20]. Nevertheless, open-source software suffers from the following problems: SPH software is typically developed in low-level languages such as C/C++ or FORTRAN [12–18], resulting in a complex structure, poor legibility, and difficulty later maintaining and improving SPH programs; software developed in high-level languages such as Python and MATLAB, whose code is readable, has limited computational efficiency [19,20]. An overview of the above, only a few researchers have dedicated themselves to developing open source

* Corresponding authors.
E-mail addresses: jiayu.qin@cugb.edu.cn (Jiayu Qin), gang.mei@cugb.edu.cn (Gang Mei).

https://doi.org/10.1016/j.softx.2022.101151

packages of the SPH method that take into account computational efficiency and code readability.

From the perspective of computer languages, the Julia language, which has emerged in recent years, is a user-friendly and fully open-source computer programming language created with high-performance computing in mind [21]. The Julia language finds a balance between a statically compiled language and a dynamically typed language, making the code simple and easy to read while still having similar performance to a static language [22]. In this paper, we designed and developed an open-source software package for the SPH method based on the Julia language, juSPH, to be used for the two-dimensional dam-break problems. The algorithm structure is designed to solve the large-scale two-dimensional dam-break problems by combining the principle of SPH and the characteristics of parallel computing in the Julia language. The juSPH software can work on single-core CPUs as well as multi-core CPUs.

2. Software description

2.1. SPH methodology in juSPH

The juSPH software computes the numerical solution of the Euler equation for an ideal fluid in the Lagrangian formulation, including the continuity equation

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{v} \tag{1}$$

and the momentum equation

$$\frac{d\mathbf{v}}{dt} = -\frac{\nabla p}{\rho} + \mathbf{g} \tag{2}$$

where $\mathbf{v}$ is the velocity vector, ρ is the density, p is the pressure, and $\mathbf{g}$ the body force.

The incompressible fluid is simulated with the SPH weakly-compressible method by introducing the equation of state and estimating the pressure through the density field [23]

$$p(\rho) = \frac{\rho_0 c^2}{\gamma}\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] + \chi \tag{3}$$

where ρ_0, c_0 and χ are the reference density, the artificial sound speed, and the background pressure, respectively. γ is set to 7 for water and is a parameter that must be calibrated for each fluid. When Eq. (3) applies free-surface flow, the background pressure χ is set to zero.

For the SPH continuity equation discretization, we choose the simpler form [24]. The discretized form of the continuity Eq. (1) is shown in Eq. (4) and is implemented in the file "density_derivative.jl".

$$\frac{d\rho_a}{dt} = \rho_a\sum_b \frac{m_b}{\rho_b}\mathbf{v}_{ab}\cdot\nabla_a W_{ab} \tag{4}$$

where $\mathbf{v}_{ab} = \mathbf{v}_a - \mathbf{v}_b$ and $\nabla_a W_{ab} = \nabla_a W(\mathbf{r}_a - \mathbf{r}_b, h)$.

The momentum Eq. (2) is discretized with the volume element $V = m/\rho$, and an artificial viscosity term is introduced to maintain numerical stability. This step is implemented in the file "velocity_derivative.jl", the acceleration due to the pressure gradient is approximated as shown in Eq. (5) [25], and the acceleration resulting from artificial viscosity is approximated as shown in Eq. (6) [26].

$$\frac{d\mathbf{v}_a}{dt} = -\frac{1}{m_a}\sum_b (V_a^2 + V_b^2)\frac{\rho_b p_a + \rho_a p_b}{\rho_a + \rho_b}\nabla_a W_{ab} \tag{5}$$

$$\frac{d\mathbf{v}_a}{dt} = -\sum_b m_b \alpha h_{ab} c_{ab}\frac{(\mathbf{v_a} - \mathbf{v_b})\cdot(\mathbf{r_a} - \mathbf{r_b})}{\rho_{ab}(|\mathbf{r_a} - \mathbf{r_b}|^2 + \varepsilon h_{ab}^2)}\nabla_a W_{ab} \tag{6}$$

where α is the artificial viscosity parameter. The coefficients h_{ab}, c_{ab} and ρ_{ab} are the averages of the smoothing length, the sound speed and the density in between particles a and b, respectively. The parameter ε is usually taken as 0.01 to ensure that the denominator is non-zero.

The smooth kernel function chosen for the software is the quantic spline smooth kernel function [27], the support domain of this function is 3 h, see Eq. (7). The calculation of the smooth kernel function is implemented in the file "kernel_function.jl". The comparatively large support domain of this function enhances the accuracy and stability of the interpolation, which is critical for achieving the boundary conditions.

$$W(q,h) = \alpha_d \begin{cases} (3-q)^5 - 6(2-q)^5 + 15(1-q)^5 & 0 \le r < 1 \\ (3-q)^5 - 6(2-q)^5 & 1 \le r < 2 \\ (3-q)^5 & 2 \le r < 3 \\ 0 & 3 \le r \end{cases} \tag{7}$$

where $q = |\mathbf{x}_{ab}|/h$ is the ratio between the distance of particles a and b and the smoothing length, α_d is the dimensional normalizing factor defined by $\alpha_d = [1/(120h), 7/(478\pi h^2), 1/(120\pi h^3)]$ for one-, two- and three-dimensions, respectively.

