Parallel and Meshfree: new frontiers of CFD

L. A. Barba*

**Department of Mechanical Engineering, Boston University, Boston, MA 02215*

Abstract: For the best part of 50 years, advances in algorithms and the increasing computing power have made computational fluid dynamics, CFD, a mature discipline. Which are some remaining frontiers of the discipline? It is common knowledge that one of the challenges in fluid simulation continues to be the need to straddle many scales. New computational methods still need to be developed that are able to adapt to the many scales of a problem. Another frontier recently opened is the development of hardware-aware software. Multi-core computers are on everyone's desktops nowadays, and a growing trend in using graphics cards and other specialized hardware is buzzing. In all of these frontiers, there is great potential for meshfree methods. Particle-type formulations for CFD offer an alternative which is low in numerical diffusion, devoid of numerical dispersion and stability constraints. Also, meshfree methods offer a natural adaptivity in situations where mesh generation is a huge burden (*e.g.* moving boundaries). Meshfree methods could be especially well-suited to exploit the new hardware technologies entering the scene. I will present an overview of some aspects of meshfree simulation in fluid dynamics, and recent progress in algorithms and parallel implementations, including novel hardware.

Keywords: meshfree, particle method, parallel computing, novel hardware

1 INTRODUCTION

In an often cited complaint of computational fluid dynamics practitioners, the generation of a good quality computational mesh is estimated to be maybe 80% of the cost of a simulation cycle. Perhaps this is today an exaggeration, as the quote has been repeated for years without editing the estimate. Or perhaps not. As computing power has grown exponentially, more and more complex problems are attempted by the community. Presently, many of the cutting-edge results in fluids simulation involve highly irregular immersed shapes or fluid conduits (*e.g.*, in biological fluid dynamics applications), or moving boundaries. In these types of problems not only the generation but the subsequent maintenance of a good mesh can be an onerous part of the job.

Before we begin discussing some recent research progress in the development and implementation of meshfree methods, maybe it is a good idea to agree on some basic terminology. What is a *mesh*? First, it is a means by which a continuum problem is translated into a finite description—but of course meshfree method require this, too. Second, it is a geometric representation of the object(s) that is(are) to be analyzed by the CFD program. Again, meshfree methods need something like this also. What characterizes the mesh-based approach in particular is that the above are provided by a set of nodes and the interconnectivity of these nodes, which has to be maintained. Meshfree approaches aim at utilizing the set of nodes but doing away with the connectivity and its maintenance over time. The simplification that this introduces in the geometric representation of objects and surfaces is substantial.

During the celebration of the 75th anniversary of the Graduate Aeronautical Laboratories in Caltech (November 2003), I had the privilege of meeting Professor Milton Van Dyke[1]. When he asked me what I was working on, and I explained about the efforts to simulate fluid flow without using a mesh, he replied: "It sounds like magic." Today, a critical mass of researchers are demonstrating that magic in a variety of applications, including both fluids simulation and computational mechanics. The list of variations that have been under development is dizzying: from the oldest vortex particle method [4] and smooth particle hydrodynamics method, SPH [11], to diffuse element method [16], element-free Galerkin [3], reproducing kernel particle method, RKPM [15, 14], hp-clouds [9, 10], and a half-dozen more.

It would be too ambitious to try to give a real overview of meshfree methods on this occasion. Instead, I will concentrate on three topics. First, how radial basis function (RBF) methods have been combined with the vortex particle method to produce a fully meshfree system of direct numerical simulation. Second, the importance of clever algorithms to provide efficiency and ensure that these methods are competitive with mainstream CFD schemes—I will focus on

[1] Author of the classic book "An Album of Fluid Motion".

the fast multipole method as a paradigm of $\mathcal{O}(N)$ methods with multi-resolution capability. Third, I will discuss innovations for using new hardware to implement numerical algorithms. This last topic will bring up the possibility that meshfree and particle methods can take better advantage of the massively parallel hardware, such as graphic card accelerators.

