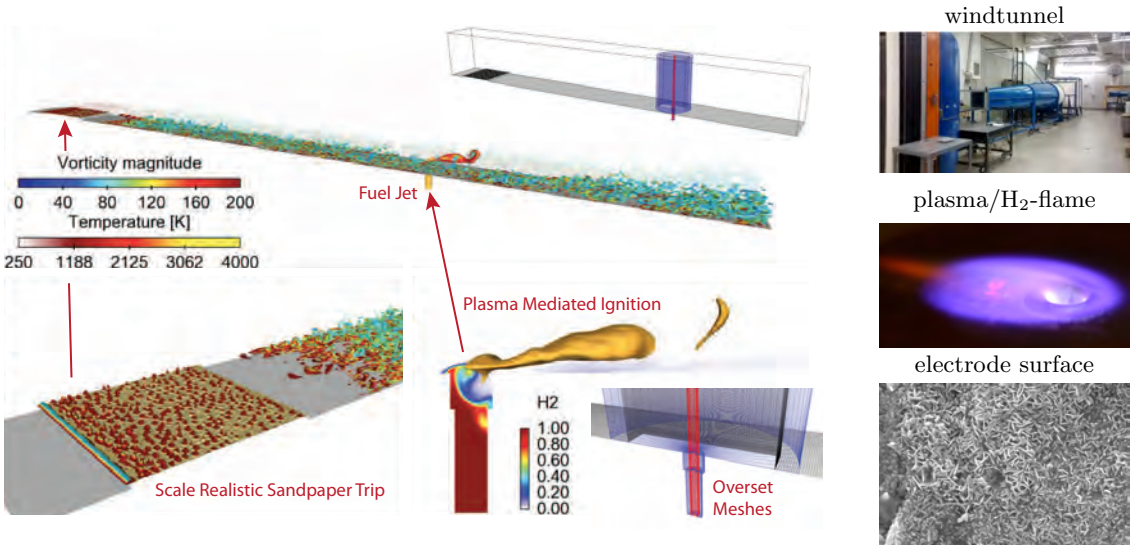
Multi-physics Turbulence Simulation: A Holistic Approach to Performance

J. B. Freund, D. J. Bodony & W. D. Gropp
XPACC: The Center for Exascale Simulation of Plasma-coupled Combustion
The University of Illinois at Urbana–Champaign
{*jbfreund,bodony,wgropp*}*@illinois.edu*

In some sense, simulating just turbulence is now easy, at least for Newtonian fluids in simple geometries. The governing equations are effectively exact, and modern computers can readily represent sufficient scales for many applications. However, maintaining performance in complex geometries and for physics-coupled turbulence is decidedly non-trivial, especially with increasing hardware complexity. Yet the benefits of performance are obvious for predictive simulations with high fidelity and the added value of quantified uncertainty. Performance can also reduce both the dollar cost and energy use of simulation. We feel that achieving performance with complex code on modern and forthcoming systems will require a holistic approach, in which performance is a key consideration at every stage from model selection and discretization through the use with uncertainty quantification. Importantly, strategies must also not restrict the utility for the computational scientists designing and using code. We use our efforts in XPACC, our PSAAP2 center on predictive simulation of plasma-coupled turbulent combustion, to discuss and motivate our integrated approach to performance.

Our discretization is based on intercommunicating **overset structured meshes**, which was selected to balance utility, performance, and fidelity. The regularity of the mesh-associated stencil-based operators aids mapping floating-point operations to computational units, and structured meshes easily support high-resolution non-dissipative discretizations without the chore of global mesh generation. The implementation is also designed for flexible performance. The discrete operators (*e.g.* derivatives) are defined to expose opportunities for concurrency, decomposition, and scheduling inherent in governing equations.

We feel that having an **explicit performance model** is essential for optimization and algorithm selection. The earliest scale-resolved turbulence simulations in the 1980's utilized hardware with remarkable efficiency in part because they had a performance target, albeit a relatively straight-



Plasma mediated ignition of a fuel jet in a turbulent boundary layer: physics-coupled turbulence simulation with overset meshes and the experimental facility at different scales.

forward one for the simple algorithms and machines of the day. An explicit performance model is harder to craft but is even more important for today's simulations. Without guidance, improvements will be haphazard at best, with many missed opportunities. To be useful, such a model starts from the operations needed for the calculation. Empirical measures, such as parallel scaling, can be simple to evaluate but provide at best an incomplete measure of true performance, and it is well understood that poor absolute performance can actually benefit scalability! Cache utilization and fraction-of-peak performance metrics are more direct, but do not necessarily show that the numerical discretization was implemented efficiently in the first place or how to improve it. Furthermore, corresponding empirical measures might be harder to use or not exist on future architectures.

Realizing performance by exploiting concurrency, vectorization, scheduling, and other opportunities will involve highly specialized code, which is impractical to write by hand for complex algorithms. Our strategy is therefore based upon **source-to-source transformation** of natural 'golden copy' code—either legacy or new code designed more-or-less within the current experience of the computational scientist—to expose parallelism, scheduling opportunities, vectorization, and memory locality with tiled data structures. These transformations can thus continue to leverage the best available compilers for any particular system. Additional tools will pursue runtime recompilation, again leveraging available tools when possible, and load balancing via fine- and coarse-grained overdecomposition for additional performance gains.

A key challenge is anticipating the best strategies to pursue, especially since what success 'looks like' for different algorithms will undoubtedly evolve with algorithm and hardware development. Because of this, we are pursuing a evolutionary approach based on a **suite of interoperable tools**. Importantly, the tools remain optional: the 'golden copy' source code minimally requires only mature technology (just Fortran2003 and MPI for our code *PlasComCM*). Thus, there is little barrier to adoption, tools can be selected (or not) incrementally based on their utility, other available tools can be integrated into the suite, and there is no single point of failure in the overall approach. Flexible annotations are being designed so that a knowledgeable user can add additional information (also optional) to the 'golden copy' to inform subsequent transformations.

Prediction uncertainty will remain a concern, especially when more physics is added atop the complexity of turbulence, and is part of the overall performance challenge. For standard turbulence, the general rule is simple: more resolved scales implies more fidelity. There is no similarly simple guideline for multi-physics coupled turbulence. Uncertainty quantification is therefore expected to become an increasingly common component of turbulence simulations. This, of course, adds value to any prediction. It can also lead to huge computational savings by indicating what expensive physics models are needed for any particular quantity of interest. Furthermore, **UQ itself can also be exploited for performance**. Since it invariably requires repeated 'function' evaluations, its inherent redundancies potentially expose additional performance opportunities (*e.g.* vectorization, scheduling, concurrency) that can be harnessed by transformation tools.

In summary, we are pursuing performance at every level, from the models, to the turbulence-resolving discretizations, to their implementation, and on to their use within a UQ framework. Explicit performance models are essential to define and guide progress, and an evolutionary process will have the advantage of being both general and robust in the face of both growing code and hardware complexity. Thus, we are developing multiple and optional interoperating tools that can be applied within current coding practices. These tools will realize diverse performance opportunities, while mitigating risk, leveraging existing code and other tools, and minimizing the burden of adoption.

This material is based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number DE-NA0002374.