The linked list search method [28] is chosen for the nearest neighbor particle search in the juSPH software, and the cell-linked list approach (CLL) is used to create a list of neighbors. The approach is implemented in two steps. First, the steps to create the particle list of the cell and the adjacent cell list of the cell are implemented in the files "link_list.jl" and "get_adjacentBins.jl". Second, the neighborhood search for different particles is implemented in the files "boundary_update.jl", "velocity_derivative.jl" and "density_derivative.jl".

The juSPH software performs time integration of the equations of motion with the velocity-Verlet time integration algorithm [29], and this type of time integration is particularly suitable for SPH programs, because it is energy efficient and relatively simple to implement, in the file "SPH.jl". Specific forms are as follows:

$$\mathbf{v}_a^{n+\frac{1}{2}} = \mathbf{v}_a^n + \frac{\Delta t}{2}\left(\frac{d\mathbf{v}_a}{dt}\right)^n \tag{8}$$

$$\mathbf{r}_a^{n+\frac{1}{2}} = \mathbf{r}_a^n + \frac{\Delta t}{2}\mathbf{v}_a^{n+\frac{1}{2}} \tag{9}$$

$$\rho^{n+1} = \rho^n + \Delta t\left(\frac{d\rho}{dt}\right)^{n+\frac{1}{2}} \tag{10}$$

$$\mathbf{r}_a^{n+1} = \mathbf{r}_a^{n+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{v}_a^{n+\frac{1}{2}} \tag{11}$$

$$\mathbf{v}_a^{n+1} = \mathbf{v}_a^{n+\frac{1}{2}} + \frac{\Delta t}{2}\left(\frac{d\mathbf{v}_a}{dt}\right)^{n+1} \tag{12}$$

It is essential to limit the size of the time step according to some criteria to sustain the stability of the numerical integration [30]. The time step Δt is evaluated in each iteration according to three conditions [31]: the CFL condition $\Delta t \le 0.25\frac{h}{c_{\max}+|\mathbf{v}_{\max}|}$, the viscous condition $\Delta t \le 0.125\frac{h^2}{\nu}$, where an equivalent effective physical kinematic viscosity [32] can be calculated from $\nu = \frac{1}{2(d+2)}\alpha h_{ab}c_{ab}$, and the body force condition $\Delta t \le 0.25\left(\frac{h}{|\mathbf{g}|}\right)^{\frac{1}{2}}$. The minimum value of the above three conditions is chosen as the time step to globally satisfy all conditions. The calculation of the time step is implemented in the file "step_size.jl".

Given the boundary condition, we choose a generalized wall boundary condition [31]. The method uses dummy particles to