2 FULLY MESHFREE VORTEX PARTICLE METHOD FOR FLUID SIMULATION

The vortex particle method is used to simulate incompressible fluid flow based on the vorticity formulation of the Navier-Stokes equation. A particle-like approximation is used on the vorticity field $\omega = \nabla \times u$, as follows,

$$\omega(x,t) \approx \omega_\sigma(x,t) = \sum_{i=1}^{N} \gamma_i \zeta_\sigma \left(x - x_i(t)\right). \tag{1}$$

Here, ζ_σ is the particle basis function, γ_i are the particle weights, and the scattered particles located at the points x_i are convected by the fluid velocity u to satisfy vorticity transport. In this Lagrangian formulation, there is an incremental loss of spatial accuracy as the x_i become disordered. This effect is not catastrophic, but it can quickly bring accuracy down to unacceptable levels. The reason is that as the vortex particles follow the flow strain, they cluster together in some areas and open gaps in other areas, and so they are unable to represent a continuous field. This, added to the fact that convergence proofs for the vortex method relied on the assumption of particle overlap—$h/\sigma < 1$ for h the average particle separation and σ their length scale—, meant that only short-time calculations were advised. This situation changed when an interpolation scheme for particle weights was first introduced [13] in what is now called 'particle redistribution' or 'remeshing'. Particle remeshing allowed for the first time long simulations of high-Reynolds number flow with the vortex method. However, it introduced back the mesh into the numerical system, and with it numerical diffusion and issues related to accommodating irregular boundaries. Nevertheless, many excellent results were produced using vortex methods with remeshing, such as [17, 6].

A fully meshfree alternative, in the sense that nodes can be scattered and no regularity requirement is introduced in the formulation, was introduced in [2], where re-location of the particles is accomplished via radial basis function interpolation. When a function f is only known at scattered points $X = \{x_1, \cdots, x_N\}$, it can be approximated by a linear superposition of radial basis functions (RBFs), $\phi_i = \phi(x - x_i)$, as,

$$f(x) \approx f_\phi(x) = \sum_{i=1}^{N} \alpha_i \phi(x - x_i) \tag{2}$$

The parallel between (2) and (1) is obvious. The crux of the RBFmethod is finding the appropriate values of the weights, α_i, so that the representation (2) approximates well the function f. To find the weights, one uses the known values of the function at the data locations, $f(x_j)$ with $j = 1 \cdots N$, and solves a collocation problem:

$$f(x_j) = \sum_{i=1}^{N} \alpha_i \phi(x_j - x_i) \qquad \text{or} \quad \vec{f} = \Phi_{ij}\vec{\alpha} \tag{3}$$

Solving such a system can be computationally expensive—$\mathcal{O}(N^2)$ if using an iterative method, due to the matrix-vector multiplications with a dense matrix—and is moreover complicated by the fact that the matrix Φ_{ij} is ill-conditioned for most choices of ϕ. Compact support basis functions have been developed to deal with these issues, but sacrifice accuracy and convergence rate, in exchange for computational efficiency.

3 COMPUTATIONAL CHALLENGES

Solving the RBF collocation problem, as described above, involves a linear system with a fully-populated and ill-conditioned matrix. Clearly this is a challenge when N is large. Another classic challenge for particle methods is the calculation of pair-wise interactions, in the case of the vortex method, to obtain the velocity of the particles from the vorticity. Using the Biot-Savart law of vorticity dynamics, the velocity of the particles is given by,

$$u(x_j) = \sum_{i=1}^{N} \gamma_i \mathbb{K}_\sigma(x_j - x_i) \tag{4}$$

where the kernel $\mathbb{K}_\sigma$ is related to the particle basis function ζ_σ via the Biot-Savart integral. A common choice for the particle basis function is a Gaussian:

$$\zeta_\sigma(x) = \frac{1}{2\pi\sigma^2}\exp\left(\frac{-|x|^2}{2\sigma^2}\right), \qquad \text{and} \qquad \mathbb{K}_\sigma(x) = \frac{1}{2\pi|x|^2}(-x_2, x_1)\left(1-\exp\left(-\frac{|x|^2}{2\sigma^2}\right)\right). \tag{5}$$

For this choice, the kernel $\mathbb{K}_\sigma$ does not decay fast enough that it can be neglected in the far field, and thus the evaluation of the velocity (4) requires $\mathcal{O}(N^2)$ operations. This was for quite some time an impediment to the adoption of the particle method, as it was simply not competitive compared with mainstream CFD schemes.

Two types of methods have been developed to more efficiently compute the particle interactions. One relies on interpolating the particle information onto a mesh and solving for the velocity field efficiently on the mesh, to then interpolate the computed field back to the particle locations. This approach is known as the vortex-in-cell method [7, 5]. As in the process of particle remeshing described in the previous section, the vortex-in-cell calculations introduce a mesh into the numerical system. The second method for the efficient computation of the particle interactions relies on a hierarchical decomposition of space, to identify what is 'near' and what is 'far' from an evaluation point, and applies series expansions to represent clusters of particles in the far domain; it is essentially meshfree. The archetype of this second type of method is the Fast Multipole Method, FMM [12].

