SANDIA REPORT

SAND2018-10100
Unlimited Release
Printed September 10, 2018

ASC ATDM Level 2 Milestone #6358: Assess Status of Next Generation Components and Physics Models in EMPIRE

Matthew T. Bettencourt, Richard M. J. Kramer, Keith L. Cartwright, Edward G. Phillips, Curtis C. Ober, Roger P. Pawlowski, M. Scot Swan, Irina Tezaur, Eric Phipps, Sidafa Conde, Eric Cyr, Craig D. Ulmer, Todd H. Kordenbrock, Scott L. Levy, Gary J. Templet, Jonathan J. Hu, Paul T. Lin, Christian A. Glusa, Christopher M. Siefert, Micheal W. Glass

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2018-10100
Unlimited Release
Printed September 10, 2018

ASC ATDM Level 2 Milestone #6358: Assess Status of Next Generation Components and Physics Models in EMPIRE

Matthew T. Bettencourt, Richard M. J. Kramer, Keith L. Cartwright, Edward G. Phillips
Electromagnetic Theory, 1352

Curtis C. Ober, Roger P. Pawlowski, M. Scot Swan
Multiphysics Applications, 1446

Scott L. Levy
Scalable System Software, 1423

Irina Tezaur
Quantitative Modeling and Analysis, 8754

Eric Phipps
Optimiztation and UQ, 1441

Sidafa Conde, Eric Cyr
Computational Mathematics, 1442

Todd H. Kordenbrock
Scalable Analysis and Viz, 1461

Gary J. Templet, Craig D. Ulmer
Scalable Modeling and Analysis Systems, 8753

Jonathan J. Hu, Paul T. Lin, Christian A. Glusa
Scalable Algorithms, 1426

Christopher M. Siefert
Computational Multiphysics, 1443

Micheal W. Glass
Computational Simulation Infrastructure, 1545

Sandia National Laboratories
P.O. Box 5800, MS-1196
Albuquerque, NM 87185-1196
mbetten@sandia.gov

Abstract

This report documents the outcome from the ASC ATDM Level 2 Milestone 6358: Assess Status of Next Generation Components and Physics Models in EMPIRE. This Milestone is an assessment of the EMPIRE (ElectroMagnetic Plasma In Realistic Environments) application and three software components. The assessment focuses on the electromagnetic and electrostatic particle-in-cell solutions for EMPIRE and its associated solver, time integration, and checkpoint-restart components. This information provides a clear understanding of the current status of the EMPIRE application and will help to guide future work in FY19 in order to ready the application for the ASC ATDM L1 Milestone in FY20. It is clear from this assessment that performance of the linear solver will have to be a focus in FY19.

Acknowledgment

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. Portions of this research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

Contents

List of Figures

Chapter 1

Introduction

The EMPIRE (ElectroMagnetic Plasma In Realistic Environments) code was the focus of an ATDM Level 2 Milestone in FY18. EMPIRE is a code designed to simulate plasma environments under various different temperatures, pressures and magnetic fields. This milestone is focused on the low-density regime and employs the particle-in-cell formulation. This report documents the current status of the EMPIRE code, performance, verification status and features. The remainder of this section gives an overview of the milestone and of the EMPIRE code and core algorithm, with following sections on performance status, verification status, time integration and output status.

1.1 Milestone description

The milestone is intended to assess the status of the EMPIRE application. The description of the milestone is as follows:

> Assess the status of a subset of the Next Generation Components and physics capabilities to be used for the FY20 L1 Milestone for the ionizing electromagnetic pulse (SREMP/SGEMP) ATDM application code (EMPIRE). The assessment will focus on the electromagnetic and electrostatic particle-in-cell solutions for EMPIRE and its associated solver, time integration, and checkpoint-restart components. A representative problem will be used to assess current status. The assessment will include code verification, performance, and portability across available ATS and CTS architectures.

The completion criteria of the milestone is as follows:

> Documentation of the status of code verification, performance, and portability on available CTS and ATS architectures for the described assessment.

This report, and accompanying PowerPoint file, are the documentation of the milestone and represent the status at the end of FY18 for the particle-in-cell portion of the code. In addition the EMPIRE and FAODEL repos have been tagged with a version tag FY18Milestone and a correspondin gTrilinos SHA of 8d652ca.

1.2 EMPIRE Particle-in-Cell Algorithm

1.2.1 Description

Particle-in-Cell (PIC) algorithms are designed to solve numerically the Klimontovich equation for plasma dynamics. The plasma is represented as a collection of discrete particles, each with its own mass, momentum, and charge, which approximate a probability distribution function as a series of delta functions across all particles i.

$$f = \sum_{i=1}^{N_p} f_i = \sum_{i=1}^{N_p} \delta(x - x_i)\delta(v - v_i). \tag{1.1}$$

The distribution f_i for each particle is updated using Newton's law and the Lorentz force equation:

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i, \qquad \frac{d\vec{v}_i}{dt} = \frac{q_i}{m_i}\left(\vec{E}(x_i) + v_i \times \vec{B}(x_i)\right), \tag{1.2}$$

where $\vec{E}$ and $\vec{B}$ are the electric and magnetic fields, respectively. Assembling these equations yields the Klimontovich equation for each particle's dynamics, which resembles the Boltzmann equation:

$$\frac{\partial f_i}{\partial t} + \frac{\vec{v}_i}{m} \cdot \nabla f_i + \frac{q_i}{m_i}\left(\vec{E}(x_i) + v_i \times \vec{B}(x_i)\right)\frac{\partial f_i}{\partial \vec{v}} = 0. \tag{1.3}$$

With known electric and magnetic fields, this fully described the particle evolution. Maxwell's equations provide the coupling between the movement of the charged particles and the electric and magnetic fields. They consist of Gauss' Law,

$$\nabla \cdot \vec{E} = \frac{\rho}{\varepsilon_0}, \tag{1.4}$$

the magnetic divergence constraint,

$$\nabla \cdot \vec{B} = 0, \tag{1.5}$$

Faraday's law,

$$\frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E}, \tag{1.6}$$

and Ampere's law,

$$\frac{\partial \vec{E}}{\partial t} = \frac{1}{\mu\varepsilon}\nabla \times \vec{B} - \frac{1}{\varepsilon}\vec{J}, \tag{1.7}$$

where ρ and $\vec{J}$ are the charge and current densities, and ε and μ are the permittivity and permeability of the background medium (typically vacuum). The particles couple back to Maxwell's equations through the charge and current densities, given by

$$\rho = \sum_{i=1}^{N_p} q_i f_i, \qquad \text{and} \qquad \vec{J} = \sum_{i=1}^{N_p} q_i v_i f_i. \tag{1.8}$$

Note that the coupling can be simplified in the electrostatic approximation, which reduces Maxwell's equations to only Gauss' Law (1.4).

Traditional PIC methods as shown in [1] and [2] make the following assumptions:

1. Particles are grouped into super particles that represent a large number of particles
2. Electric and magnetic fields are solved on a mesh
3. Leap-frog time integration with linear fields
4. Debye length and plasma period are resolved.

EMPIRE uses the traditional operator-split time integration method for particles and fields, where the fields are solved staggered by a half step from the particles with one way coupling between the two, as shown in Figure 1.1.

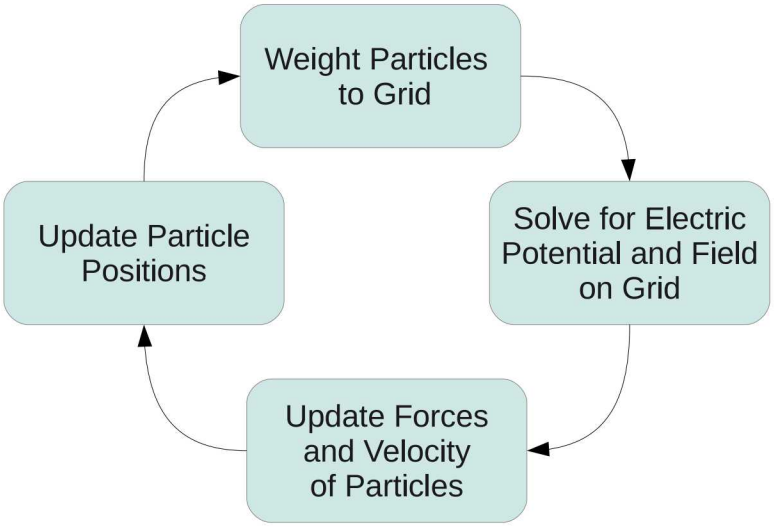


Figure 1.1: Time Integration Loop

EMPIRE uses the finite-element method (FEM) [3] for solving Maxwell's equations, with a compatible edge/face discretization for the electric and magnetic fields to enforce the magnetic divergence constraint (1.5) by construction [4]. A more complete description of the FEM can be found in the EMPIRE theory manual and is not repeated here; however, the unique components of EMPIRE will be highlighted below. EMPIRE utilizes the Trilinos package Panzer [5] for the infrastructure required to implement the FEM in software.

1.2.2 Weighting Algorithms

As shown in Figure 1.1, there are four main stages in the time integration step. Much of the accuracy of a PIC method comes from appropriate choice of the weighting algorithm for transferring the fields to and from the mesh. The current and charge of the particles needs to weight into the residual of the weak form of Maxwell's equations. Particles are represented as delta functions in both space and time, and therefore, the weak form involves integration of these delta functions against a test function.

Weighting of the charge density involves weighting each particle i on element j against test function k,

$$\int_{\Omega_j} \rho \psi_k dV = \sum_{i=1}^{N_p} \int_{\Omega_j} q_i \delta(\vec{x} - \vec{x}_i) \psi_k dV = \sum_{i=1}^{N_p} q_i \psi_k(\vec{x}_i). \tag{1.9}$$

This replaces the integration of the charge over the volume with a single evaluation of the test function at the point of the particle, and requires that the evaluation of the basis functions at arbitrary points needs to be efficient and performant. Weighting the current follows a similar approach, except that the basis and test functions are the edge basis, $\hat{e}$, rather than the nodal basis:

$$\int_{\Omega_j} \vec{J} \hat{e}_k dV = \sum_{i=1}^{N_p} \int_{\Omega_j} \int_{n\triangle t}^{(n+1)\triangle t} q_i \vec{v}_i \cdot \hat{e}_k dV. \tag{1.10}$$

The time integration is done numerically, which for second order integration on simplices reduces to the midpoint rule. Therefore, the current weighting results in

$$\int_{\Omega_j} \vec{J} \hat{e}_k dV = \sum_{i=1}^{N_p} \triangle t q_i \vec{v}_i(\vec{x}_i^{n+\frac{1}{2}}) \cdot \hat{e}_k(\vec{x}_i^{n+\frac{1}{2}}). \tag{1.11}$$

Once again, this reduces to a simple evaluation of the test function at the particle location, now at the midpoint of the trajectory. The trajectory must be split up so every individual element visited by a particle in a time step and each line segment of current is accrued to that element. Because of the nature of the compatible discretization, this current weighting maintains the charge density weighting and (1.4) to solver tolerance. Once the charge and current density are weighted to the mesh, Maxwell's equations can be solved.

