Software Architecture Design for Modular Multiphysics Simulations

Ayden Kish[1], John Shaw[1], Michael Lavell[1], Andrew Sexton[1], and Adam Sefkow[1]

[1]The TriForce Center for Multiphysics Modeling, a collaboration between the Departments of Mechanical Engineering, Physics, Computer Science, and the Laboratory for Laser Energetics

Summary

- Achieving the long-term objectives of the TriForce computational environment will require careful attention to software architecture and design from its earliest stages
- The TriForce Fundamental Algorithm Testing Environment (TFFate) is being developed as a stand-alone application in Python to serve as a framework for software architecture prototyping and algorithm development
- Inspired by concepts in general code design and software architecture [1], the principle of an "architectural hierarchy" specific to scientific code development is proposed
- Extensive use of Abstract Base Classes (ABCs) allows for implementation within the architectural hierarchy without sacrificing user extensibility [2]

Objectives of TFFate

For TriForce:

- Provides a staging ground in Python for the addition of new functionalities and algorithms to the primary TriForce computing environment in C++
- Provides a prototype for the long-term architecture of the TriForce computing environment

For the Learner:

- The flow of the code and the role of its parts can be understood using only high-level knowledge of the simulation archetype (PIC, FCV, SPH, etc.)
- Individual parts of the code can be read and understood without detailed knowledge of the rest of the code

1. Simplicity of form without loss of function

For the Developer:

- Individual parts of the code can be added to or modified with as few changes as possible
- Extensive testing suite, along with test-driven development streamlines bug squashing

2. Simplicity of change without loss of stability

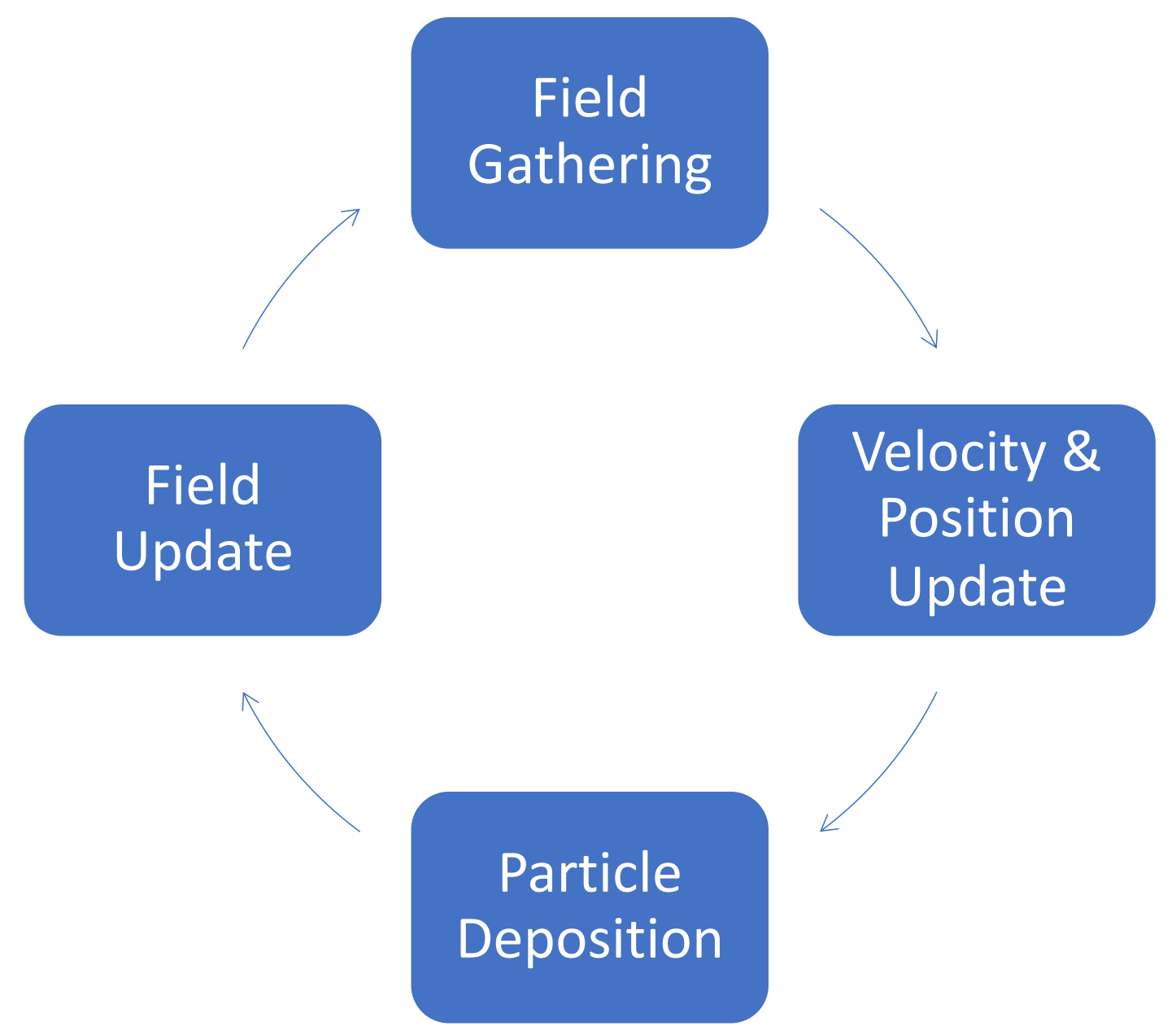
For the User:

