• 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 software architecture and design from its earliest stages

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

For the Learner:
- Individual parts of the code can be read and understood without detailed knowledge of the rest of the code
- Enables bug squashing at a granularity that suits the needs of the learner

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

For the Developer:
- Enables bug squashing at a granularity that suits the needs of the learner
- Individual parts of the code can be read and understood without detailed knowledge of the rest of the code

For TFFate:
- The key for this code and for reliance of its parts can be understood using only high level knowledge of the simulation archetype (PIC, SPH, etc.)

• Field-based representations and methods allow for changes to be localized and freedom of non-physics simulation capabilities for hybrid schemes allow for detailed knowledge of the rest of the code

For the Architectural Hierarchy:
- User-facing configuration directly via Python code or indirectly via Input Deck
- Converter implementations of explicit momentum- and energy-conserving particle pushers to TFFate
- Compartmentalization allows for simplicity of control without loss of freedom of implementation and optimization

For Next Steps:
- Finalize v1.0 of the TFFate architecture and testing
- Convert existing implementations of explicit momentum- and energy-conserving particle pushers to TFFate
- Implementation of fully implicit algorithms, such as those by Chen, et. al. [6]
- Investigation of methods allowing for under-resolving of the cyclotron frequency

References

Acknowledgments
This material is based upon work supported by the United States Department of Energy ARM-E under Award No. DE-AC02-06CH11357, and NNSA under Award No. DE-NA0003595.