With the solution to the linear system, we either have nodal values for the potential ϕ in the electrostatic approximation, or edge $\vec{E}$ and face $\vec{B}$ values that must be weighted back to the nodal basis values according to

$$\int_{\Omega_j} \left(\vec{E}_e - \vec{E}_n \right) \hat{\psi}_k dV = 0, \tag{1.12}$$

where $\vec{E}_e$ is the edge representation of the field, $\vec{E}_n$ is the nodal basis and ψ_k is the test function, which may be different from the basis function for $\vec{E}_n$. EMPIRE has two different weighting methods available that differ in the test function against which the projection is integrated: a consistent mass formulation and a lumped mass formulation. The consistent mass formulation has the effect of steepening the solution, whereas the lumped mass formulation has a smoothing effect. Note that the lumped approach is analogous to what is done in traditional PIC methods. From the nodal field representation, the field value at the particle location is calculated simply be applying the nodal basis:

$$\vec{E}(\vec{x}_i) = \sum_{k=1}^{N_b} e_k \psi_k(\vec{x}_i), \tag{1.13}$$

where e_k is the solution value from the projection equation (1.12).

1.2.3 Particle Dynamics

The electric and magnetic field values weighted to the particles define the forces acting upon them. The particles can then be accelerated and moved using (1.2). The acceleration is calculated simply from $F = ma$, but since the force is a function of the particle velocity from (1.2), the velocity cannot be updated using the old force. In 1970, Boris described an elegant alternative, which is now commonly known as the Boris Method. Boris method is the de facto standard for particle pushing in plasma simulation codes. Specifically, we are solving

$$\frac{\vec{v}^{n+1/2} - \vec{v}^{n-1/2}}{\triangle t} = \frac{q}{m}\left[\vec{E}^n + \frac{\vec{v}^{n+1/2} + \vec{v}^{n-1/2}}{2} \times \vec{B}^n\right] \tag{1.14}$$

where Boris noticed that the electric field can be eliminated by defining

$$\begin{aligned} \vec{v}^{n-1/2} &= \vec{v}^- - \frac{q\vec{E}^n}{m}\frac{\Delta t}{2}, \\ \vec{v}^{n+1/2} &= \vec{v}^+ + \frac{q\vec{E}^n}{m}\frac{\Delta t}{2}. \end{aligned} \tag{1.15}$$

Substituting these definitions into (1.14), we obtain the pure rotational motion

$$\frac{\vec{v}^+ - \vec{v}^-}{\Delta t} = \frac{q}{2m}\left(\vec{v}^+ + \vec{v}^-\right) \times \vec{B}^n. \tag{1.16}$$

Boris next utilized some basic geometry (see Figure 4-4a in Birdsall, p. 62) to derive the expression for performing the rotation. The first step is to find the vector bisecting the angle formed by the pre- and the (yet computed) post-rotation velocity. The angle through which the velocity will rotate in the given time step is, from geometry, $\tan(\theta/2) = -(qB/m)\Delta t/2$. The vector form of this is $\mathbf{t} \equiv -\hat{\mathbf{b}}\tan\theta/2 = (q\mathbf{B}/m)\Delta t/2$. The bisector vector (v prime) is then

$$\mathbf{v}' = \mathbf{v}^- + \mathbf{v}'^- \times \mathbf{t}. \tag{1.17}$$

This $\mathbf{v}'$ vector is perpendicular to both the magnetic field (the vector $\mathbf{t}$) and the vector from $\mathbf{v}^-$ to $\mathbf{v}^+$, the post-rotation velocity we are seeking. This connecting vector is again obtained from geometry as the cross product of $\mathbf{v}'$ and a new vector $\mathbf{s}$. This vector $\mathbf{s}$ is just a version of the rotation vector $\mathbf{t}$ scaled to satisfy the requirement that magnitude of velocity remains constant in the rotation. Mathematically,

$$\mathbf{v}^+ = \mathbf{v}^- + \mathbf{v}' \times \mathbf{s} \tag{1.18}$$

where

$$\mathbf{s} = \frac{2\mathbf{t}}{1+t^2} \tag{1.19}$$

To implement the Boris method, first obtain v^- by adding half acceleration to the initial velocity, per (1.15). Then perform the full rotation according to (1.18) and (1.19). Finally, add another half acceleration, as given by (1.15). From the new velocity at time $n+\frac{1}{2}$, the updated particle position is simply

$$\vec{x}^{n+1} = \vec{x}^n + \triangle t\vec{v}^{n+\frac{1}{2}}. \tag{1.20}$$

Chapter 2

Performance

Mission requirements for running production problems effectively at scale impose a performance threshold for the EMPIRE application to average throughput of 10 time-steps per second, and ideally 30 time-steps per second. This chapter describes progress towards that goal, focusing on the PIC algorithm, linear solver, and input/output performance.

The goal of 10 time-steps per second is based on two criteria. First to reproduce the performance that we already have in Aleph (electrostatically) and EMPHASIS (electromagnetically) both MPI only codes. For Aleph there were timings using a mesh similar to the EMPIRE performance simulations and run on Serrano[1], (EMPIRE small) 467k elements, 128 cores, 50 time-steps per second; (EMPIRE medium) 1.8M elements, 512 cores, 33 time-steps per second, and (EMPIRE large) 7.6M elements, 2048 cores, 20 time-steps. For EMPHASIS, also run on Serrano, with nearly identical meshes with a CFL=1 and 64 particles per cell. The timings for, small, medium, and large shows performance going from 35 time-steps per second for the small problem with 128 cores (4 nodes which is not using the full number of cores on each node), 30 times-steps per second for the medium problem with 1024 cores (32 nodes), 10 time-steps per second for the large problem with 4096 cores (128 nodes, half the number or cores for weak scaling because of waiting time in the queue). A better understanding for our current code scaling would be useful, but this is what we have at this time and will be improved for next years milestone. Since the hardware and software environment are different than trinity detailed comparison should not be done. However, it does show through a couple thousand cores 10s of time-steps per second are possibility. The other half of this rational is based on our target simulation for the L1 milestone in FY20. For this simulation, it would be nice to finish the million time-step simulations in one day (11.6 time-steps per second). The target simulation can be changed to have 2 million or 1/2 million time-step by just having better resolution or questionable resolution. The wall clock time could also increased to 2 days, so maybe we could get by with 3 time-step per second but with all the other uncertainty and risk aiming for the bare minimum with questionable resolution is not a good idea.

The performance of the EMPIRE application was tested using a realistic geometry in both electrostatic and electromagnetic regimes. This geometry is affectionately known as the blob problem and is shown in Figure 2.1. This geometry was created for this project to represent a realistically sized domain, similar to the Aleph benchmarking problem using in their strong/weak scaling studies. This domain is filled with a quasi-neutral plasma (electrons and hydrogen ions) of density 10^{16} m^{-3} and a temperature of $\sim$10 eV.

[1] 1,122 nodes, 40,392 cores, 2.1 GHz Intel Broadwell E5-2695 v4, 128 GB per node

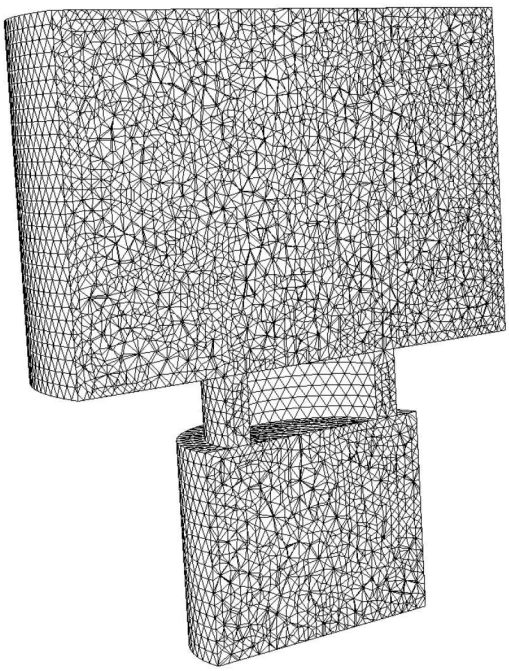


Figure 2.1: Coarsest “blob” mesh for performance studies

Size	# of Elements	# of Nodes	# of Edges	# of Faces	# of Particles	Particles/element
S	337k	60k	406k	683k	16M	47.5
M	2.68M	462k	3.18M	5.40M	128M	47.8
L	20.7M	3.51M	24.4M	41.6M	1B	49.5
XL	166M	27.9M	195M	333M	8.2B	49.4
XXL	1.332B	223M	1.56B	2.67B	65.6B	49.2

Table 2.1: Performance test sizes for scaling study.

This problem was used for both a weak and strong scaling studies. The size of the problems considered for these studies are shown in Table 2.1. As can be seen, each refinement level is roughly a factor of 8 times larger in mesh and particle count.

We do a combination of weak and strong scaling tests. Each size problem (S, M, ...) is run over four doublings on Trinity nodes, starting at 1 node for the smallest problem and at 8 times as many nodes for each successively larger problem. For example, the medium problem is run on 8, 16, 32 and 64 nodes, etc. This is done for both the Knights Landing (KNL) and the Haswell (HSW) partitions of the machine. These meshes are run for only 100 time-steps.

The first analysis breaks down the total runtime into two major components, the linear solver time, and the particle push time. Figure 2.2 shows total, linear solve, and particle push times for both EM and ES solutions on both HSW and KNL.