- Existing input decks can be read, understood, and modified easily
- For experienced users, simplicity of control does not mean loss of granularity

3. Simplicity of control without loss of granularity

PIC Archetype

Core Particle-in-cell (PIC) Loop



Field Gathering:

- Interpolating field values from the grid to the particles to calculate electromagnetic forces

Velocity & Position Update:

- Integrating the Newton-Lorentz equations of motion to update particle v(t) and x(t)

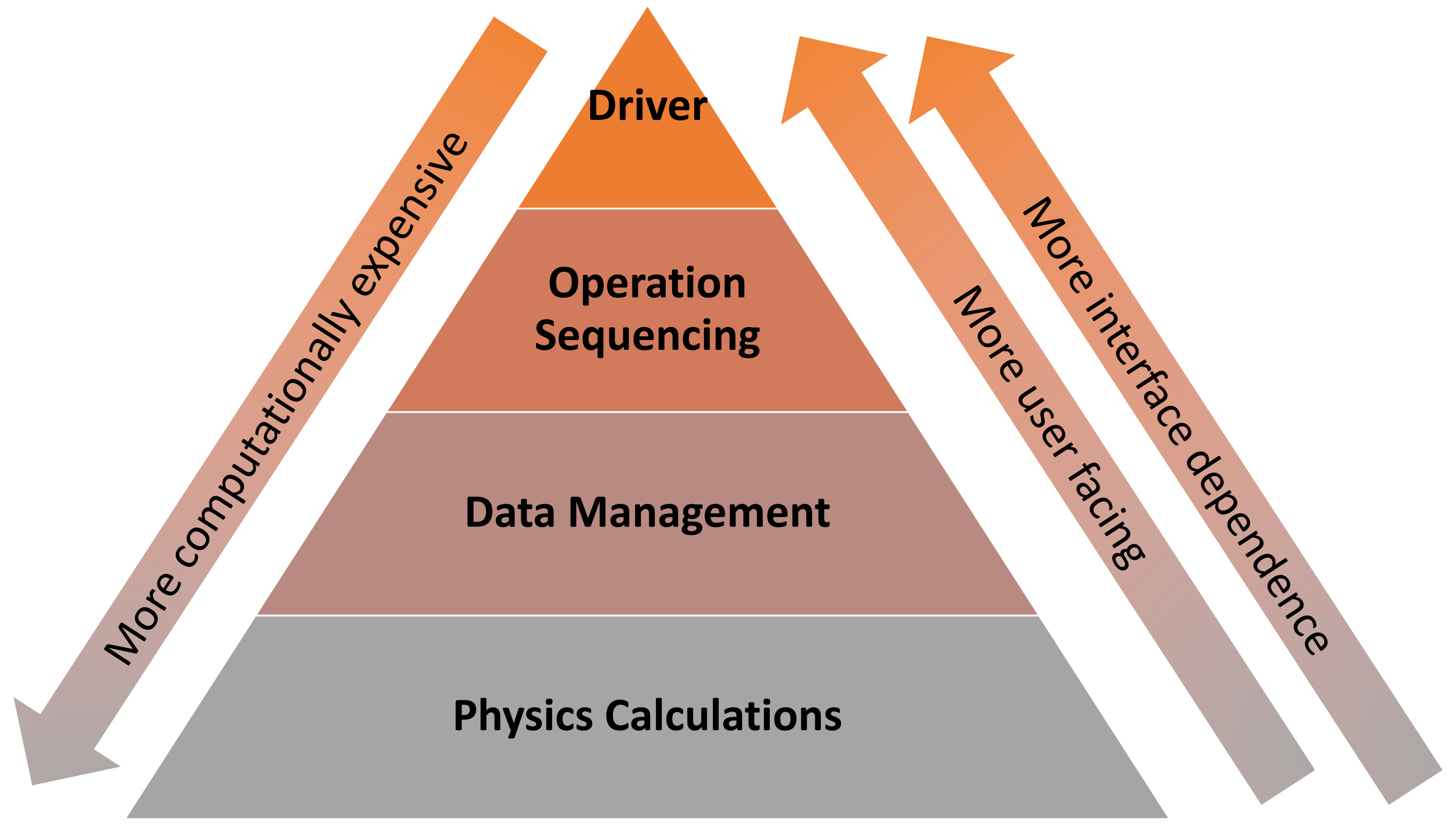
Particle Deposition:

- Interpolating particle charge/current density contributions to the grid

Field Update:

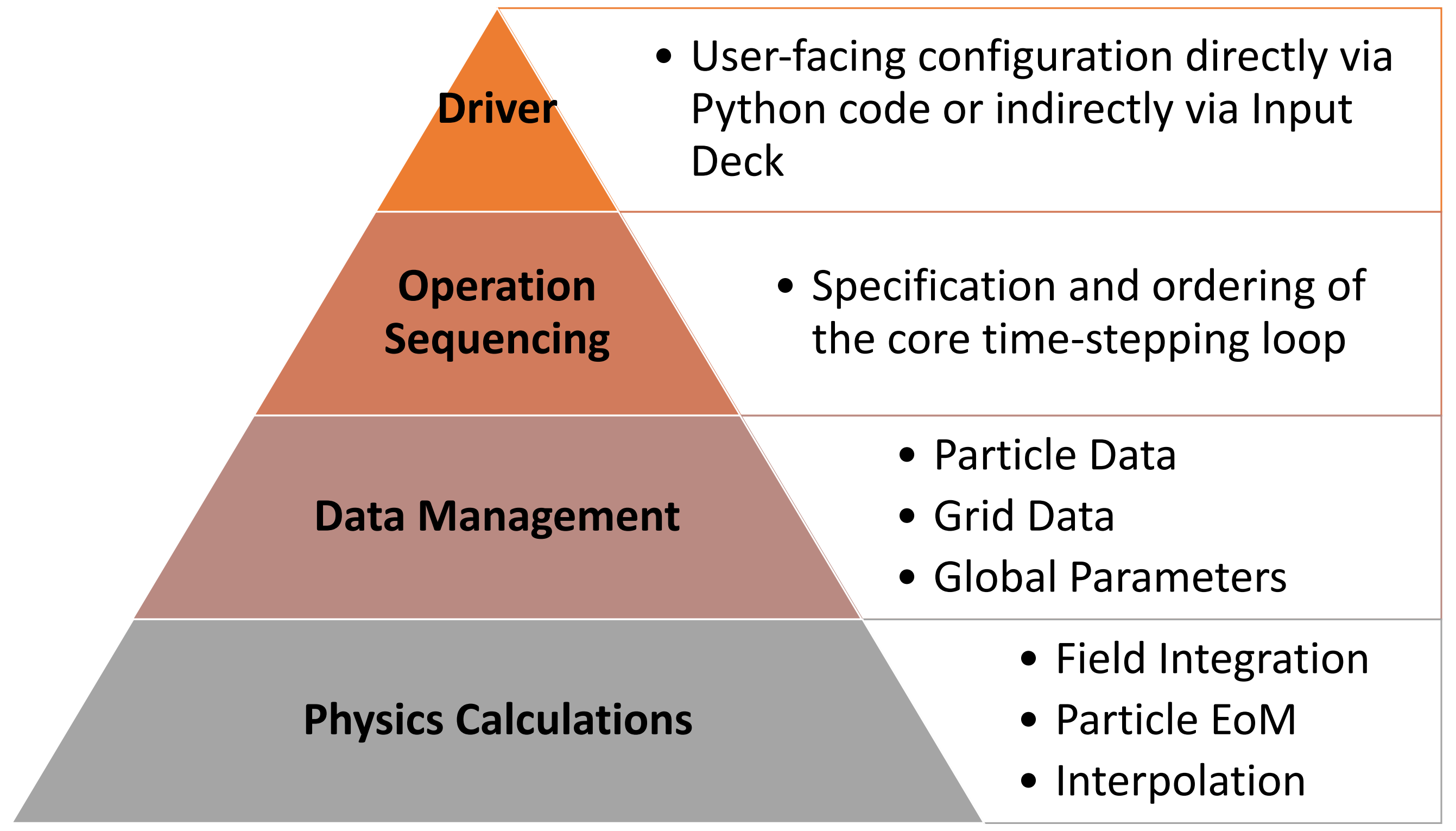
- Integrating Maxwell's equations to update the electric and magnetic fields