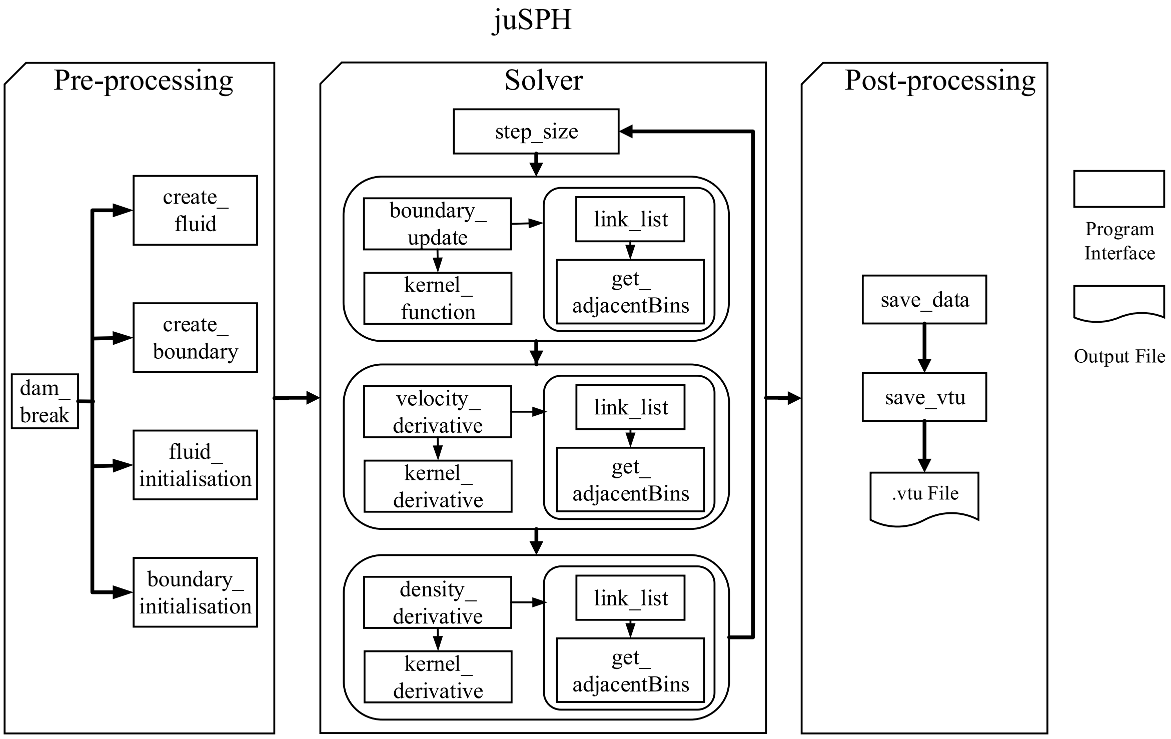


Fig. 1. The architecture of the juSPH software.

describe approximately the interface with the fluid phase and the boundary. This boundary condition is implemented in the file "boundary_update.jl". The smooth fluid phase velocity field is first extrapolated to the dummy particle location to establish the non-slip condition, and the wall particle velocity is

$$v_w = 2v_a - \frac{\sum_b v_b W_{ab}}{\sum_b W_{ab}} \tag{13}$$

The method is based on the balance of forces on the wall and calculates the pressure to maintain the impermeability of the rigid wall. For wall particles, the pressure of the wall particles under the influence of multiple fluid particle pressures is

$$p_w = \frac{\sum_f p_f W_{wf} + \sum_f \rho_f \mathbf{g} \cdot (\mathbf{r}_w - \mathbf{r}_f) W_{wf}}{\sum_f W_{wf}} \tag{14}$$

We also acquire dummy particles' density from the pressure p_w as

$$\rho_w = \rho_{0,b}\left(\frac{p_w - \chi}{p_{0,b}} + 1\right)^{\frac{1}{\gamma}} \tag{15}$$

2.2. Software architecture

The juSPH software consists of three modules: pre-processing, solver, and post-processing, see the architecture of the juSPH software in Fig. 1.

2.2.1. Pre-processing

The parameter initialization process is implemented by code and includes the input of Initial sizes of the two-dimensional dam-break model, initial geometric configuration of the particles, particle properties, and other simulation control parameters. The main script main.jl calls dam_break.jl to generate a numerical model of a two-dimensional dam break, dam_break.jl calls create_fluid.jl and create_boundary.jl to create fluid particles and boundary dummy particles respectively, calls fluid_initialization.jl to initialize the coordinates, velocity, mass, density, and pressure of fluid particles, and calls boundary_initialization.jl to initialize the coordinates, velocity, and pressure of boundary dummy particles.

2.2.2. Solver

The solver module includes the main modules of the SPH numerical simulation and implements a velocity-Verlet time integration algorithm in the time integration module. The following post-processing modules need to be included in the solver module. The solver module designs parallel algorithms for computational fluid dynamics problems with large-scale particle numbers and uses parallelism to significantly improve solution efficiency.

Considering the stability and efficiency of the program, we utilize multi-core processing to achieve parallelism in the SPH algorithm [33]. As shown in Fig. 2 (top right), the multi-core computer programs the task and redistributes it to the individual processes to start the computation. In the Julia language, setting the array to the "SharedArrays" type allows each process to access the entire array, using the "SharedArrays" and "@distributed" macro, distributed computing assigns tasks to individual threads for parallel computation in for-loops, depending on the number of processes and the total number of tasks.

The solver module is implemented in a time integration module with specific steps consisting of multiple loop steps, each loop step can be further divided into several steps. As shown in Fig. 2 (left), we set "particles" to the "SharedArrays" type, and then the steps of updating the velocity of fluid particles, updating the position of fluid particles, updating the density of fluid particles, updating the pressure of fluid particles, updating the position of fluid particles and updating the velocity of fluid particles are computed in parallel using multi-core processing to improve efficiency. The for-loop of file "boundary_update.jl", file "velocity_derivative.jl" and file "density_derivative.jl" also make use of multi-core processing to improve computational efficiency.

In the meantime, from the process diagram of the solver, it can be seen that the neighborhood particle search takes at least