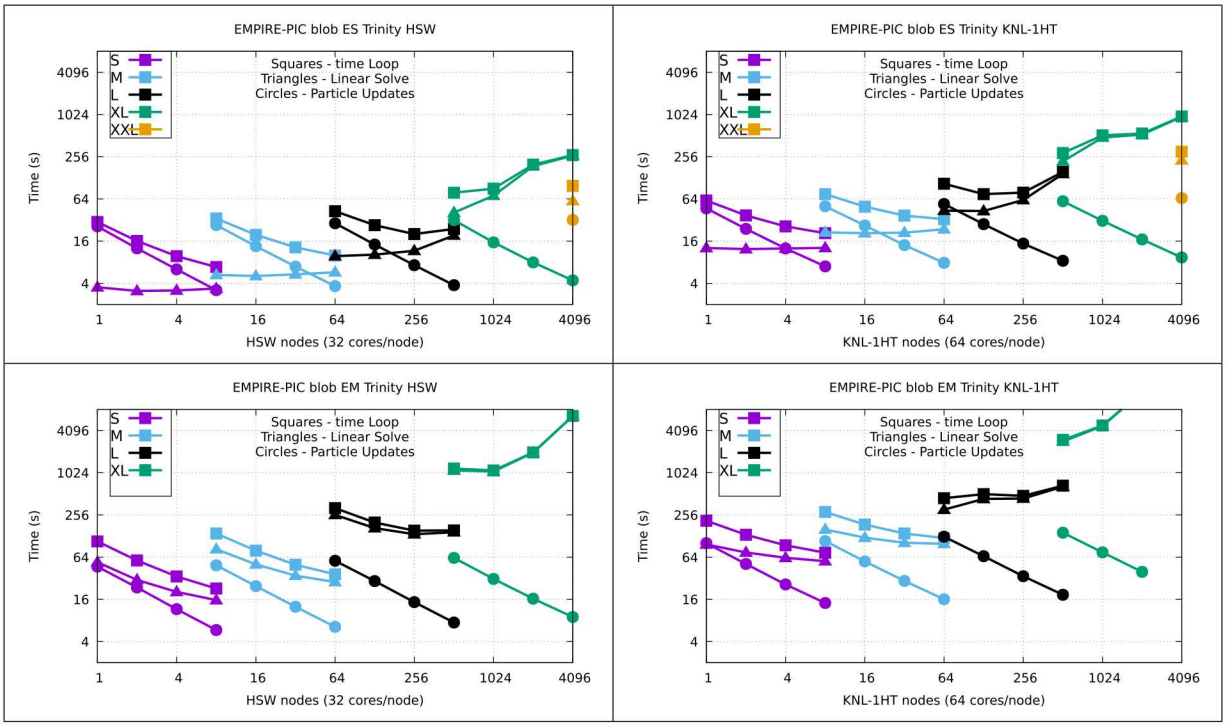


Figure 2.2: Total timing results for EMPIRE on HSW (left) and KNL (right) running the ES solver (top) and the EM solver (bottom)

The use of hyper-threads was explored for the KNL architectures. Hyper-threads are a hardware abstraction where one oversubscribes a physical core to hide memory latency. The core can swap between hyper-threads in a few cycles, switching to a thread which is ready to perform a computation. Figure2.3 shows the effects of hyper-threads. As can be seen, the additional threads speed up the particle work while hurting the solver work. This is somewhat expected and PIC, especially on unstructured meshes, has irregular memory access patterns. The overall effect is up to a 40% savings in the particle updates, while costing a factor of two on the solver side.

It can be seen that we are within the mission requirement threshold for run time on the very small problems, but as much as a factor of 400x away from that threshold for problems at scale. Unlike a traditional PIC code, which is particle dominated, here the linear solver time dominates the solution for all but the smallest core counts. The details for the linear solver will be provided in section 2.2 with the particle update times being demonstrated in section 2.1, with I/O details will be provided in section 5.

2.1 PIC Performance

This section details the performance of the sections of the code specific to the PIC algorithm, in particular the particle move (including current weighting), particle sorting, charge weighting, and field weighting algorithms. Figure 2.4 shows the scaling results.

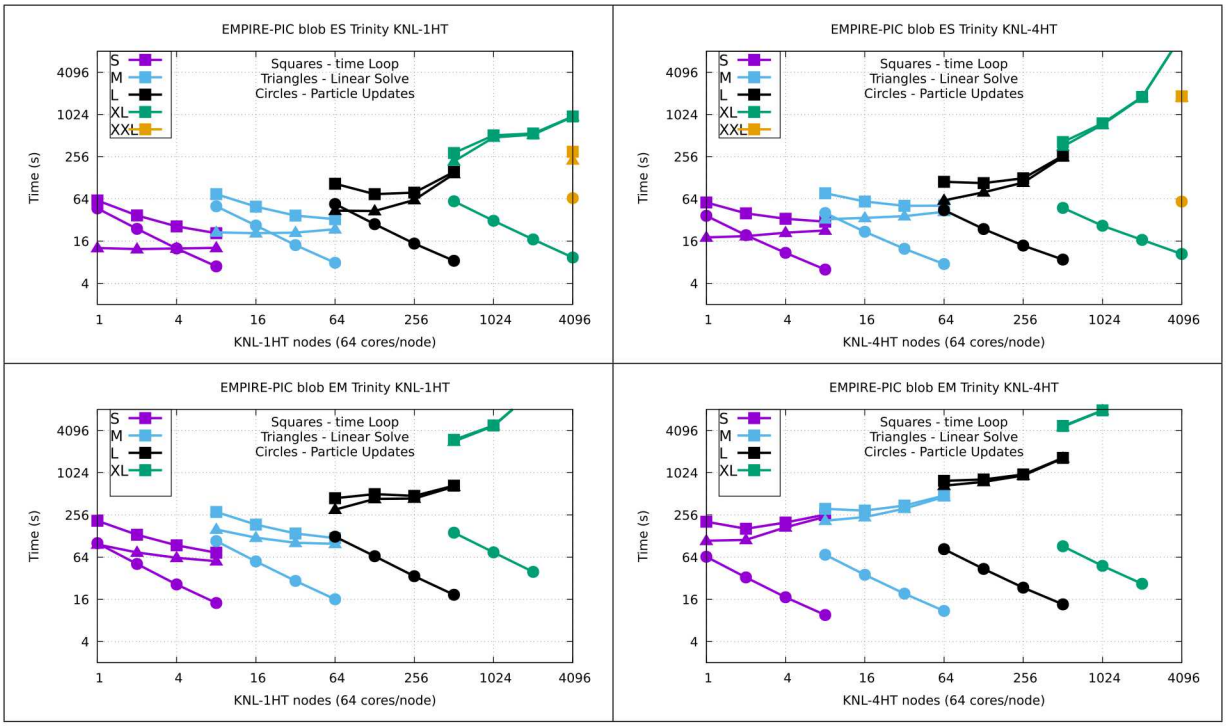


Figure 2.3: Total timing results for EMPIRE on KNL with 1 hyper-threads (left) and 4 hyper-threads (right) running the ES solver (top) and the EM solver (bottom)

The effect of hyper-threads can be seen in Figure 2.5.

From the top level results some general conclusions can be drawn:

1. The particle update weak scales with near ideal weak-scaling efficiency.
2. The particle update strong scales over 3 doublings with near-ideal scaling efficiency.
3. There is no evidence that it will not continue to strong and weak scale.
4. The move is the most expensive part of the code.
5. The particle update benefits from hyper-threads.

Breaking down the results, it is clear that the particle move is the dominant feature of the total PIC algorithm time. Each particle is associated with a region of physical space, which EMPIRE decomposes amongst MPI partitions. When a particle reached the end of a partition boundary, this particle needs to be migrated to a neighboring processor. The particle move timer consists of both the actual move step combined with the cost of this particle migration.

The actual move step consists of advancing each particle's position from time $n \rightarrow n+1$, and for electromagnetics, weighting currents to the mesh along the way. The algorithm requires moving each particle segment-by-segment, stopping at every element boundary and accruing current to the

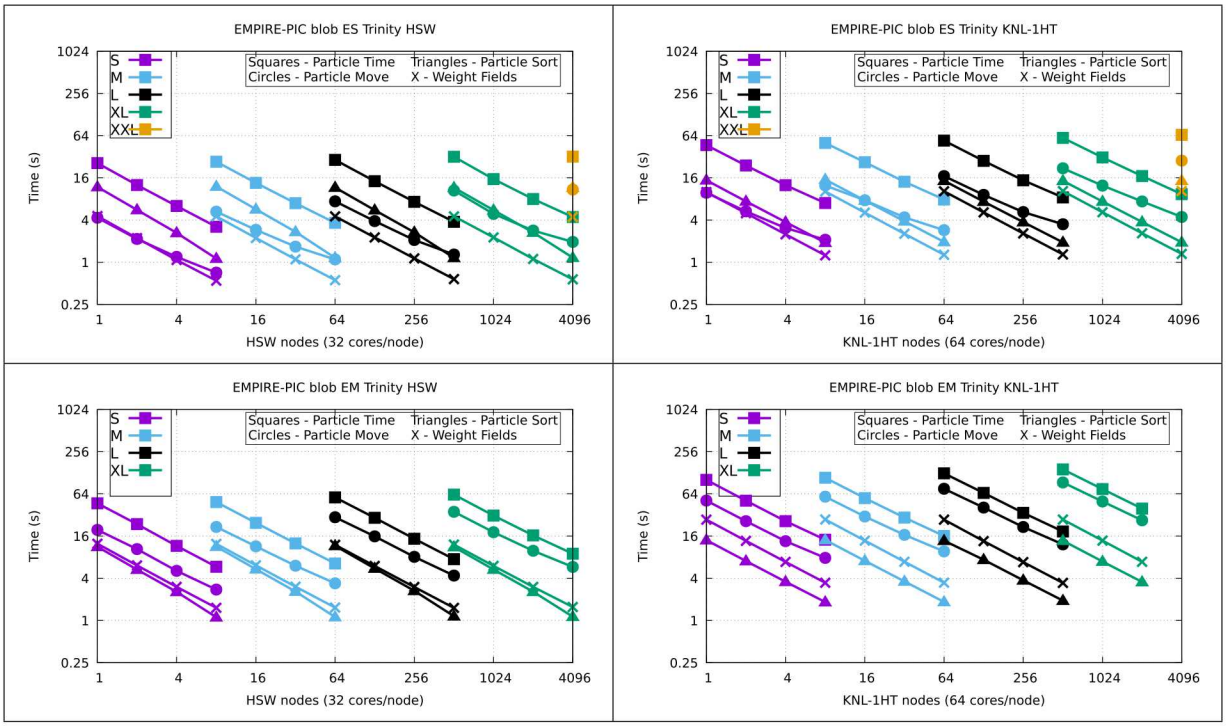


Figure 2.4: Particle timing results for EMPIRE on HSW (left) and KNL (right) running the ES solver (top) and the EM solver (bottom)

mesh element-by-element. This is a costly operation, as determining the time at which a particle departs an element involves evaluation of several dot products of the particle velocity with the element face normals, evaluation of conditionals, and division operations. None of this readily vectorizes or takes advantage of streaming instruction sets. Future work will be conducted towards exploring alterations to the move algorithm that could take advantage of these specialized hardware routines.