The FMM algorithm can be rather complex to fully grasp, and difficult to program, which has been a barrier for adoption and arguably diminished its impact. Moreover, parallelization can be quite tricky. To aid in a more widespread adoption of the FMM, we have recently developed PetFMM[2], a parallel fast multipole library implemented within the framework of PETSc[1]. The current version consists of a complete implementation of the FMM in parallel, with the feature of providing load balancing automatically via an optimization approach [8]. A prominent feature of this software is that it is designed to be extensible—the goal is to effectively unify efforts involving several algorithms which are based on the same principles of the FMM. This should be a significant contribution to the meshfree and particle methods community.

Going back now to the problem of solving the ill-conditioned RBF interpolation problem, we also have some significant progress to report. In [18] we developed a domain decomposition approach with an added mean field, designed to take advantage of the rapid decay of the Gaussian basis function. The algorithm was shown to have excellent algorithmic efficiency, converging rapidly to machine precision, and an $\mathcal{O}(N)$ complexity and storage requirement. At this time, the research code for this fast RBF method is available from Google Code[3], but we are initiating the development of a parallel version which will be shared in an open source framework similar to the PetFMM software.

4 HARDWARE ACCELERATION

A new wave in scientific computing has the community buzzing: the use of graphics processors for programmable tasks, with the aim of exploiting their massively parallel architecture. The huge attention that GPUs are attracting stem from the large computing capacity and the low cost of these devices. When GPUs are used as accelerators for CPUs, the hardware configuration is often referred to as a hybrid system. The performance achieved from hybrid systems far exceeds that of today's best X86 processor. Therefore, the potential for order of magnitude speed-ups is enticing many venturesome researchers to attempt implementation of their algorithms for such systems.

Not all problems are massively parallel, however. Indeed, many algorithms do not map well to the GPU architecture. Often 'porting' a code from a serial version is inefficient and complicated. Rethinking algorithms might be required, so that they are more suitable for the new architecture. Because the GPU has been developed for graphics computations, the hardware itself is very different from a normal computer. Programming for the massively parallel GPU is difficult—the number of parallel executions can easily be in the order of several thousand kernels (a kernel is a small function). Imagine that programming in parallel for clusters of computers is already a challenge!

Our approach to the task of implementing problems for systems with GPU acceleration is to think about them as 'heterogeneous computing' applications. The philosophy of heterogeneous computing is reducing the overall execution time by using different types of hardware available within the same system. Commonly, algorithms do not fit only a single model (they can have some parts that require sequential execution, and others that are more suitable for parallel execution), so identifying and exposing parallelism in an algorithm is the first step. Often, the sequential algorithm will need to be re-thought. As a result, it is possible that a less 'efficient' but more parallel algorithm is produced.

[2] PetFMM stands for 'portable extensible toolkit for FMM', as in PETSc, which is 'portable extensible toolkit for scientific computing'.
[3] The code is found at `http://code.google.com/p/pyrbf/`.

For example, in an N-body simulation, where all the particles in the system interact with each other, if the interactions are symmetric—the interaction (a, b) is equal to the interaction (b, a)—a sequential code can take advantage of the symmetry to reduce the operations. In the GPU, the serial optimization based on symmetry is actually more complicated to implement and could also be slower than computing all the interactions directly!

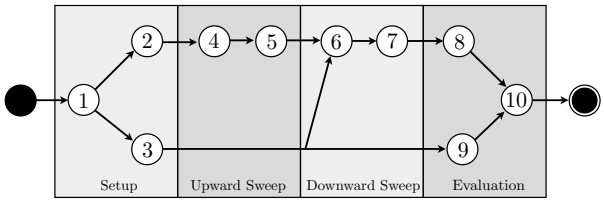

Figure 1. The tasks in the DAG are: ① Tree construction ② Particle clustering ③ Listing of cluster interactions ④ Particle to multipole calculations ⑤ Multipole to multipole translations ⑥ Multipole to local transformation ⑦ Local to local translation ⑧ Local to particle evaluation ⑨ Near domain evaluation ⑩ Adding near and far field contributions.

As mentioned above, the first step is exposing parallelism in the algorithm. In the second step, we implement and distribute the code between the CPU and the GPU. Our goal is to efficiently distribute the work, taking advantage of the fact that CPU and GPU could also work in parallel. As an example, consider the FMM algorithm. If we look at the FMM from a sequential point of view we have four sequential stages: setup, upward sweep, downward sweep, and evaluation. But the algorithm can be further analyzed for more parallelism. We use a Directed Acyclic Graph (DAG) representation of the algorithm to express the finer-parallelism between tasks, as shown in Figure 1. This tool makes distributing the work evenly between the CPU and GPU a more manageable task.