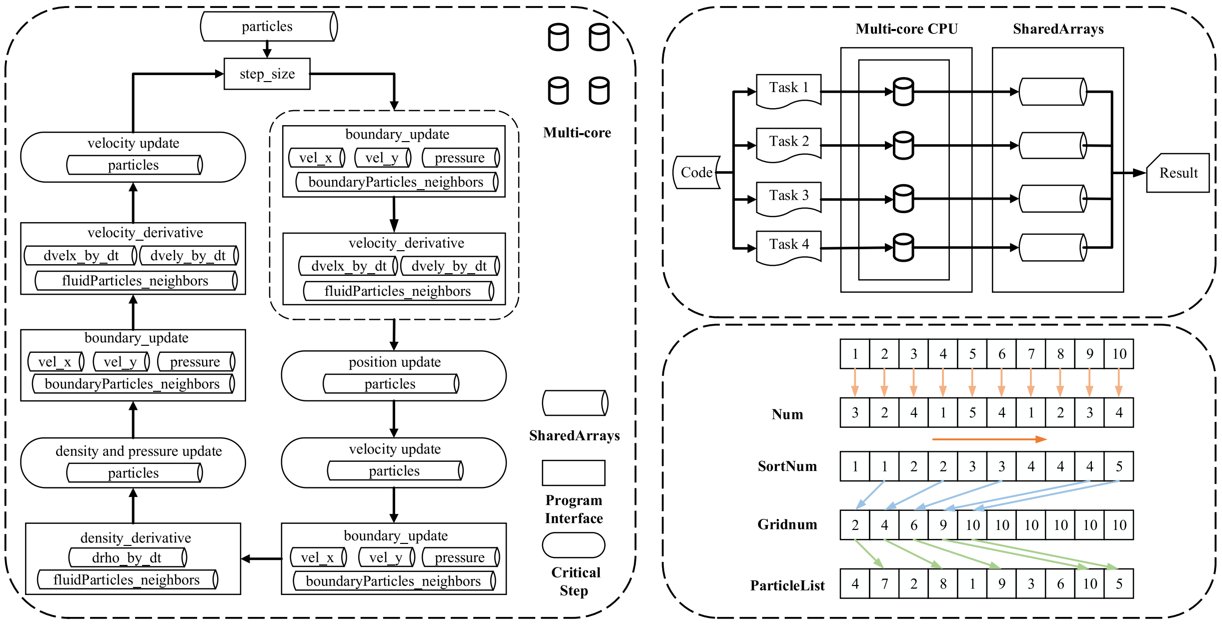


Fig. 2. Left: Process of the solver. Top right: Workflow diagram of multi-core CPU computing in Julia language. Bottom right: Index sorting data structure. The top row illustrates the particle number, where the number indicates the corresponding cell index.

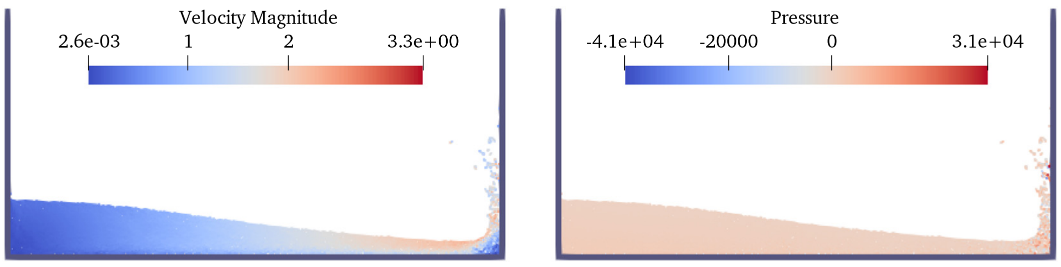


Fig. 3. Visualization of velocity (left) and pressure (right) of fluid particles.

4 times in one loop step. Therefore, it is necessary to design an efficient parallel linked list search algorithm. On the one hand, we use multi-core processing to achieve parallelism in the linked list search algorithm. On the other hand, we design an index sorting data structure without opening another lookup table, as shown in Fig. 2 (bottom right), using an index sorting data structure to obtain the list "PartlicleList" of particles stored in each grid, which has the advantage of improving cache hit rates and reducing memory transfers for implementations on parallel architectures with shared memory.

2.2.3. Post-processing

When the time step specifies a time step or within a certain time interval, the result of the calculation is output to additional files for subsequent analysis or post-processing. To plot the results of these numerical calculations, we use the open-source advanced visualization software ParaView [34] for visualization. First, the fluid particles' coordinates and velocities, densities, and pressures are output in VTU format files by the "save_vtu" function, and then these VTU format files are imported into ParaView for visualization (Fig. 3).

2.3. Software functionalities

After running the juSPH software package, the flow field morphology, velocity distribution, and pressure distribution of the breached dam at different moments can be obtained, which can well simulate the two-dimensional large deformation free-surface dramatically changing water flow problems. The juSPH software can be integrated with Visual Studio Code to build the run-time environments, the simple code and fast computational efficiency of juSPH software, which can run on multi-core CPUs, enables the simulation of large-scale particle number dam-break problems.

3. Illustrative examples

In this section, we verified the accuracy and evaluated the efficiency of the juSPH software through an example of a dam-break experiment. The experimental data of ETSIN [35] are selected for calculation, and the experimental model parameters are shown in Fig. 4(a). The size of the numerical water tank is 1.61×0.8(m), and the size of the water body is 0.6×0.3(m). To evaluate the computational efficiency of the juSPH software, the test environment is shown in Table 1.

3.1. Verification of the accuracy of juSPH

In the numerical model, the boundary conditions are set on the left, bottom, and right sides of the numerical water tank, and each side of the boundary is discretized into three layers of dummy particles, and the specific parameter settings for the numerical model of the dam break are shown in Table 2.