As can also be seen in the plots, the move for the electrostatic case is cheaper than the move for the electromagnetic cases. This is because in the electrostatic case current is not weighted to the mesh, which can be an expensive step, on par with the cost of moving the particle.

Particle sorting at one time was the most expensive part of the PIC algorithm. This was especially true on GPU systems where the sorting operation occupied 91% of the particle update time. This has been drastically improved (12x on P100) by adopting the parallel merge sort from the CUDA Thrust library. This also reduced the sort cost on CPU systems where it now one of the smaller contributions to the total time.

Overall, the particle update, which dominates the runtime of PIC codes at Sandia and elsewhere, shows that the EMPIRE implementation both strong and weak scales to 4096 nodes (256k cores). It is believed that it will not be the bottleneck in achieving the target of 10 time steps per second for the 2020 L1 Milestone.

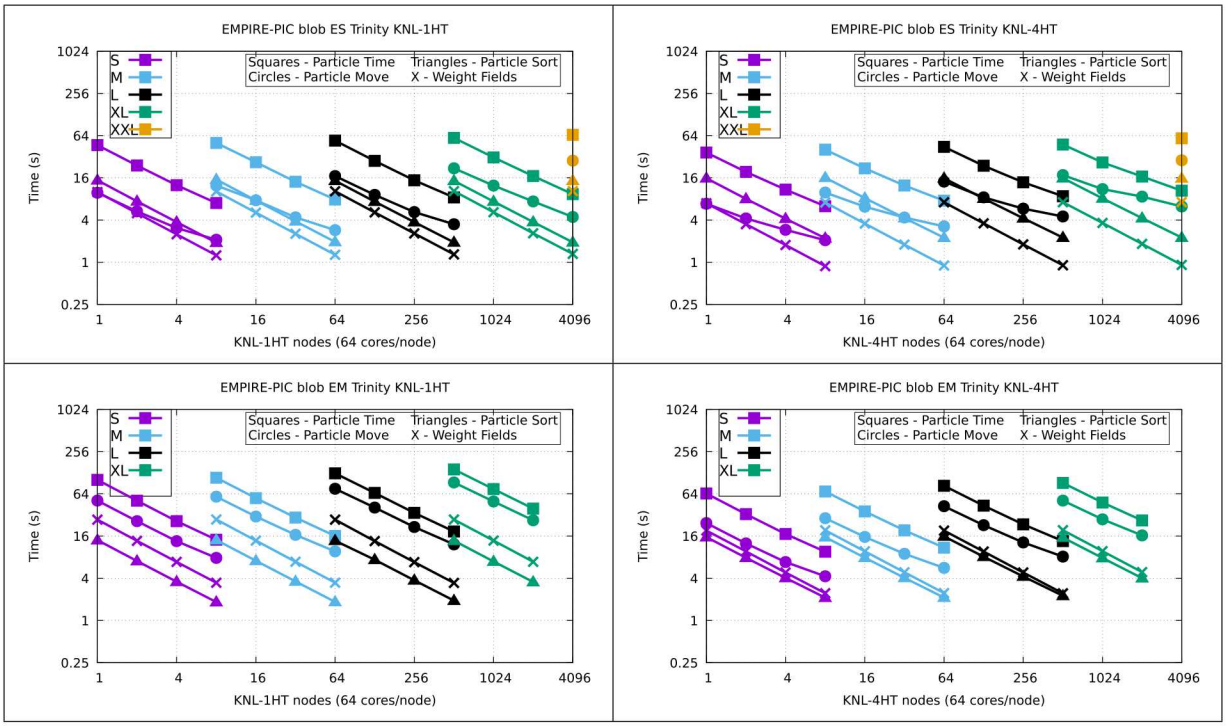


Figure 2.5: Particle timing results for EMPIRE on KNL, 1 hyper-thread (left) and 4 hyper-threads (right) running the ES solver (top) and the EM solver (bottom)

2.2 Linear Solvers

In this section we first describe the linear systems that are solved in the finite element phase of the blob simulations. We then outline the overall solution method and specific underpinning linear solver algorithms. Results are given over a range of processor core counts, as well as descriptions of improvements throughout the milestone. Finally, we describe ongoing and future work to further improve linear solver performance.

There are two main system types in EMPIRE that depend on linear solvers in Trilinos, an electrostatics(ES) problem and an electromagnetic (EM) problem.

2.2.1 ES Problem

For the electrostatics problem, the linear system is the following four-by-four blocked linear system:

$$\begin{pmatrix} A & 0 & 0 & 0 \\ G_x & Q & 0 & 0 \\ G_y & 0 & Q & 0 \\ G_z & 0 & 0 & Q \end{pmatrix} \begin{pmatrix} \phi \\ E_x \\ E_y \\ E_z \end{pmatrix} = f, \tag{2.1}$$

where A is a Poisson matrix; G_x, G_y, and G_z are gradient components; Q is a (possibly lumped) mass matrix; E_x, E_y, and E_z are electric field components; and the right-hand side f is the charge from the prior particle solve.

2.2.1.1 Initial ES Results

Initially, the four-by-four ES blocked system was explicitly assembled and solved at each time step. The global Krylov solver is GMRES, preconditioned by a block Gauss-Seidel (GS) iteration. The Q blocks in 2.1 are point-diagonal and can be trivially inverted. The A block is inverted using a single V-cycle of smoothed aggregation algebraic multigrid (SA-AMG). Thus, the dominant cost of each ES GMRES iteration, apart from the matrix-vector multiply, should be the solution of a Poisson matrix.

ES solver results from 14-May-2018 are given in Figure 2.6. Each color denotes the strong

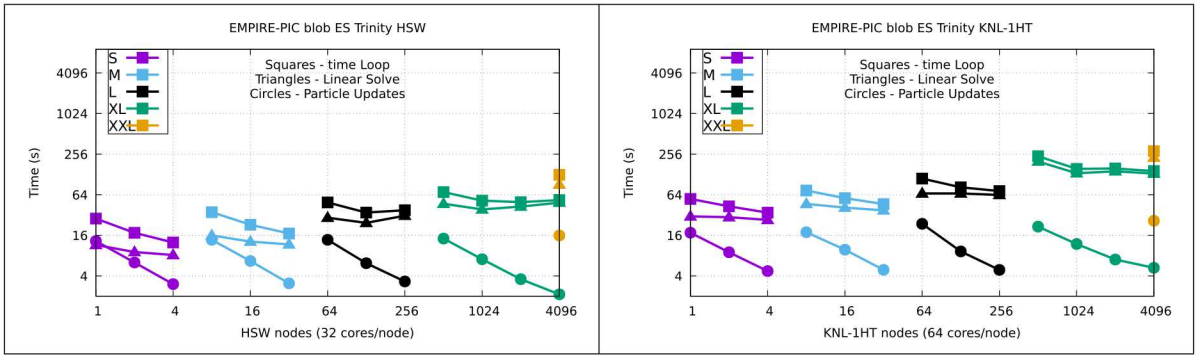


Figure 2.6: ES linear solver results (triangle symbols) on Trinity from 14-May-2018.

scaling of various EMPIRE solve components for a particular resolution of the blob problem on Trinity. (See Table 2.1 for mesh statistics.) For the Haswell results, there are 2 MPI processes per node and 16 threads per MPI process. For the KNL results, there are 4 MPI processes per node and 16 threads per MPI process. Weak scaling performance can be seen by comparing the first dot in each curve (respectively, second or third). Of interest for the current discussion are the fine dashed lines, which are the timings for the linear solve.

We note that there is at best very modest benefit in strong scaling for all meshes. For the L and XL meshes, however, we see the scaling curve turning up for anything past two times the original number of compute nodes. This can be explained by considering the results of a standalone matrix-vector strong-scaling study on mutrino's Haswell partition, given in Figure 2.7. In this experiment, matrix-vector performance was measured for a 50^3 Poisson matrix. The experiment uses one MPI rank per node, with no threading, which should be the case with the largest computation to communication ratio. The blue line represents measured performance, the green line represents a simple three parameter model[2], and the red line represents the model's latency. At 7800 DOFs per MPI rank (16 nodes), the efficiency is at 64%, i.e., latency is dominating. Adding threads per MPI

[2]latency, bandwidth, and local work

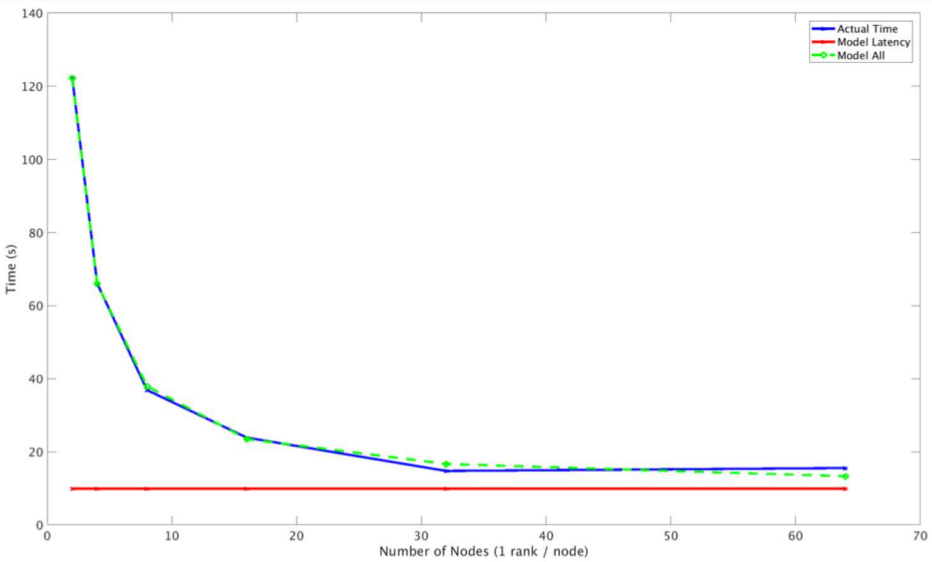


Figure 2.7: Observed and modelled matrix-vector multiply strong-scaling performance on mutrino's Haswell partition, 50^3 row Poisson matrix, 1 MPI rank/node, no threading.

rank will only cause the latency to dominate at lower node counts. As seen in Table 2.2, for the ES blob studies, the work per thread starts at approximately 1800 DOF per thread. This is well below the point measured on mutrino where latency begins to dominate.

