Original software publication

FLeCSPH: The next generation FleCSIble parallel computational infrastructure for smoothed particle hydrodynamics

Julien Loiseau [a,*], Hyun Lim [a,b,c], Mark Alexander Kaltenborn [b,c,d], Oleg Korobkin [a,c], Christopher M. Mauney [c,e], Irina Sagert [b,c], Wesley P. Even [b,c,f], Benjamin K. Bergen [a]

[a] *Applied Computer Science Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[b] *The Computational Physics and Methods Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[c] *Center for Theoretical Astrophysics, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[d] *Department of Physics, The George Washington University, Washington, DC 20052, USA*
[e] *HPC Environments Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[f] *Department of Physical Sciences, Southern Utah University, Cedar City, UT 84720, USA*

ARTICLE INFO

ABSTRACT

FLeCSPH is a smoothed particle hydrodynamics simulation tool, based on the compile-time configurable framework FLeCSI. The asynchronous distributed tree topology combined with a fast multipole method allows FLeCSPH to efficiently compute hydrodynamics and long range particle–particle interactions. FLeCSPH provides initial data generators, particle relaxation techniques, and standard evolution drivers, which can be easily modified and extended to user-specific setups. Data input/output uses the H5part format, compatible with modern visualization software.

Code metadata

Current code version	Version 1.2.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_268
Legal Code License	BSD 3-Clause License
Code versioning system used	git
Software code languages, tools, and services used	C++17, MPI, OpenMP.
Compilation requirements, operating environments & dependencies	OS: Linux/OSX. DEP: FleCSI, HDF5, GSL
Link to developer documentation/manual	github.com/laristra/flecsph/wiki
Support email for questions	flecsph-support@lanl.gov

1. Motivation and significance

FLeCSPH is an open-source distributed smoothed particle hydrodynamics (SPH) code built on top of the Flexible Computational Science Infrastructure (FLeCSI) [1] developed at Los Alamos National Laboratory (LANL). FLeCSI is a task-based runtime abstraction layer that provides a seamless programming model for distributed-memory tasks (Legion [2]), and fine-grained data-parallel kernels (Kokkos [3]), with several core topology types that can be statically specialized to support a variety of applied methods. When using FLeCSI, FLeCSPH has the potential to separate the application implementation from the details of machine architecture, although currently only MPI back end is implemented.

SPH is a mesh-free Lagrangian method that solves the partial differential equations of hydrodynamics by discretizing the flow with a set of fluid elements called particles [4,5]. SPH has several advantages including handling complex geometries and support for true vacuum conditions. Conservation of mass is included by construction, and conservation of linear momentum, angular momentum, and energy can be implemented up to machine precision. Since the quantities are stored in the moving particles, SPH has the advantage of exact and automatic advection. Furthermore, the same tree structure used for determining particle neighbors can be employed for computing gravitational forces, as described for FleCSPH in Section 2. SPH particles can carry the stress history of the material to determine damage, model fracture, and fragmentation of solids [6]. Complicated processes in reacting

---

flows are easily incorporated into an SPH model [7]. Overall, many codes implementing the SPH method have emerged in recent years, attesting to its popularity and efficiency [8–16].

In this paper, we outline the main features of FleCSPH. The initial version of this software is described in [17], while subsequent extensions are presented here.

2. Software description

FleCSPH is written in C++ for UNIX computing and supercomputing platforms taking advantage of modern features of C++ and the Standard Template Library (STL) [18]. CMake [19] provides the build system and FleCSPH can be integrated with Spack [20] to construct build- and run-time environments. Particle data is output in the H5part format [21], compatible with modern visualization software, such as SPLASH [22], Paraview [23], or VisIt [24]. Simulation checkpoint and restart is supported.

2.1. Software architecture

Drivers and initial data generators. The main set of user "apps" are *drivers*, which implement evolution equations, and initial data *generators*, which produce initial particle configurations. Two drivers are provided to evolve hydrodynamics with and without gravity. Users can also iterate on the set of existing drivers for more advanced physics models with different numerical evolution schemes and create custom particle system generators. A suite of initial data generators is provided including: five standard Sod shock tubes, Sedov blast wave, Noh implosion, as well as Rayleigh–Taylor and Kelvin–Helmholtz instability.

Physics functionalities. An SPH "particle" is implemented as a class body_u, templated on the problem dimension and basic floating-point type. 'body' refers to a single body in N-body system and '_u' stands for 'unspecialized.' Elements of this class consist of various particle physical properties, e.g., density and velocity, with standard mutators and accessors to allow customized access control. FleCSPH offers a selection of kernels, two SPH formulations, several viscosity prescriptions, particle relaxation mechanism, external conservative forces, and multiple equations of state (EOSs). The list of options for physics functionalities is described in the developer's notes on the project wiki page. These functionalities are sorted into different C++ header files.

Parameter files. The choice of physical and numerical methods is made at runtime based on the options supplied by an ASCII parameter file with a key–value syntax. The parameter file specifies options such as the number of particles, SPH kernels, and boundary conditions. The complete list of options is located on the wiki page. FleCSPH parameter files are concise and human-readable records of simulation conditions, allowing for simple reproducibility.