Architectural Hierarchy



- Using such a hierarchy as a guiding principle, clarity of structure and modularity of function become natural
- Compartmentalization allows for changes to be localized and encourages modules to be individually intelligible
- Minimizing the impact of intra-code design choices on physics modules allows freedom of implementation and optimization

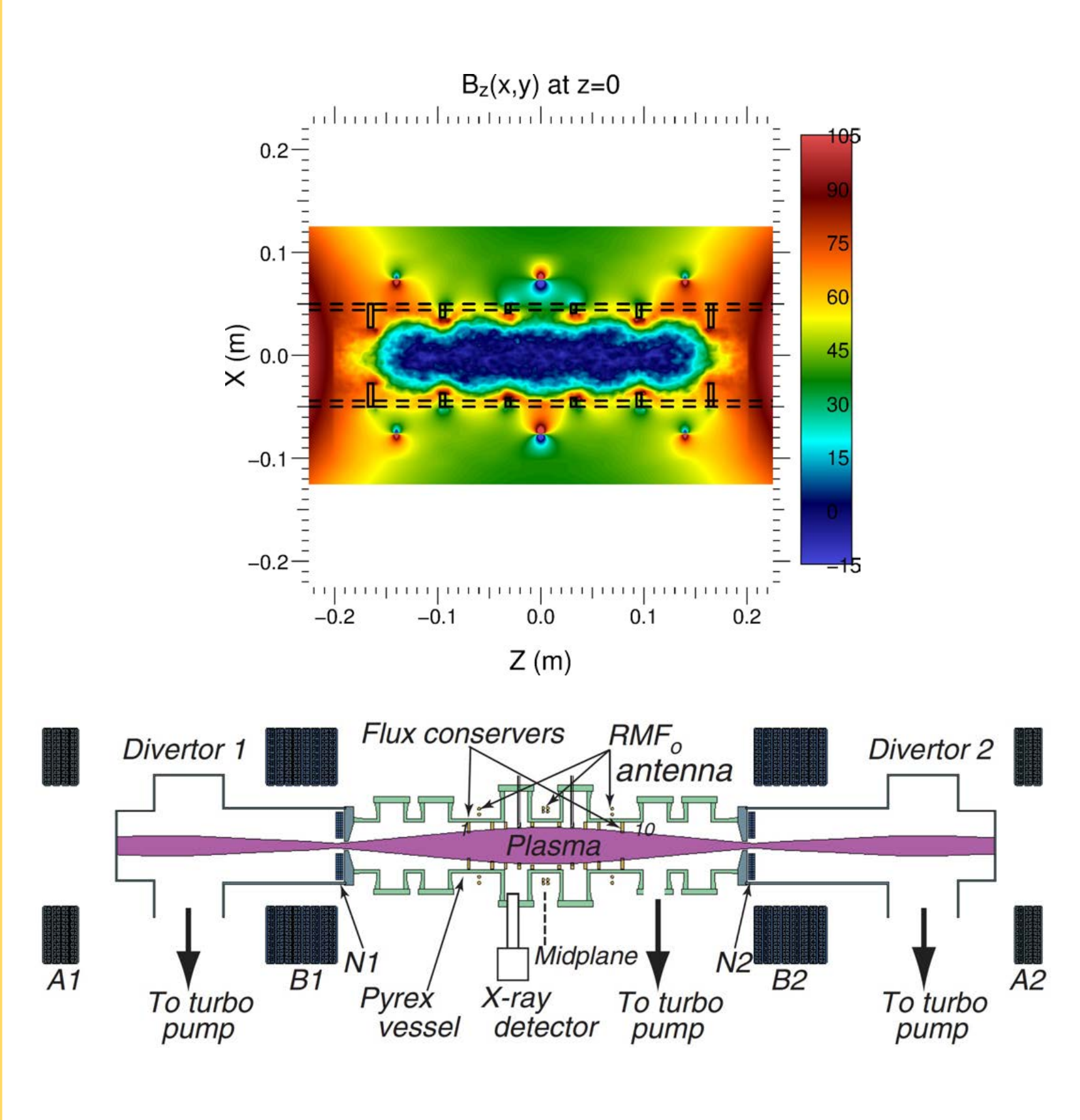
Implementing PIC using the Architectural Hierarchy