Refinement Level	blob mesh				
	S	M	L	XL	XXL
base	1886	1805	1712	1700	1701
2x threads	943	903	856	850	851
4x threads	472	451	428	425	425

Table 2.2: DOFs per thread in EMPIRE ES Poisson linear solve for various meshes.

2.2.1.2 Current ES Results

Two major changes were made in the ES linear solve. First, profiling revealed nonoptimal storage choices for the blocked systems and inversion of the diagonal blocks. This meant that multiplying by a zero or diagonal block required the same time as multiplication with the A block in (2.1). Second, (2.1) was refactored to separate the projection steps from the Poisson solve, as well as simplify them. Thus, only the Poisson system now needs to be solved iteratively. Prior to this refactor, the linear system was nonsymmetric and thus required a GMRES Krylov method. After the refactor, Conjugate Gradients (CG) can be used, as the Poisson system is symmetric.

Table 2.3 compares linear solver times between May and August on Trinity/Haswell. For the largest XXL mesh, there is an approximately 50% improvement in solver time. Figure 2.8 shows

mesh	May	Current	Speedup
S	11.1	3.3	2.1
M	15.6	5.3	2.9
L	28.8	9.6	3.0
XL	46.7	40.4	1.2
XXL	88.5	59.1	1.5

Table 2.3: Comparison of ES Trinity/Haswell total linear solver times from May to August.

the effects of these changes for the total simulation from 22-August-2018. The Trinity/KNL HT= 1

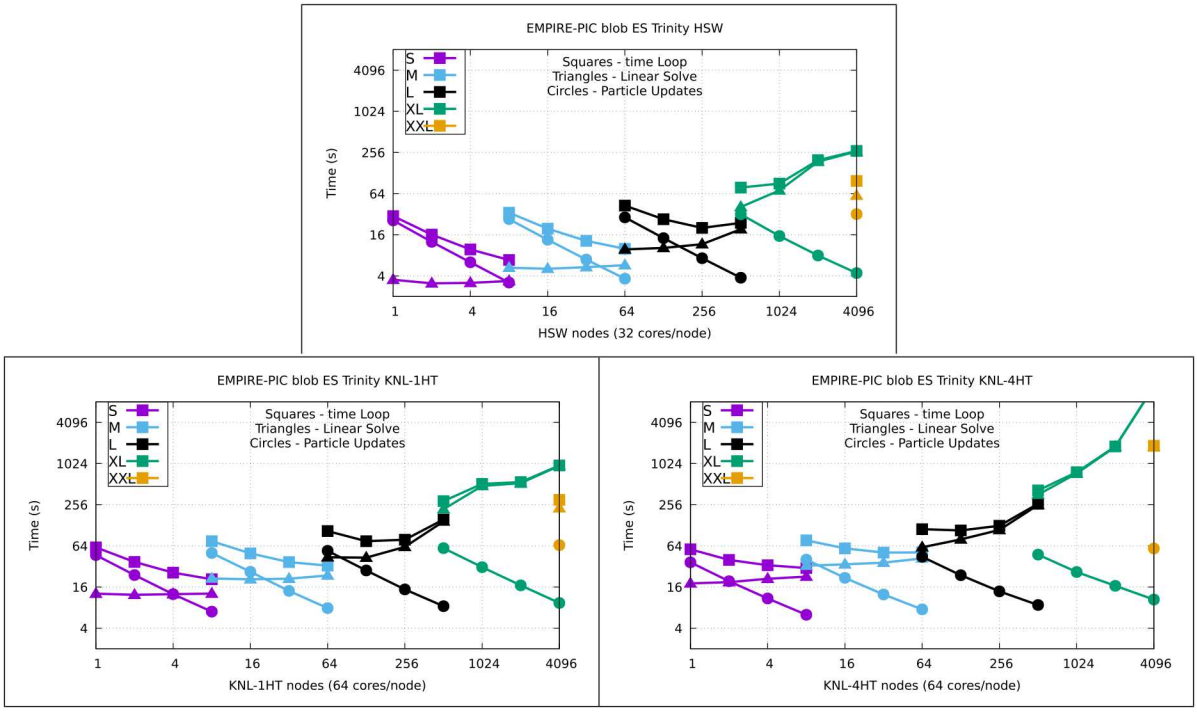


Figure 2.8: ES blob simulation, Trinity, 22-August-2018.

results show speedups of 2.3, 2.2, and 1.8 for S,M, and L meshes, respectively. For the XL and XXL meshes, however, there is no observable speedup.

2.2.2 EM Problem

The electromagnetics solve can be formulated as the following 8×8 block system

$$\begin{pmatrix} Q_n & & & & & & -Q_x^E & \\ & Q_n & & & & & -Q_y^E & \\ & & Q_n & & & & -Q_z^E & \\ & & & Q_n & & & & -Q_x^B \\ & & & & Q_n & & & -Q_y^B \\ & & & & & Q_n & & -Q_z^B \\ & & & & & & Q_B & K \\ & & & & & & -K^T & Q_E \end{pmatrix} \begin{pmatrix} E_x \\ E_y \\ E_z \\ B_x \\ B_y \\ B_z \\ B \\ E \end{pmatrix} = g, \tag{2.2}$$

where Q_B is a face-based mass matrix; Q_E is an edge-based mass matrix; the Q_n are nodal projection terms; K is the weak curl; and g is the current density from the prior particle solve.

2.2.2.1 Initial EM Results

Initially, the eight-by-eight EM blocked system was similarly explicitly assembled and solved at each time step. The global linear solver is GMRES, again preconditioned by a block GS iteration. The first six diagonal blocks of (2.2) are projection matrices. The last two diagonal blocks of (2.2) are edge and face mass matrices, respectively. The preconditioner is a Schur complement approach. In this approach, the diagonal blocks are trivially invertible. The mass matrix is solved iteratively using a single V-cycle of SA-AMG, and the resulting curl-curl matrix in the Schur complement approximation is iteratively solved using algebraic multigrid.

Initial results from 11-June-2018 are given in Figure 2.9 for the EM problem.

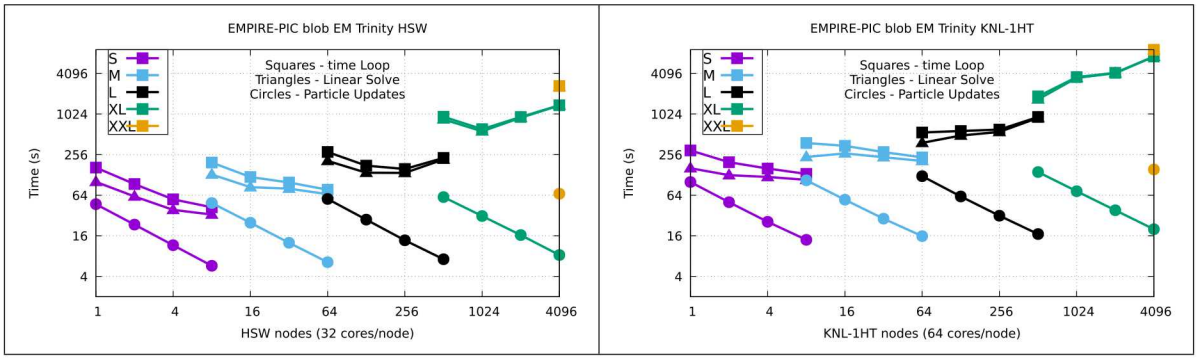


Figure 2.9: Linear solver results (triangle symbols) for EM blob simulation on Trinity from 11-June-2018.

2.2.3 Current EM Results

Subsequent to the 11-June-2018 results, the EM block linear system was reformulated to separate the projection steps from the E and B field solves, resulting in the following two-by-two block system:

$$\begin{pmatrix} Q_B & K \\ -K^T & Q_E \end{pmatrix} \tag{2.3}$$

which can be rewritten as

$$\begin{pmatrix} I & 0 \\ -K^T Q_B^{-1} & I \end{pmatrix} \begin{pmatrix} Q_B & K \\ 0 & S_E \end{pmatrix}, \tag{2.4}$$

where

$$S_E = Q_E + K^T Q_B^{-1} K. \tag{2.5}$$

The two-by-two block system (2.3) is solved using GMRES preconditioned by the block preconditioner

$$\begin{pmatrix} \hat{Q}_B & K \\ 0 & \hat{S}_E \end{pmatrix} \tag{2.6}$$

where $\hat{Q}_B$ is an SA-AMG solver and $\hat{S}_E$ is a Maxwell-specific algebraic multigrid solver [6] applied to an approximation of S_E.

Figure 2.10 shows weak and strong-scaling Trinity results resulting from the EM reformulation. We first note that the Schur complement solver is an algebraic multigrid solver, RefMaxwell [6].