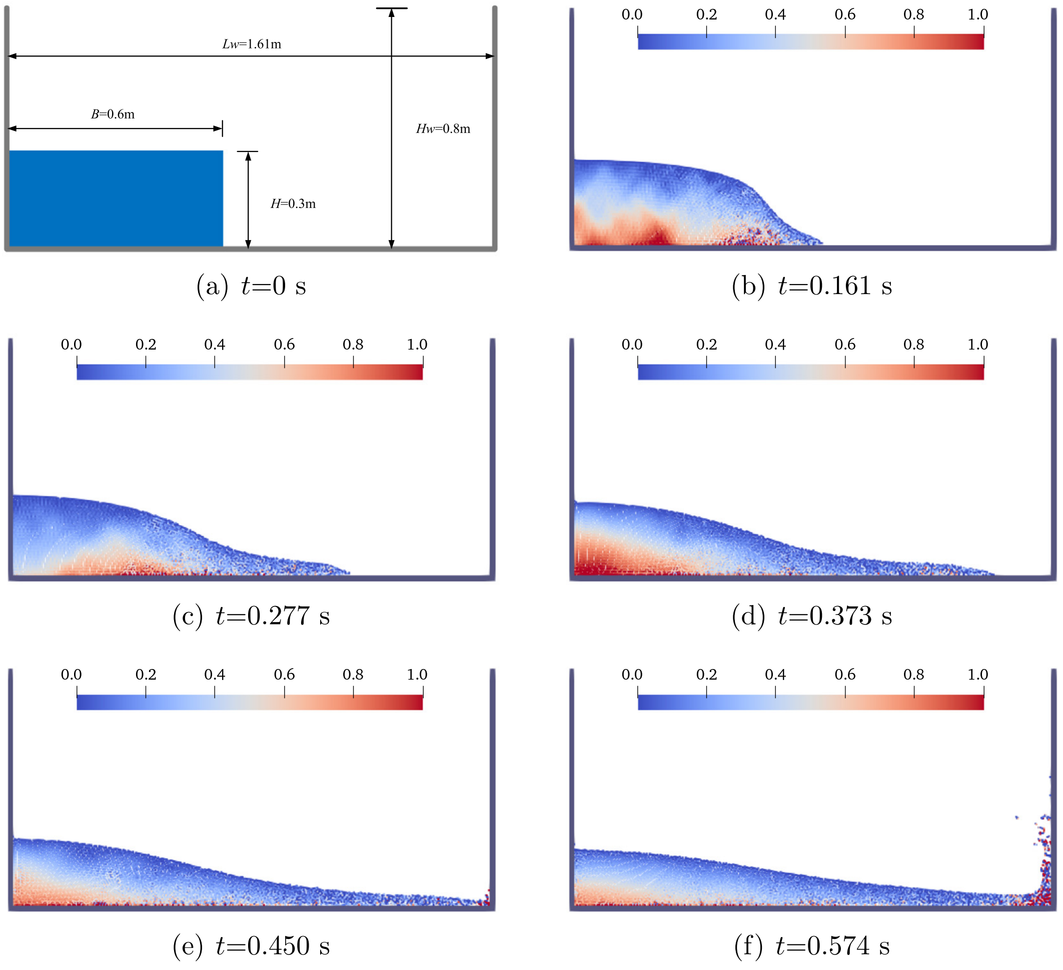


Fig. 4. (a) The initial size setting of the two-dimensional dam break model, (b)–(f) The snapshot of dam break motion in which fluid particles are colored by the dimensionless pressure $P/\rho gH$.

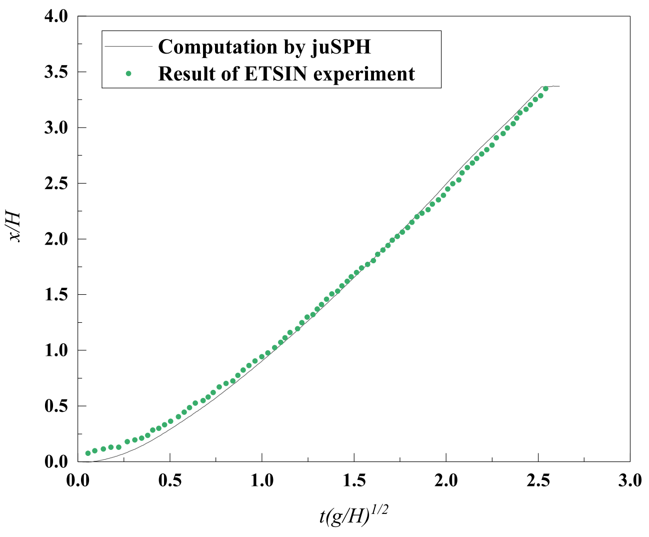


Fig. 5. Comparison of the position of the leading edge of the water column.

