Plug and Play Approach to Validation of Particle-Based Algorithms

Giovanni Lapenta and Stefano Markidis

Los Alamos National Laboratory, Los Alamos, NM 87545, USA
{lapenta, markidis}@lanl.gov

Abstract. We present a new approach for code validation. The approach is based on using particle-based algorithms to simulate different levels of physical complexity. We consider here heat and mass transfer in a multicomponent plasma at the kinetic and fluid level. By representing both levels using particle methods we can design a component based software package, Parsek, to conduct validation using a plug and play approach. With the plug and play paradigm, different components representing different physical descriptions but all based on a common particle algorithm can be interchanged without altering the overall software architecture and the overall algorithm. The advantage of the plug and play approach is that validation can be conducted for each component and switching between physical descriptions requires the validation of just the affected components, not entire codes.

1 Introduction

There exist Particle-based algorithms for every level of description, from the quantum ab-initio molecular dynamics (MD), to classic MD, to kinetic particle in cell (PIC) to fluid particle-based algorithms (e.g. fluid PIC, SPH, vortex methods). While other methods compete with particle-based methods at the macroscopic fluid level, particle-based methods are nearly universally used at the kinetic and MD level.

The availability of a common class of algorithms at all levels of description provides a unique possibility inaccessible to other classes of algorithms. All particle-based algorithms share a common software infrastructure made of particles and interactions (either pair-interactions or interactions mediated by a field discretized on a grid). Using a particle-based approach at all levels it is possible to design a single software package based on a component architecture. All the levels of description are formulated using the same classes of software components (e.g. particle movers or components for particle-particle interactions). Within a single software package it is possible to go across all levels of description simply activating a component in place of another, for example a kinetic particle mover (based on Newton's equations for the particle position and velocity) can be replaced by a fluid particle mover (where the motion is computed according to the local flow speed rather than the particle velocity).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 88–95, 2005.

We term the approach *plug and play validation* because it allows to validate coarser levels of description with finer levels of description by plugging in components and testing them in use.

In the present work, we follow the standard nomenclature [2] of terming the process of *verification* as the task of making sure that the equations of the model are solved correctly and the process of *validation* as the task that the model equations represent with fidelity the actual physics of the system under investigation. Our focus here is on validation alone.

The approach proposed here brings validation to a new level because it not only validates the prediction of coarse levels by testing their results. It also validates coarse levels by directly testing the assumptions behind them (e.g. equations of state or closure relations) validating their assumptions using the information from finer levels of description.

While our focus is on validation, we remark that the approach proposed has also the advantage of simplifying verification because all levels are present within a common software infrastructure and verification can be done component by component. The alternative of using totally different codes and algorithms for the different levels clearly requires a much more extensive effort of verification.

We present a specific example. We consider the heat conduction in a system of multiple ionized gases. The fluids are not in local thermodynamic equilibrium and each has its own temperature and velocity. The fluid closures for this case are not consolidated. We test a model we recently developed based on the 13 moment approach:

$$\mathbf{q}_s = \sum_t \lambda_{st} \nabla T_t \tag{1}$$

where $\mathbf{q}_s$ is the heat flux in species s due to temperature gradients in all other species T_t. The coefficients λ_{st} can be computed from the kinetic model [1]. We will simulate the system at the kinetic and at the fluid level using a single pilot code based on particle methods both for the kinetic and the fluid model. The plug and play approach is used in the design of the pilot code and in the execution of the validation protocol. The process of validation proposed here would determine the validity of the fluid model used (eq. 1) and of the formulation of the transport coefficients λ_{st}.

2 The Physical Challenge

Plasma physics provides an ideal test-bed for developing and testing multiscale, multiphysics algorithms. Plasma physics intrinsically includes two level of multiplicity. First, the plasma species have intrinsically different scales. The electrons are lighter (in a typical hydrogen plasma, the ion mass is approximately 1836 times larger than the electron mass). Second, two types of interactions occur in a plasma: short range and long range. While in a typical gas the interactions are only short range (collisions), in plasmas the presence of charge in the components introduces also a long range force due to collective electromagnetic interactions.

Plasma physics encompasses the whole range of physical scales and models known to mankind. Even not including the topic of quark-gluon plasmas, proper plasma physics ranges from quantum many-body systems to cosmological general relativity models on the structure of the universe. We propose that the approach described here can indeed in principle cover the whole spectrum of physics at these two ends and everything in between. Of course realistically, we need to limit the scope here. And we decide to limit the scope to just two levels of description: the kinetic level and the fluid level.

At the kinetic level, a N-body system is described through the evolution of the distribution function in phase space. The classical formalism for this level of description is the Boltzmann equation.

At the fluid level, a N-body system is described through the evolution of the moments of the distribution function, such as density, velocity and temperature. A great variety of fluid models can be derived from the kinetic level taking a different number of moments of the distribution.