The approach to exploitation of novel hardware just described is currently being used to produce a GPU implementation of the FMM algorithm; this is work in progress. But it also has revealed the following: due to the fact that particle methods are characterized by a higher computational intensity, they have greater potential to take advantage of the massively parallel hardware. In contrast, mainstream mesh-based CFD methods, such as finite difference, are not computationally intensive, and will have a harder time exploiting the novel hardware paradigm introduced by GPUs.

5 CONCLUDING REMARKS

Meshfree methods have been gaining increasing levels of interest in recent years. In computational fluid dynamics, so far the most prevalent meshfree methods are the vortex particle method and smooth particle hydrodynamics methods. An impediment to the wider adoption of these methods for quite some time was the perceived computational expense, in comparison with more mainstream CFD schemes. Currently, efficient algorithms are becoming available that are able to make meshfree methods competitive in terms of computational complexity. Recent progress includes the development of fast solution algorithms for the Gaussian radial basis functions used in vortex methods (in SPH it has become customary to use compact support basis functions, but potentially Gaussians could be used instead with the new efficient algorithms). Drives to produce software libraries to aid in the adoption of meshfree methods, aggregating the experience of many researchers through the open source model, will also contribute to the maturation of the field. As this happens, we will be able to explore with meshfree methods some of the remaining frontiers of CFD:

- The need to straddle many scales in a simulation—particles of variable scale and density can be used quite naturally to resolve different scales in the computational domain.
- Algorithms that are able to detect and adapt to a solution—meshfree methods naturally can adapt. Means of detecting features in a solution remain a challenge.
- Hardware-aware software—meshfree methods could be especially well-suited to exploit the emerging massively parallel architectures, such as GPU.
- The capacity to tackle problems wit complex/moving geometry—enforcing boundary conditions with meshfree methods is still an area where ample progress needs to be made. As this progress is made, the new formulations will be easily extended to complex/moving boundaries, due to the simplified geometry descriptions allowed by the meshfree model.

REFERENCES

[1] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. Curfman-McInnes, B. F. Smith, and H. Zhang. PETSc User's Manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2002.

[2] L. A. Barba, A. Leonard, and C. B. Allen. Advances in viscous vortex methods — meshless spatial adaption based on radial basis function interpolation. *Int. J. Num. Meth. Fluids*, 47(5):387–421, 2005.

[3] Ted Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *Int. J. Num. Meth. Eng.*, 37(2):229–256, 1994.

[4] A. J. Chorin. Numerical study of slightly viscous flow. *J. Fluid Mech.*, 57:785–796, 1973.

[5] G.-H. Cottet. Convergence of a vortex-in-cell method for the two-dimensional Euler equations. *Math. Comp.*, 49(180):407–425, 1987.

[6] G.-H. Cottet and P. Poncet. Particle methods for direct numerical simulations of three-dimensional wakes. *J. Turbulence*, 3:038, 2002.

[7] B. Couet, O. Buneman, and A. Leonard. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *J. Comp. Phys.*, 39(2):305–328, 1981.

[8] Felipe A. Cruz, L. A. Barba, and Matthew G. Knepley. PetFMM — a dynamically load-balancing parallel fast multipole library. To be submitted; preprint available `http://people.bu.edu/labarba/pubs/CruzBarbaKnepley2009.pdf`, 2009.

[9] C. A. Duarte and J. Tinsley Oden. An h-p adaptive method using clouds. *Comp. Meth. Appl. Mech. Engrg.*, 139(1–4):237–262, 1996.

[10] C. A. Duarte and J. Tinsley Oden. h-p clouds—an h-p meshless method. *Numer. Meth. Partial Diff. Eq.*, 12(6):673–705, 1996.

[11] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Notices R. Astron. Soc.*, 181:375–389, 1977.

[12] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[13] P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 296:1–38, 1995.

[14] Wing Kam Liu, Yijung Chen, R. Aziz Uras, and Chin Tang Chang. Generalized multiple scale reproducing kernel particle methods. *Comp. Meth. Appl. Mech. Engrg.*, 139(1–4):91–157, 1996.

[15] Wing Kam Liu, Sukky Jun, Shaofan Li, Jonathan Adee, and Ted Belytschko. Reproducing kernel particle methods for structural dynamics. *Int. J. Num. Meth. Eng.*, 38(10):1655–1679, 1995.

[16] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Comp. Mech.*, 10(5):307–318, 1992.

[17] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Application to the sphere at Re=300, 500 and 1000. *J. Comp. Phys.*, 178:427–463, 2002.

[18] C. E. Torres and L. A. Barba. Fast radial basis function interpolation with Gaussians by localization and iteration. In press: *J. Comput. Phys.*, doi:10.1016/j.jcp.2009.03.007, 2009.