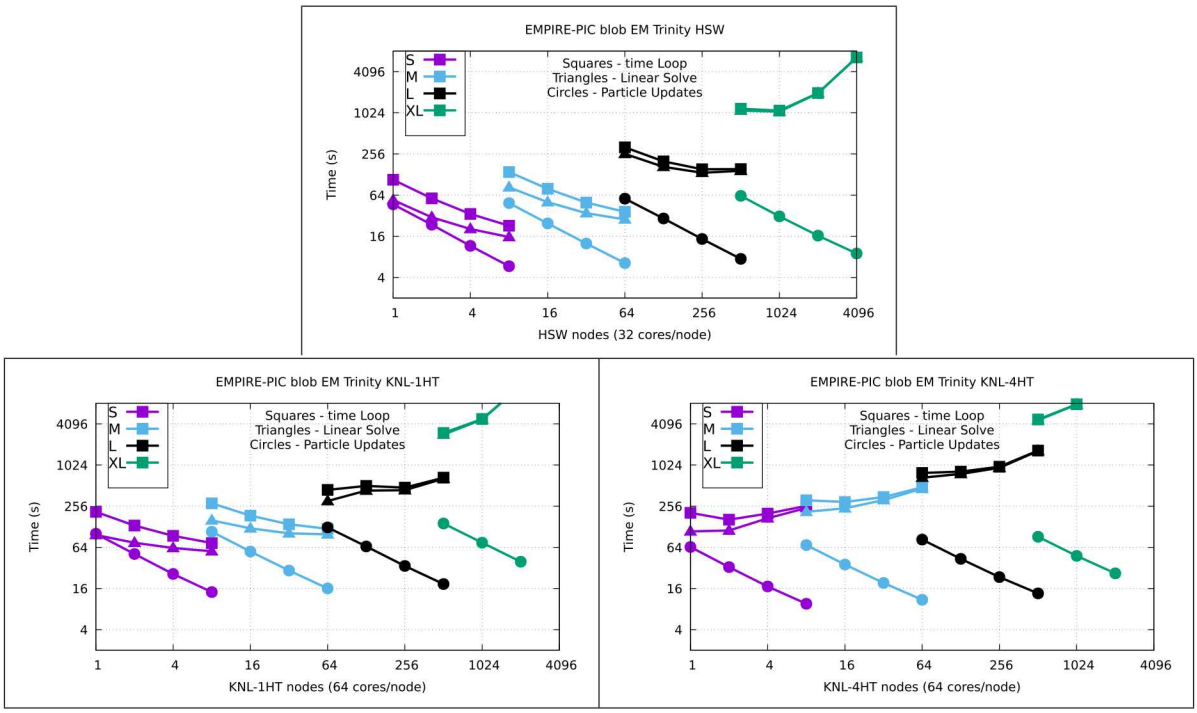


Figure 2.10: EM blob simulation, Trinity/Haswell, 22-August-2018.

The strong-scaling on Trinity/Haswell shows some benefit for meshes S, M, and L meshes, and on Trinity/KNL for S and M meshes. Strong scaling appears to be detrimental for XL and XXL.

However, a more pressing issue is growth in iterations, as seen in Figure 2.11. Iteration counts are approximately 2.5x higher when moving from L to XL. Additionally, iterations are growing at time advances.

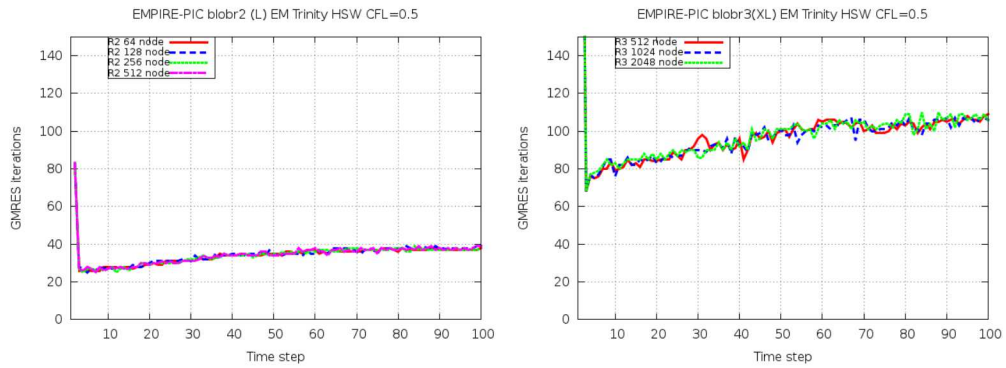


Figure 2.11: GMRES iterations per time step for EM blob L and XL, Trinity/Haswell, 22-August-2018.

2.2.4 Path Forward

We propose an approach to solve (2.3) as follows. Instead of solving iteratively with GMRES, (2.3) can be solved exactly with upper and lower triangular solves based on the factorization (2.4). This requires inverting the diagonal blocks of the upper triangular factor in (2.4). The inverse of the $(1,1)$ block Q_B can be approximated with preconditioned CG. The $(2,2)$ diagonal term S_E will be formed exactly, and its inverse approximated with CG preconditioned with Maxwell-specific AMG. This has two main advantages:

1. Global GMRES is no longer necessary, which avoids long term recurrences and eliminates associated orthogonalization costs.

2. When a strong form of the curl C is available, assembly of $K = Q_B C$ and $K^T = C^T Q_B$ will no longer necessary, and the Q_B system will only need to be inverted once (rather than twice).

The proposed solve has been implemented in Trilinos and preliminary testing done using the Panzer stand-alone driver `mini-EM`. Initial comparative results using `mini-EM` are given in Figure 2.12. `mini-EM` was built with OpenMP enabled and run in two configurations: 32 MPI processes per node, one thread per MPI process; and 2 MPI processes per node, 16 threads per MPI process. The Q_B block is solved with diagonally-scaled CG. The smoother used in the S_E solve was Chebyshev smoother, which matches that used in the Trinity runs. Note that the proposed block solve is approximately 5x faster faster than the current solve for the S, M, and L runs. Future work includes

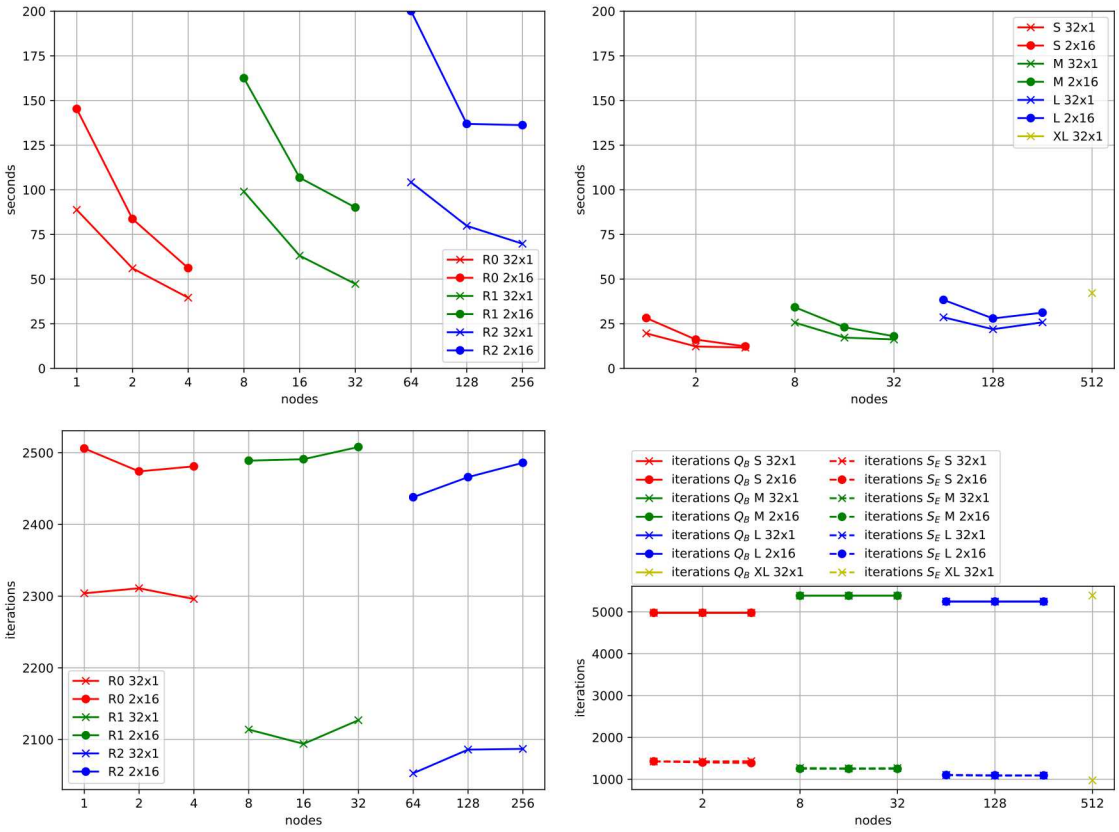


Figure 2.12: Timing and iteration comparison of current (left) and proposed block (right) solver in `mini-EM` for blob meshes S, M, L, and XL on NERSC's Cori Haswell partition.

modifying the EMPIRE solver interfaces appropriately and testing within EMPIRE, as well as implementing a strong form of the curl operator.

Chapter 3

Time Integration

The EMPIRE code will require a suite of time-integration strategies both for the stand-alone EM-PIC solves and when coupled to EMPIRE-Fluid. Currently for EM-PIC, EMPIRE is using a Leapfrog scheme for particle advances and a backward Euler implicit stepper for the electromagnetic field solves. The expectation is for components to provide a larger selection of routines including higher-order methods and IMEX capabilities for coupling with the fluids code.

As part of this milestone, the Tempus time-integration package in Trilinos is being integrated into EMPIRE and run on several problems of interest. The current list of development tasks for this milestone includes

- Implement Leapfrog stepper as a general capability in Tempus.
- Integrate Tempus into EMPIRE.
- Evaluate accuracy and performance on a simple unit tests.

Additionally, Tempus has current and developing capabilities that will be of use in the future to EMPIRE.

- Partitioned IMEX scheme stepper (EMPIRE-Hybrid)
- Provide second time-derivative steppers (second-order electromagnetic solves)
- Sensitivity analysis
- Embedded error analysis (variable time-stepping)

Tempus has many contributors for the capabilities reported below, and they include

- Integrating Tempus into EMPIRE - Roger Pawlowski, Edward Phillips, Curtis Ober
- Partitioned IMEX scheme stepper - Eric Cyr, Curtis Ober
- Provide second time-derivative steppers - Irina Tezaur
- Sensitivity analysis - Eric Phipps
- Embedded error analysis - Sidafa Conde

3.1 Background

Tempus is a software framework that provides a general infrastructure for the time evolution of ordinary-differential equations (ODEs), and partial-differential equations (PDEs) through a variety of general integration schemes, which can range from small systems of equations to large-scale transient simulations requiring exascale computing. Examples of time-integration methods available are

- Forward Euler
- Backward Euler
- Backward Differentiation Formula (BDF2)
- Leapfrog
- Trapezoidal
- Explicit Runge-Kutta
- Diagonally Implicit Runge-Kutta (DIRK) methods
- Newmark-β
  - Explicit A-form
  - Implicit A-form
  - Implicit D-form
- Hilber-Hughes-Taylor (HHT-α)
- Implicit/Explicit Runge-Kutta (IMEX-RK) methods
- Partitioned IMEX-RK methods
- First-order Operator Split

Forward sensitivity analysis is available for many of these steppers, and the Runge-Kutta methods have embedded error analysis capabilities to support variable time-step control. More details are provided in later sections.

The above milestone tasks fall into two categories, (1) developing and testing capabilities with Tempus for future use, and (2) demonstration of some these capabilities within EMPIRE. In the following sections, we discuss the Tempus and EMPIRE tests used for order of accuracy assessment and their associated results.