A well known feature of fluid models and their derivation from kinetic models is the need for closure relations. The moment formulation does not form a closed system of equations per se. The equation for the evolution of a moment of any order generally involves higher order moments. Relationships expressing higher order moments in terms of lower order moments are called *closure relations* and are needed to turn the fluid model into a well posed closed problem.

Examples of closure relations are the equations of state: $p = p(n, T)$ and relationships expressing the heat flux or the stress tensor in terms of lower order moments. A classic example is the Fourier law: $q = -k\nabla T$, generalized in our case to eq. (1).

In the present work we consider an initially uniform system with density $n_s = 1$ for all species s, at rest $\mathbf{v}_s = 0$. But we assume that the system is not in thermodynamic equilibrium and each species is initially set up with a uniform temperature gradient. The electrons represent a neutralizing background.

A complete description of the physics and of the mathematical models used can be found in Ref. [1]. The kinetic model can be formulated with the Boltzmann equation:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} + \frac{q_s}{m_s}\mathbf{E} \cdot \frac{\partial f_s}{\partial \mathbf{v}} = St(f_s) \tag{2}$$

for each species s, including electrons and ions. Charge imbalance could be created in the transient evolution, but we assume that magnetic fields are neglected. This latter assumption is not realistic in most laboratory experiments but it is acceptable within the scope of the present work.

The fluid model is based on a continuity, momentum and energy equation for each species. We use the standard textbook formulation of multiple fluid models [3]. However, we investigate the issue of how many moments are required to represent the physics correctly and what closure relations can be used. We base our fluid analysis on our recent work on this topic [1] and on the classic 13-moment approach [3].

The model poses a formidable challenge, a challenge that we try to address with particle-based algorithms.

3 Particle-Based Method Meets the Challenge

We consider three classes of particle-based methods. We categorize them according to the level of physical description.

First, we consider molecular dynamics (MD), MonteCarlo (MC), or generally particle-particle (PP) methods, typically based on a direct mathematical representation of the physical system. In this class of methods, the particles represent a physical entity: a particle, a molecule or a cluster of atoms and molecules (such as a nanoparticle or a subunit of it). The interactions among the computational particles mimic as best we know the actual physical interaction among the real physical objects.

Second, we consider kinetic particle-mesh (PM) methods, often referred to as particle in cell (PIC) methods. This class differs from the PP approach in two ways: 1) While the PP approach uses the direct simulation of N-bodies through their fundamental laws of motion, the PM approach studies the N-body problem at the kinetic level, as described by the Boltzmann transport theory of classical or relativistic N-body systems. In PM, the computational particle represents an element of phase space. 2) While the PP approach handles only particles and their pair interactions, the PM approach introduces average fields and for their discretization introduces a computational mesh.

Third, we consider fluid particle-based methods (fluid PIC, SPH, vortex methods, ...). While the data structure and the software engineering issues typical of fluid PIC are similar to kinetic PIC, the mathematical and physical bases for it are entirely different. In fluid PIC, the computational particle represents an element of the fluid (or solid) in the coordinate space (rather than of the phase space) and its significance is similar to that of a fluid element in a common Lagrangian scheme. Indeed, the fluid PIC method can be viewed as a peculiar implementation of the arbitrary Lagrangian Eulerian (ALE) approach, widely used in industrial engineering applications.

In our study PP approaches are used to handle short range interactions at the kinetic level, PM methods are used to treat long range interactions at the kinetic level. The combined used of PP and PM methods is often referred to as PPPM or P^3M. Fluid PIC is used to handle the fluid description.

4 Plug and Play Approach to Validation

We can identify two primary advantages of particle-based methods in addressing the issue of validation.

First, there is a direct relationship of particle-based algorithms with the physical system under investigation. Computational particles can be directly linked

with physical observables (particle themselves in PP, elements of the phase space in PM and parcels of fluid in fluid PIC). Furthermore, the interactions among the particles (either pair interactions or field-mediated interactions) are direct mathematical constructs of actual measurable interactions in the real world.

Second, particle-based algorithms use the same data structure and the same software architecture to treat all different levels of description, from the fundamental quantum level of ab-initio quantum MD, to classical PP methods, to kinetic PIC, to fluid PIC. All levels of description of N-body systems are covered by particle-based algorithms. No other class of numerical algorithms spans the same range. Based on the two properties above, we can propose that particle-based algorithms provide a unique prototyping environment for Validation & Verification (V&V).

We propose a line of investigation closed on the particle-based algorithms themselves and investigates for a specific system the limits of validity, the fidelity and the predictive potency of different levels of description. Having, as particle-based algorithm do have, the most fundamental level of description (kinetic for weakly coupled systems and molecular dynamics for strongly coupled systems), one has the exact correct theoretical answer based on our most complete and modern understanding of the physics world. By reducing the complexity and stepping down the ladder of further approximations and simplifications, we can address the question of "what is the simplest model to capture a given process".