Tree topology. FleCSPH uses a hashed tree [25,26], also known as a binary tree, quadtree, or octree in 1, 2, or 3 dimensions, respectively. Space-filling curves are used for the domain decomposition and the hash-table construction. This allows for finding the parent or children of a node in $O(1)$ on average. Both Morton [27] (Z-order) and Hilbert space filling curves are implemented, which show faster computation of keys or better particle locality, respectively.

Fig. 1 diagrams the tree topology. The central data structure of the tree is a hash table. It stores cells, denoted *hcells*, which can represent both a particle and a node. An *hcell* is identified by a binary-string *key* determined from the space-filling curve and the type of cell they represent: for particles, the key is computed directly from their coordinates; for nodes, it is computed based on the position of the node center of mass (CofM). To resolve any key conflicts, each particle is assigned a unique ID. The hashing function to distribute the *hcell* takes the N last bits of the keys, offering a perfect distribution at the bottom and contiguous storage at the top of the tree, which is accessed more often. This representation allows for the parent or children of a branch to be easily located by adding or removing a digit in their key and accessing the hash table, an order $O(1)$ on average operation.

Creating the distributed tree requires several steps. First, the particles are added individually to the tree, and the necessary nodes are refined until the particles are isolated each in their own node. The node is refined into 2^D sub-entities, where D is the dimensionality. After the local trees have been created, information about the top part is exchanged across the MPI ranks. Using a hypercube communication pattern, all the ranks share the useful data resulting in a tree where each unknown node belongs to only one other MPI rank.

FleCSPH implements hydrodynamics with SPH, and gravity with the fast multipole method (FMM). While SPH only computes short-range forces, gravity is a long-range interaction, requiring global communication. Both cases of short- and long-range forces are handled by the same particle tree, but two different types of tree traversals are used: SPH and FMM.

The tree traversal is performed on a group of particles in the same node using the CofM boundary information to prune empty areas of the tree. A list of neighbors per particle is built, and the physics functions are called during the traversal. When encountering a non-local node, the ranks use asynchronous MPI to request missing information from the owner, while continuing the traversal on other particles. When the data arrives, it is added to the tree and used to complete the neighbors list.

2.2. Software functionalities

2.2.1. SPH formulation in FleCSPH

FleCSPH numerically solves Euler equations of ideal fluid in their Lagrangian formulation, expressing conservation of mass, momentum and energy:

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

$$\frac{d\vec{v}}{dt} = -\frac{\nabla P}{\rho} + \vec{g}, \tag{2}$$

$$\frac{du}{dt} = \left(\frac{P}{\rho^2}\right)\frac{d\rho}{dt}, \tag{3}$$

where ρ is density, $d/dt = \partial_t + \vec{v} \cdot \nabla$ is convective derivative, $\vec{v}$ is fluid velocity, u is specific thermal energy, P is pressure, and $\vec{g}$ is an external acceleration. The gravitational acceleration is determined by the fluid self-gravity, an external gravitational field, or both.

For SPH discretization, we use one of the simplest formulations [28]. The Euler equations are discretized with the volume element $V_b = m_b/\rho_b$, and an artificial viscosity term Π_{ab} is added:

$$\frac{du_a}{dt} = \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{1}{2}\Pi_{ab}\right) \vec{v}_{ab} \cdot \nabla_a W_{ab}, \tag{4}$$

$$\frac{d\vec{v}_a}{dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} + \Pi_{ab}\right) \nabla_a W_{ab} + \vec{g}_a, \tag{5}$$

where $W_{ab} = W(|\vec{r}_a - \vec{r}_b|, h_{ab})$ and $h_{ab} = (h_a + h_b)/2$. The density is not evolved but reconstructed from the particle positions:
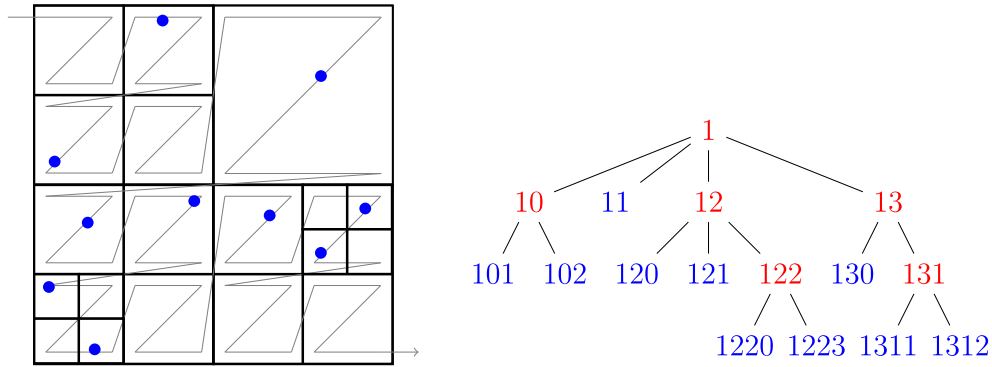
$$\rho_a = \sum_b m_b W_{ab}. \tag{6}$$

Fig. 1. Tree topology representation with the domain decomposition using the Z-order, the tree representation of this domain using associate keys for the entities.