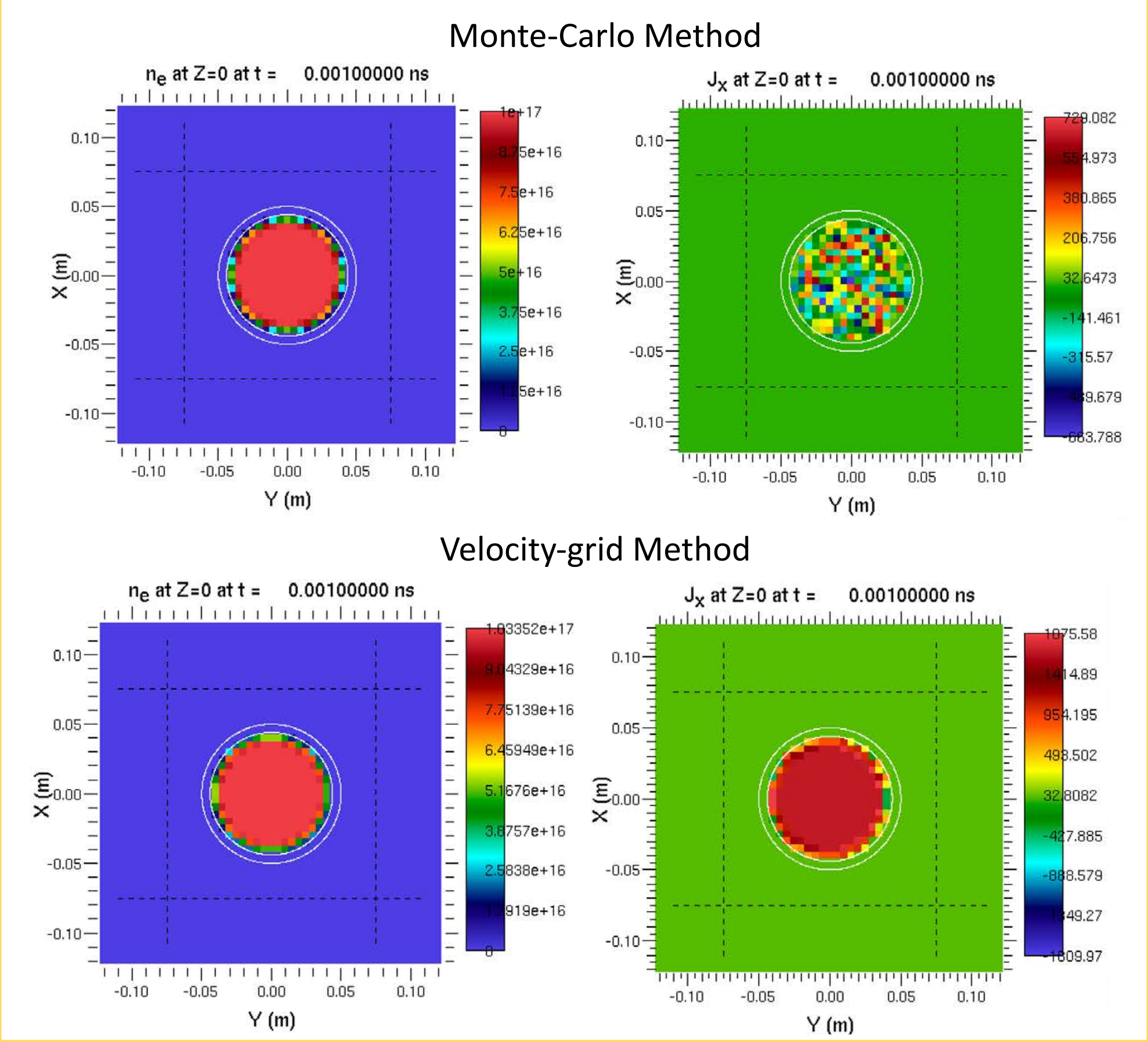
Driver	User-facing configuration directly via Python code or indirectly via Input Deck
Operation Sequencing	Specification and ordering of the core time-stepping loop
Data Management	Particle Data Grid Data Global Parameters
Physics Calculations	Field Integration Particle EoM Interpolation