Within the particle-based approaches, this can be accomplished without changing the overall software architecture, with as simple a switch as changing for example the equations of motion from relativistic to classic, or the field equations from fully electromagnetic to electrostatic. We propose here a plug and play component based software architecture where the properties of the particles and of their interactions through pair forces or through fields can be exchanged by replacing a software component with a compatible one based on a different level of sophistication.

No doubt, this has a strong impact on the cost of the simulations or on the time scales and length scales involved. But if a system is chosen where the most fundamental approach is feasible, stepping down the ladder of approximate models will be progressively easier and more feasible. We note that the plug and play approach just outlined is not feasible with other non particle-based approaches. For example Eulerian methods would not have an equivalent component to be replaced with at the kinetic level (at least a mature one) and even less so at the molecular dynamics level. In that case, the plug and play approach would not be feasible and different codes would need to be used instead making the V&V much more difficult and time consuming, as all parts of the code would need V&V, not just the new component plugged in. Different software architectures would be involved, adding a level of uncertainty and clouding the issue. Particle-based methods allow the plug and play approach primarily because they work as the physics system work allowing a direct comparison with them.

4.1 Specific Example

Let us consider an example: the formulation of heat conduction in hydrodynamics simulations. We propose to start at the fundamental kinetic level with a PIC simulation (assuming a weakly coupled system, but the same approach would function by starting from a PP approach for high energy density systems) where a first principle approach provides the exact heat transfer by taking the appropriate moments of the particle distribution function.

The heat conduction is defined as the third order moment of the particle distribution function:

$$q_{hs} = \frac{m_s}{2} \sum_{k=1}^{3} \int c_{sh} c_{sk} c_{sk} f_s d\mathbf{v}_s \tag{3}$$

where $\mathbf{c}_s$ is the random velocity (difference between actual velocity and average fluid velocity) for species s, h and k label the coordinates.

From a kinetic description, the heat flux can be computed directly using eq. (3).

In the fluid model, heat flux is computed using the 13-moment fluid closure and is defined as in eq. (1) where the transport conefficients λ_{st} are provided by theoretical estimates [1].

In our plug and play validation approach, we replace the components for kinetic description with components for fluid descriptions based on different level of sophistication and we test not only whether the predictions are still valid, but also whether the assumptions behind each reduced model is correct. Moreover, this can be done one component at a time, focusing on the effect of only replacing one component within a common code without worrying at the same time with having replaced a whole code with another. The approach proposed is a striking alternative to the classic approach of just validating closure relations by testing their predictions when applied in existing fluid codes. Here besides doing that, we also test directly the assumptions behind the closures, thanks to the availability of the more fundamental kinetic (or MD) level. This provides information on what is not valid in certain closures allowing us to improve them.

5 Implementation in Parsek

To implement the plan discussed above and demonstrate the feasibility of the plug and play approach for the validation of particle-based algorithms, we have developed a component based package called Parsek.

Parsek uses components for each of the four classes of components: particles, fields, interactions between particle and interactions between fields.

Each component is implemented in an object oriented language. We considered here Java and C++. Two approaches have been tested for object orientation: coarse grained or fine grained [4].

In the fine grained approach each object corresponds to the smallest physical entity: particle or cell. In the coarse grained approach each object represents

a population of the smallest physical entities: particle species or grid. The fine grained approach is composed of small objects and uses arrays of objects (or pointers to them) to describe the whole population of them (species for the particles and grid for the cells). The coarse grained approach, instead, uses arrays of native double precision numbers to represent the population and the whole population is the object. Figure 1-a illustrates the difference between these two approaches. The great advantage of the latter approach is the avoidance of arrays of objects (or pointers to them), resulting in a more efficient use of the cache on processors such as the Pentium family.

Figure 1-b proves indeed that a significant advantage can be gained by using a coarse grained approach. A similar study has been conducted in a previous study using similar Java implementations [4] and is reported in Fig. 1-b. For our pilot code Parsek, we settled on the coarse grained approach.