For the viscous stress tensor Π_{ab}, we follow the standard prescription [28,29]. Alternatively, FleCSPH features an implementation of the so-called thermokinetic formulation [28], in which the total particle energy is evolved: $e_a = u_a + \frac{1}{2}v_a^2$. Corresponding discretized version of the energy equation reads,

$$\frac{de_a}{dt} = -\sum_b m_b \left(\frac{P_a \vec{v}_b}{\rho_a^2} + \frac{P_b \vec{v}_a}{\rho_b^2} + \frac{\vec{v}_a + \vec{v}_b}{2} \Pi_{ab} \right) \cdot \vec{\nabla}_a W_{ab}. \qquad (7)$$

Störmer–Verlet algorithm is used for time integration [30].

An adaptive smoothing length h_a allows placing particles where more resolution is needed [28,31,32]. The smoothing length is adapted according to the expression:

$$h_a = C\eta \left(\frac{m_a}{\rho_a} \right)^{1/D}, \qquad (8)$$

where D is dimensionality, C is a kernel-dependent normalization constant, and η is a user-specified number of neighbors [28]. In this method, the number of neighbors for hydrodynamic interactions remains approximately constant for all particles and times.

2.2.2. Computing gravitational force via FMM

FleCSPH uses the FMM approximation [33] to treat gravitational interactions, following [34], up to the first order in the Taylor expansion. Without approximation, pairwise N-body interactions have $O(N^2)$ computational complexity. FMM replaces gravitational forces between individual particles in distant nodes with symmetrized node-node interactions [34]. Nodes are accepted for node-node interaction, if they satisfy the so-called "Minimal Acceptance Criterion" (MAC, see Fig. 2).

Attraction of a distant node is approximated by a series of multipoles. Because of the symmetry in nodal interactions, FMM conserves linear momentum. In the implemented first order of the multipole expansion, angular momentum is conserved exactly.

The gravitational acceleration $\vec{g}_a$ of a particle a is computed as the sum of attractions from all cells passed during the tree traversal while satisfying the MAC angle with respect to the particle:

$$\vec{g}_a = -\sum_{\text{MAC}(A,B)} \frac{GM_B}{|\vec{R}_{AB}|^3} \vec{R}_{AB}, \qquad (9)$$

where G is the Newton's gravitational constant, A is the cell containing particle a, $a \in A$, and B is the other cell, $a \notin B$. $\vec{R}_A$ and $\vec{R}_B$ are the centers of mass for A and B, respectively. $\vec{R}_{AB} = \vec{R}_A - \vec{R}_B$ is the distance vector, and M_B is the mass of B.

When $\theta_{\text{MAC}} = 0$, Eq. (9) reduces to the exact expression for Newtonian interactions between individual particles.

3. Illustrative examples

In this section, we provide several test cases to demonstrate validation and functionalities. Detailed instructions for all test cases can be found in the wiki page.

3.1. Basic hydrodynamics problems: Sod shock tube test

The Sod shock tube is a standard test with a classical Riemann problem with the following initial parameters:

$$(\rho, v, p)_{t=0} = \begin{cases} (1.0, 0.0, 1.0) & \text{if} \quad 0.0 < x \le 0.5 \\ (0.125, 0.0, 0.1) & \text{if} \quad 0.5 < x < 1.0. \end{cases} \qquad (10)$$

This leads to the development of a shock front, which propagates from high-density into low-density regions, and followed by a contact discontinuity. A density rarefaction wave propagates into the high-density region.

Fig. 3 shows a 1D Sod shock tube with 10,000 particles. Density (ρ), pressure (P), specific internal energy (U), and velocity (v_r) are plotted at four different times. The dashed black lines are analytic solutions. Despite small deviations appearing near shock and contact discontinuities, our results agree well with the analytic solution.

3.2. Testing gravity: Stellar oscillations

As a demonstration of the numerical methods for self-gravitating fluids, we present evolution of a stable isolated star in equilibrium. Truncation error in the initial configuration triggers small oscillations of the star, most notably its fundamental radial mode and the first few overtones [35]. These oscillations are damped by the viscosity during the evolution. This test checks consistency and conservation properties for the coupled hydrodynamics and gravity. We compare conservation of energy, momentum, and angular momentum, computed using the FMM approximation, and using the exact pairwise N-body particle interactions with $O(N^2)$ complexity. For the initial data, we solve Lane–Emden equation [36] for polytrope with $\Gamma = 5/3$, $K = 10^{12}$ (in CGS units), and central density $\rho_c = 5.2 \times 10^6 \, \text{g cm}^{-3}$. This results in a polytrope resembling a white dwarf, with mass $0.2\,M_\odot$ and radius 4790 km and period of adiabatic oscillations in the fundamental mode for this kind of polytrope is 7.75 s (see e.g. [35], Table 8.1).

Fig. 4 shows a star discretized with 14,993 equal-mass particles evolved using the thermokinetic formulation. It demonstrates perfect conservation of linear and angular momenta to machine precision. The top right panel shows the relaxed particle configuration. The top left panel displays the evolution of the gravitational energy (normalized to its initial value), for the

```
let Q := empty queue of cell-cell interactions
Q -> push pair (root, root)
while Q is not empty:
    Q -> pop pair of nodes (A, B)
    if A == B:
        find all children of A
        Q -> push all combinations of children pairs (A1, A2)
    else: # A =/= B
        if pair satisfies MAC:
            for all particles in A:
                add attraction from cell B
            end for
        else:
            if radius(A) > radius(B):
                for all children A' of A: Q -> push (A', B)
            else
                for all children B' of B: Q -> push (A, B')
            end if
        endif
    end if
end while
```
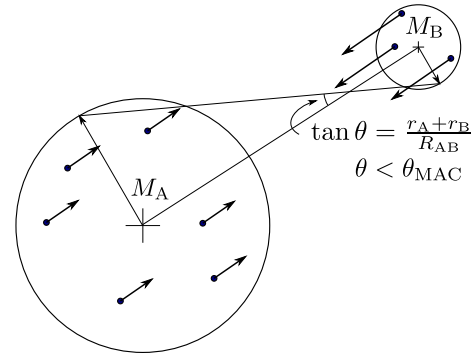
$$\tan\theta = \frac{r_A + r_B}{R_{AB}}$$
$$\theta < \theta_{MAC}$$