- Dual configuration schemes allow for simplicity of control without loss of control
- User-defined Sequencers allow simulation archetype to be altered, paving the way for hybrid simulation capabilities
- Compartmentalization of non-physics functionality allows physics algorithms to support modularity

Other Work by TCMM



- TriForce's Library for Integrated Numerical Kinetics (TFLink) is being used in collaboration with PPPL and Princeton Satellite Systems to simulate the PFRC-1 [3] device en route to simulations of PFRC-2/3 (lower figure adapted from [4])


Monte-Carlo Method

Velocity-grid Method

- Novel particle initialization and weighting schemes for reducing particle noise are also being investigated

Next Steps

- Finalize v1.0 of the TFFate architecture and testing suite
- Convert existing implementations of explicit momentum- and energy-conserving particle pushers to TFFate
- Begin implementation of semi-implicit algorithms, such as that developed by Lapenta, et. al. [5]
- Implementation of fully implicit algorithms, such as those by Chen, et. al. [6]
- Investigation of long-time-scale energy conservation
- Investigation of methods allowing for under-resolving of the cyclotron frequency

References

[1] K. Reitz and T. Schlusser, *The Hitchhiker's Guide to Python* (O'Reilly, Sebastopol, 2016), Chap. 4-6.

[2] Brightcove Inc., I. Pouzyrevsky, R. Smith, W. Modderman-Lenstra, D. Kaarsemaker (2021) Diamond source code (v4.0.515) [Open-source Code]. https://github.com/python-diamond/Diamond.

[3] D.R. Welch, S.A. Cohen, T.C.Genoni, and A.H. Glasser, Phys. Rev. Lett. **105** Article No. 015002 (2010). doi: https://doi.org/10.1103/PhysRevLett.105.015002.

[4] S.A. Cohen, B. Berlinger, C. Brunkhorst, A. Brooks, N. Ferraro, D.P. Lundberg, A. Roach, and A.H. Glasser, Phys. Rev. Lett. **98**:14 Article No. 145002 (2007). doi: https://doi.org/10.1103/PhysRevLett.98.145002.

[5] G. Lapenta, J. Comput. Phys. **334**, 349 (2017).
doi: http://dx.doi.org/10.1016/j.jcp.2017.01.002.

[6] G. Chen, L. Chacón, C. A. Leibs, D. A. Knoll, and W. Taitano, J. Comput. Phys. **258**, 555 (2014).
doi: http://dx.doi.org/10.1016/j.jcp.2013.10.052.

Acknowledgments

This material is based upon work supported by the United States Department of Energy ARPA-E under Award No. DE-AR0001272, OFES under Award No. DE-SC0017951, and NNSA under Award No. DE-NA0003856.

UNIVERSITY *of* ROCHESTER