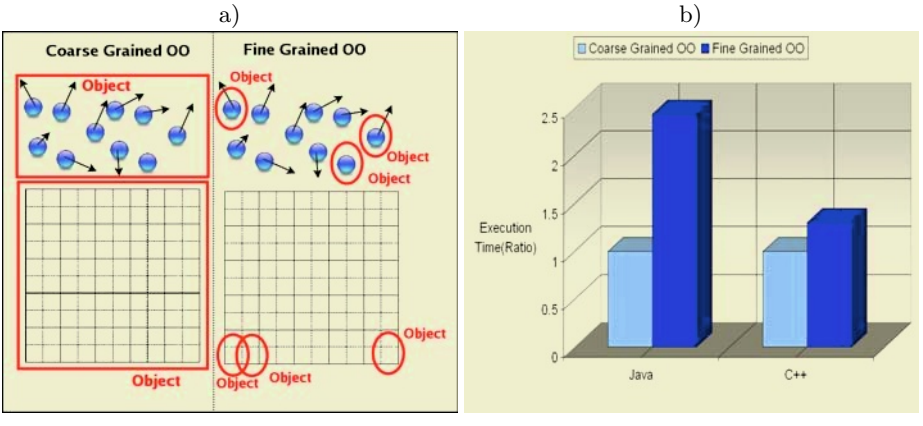

Fig. 1. Coarse grained and a fine grained object oriented versions (a). Comparison of fine and coarse object orientation performance in Java and C++ (b)

The computational challenge posed by the task under investigation (and particularly by the kinetic level) requires the use of a fully parallelized version of the algorithms. We adopted a 3D domain decomposition where portions of the systems including the cells and the particles in them are apportioned to a processor. The requirement to keep the particles and the cells belonging to the same physical domain in the same processor is due to the need to interpolate the information between particles and cells to compute the interactions. A consequence of this choice is the need to dynamically allocate the particles to the processors as the particles cross domain boundaries. In the examples considered here the system is 1D and is obtained as a limit of the 3D case by taking only once cell in the y and z direction and applying periodic boundary conditions.

We have considered two parallelization environments: message passing and multithreading. Message passing is a widely used programming paradigm and it is based on the use of libraries such as MPI. We have developed a version

of our software package, Parsek, in C++ using the MPI library to handle the communication. Figure 2 shows the scaling obtained in a typical problem by the C++/MPI version of Parsek. The test were conducted on a distributed memory cluster of INTEL Xeon processors.

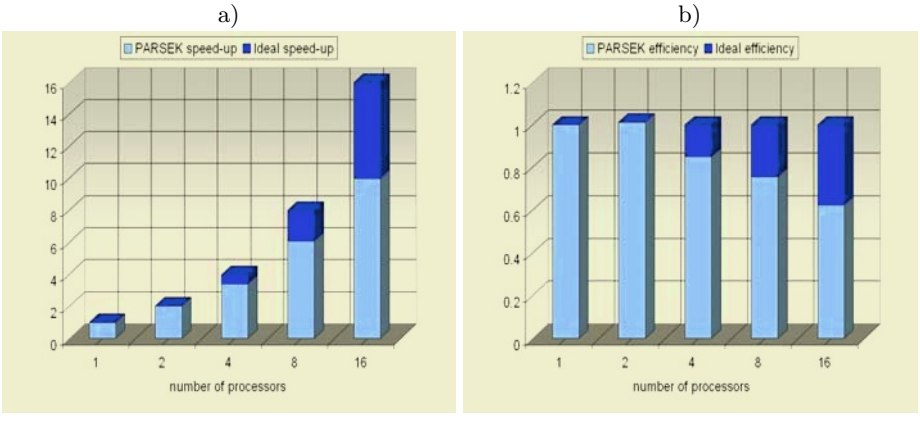

Fig. 2. Parallel speed-up (a) and parallel efficiency (b) of a C++ version with parallelization based on the MPI library. Study conducted on a distributed memory cluster

As an alternative, we considered also the multithreading approach typical of grid-based languages such as Java. The tests (not reported here) were conducted on a shared memory four processor SUN workstation and the observed scaling were very efficient. However, when the same approach is attempted on a cluster machine, native Java multithreading cannot be used and tools such as Java Party need to be used [5].

References

1. L. Abrardi, G. Lapenta, C. Chang, *Kinetic theory of a gas mixture far from thermodynamic equilibrium: derivation of the transport coefficients*, LANL Report, LA-UR-03-8334.
2. S. Schleisinger, R.E. Crosbie, R.E. Gagne, G.S. Innis, C.S. Lalwani, J. Loch, R.J. Sylvester, R.D. Wright, N. Kheir, D. Bartos, *Terminology for Model Credibility*, **Simulation**, 32:103, 1997.
3. J.M. Burgers, *Flow equations for composite gases*, Academic Press, New York, 1969.
4. S. Markidis, G. Lapenta, W.B. VanderHeyden, Z. Budimlić, *Implementation and Performance of a Particle In Cell Code Written in Java*, **Concurrency and Computation: Practice and Experience**, to appear.
5. N. T. Padial-Collins, W. B. VanderHeyden, D. Z. Zhang, NM E. D. Dendy, D. Livescu ,*Parallel operation of cartablanca on shared and distributed memory computers*, **Concurrency and Computation: Practice and Experience**, 16(1):61-77, 2004.