Fig. 2. Fast Multipole Method: pseudocode algorithm and an illustration of the symmetric gravitational interaction between two nodes. Node proximity is quantified by angle θ: if $\theta > \theta_{MAC}$, larger node is split up into child nodes to refine the interaction.
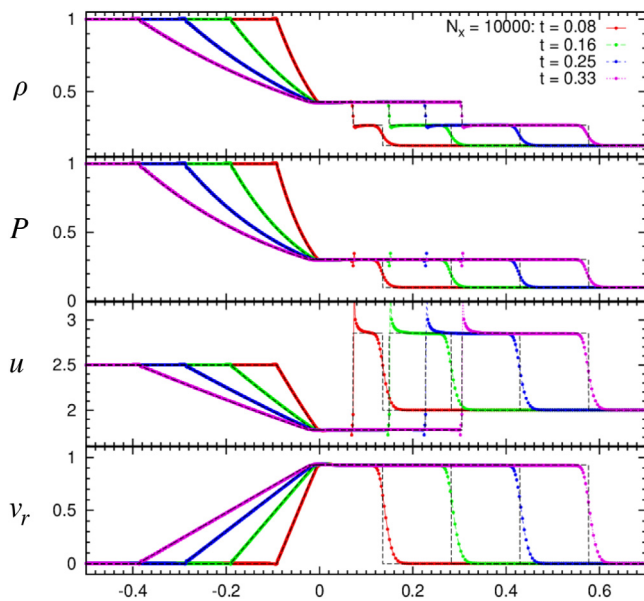

Fig. 3. One-dimensional Sod shock tube with 10,000 particles. Panels, from top to bottom: density, pressure, specific internal energy, and velocity. Each panel contains four different times. Dashed black lines show analytic solution.

exact N-body gravity with pairwise interactions, and FMM implementation with different values of $\tan\theta_{MAC}$. Gravitational energy oscillates with the fundamental frequency. Evolution of the gravitational energy differs for the different values of MAC, but the results approach the N-body case when the MAC angle decreases. Since the $O(N^2)$ algorithm computes gravitational forces exactly, this provides validation of the FMM implementation. The FMM algorithm could be extended to higher orders in Taylor expansion [34,37] to provide better agreement with the exact N-body scheme. However, conservation of angular momentum breaks down for higher orders, and special techniques are needed to recover it [38].

3.3. Integrate example: White dwarf binary

A prominent application for SPH is stellar mergers [39–42]. Here, we present a binary white dwarf(WD) simulation. To set up the system spherical particle distributions are generated for the individual WDs, similar to the setup in Section 3.2. The configurations are then placed on a Keplerian orbit.

Fig. 5 shows four frames of the merging double WD binary. The co-rotating system is composed of 21,295 particles using the zero temperature WD EOS [43] and has an initial period of 40.1 s. In order to limit the simulation time for this example the particle number has been lowered and the orbit is chosen such that the stars immediately being transferring mass and merge within a single orbital period. In nature the stars would begin to slowly transfer mass at a wider orbit and this process would evolve over many orbital periods, but end a qualitatively similar state.

3.4. Scalability

Fig. 6 demonstrates strong and weak scaling of FleCSPH on LANL supercomputing clusters Grizzly and Capulin. Grizzly is an 8-SU cluster running RHEL Linux v.7.7, it has dual socket 2.1 GHz 18 core Intel Broadwell E5 2695v4 processor with 45 MB of cache and 128 GB of RAM on each node. Capulin is an Advanced RISC Machine cluster (ARM) by Cray, Inc., with 56 cores in simultaneous multi-threading (SMT) mode with four threads per core and 256 GB of RAM per node. Scaling runs on Capulin were capped at 32 cores/node.

The oscillating star setup with different numbers of particles was used for both tests (see Section 3.2). For jobs with number of ranks from 1 to 32, we used a single node, and for larger jobs − multiple nodes with 32 ranks per node. This manifested in a drop in strong scaling efficiency on Grizzly between 32 and 64 ranks is due to the inter-node data transfer. On Capulin, efficiency degrades due to increased context switching between SMT threads. For the weak scaling, efficiency drops between 1 and 2 ranks due to parallel overhead, but then remains relatively flat. Overall both SPH and FMM tree traversals show comparable scaling.

Unlike the exact N-body with quadratic complexity, FMM algorithm has linear (or even sublinear) complexity [37]. For FleCSPH, this is shown in the bottom right panel of Fig. 6. The SPH tree traversal is expected to have $O(N \log N)$ complexity, which is satisfied in FleCSPH for the highest number of particles.