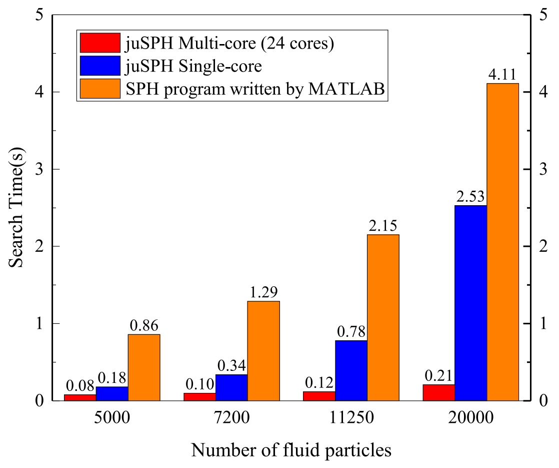
Given the above experimental conditions, some snapshots of dam break motion at different moments are shown in Fig. 4(b)–(f). As can be seen from these snapshots, the simulated and experimental results are consistent in terms of general trends. In addition to the water movement forms, Fig. 4(b)–(f) also shows the calculated pressure fields. It can also be seen from the figure that the fluid particle pressure changes with time in the simulation process of dam break. Numerical results show that the juSPH software can simulate the dam break process.

Comparing the position of the water column front is an important reference to verify the accuracy of the calculation results. Therefore, the positions of the leading edge of the water column obtained by juSPH and the experiment are compared, and the variation of the dimensionless transport distance of the water column front x/H with the dimensionless time $t(g/H)^{1/2}$ is shown in Fig. 5. It can be seen that it is consistent with the results in the literature [35], which verifies the accuracy of juSPH software.

3.2. Evaluation of the efficiency of juSPH

To evaluate the computational efficiency of the juSPH software, we also introduced a serial SPH program written in MATLAB language based on the same algorithm. Furthermore, for the same size dam-break model, four dam-break models with different numbers of particles are generated, detailed information on the particle discretization of the dam-break model is shown in Table 3.

Since the nearest neighbor particle search was carried out throughout the simulation and took the most time in the SPH



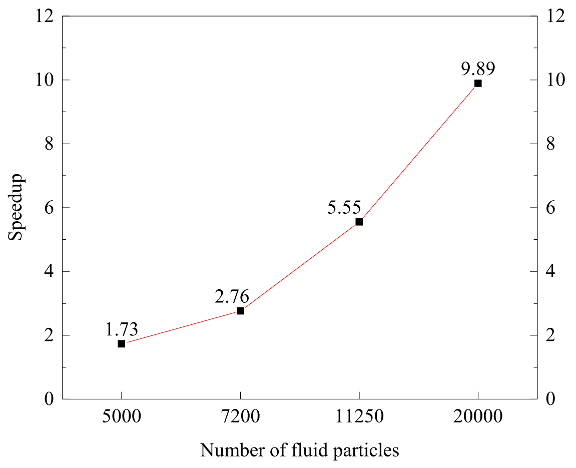


Fig. 6. Left: Comparison of the time to perform one neighborhood search among all particles for fluid particles of the dam break model with different particle numbers in the initial state. Right: The speedup of parallel juSPH on a 24-core CPU to serial juSPH for solving the Euler equation.

Table 1
Specifications of the employed workstation computer.

Specification	Detail
OS	Windows 10 Professional
CPU	Intel Xeon Gold 5118
CPU Frequency (GHz)	2.30
CPU Cores	24
CPU RAM (GB)	128
Julia Version	Julia 1.6.2
MATLAB Version	MATLAB R2013b

Table 2
The specific parameter settings for the numerical model of the dam break.

Parameter	Options
Dimension	2D
Size of the water body (m)	0.6×0.3
The density of water (kg/m^3)	1000
Artificial viscosity coefficient	0.02
Gravity acceleration (m/s^2)	9.81
Smoothing length (m)	0.006
Particle spacing (m)	0.006
Number of fluid particles	5000
Number of boundary dummy particles	1629

simulation, the time for the fluid particles to search for neighborhood particles among all particles was compared at the initial state of each dam break model, and the results of the comparison are shown in Fig. 6 (left). From the above results, it can be seen that for a dam break model consisting of 20000 fluid particles, the serial SPH program written by MATLAB takes about 4.11 s to search for the nearest neighboring particle once, while the parallel juSPH software on a 24-core CPU takes only 0.21 s to search once. The results show that the parallel juSPH software on a 24-core CPU is about 19.57 times faster than the serial SPH program written by MATLAB.

To further illustrate the computational efficiency of the juSPH software, we also compared the time taken to solve the Euler equation required to compute the same dam break model by the SPH program written by MATLAB and Julia language based on the same algorithm, the results of the comparison are shown in Table 3, the speedup of parallel juSPH on a 24-core CPU to serial juSPH for solving the Euler equation is illustrated in Fig. 6 (right). The results show that the parallel juSPH is much faster than the serial juSPH and the serial SPH program written in MATLAB. As the number of fluid particles increases, the time speedup ratio of parallel computation increases as well. When the number of fluid

Table 3
Comparison of the time (/h) required to solve the Euler equation for the four dam-break models using (1) parallel juSPH on a 24-core CPU, (2) serial juSPH on a single-core CPU, and (3) the serial SPH program written in MATLAB.