3.2 Tempus Tests

The following tests cover a wide range of time-integration needs (linear equations with analytic solutions to nonlinear equations and second-order ODEs), and are extensively used in Tempus testing. Here we give a description of each test used in testing the steppers discussed in later sections.

3.2.1 SinCos Test

This second-order ODE is a canonical Sine-Cosine differential equation

$$\ddot{\mathbf{x}} = -\mathbf{x}$$

with a few enhancements, which can also be used to test a system of first-order ODEs. We start with the exact solution to the differential equation

$$\begin{aligned} x_0(t) &= a + b * \sin((f/L) * t + \phi) \\ x_1(t) &= b * (f/L) * \cos((f/L) * t + \phi) \end{aligned}$$

then the form of the system of first-order ODEs is

$$\begin{aligned} \frac{d}{dt} x_0(t) &= x_1(t) \\ \frac{d}{dt} x_1(t) &= \left(\frac{f}{L}\right)^2 (a - x_0(t)) \end{aligned}$$

where the default parameter values are $a = 0$, $f = 1$, and $L = 1$, and the initial conditions

$$\begin{aligned} x_0(t_0 = 0) &= \gamma_0 [= 0] \\ x_1(t_0 = 0) &= \gamma_1 [= 1] \end{aligned}$$

determine the remaining coefficients

$$\begin{aligned} \phi &= \arctan(((f/L)/\gamma_1) * (\gamma_0 - a)) - (f/L) * t_0 [= 0] \\ b &= \gamma_1 / ((f/L) * cos((f/L) * t_0 + \phi)) [= 1] \end{aligned}$$

A typical solution for the SinCon test is shown in Fig. 3.2.

3.2.2 Partitioned van der Pol model

This problem is a canonical equation of a nonlinear oscillator [7, pp. 111-115] [8, pp. 4-5] for an electrical circuit. In implicit ODE form, $\mathscr{F}(\dot{x}, x, t) = 0$, the scaled problem can be written as

$$\begin{aligned} \dot{x}_0(t) - x_1(t) &= 0 \\ \dot{x}_1(t) - [(1 - x_0^2) x_1 - x_0]/\varepsilon &= 0 \end{aligned}$$

where the initial conditions are

$$x_0(t_0 = 0) = 2$$
$$x_1(t_0 = 0) = 0$$

and the initial time derivatives are

$$\dot{x}_0(t_0 = 0) = x_1(t_0 = 0) = 0$$
$$\dot{x}_1(t_0 = 0) = [(1 - x_0^2)x_1 - x_0]/\varepsilon = -2/\varepsilon$$

This test can also be used for a partitioned IMEX-RK time stepper. We need to rewrite this in the following form

$$M(z,t)\,\dot{z} + G(z,t) + F(z,t) = 0,$$
$$\mathscr{G}(\dot{z},z,t) + F(z,t) = 0,$$

where $\mathscr{G}(\dot{z},z,t) = M(z,t)\,\dot{z} + G(z,t)$, $M(z,t)$ is the mass matrix, $F(z,t)$ is the operator representing the "slow" physics (and evolved explicitly), and $G(z,t)$ is the operator representing the "fast" physics. For the van der Pol problem, we can separate the terms as follows

$$z = \left\{ \begin{array}{c} y \\ x \end{array} \right\} = \left\{ \begin{array}{c} x_0 \\ x_1 \end{array} \right\}, \quad F(z,t) = \left\{ \begin{array}{c} F^y(x,y,t) \\ F^x(x,y,t) \end{array} \right\} = \left\{ \begin{array}{c} -x_1 \\ x_0/\varepsilon \end{array} \right\},$$

and

$$G(z,t) = \left\{ \begin{array}{c} 0 \\ G^x(x,y,t) \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ -(1 - x_0^2)x_1/\varepsilon \end{array} \right\}$$

where $M(z,t) = I$ is the identity matrix.

Thus the explicit van der Pol model formulated for the partitioned IMEX-RK is

$$F^y(x,y,t) = \dot{x}_0(t) - x_1(t) = 0$$
$$F^x(x,y,t) = \dot{x}_1(t) + x_0/\varepsilon = 0$$

and the implicit van der Pol model formulated for the partitioned IMEX-RK is

$$G^x(x,y,t) = \dot{x}_1(t) - (1 - x_0^2)x_1/\varepsilon = 0$$

Noting that $G^y(x,y,t) = \dot{x}_0(t) = 0$ is not needed. In Figure 3.1, a typical solution is shown.

3.2.3 Damped Harmonic Oscillator Test

Consider the ODE:

$$m\ddot{x} + c\dot{x} + kx = f$$

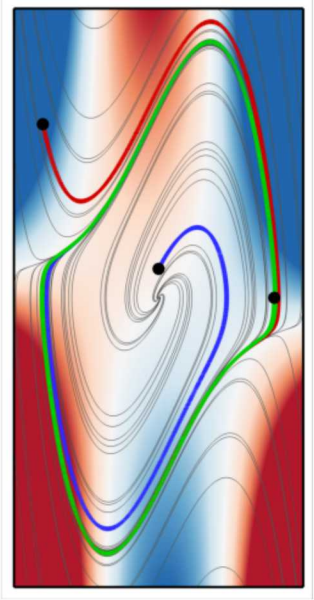


Figure 3.1: Tempus time integration of Van der Pols strange attractor and limit cycle. All initial conditions reach the limit cycle (green), either from small initial values (blue) or large initial values (red).

where $k \geq 0$ is a constant, c is a constant damping parameter, f is a constant forcing parameter, and $m > 0$ is a constant mass parameter, with initial conditions are:

$$\begin{aligned} x(0) &= 0 \\ \dot{x}(0) &= 1 \end{aligned}$$

It is straight-forward to show that the exact solution to this ODE is:

$$\begin{aligned} x(t) &= t(1+0.5\tilde{f}t), && if \;\; k = c = 0 \\ &= \frac{(\tilde{c}-\tilde{f})}{\tilde{c}^2}(1-e^{-\tilde{c}t}) + \frac{f}{c}t, && if \;\; k = 0, c \neq 0 \\ &= \frac{1}{\sqrt{\tilde{k}}}\sin(\sqrt{\tilde{k}}t) + \frac{f}{k}\left(1-\cos(\sqrt{\tilde{k}}t)\right), && if \;\; k > 0, c = 0 \end{aligned}$$

where $\tilde{c} \equiv c/m$, $\tilde{k} \equiv k/m$ and $\tilde{f} \equiv f/m$. While it is possible to derive the solution to this ODE for the case when $k > 0$ and $c \neq 0$, we do not consider that case here. When $c = k = 0$, $m = 1$, and $f = -1$, our ODE simplifies to a canonical differential equation model of a ball thrown up in the air, with a parabolic trajectory solution, namely

$$x(t) = t(1-0.5t)$$

where $t \in [0,2]$. When $c = f = 0$ and $m = k = 1$, this test is equivalent to the SinCos model.

3.3 Results

Here we show results for new steppers available for EMPIRE, and have been demonstrated with Tempus regression and verification testing. We also show that EMPIRE can utilize several steppers and achieve higher-order time integration.

3.3.1 Tempus Results

3.3.1.1 Leapfrog Stepper

The Leapfrog stepper is used to solve the following governing equation

$$\mathbf{M}\ddot{\mathbf{x}}^{n+1} + \mathbf{K}\mathbf{x}^{n+1} = \mathbf{F}\left(t^{n+1}\right) \tag{3.1}$$

which is similar to the governing equation for Newmark-β, except $\mathbf{C} = 0$. Solving Eq. (3.1) for $\ddot{\mathbf{x}}^{n+1}$

$$\ddot{\mathbf{x}}^{n+1} = \mathbf{M}^{-1}\left[\mathbf{F}\left(t^{n+1}\right) - \mathbf{K}\mathbf{x}^{n+1}\right] = f(\mathbf{x}^{n+1}, t^{n+1})$$

we can note that this is an explicit update for $\mathbf{x}^{n+1}$. The familiar steps to solve this can be written

$$\begin{aligned}
\mathbf{x}^{n+1} &= \mathbf{x}^{n} + \Delta t^{n}\dot{\mathbf{x}}^{n+1/2} \\
\ddot{\mathbf{x}}^{n+1} &= f(\mathbf{x}^{n+1}, t^{n+1}) \\
\dot{\mathbf{x}}^{n+3/2} &= \dot{\mathbf{x}}^{n+1/2} + \Delta t^{n}\ddot{\mathbf{x}}^{n+1}
\end{aligned}$$

To start up Leapfrog, we need to take a half step for $\dot{\mathbf{x}}$

$$\dot{\mathbf{x}}^{n+1/2} = \dot{\mathbf{x}}^{n} + \frac{\Delta t^{n}}{2}\ddot{\mathbf{x}}^{n}$$

where $\ddot{\mathbf{x}}^{n} = f(\mathbf{x}^{n}, t^{n})$, and similarly to finish a time step, we only need to take a final half step for $\dot{\mathbf{x}}$

$$\dot{\mathbf{x}}^{n+1} = \dot{\mathbf{x}}^{n+1/2} + \frac{\Delta t^{n}}{2}\ddot{\mathbf{x}}^{n+1}$$

In Figure 3.2b, the order of accuracy for the Leapfrog stepper is shown over several orders of magnitude.