The timing of FMM algorithm strongly depends on the MAC angle. Top left panel in Fig. 6 shows the timing of a single FMM tree traversal as a function of MAC. In the extremes, MAC = 0 reduces to the exact N-body, while MAC = 1 corresponds to nodes in contact, i.e., accepting any nodes that do not intersect. The speedup between the extremes for 38,000 particles is almost three orders of magnitude, which will be even higher for larger number of particles. This comes at a moderate price, because using higher MAC reduces code accuracy, as shown in Fig. 4.
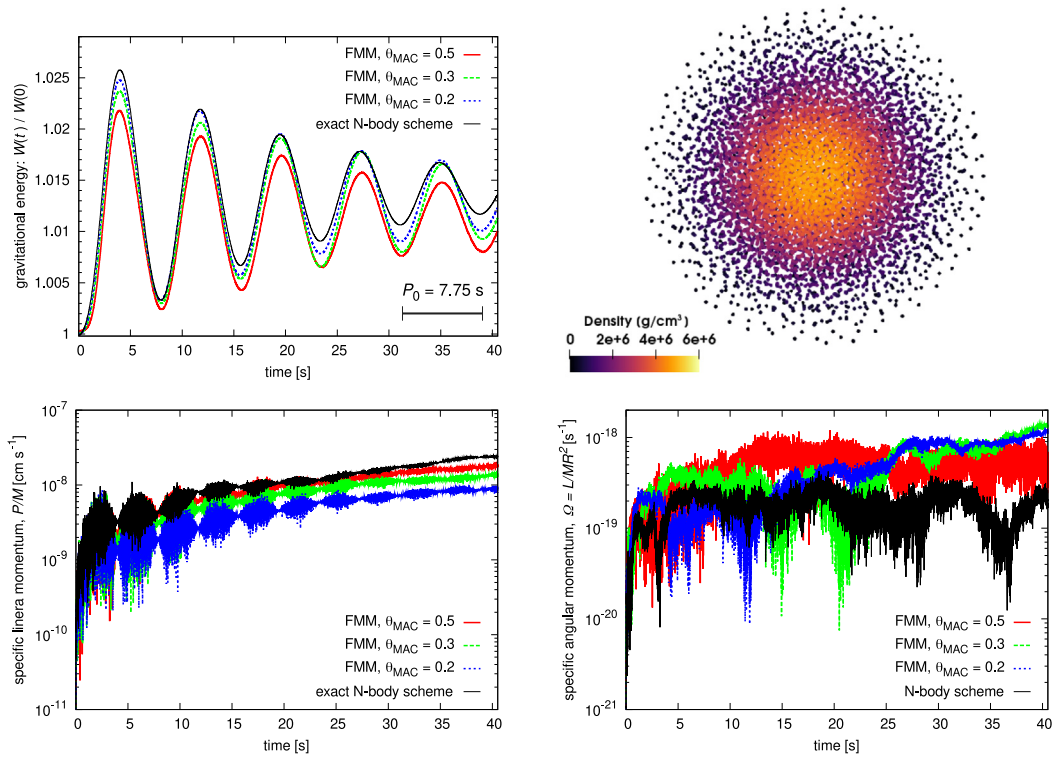
Fig. 4. Oscillations of a star near equilibrium (14,993 particles). Top left panel: gravitational energy evolution for the exact N-body gravity and the FMM approximation with three different MAC values: $\tan\theta_{MAC} = 0.2$, 0.3 and 0.5. Top right panel: 3D rendering of particle positions with ParaView. Bottom row: evolution of specific linear (right) and angular momenta (left).
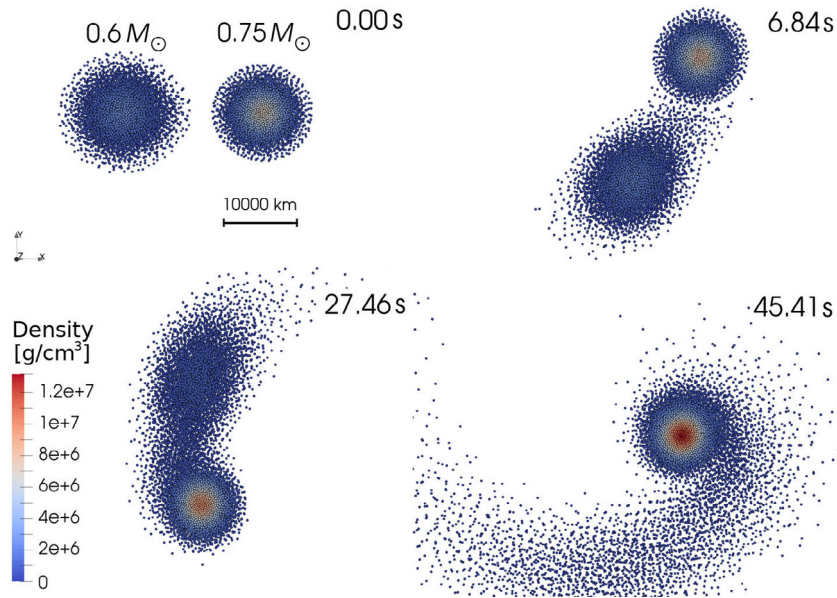

Fig. 5. The density of particles in a binary white dwarf merger.