Particle discrete details			Comparison of the time (/h)		
Particle spacing(m)	Fluid particles	Boundary particles	juSPH Single-core	juSPH 24-core	SPH code MATLAB
0.006	5000	1629	2.61	1.50	15.94
0.005	7200	1944	5.52	2.00	N/A
0.004	11250	2427	17.83	3.21	N/A
0.003	20000	3231	69.21	7.00	N/A

particles is 20000, the parallel juSPH is 9.89 times faster than the serial juSPH on a 24-core CPU.

4. Impact

The juSPH software is a tool dedicated to the large-scale application of SPH simulations designed to explore massively particle parallel techniques. It opens up a new path for researchers to write efficient programs that will be executed on a large scale for fields that could benefit from meshless methods. Its modular design enables researchers to expand it for other field applications. We believe that juSPH can aid researchers in furthering their research; for example, juSPH can be further applied to the field of geological hazard, such as simulating the evolution process of dam break floods and the movement process of diluted debris flow, which is of great significance for disaster scope prediction and rational prevention.

5. Conclusions

This paper presents a Julia-based package of parallel SPH named juSPH, which can easily simulate the movement process of two-dimensional dam-break water flow and flexibly visualize the results of simulation using modern visualization software Paraview. As opposed to various works in the literature, the present package is considered in both code legibility as well as computational efficiency. In addition, the architecture and functions of the juSPH software are very easily modular, and the code is highly readable to facilitate further development.

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

Data will be made available on request.

Acknowledgments

This work was jointly supported by the Fundamental Research Funds for China Central Universities (Grant Numbers: 2652018091), and the National Natural Science Foundation of China (Grant Numbers: 11602235 and 41772326). The authors would like to thank the editor and the reviewers for their valuable comments.