4. Impact & conclusion

FʟᴇCSPH is designed to be a performance portable particle hydrodynamics simulation tool, oriented to explore modern heterogeneous architectures and massive parallelism. It opens an easy avenue for researchers to write efficient applications that will perform at scale, for those areas that can benefit from mesh-free methods. Its modular design allows users to extend the initial hydrodynamics + gravity suite with a variety of other multi-physics applications. In this work, we demonstrate the structure and capabilities of FʟᴇCSPH using several examples of standard hydrodynamic tests and a few astrophysical applications.

FleCSPH was designed from conception to use the compile-time configurable framework FleCSI, with its functional programming model for execution, control, and data abstractions that are consistent both with MPI and with task-based runtime systems such as Legion (distributed) or Kokkos (node-level). However, the current version of FleCSPH does not take advantage of all features provided in FleCSI: it is limited to only having the support for the
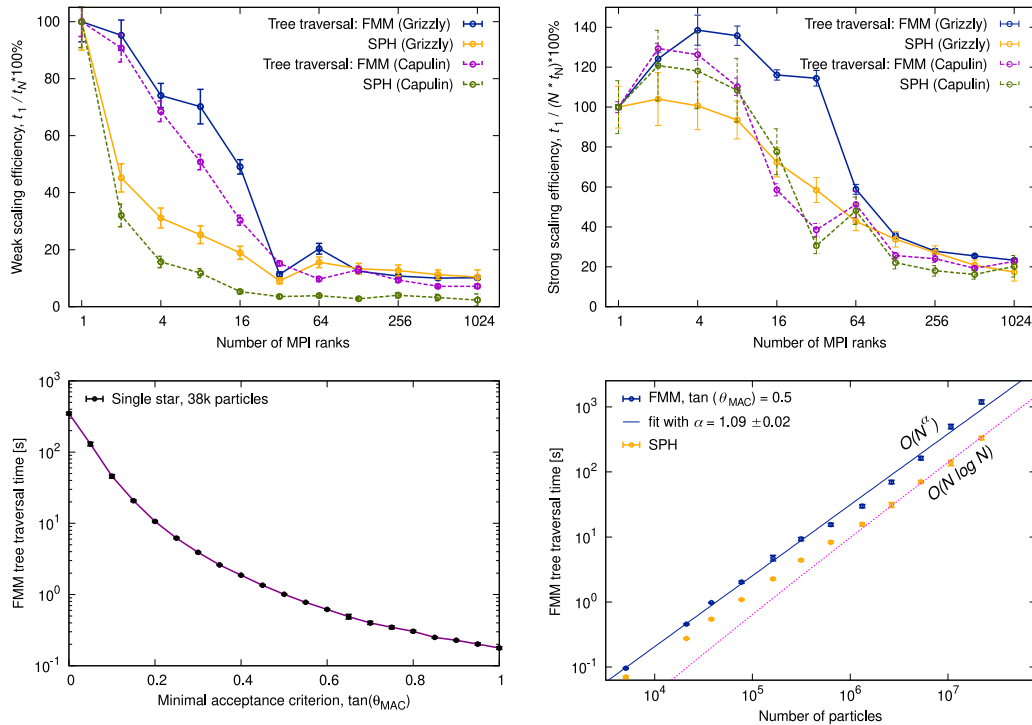
Fig. 6. Top row: weak (left) and strong (right) scaling efficiency for FMM and SPH tree traversal time, on two clusters (Grizzly and Capulin). Weak scaling used 5000 particles per rank and strong scaling used 10^7 particles. Bottom left: time spent per iteration as a function of MAC. Bottom right: timing for a single FMM and SPH tree traversal on a single node as a function of the number of particles N, as well as fits with power law N^α and $N \log N$. Fits demonstrate asymptotic complexity of $O(N)$ for FMM and $O(N \log N)$ for SPH tree traversal implementations.

MPI backend. Future development plans include incorporating more FleCSI functionalities into FleCSPH.

As its nearest goal, FLECSPH will address problems in astrophysics that involve highly irregular morphologies and are sensitive to conservation properties. Examples of such problems include mergers of binary white dwarfs or neutron stars, tidal disruptions of stars by black holes, fallback accretion, planetary impacts, and more. FleCSPH is currently being used by scientist at LANL to address these research topics and is also being used as an educational tool for participants in LANL's computational science and physics student programs.

With the recent advances in SPH techniques, new codes have been developed [9,10,13], bringing the fidelity needed to resolve some of these difficult research problems. FLECSPH will feature performance portability in an open-source environment, which will allow its users to study the big open questions in astrophysics and hydrodynamics at scale. Several natural avenues for future development include new multi-physics applications, such as Lagrangian magnetohydrodynamics [44], radiation transport in the flux-limited diffusion approximation [45], and coupling to other methods such as Monte Carlo method [46].

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.

Acknowledgments

References

[1] Bergen BK, Charest MRJ. Flexible Computational Science Infrastructure (FleCSI): Overview & Application Progress. Tech. rep., Los Alamos, NM (United States): Los Alamos National Lab.(LANL); 2017.

[2] Bauer M, Treichler S, Slaughter E, Aiken A. Legion: Expressing locality and independence with logical regions. In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE; 2012, p. 1–11.

[3] Edwards HC, Trott CR, Sunderland D. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. J Parallel Distrib Comput 2014;74(12):3202–16. http://dx.doi.org/10.1016/j.jpdc.2014.07.003, Domain-Specific Languages and High-Level Frameworks for High-Performance Computing. URL http://www.sciencedirect.com/science/article/pii/S0743731514001257.

[4] Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars.. Mon. Not. R. Astron. Soc 1977;181:375–89. http://dx.doi.org/10.1093/mnras/181.3.375.

[5] Lucy LB. A numerical approach to the testing of the fission hypothesis.. Astron. J 1977;82:1013–24. http://dx.doi.org/10.1086/112164.

[6] Benz W, Asphaug E. Simulations of brittle solids using smooth particle hydrodynamics. Comput Phys Comm 1995;87(1):253–65. http://dx.doi.org/10.1016/0010-4655(94)00176-3, Particle Simulation Methods. URL http://www.sciencedirect.com/science/article/pii/0010465594001763.

[7] Tartakovsky AM, Meakin P, Scheibe TD, West RME. Simulations of reactive transport and precipitation with smoothed particle hydrodynamics. J Comput Phys 2007;222(2):654–72. http://dx.doi.org/10.1016/j.jcp.2006.08.013, URL http://www.sciencedirect.com/science/article/pii/S002199910600372X.

[8] Crespo AJC, Domínguez JM, Rogers BD, Gómez-Gesteira M, Longshaw S, Canelas R, Vacondio R, Barreiro A, García-Feal O. 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.

[9] Rosswog S. The Lagrangian hydrodynamics code MAGMA2. 2019, arXiv e-prints arXiv:1911.13093.

[10] Price DJ, Wurster J, Tricco TS, Nixon C, Toupin S, Pettitt A, Chan C, Mentiplay D, Laibe G, Glover S, et al. Phantom: A smoothed particle hydrodynamics and magnetohydrodynamics code for astrophysics. Publ Astron Soc Aust 2018;35:e031. http://dx.doi.org/10.1017/pasa.2018.25.

[11] Springel V. The cosmological simulation code GADGET-2. Mon. Not. R. Astron. Soc. 2005;364(4):1105–34. http://dx.doi.org/10.1111/j.1365-2966.2005.09655.x, arXiv:astro-ph/0505010.

[12] Ramachandran P, Puri K, Bhosale A, Dinesh A, Muta A, Negi P, Govind R, Sanka S, Pandey P, Kaushik C, Kumar A, Sen A, Kaushik R, Patil M, Tavker D, Menon D, Kurapati V, Sebastian AS, Dutt A, Agarwal A. PySPH: a python-based framework for smoothed particle hydrodynamics. 2019, arXiv e-prints arXiv:1909.04504.

[13] Owen J. An open-source project for modeling hydrodynamics in astrophysical systems. Comput Sci Eng 2001;3(06):54–9. http://dx.doi.org/10.1109/5992.963428.

[14] Cherfils J, 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.

[15] Hérault A, Bilotta G, Vicari A, Rustico E, Del Negro C. Numerical simulation of lava flow using a GPU SPH model. Ann Geophys 2011;54(5):600–20. http://dx.doi.org/10.4401/ag-5343.

[16] Cercos-Pita J. AQUAgpusph, a new free 3D SPH solver accelerated with OpenCL. Comput Phys Comm 2015;192(C):295–312. http://dx.doi.org/10.1016/j.cpc.2015.01.026.

[17] Loiseau J, Lim H, Bergen BK, Moss ND, Alin F. FleCSPH: a parallel and distributed smoothed particle hydrodynamics framework based on flecsi. In: 2018 International Conference on High Performance Computing & Simulation (HPCS). 2018, p. 484–91.

[18] C++ Standards Committee. ISO/IEC DIS 14882, Standard for Programming Language C++ (C++17). Tech. rep., ISO/IEC JTC1/SC22/WG21; 2017.

[19] Martin K, Hoffman B. An open source approach to developing software in a small organization. IEEE Softw 2007;24(1):46–53.

[20] Gamblin T, LeGendre M, Collette MR, Lee GL, Moody A, de Supinski BR, Futral S. The spack package manager: bringing order to HPC software chaos. In: SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE; 2015, p. 1–12.

[21] Adelmann A, Ryne RD, Shalf J, Siegerist C. H5part: A portable high performance parallel data interface for particle simulations. In: Proceedings of the 2005 Particle Accelerator Conference. IEEE; 2005, p. 4129–31.

[22] Price DJ. SPLASH: An interactive visualisation tool for smoothed particle hydrodynamics simulations. Publ Astron Soc Aust 2007;24(3):159–73. http://dx.doi.org/10.1071/AS07022.

[23] Ayachit U. The ParaView Guide: A Parallel Visualization Application. Clifton Park, NY, USA: Kitware, Inc.; 2015, URL https://www.paraview.org/paraview-guide/.

[24] Childs H, Brugger E, Whitlock B, Meredith J, Ahern S, Pugmire D, Biagas K, Miller M, Harrison C, Weber GH, Krishnan H, Fogal T, Sanderson A, Garth C, Bethel EW, Camp D, Rübel O, Durant M, Favre JM, Navrátil P. Visit: An end-user tool for visualizing and analyzing very large data. In: High Performance Visualization–Enabling Extreme-Scale Scientific Insight. 2012, p. 357–72.

[25] Warren MS, Salmon JK. A parallel hashed oct-tree n-body algorithm. In: Supercomputing'93: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing. IEEE; 1993, p. 12–21.