References

[1] Liu MB, Liu GR. Smoothed particle hydrodynamics (SPH): an overview and recent developments. Arch Comput Methods Eng 2010;17(1):25–76. http://dx.doi.org/10.1007/s11831-010-9040-7.
[2] Lucy LB. A numerical approach to the testing of the fission hypothesis. Astrophys J 1977;82:1013–24. http://dx.doi.org/10.1086/112164.
[3] Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. Mon Not R Astron Soc 1977;181(3):375–89. http://dx.doi.org/10.1093/mnras/181.3.375.
[4] Lo EYM, Shao SD. Simulation of near-shore solitary wave mechanics by an incompressible SPH method. Appl Ocean Res 2002;24(5):275–86. http://dx.doi.org/10.1016/s0141-1187(03)00002-6.
[5] Shao SD, Lo EYM. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. Adv Water Resour 2003;26(7):787–800. http://dx.doi.org/10.1016/s0309-1708(03)00030-7.
[6] Gomez-Gesteira M, Rogers BD, Dalrymple RA, Crespo AJ. State-of-the-art of classical SPH for free-surface flows. J Hydraul Res 2010;48(sup1):6–27. http://dx.doi.org/10.1080/00221686.2010.9641242.
[7] Bui HH, Nguyen GD. Smoothed particle hydrodynamics (SPH) and its applications in geomechanics: From solid fracture to granular behaviour and multiphase flows in porous media. Comput Geotech 2021;138. http://dx.doi.org/10.1016/j.compgeo.2021.104315.
[8] Xu R, Stansby P, Laurence D. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. J Comput Phys 2009;228(18):6703–25. http://dx.doi.org/10.1016/j.jcp.2009.05.032.
[9] Antuono M, Colagrossi A, Marrone S. Numerical diffusive terms in weakly-compressible SPH schemes. Comput Phys Comm 2012;183(12):2570–80. http://dx.doi.org/10.1016/j.cpc.2012.07.006.
[10] Sun PN, Colagrossi A, Marrone S, Zhang AM. The delta plus-SPH model: Simple procedures for a further improvement of the SPH scheme. Comput Methods Appl Mech Engrg 2017;315:25–49. http://dx.doi.org/10.1016/j.cma.2016.10.028.
[11] Antuono M, Sun PN, Marrone S, Colagrossi A. The delta-ALE-SPH model: An arbitrary Lagrangian-Eulerian framework for the delta-SPH model with particle shifting technique. Comput & Fluids 2021;216. http://dx.doi.org/10.1016/j.compfluid.2020.104806.
[12] Vanaverbeke S, Keppens R, Poedts S, Boffin H. GRADSPH: A Parallel smoothed particle hydrodynamics code for self-gravitating astrophysical fluid dynamics. Comput Phys Comm 2009;180(7):1164–82. http://dx.doi.org/10.1016/j.cpc.2008.12.041.
[13] Cherfils JM, Pinon G, Rivoalen E. JOSEPHINE: A Parallel SPH code for free-surface flows. Comput Phys Comm 2012;183(7):1468–80. http://dx.doi.org/10.1016/j.cpc.2012.02.007.
[14] Gomez-Gesteira M, Rogers BD, Crespo AJC, Dalrymple RA, Narayanaswamy M, Dominguez JM. SPHysics - development of a free-surface fluid solver - part 1: Theory and formulations. Comput Geosci 2012;48:289–99. http://dx.doi.org/10.1016/j.cageo.2012.02.029.
[15] Gomez-Gesteira M, Crespo AJC, Rogers BD, Dalrymple RA, Dominguez JM, Barreiro A. SPHysics - development of a free-surface fluid solver - part 2: Efficiency and test cases. Comput Geosci 2012;48:300–7. http://dx.doi.org/10.1016/j.cageo.2012.02.028.
[16] Crespo AJC, Dominguez JM, Rogers BD, Gomez-Gesteira M, Longshaw S, Canelas R, et al. DualSPHysics: Open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). Comput Phys Comm 2015;187:204–16. http://dx.doi.org/10.1016/j.cpc.2014.10.004.
[17] Rosswog S. The Lagrangian hydrodynamics code MAGMA2. Mon Not R Astron Soc 2020;498(3):4230–55. http://dx.doi.org/10.1093/mnras/staa2591.
[18] Zhang C, Rezavand M, Zhu YJ, Yu YC, Wu D, Zhang WN, et al. SPHinXsys: An open-source multi-physics and multi-resolution library based on smoothed particle hydrodynamics. Comput Phys Comm 2021;267. http://dx.doi.org/10.1016/j.cpc.2021.108066.
[19] Parmas B, Vosoughifar HR. Novel method of boundary condition of dam-break phenomena using ghost-particle SPH. Nat Hazards 2016;84(2):897–910. http://dx.doi.org/10.1007/s11069-016-2463-1.
[20] Ramachandran P, Bhosale A, Puri K, Negi P, Muta A, Dinesh A, et al. PySPH: A python-based framework for smoothed particle hydrodynamics. Acm Trans Math Softw 2021;47(4). http://dx.doi.org/10.1145/3460773.
[21] Bezanson J, Karpinski S, Shah VB, Edelman A. Julia: A fast dynamic language for technical computing. Comput Sci 2012. http://dx.doi.org/10.48550/arXiv.1209.5145.
[22] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. Siam Rev 2017;59(1):65–98. http://dx.doi.org/10.1137/141000671.
[23] Morris JP, Fox PJ, Zhu Y. Modeling low Reynolds number incompressible flows using SPH. J Comput Phys 1997;136(1):214–26. http://dx.doi.org/10.1006/jcph.1997.5776.
[24] Monaghan JJ. Theory and applications of smoothed particle hydrodynamics. Springer Berlin Heidelberg; 2005, http://dx.doi.org/10.1007/3-540-28884-8_3.
[25] Hu XY, Adams NA. A multi-phase SPH method for macroscopic and mesoscopic flows. J Comput Phys 2006;213(2):844–61. http://dx.doi.org/10.1016/j.jcp.2005.09.001.
[26] Price DJ. Smoothed particle magnetohydrodynamics - IV. Using the vector potential. Mon Not R Astron Soc 2010;401(3):1475–99. http://dx.doi.org/10.1111/j.1365-2966.2009.15763.x.
[27] Wang ZB, Chen R, Wang H, Liao Q, Zhu X, Li SZ. An overview of smoothed particle hydrodynamics for simulating multiphase flow. Appl Math Model 2016;40(23–24):9625–55. http://dx.doi.org/10.1016/j.apm.2016.06.030.
[28] Dominguez JM, Crespo AJC, Gomez-Gesteira M, Marongiu JC. Neighbour lists in smoothed particle hydrodynamics. Internat J Numer Methods Fluids 2011;67(12):2026–42. http://dx.doi.org/10.1002/fld.2481.
[29] Verlet L. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys Rev 1967;159(1):98. http://dx.doi.org/10.1103/PhysRev.159.98.
[30] Monaghan JJ. Smoothed particle hydrodynamics. Rep Progr Phys 2005;68(8):1703–59. http://dx.doi.org/10.1088/0034-4885/68/8/r01.
[31] Adami S, Hu XY, Adams NA. A generalized wall boundary condition for smoothed particle hydrodynamics. J Comput Phys 2012;231(21):7057–75. http://dx.doi.org/10.1016/j.jcp.2012.05.005.
[32] Hu XY, Adams NA. Angular-momentum conservative smoothed particle dynamics for incompressible viscous flows. Phys Fluids 2006;18(10). http://dx.doi.org/10.1063/1.2359741.
[33] Julia 1.7 documentation. 2022, https://docs.julialang.org/en/v1/manual/distributed-computing/.
[34] Paraview. 2022, https://www.paraview.org/.
[35] Lobovsky L, Botia-Vera E, Castellana F, Mas-Soler J, Souto-Iglesias A. Experimental investigation of dynamic pressure loads during dam break. J Fluids Struct 2014;48:407–34. http://dx.doi.org/10.1016/j.jfluidstructs.2014.03.009.