[26] Warren MS. 2HOT: an improved parallel hashed oct-tree n-body algorithm for cosmological simulation. Sci Program 2014;22(2):109–24.

[27] Morton GM. A computer oriented geodetic data base and a new technique in file sequencing. Tech. rep., International Business Machines Company New York; 1966, URL https://dominoweb.draco.res.ibm.com/0dabf9473b9c86d48525779800566a39.html.

[28] Rosswog S. SPH methods in the modelling of compact objects. Living Rev Comput Astrophys 2015;1(1):1–109. http://dx.doi.org/10.1007/lrca-2015-1.

[29] Monaghan JJ. Smoothed particle hydrodynamics. Annu Rev Astron Astrophys 1992;30:543–74. http://dx.doi.org/10.1146/annurev.aa.30.090192.002551.

[30] Rosswog S. Astrophysical smooth particle hydrodynamics. New Astron Rev 2009;53(46):78–104. http://dx.doi.org/10.1016/j.newar.2009.08.007, URL http://www.sciencedirect.com/science/article/pii/S1387647309000487.

[31] Attwood RE, Goodwin SP, Whitworth AP. Adaptive smoothing lengths in SPH. Astron Astrophys 2007;464(2):447–50. http://dx.doi.org/10.1051/0004-6361:20066606.

[32] Cullen L, Dehnen W. Inviscid smoothed particle hydrodynamics. Mon Not R Astron Soc 2010;408(2):669–83. http://dx.doi.org/10.1111/j.1365-2966.2010.17158.x, URL https://academic.oup.com/mnras/article-pdf/408/2/669/18431145/mnras0408-0669.pdf.

[33] Greengard L, Rokhlin V. A fast algorithm for particle simulations. J Comput Phys 1987;73(2):325–48. http://dx.doi.org/10.1016/0021-9991(87)90140-9, URL http://www.sciencedirect.com/science/article/pii/0021999187901409.

[34] Dehnen W. A very fast and momentum-conserving tree code. Astrophys J 2000;536(1):L39–42. http://dx.doi.org/10.1086/312724.

[35] Cox JP. Theory of stellar pulsation. Princeton University Press; 1980, http://dx.doi.org/10.1002/9783527655182.ch5.

[36] Chandrasekhar S, Chandrasekhar S. An introduction to the study of stellar structure. In: Astrophysical monographs, Dover Publications; 1957, https://books.google.com/books?id=VwPLAgAAQBAJ.

[37] Dehnen W. A fast multipole method for stellar dynamics. Comput Astrophys Cosmol 2014;1(1):1. http://dx.doi.org/10.1186/s40668-014-0001-7.

[38] Marcello DC. A very fast and angular momentum conserving tree code. Astron J 2017;154(3):92. http://dx.doi.org/10.3847/1538-3881/aa7b2f.

[39] Rasio FA, Shapiro SL. Hydrodynamics of binary coalescence. II. Polytropes with gamma = 5/3. Astrophys J 1995;438:887. http://dx.doi.org/10.1086/175130, arXiv:astro-ph/9406032.

[40] Dan M, Rosswog S, Guillochon J, Ramirez-Ruiz E. Prelude to a double degenerate merger: The onset of mass transfer and its impact on gravitational waves and surface detonations. Astrophys. J. 2011;737(2):89. http://dx.doi.org/10.1088/0004-637X/737/2/89, arXiv:1101.5132.

[41] Rosswog S, Piran T, Nakar E. The multimessenger picture of compact object encounters: binary mergers versus dynamical collisions. Mon Not R Astron Soc 2013;430(4):2585–604. http://dx.doi.org/10.1093/mnras/sts708, arXiv:https://academic.oup.com/mnras/article-pdf/430/4/2585/3816752/sts708.pdf.

[42] Motl PM, Frank J, Staff J, Clayton GC, Fryer CL, Even W, Diehl S, Tohline JE. A comparison of grid-based and SPH binary mass-transfer and merger simulations. Astrophys J Suppl Ser 2017;229(2):27. http://dx.doi.org/10.3847/1538-4365/aa5bde.

[43] Chandrasekhar S. The highly collapsed configurations of a stellar mass. Mon Not R Astron Soc 1935;95:207–25. http://dx.doi.org/10.1093/mnras/95.3.207.

[44] Price DJ. Smoothed particle hydrodynamics and magnetohydrodynamics. J Comput Phys 2012;231(3):759–94. http://dx.doi.org/10.1016/j.jcp.2010.12.011, Special Issue: Computational Plasma Physics. URL http://www.sciencedirect.com/science/article/pii/S0021999110006753.

[45] Whitehouse SC, Bate MR. Smoothed particle hydrodynamics with radiative transfer in the flux-limited diffusion approximation. Mon Not R Astron Soc 2004;353(4):1078–94. http://dx.doi.org/10.1111/j.1365-2966.2004.08131.x, arXiv:https://academic.oup.com/mnras/article-pdf/353/4/1078/2988902/353-4-1078.pdf.

[46] Oxley S, Woolfson MM. Smoothed particle hydrodynamics with radiation transfer. Mon Not R Astron Soc 2003;343(3):900–12. http://dx.doi.org/10.1046/j.1365-8711.2003.06751.x, arXiv:https://academic.oup.com/mnras/article-pdf/343/3/900/3905202/343-3-900.pdf.