3.3.1.2 Partitioned IMEX Stepper

Partitioned IMEX-RK [9, 10] is similar to the IMEX-RK, except a portion of the solution only requires explicit integration, and should not be part of the implicit solution to reduce computational costs. Again our ODE can be written as

$$\begin{aligned}
M(z,t)\,\dot{z} + G(z,t) + F(z,t) = 0, \\
\mathscr{G}(\dot{z}, z, t) + F(z,t) = 0,
\end{aligned}$$

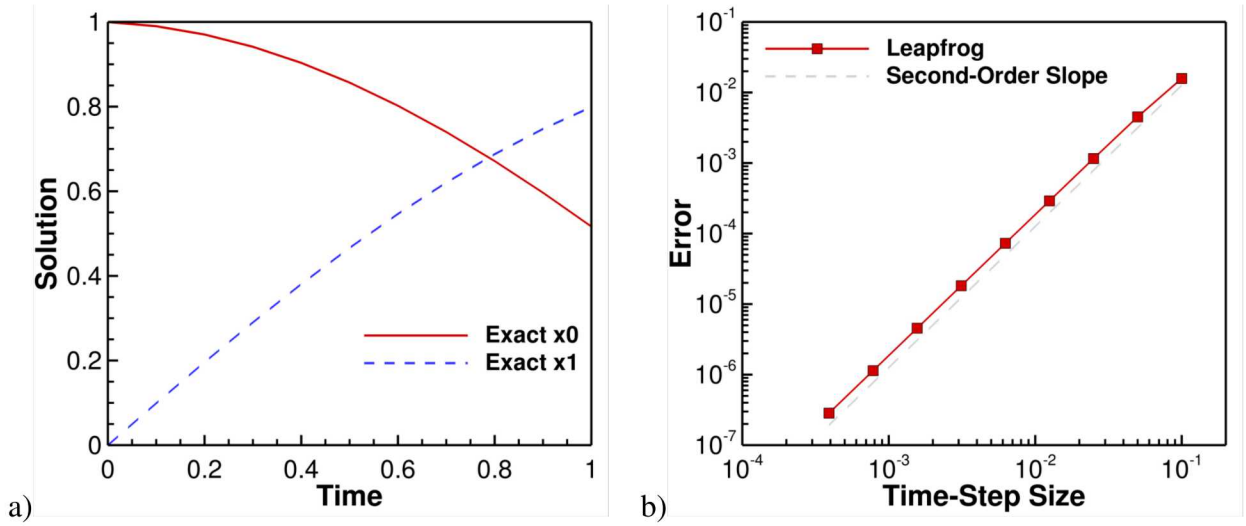


Figure 3.2: a) Exact solution for the SinCos test, and b) the order of convergence for the Leapfrog stepper.

but now

$$z = \left\{ \begin{array}{c} y \\ x \end{array} \right\}, \; F(z,t) = \left\{ \begin{array}{c} F^y(x,y,t) \\ F^x(x,y,t) \end{array} \right\}, \text{ and } G(z,t) = \left\{ \begin{array}{c} 0 \\ G^x(x,y,t) \end{array} \right\}$$

where z is the product vector of y and x, $F(z,t)$ is still the "slow" physics (and evolved explicitly), and $G(z,t)$ is still the "fast" physics (and evolved implicitly), but a portion of the solution vector, y, is "explicit-only" and is only evolved by $F^y(x,y,t)$, while x is the Implicit/Explicit (IMEX) solution vector, and is evolved explicitly by $F^x(x,y,t)$ evolved implicitly by $G^x(x,y,t)$. Note we can expand this to explicitly show all the terms as

$$\begin{aligned} M^y(x,y,t)\,\dot{y} + F^y(x,y,t) &= 0, \\ M^x(x,y,t)\,\dot{x} + F^x(x,y,t) + G^x(x,y,t) &= 0, \end{aligned}$$

or

$$\left\{ \begin{array}{c} \dot{y} \\ \dot{x} \end{array} \right\} + \left\{ \begin{array}{c} f^y \\ f^x \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ g^x \end{array} \right\} = 0$$

where

$$\begin{aligned} f^y(x,y,t) &= M^y(x,y,t)^{-1} F^y(x,y,t) \\ f^x(x,y,t) &= M^x(x,y,t)^{-1} F^x(x,y,t) \\ g^x(x,y,t) &= M^x(x,y,t)^{-1} G^x(x,y,t) \end{aligned}$$

or

$$\dot{z} + f(x,y,t) + g(x,y,t) = 0,$$

where $f(x,y,t) = M(x,y,t)^{-1} F(x,y,t)$, and $g(x,y,t) = M(x,y,t)^{-1} G(x,y,t)$. Using Butcher tableaus for the explicit terms

$$\begin{array}{c|c} \hat{c} & \hat{A} \\ \hline & \hat{b}^T \end{array} \quad \text{and for implicit terms} \quad \begin{array}{c|c} c & A \\ \hline & b^T \end{array},$$

the basic scheme for this partitioned, s-stage, IMEX-RK is

$$\begin{aligned} Z_i &= Z_{n-1} - \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij}\, f(Z_j, \hat{t}_j) - \Delta t \sum_{j=1}^{i} a_{ij}\, g(Z_j, t_j) \quad \text{for } i = 1 \ldots s, \\ z_n &= z_{n-1} - \Delta t \sum_{i=1}^{s} \left[\hat{b}_i\, f(Z_i, \hat{t}_i) + b_i\, g(Z_i, t_i) \right] \end{aligned}$$

or expanded

$$\begin{aligned} Y_i &= y_{n-1} - \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij}\, f^y(Z_j, \hat{t}_j) \qquad \text{for } i = 1 \ldots s, \\ X_i &= x_{n-1} - \Delta t \sum_{j=1}^{i-1} \hat{a}_{ij}\, f^x(Z_j, \hat{t}_j) - \Delta t \sum_{j=1}^{i} a_{ij}\, g^x(Z_j, t_j) \quad \text{for } i = 1 \ldots s, \\ y_n &= y_{n-1} - \Delta t \sum_{i=1}^{s} \hat{b}_i\, f^y(X_i, Y_i, \hat{t}_i) \\ x_n &= x_{n-1} - \Delta t \sum_{i=1}^{s} \left[\hat{b}_i\, f^x(Z_i, \hat{t}_i) + b_i\, g^x(Z_i, t_i) \right] \end{aligned}$$

where $\hat{t}_i = t_{n-1} + \hat{c}_i \Delta t$ and $t_i = t_{n-1} + c_i \Delta t$.

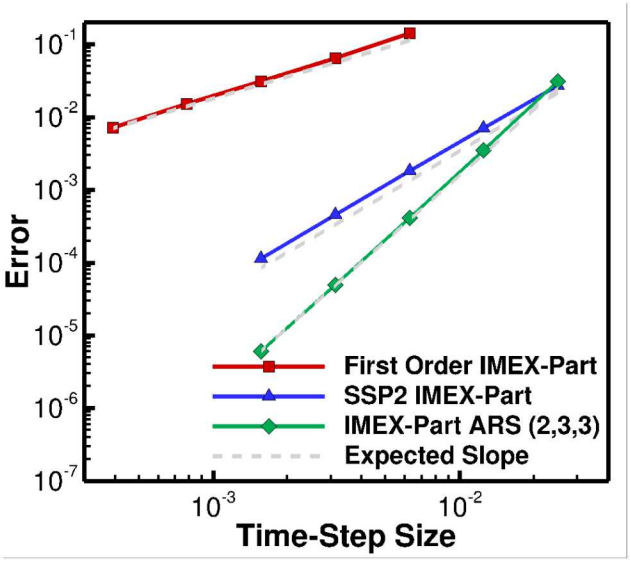


Figure 3.3: Order of convergence for the partitioned IMEX-RK stepper with the partitioned van der Pol test.

3.3.1.3 HHT-α Stepper

Here, the HHT-Alpha scheme in predictor/corrector form (see equations (10) and (13)-(19) in [11]) has been implemented in Tempus by Irina Tezaur. There are three parameters in the scheme: α_f, β and γ, all of which must be in the range $[0,1]$. When $\alpha_f = 0$, the scheme reduces to the Newmark-β scheme (see Tempus::StepperNewmark for details). Like the Newmark-β scheme, the HHT-Alpha scheme can be either first or second order accurate, and either explicit or implicit.

The Hilber-Hughes-Taylor (HHT-α) α-method evolves the following governing equation

$$\mathbf{M}\ddot{\mathbf{x}}^{n+1} + \mathbf{C}\dot{\mathbf{x}}^{n+1-\alpha_f} + \mathbf{K}\mathbf{x}^{n+1-\alpha_f} = \mathbf{F}\left(t^{n+1-\alpha_f}\right) \tag{3.2}$$

Note that in the original formulation [12] $\alpha = -\alpha_f$. The HHT-α method is second-order accurate and unconditionally stable if

$$\beta = \frac{1}{4}\left(1+\alpha_f\right)^2$$
$$\gamma = \frac{1}{2}+\alpha_f$$

If $\alpha_f = 0$, the HHT-α method is the trapezoidal rule.

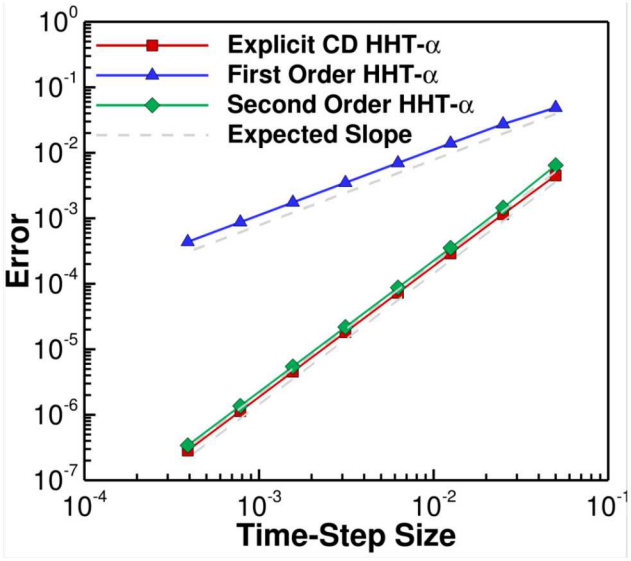


Figure 3.4: Order of convergence for the HHT-a stepper with the SinCos test.

3.3.1.4 Newmark Stepper

Here, the Newmark scheme in predictor/corrector form (see equations (34)-(35) in [13]) has been implemented in Tempus by Irina Tezaur. Newmark is second order accurate if $\gamma = 0.5$; otherwise it is first order accurate. Some additional properties about the Newmark scheme can be found at http://opensees.berkeley.edu/wiki/index.php/Newmark_Method.

Newmark has two parameters: β and γ, both of which need to be in the range $[0,1]$. Newmark can be an explicit or implicit method, depending on the value of the β parameter. If $\beta = 0$, the method is explicit. Regardless of whether the method is implicit or explicit, a linear solve is required. This linear solve can be optimized, however, for the explicit case by lumping the mass matrix.

The Newmark-β method [14] can be obtained from the HHT-α method by setting $\alpha_f = 0$ and evolves the governing equation

$$\mathbf{M}\ddot{\mathbf{x}}^{n+1} + \mathbf{C}\dot{\mathbf{x}}^{n+1} + \mathbf{K}\mathbf{x}^{n+1} = \mathbf{F}\left(t^{n+1}\right) \tag{3.3}$$

The Newmark-β method is second-order accurate and unconditionally stable if

$$\beta = \frac{1}{4}$$
$$\gamma = \frac{1}{2}$$

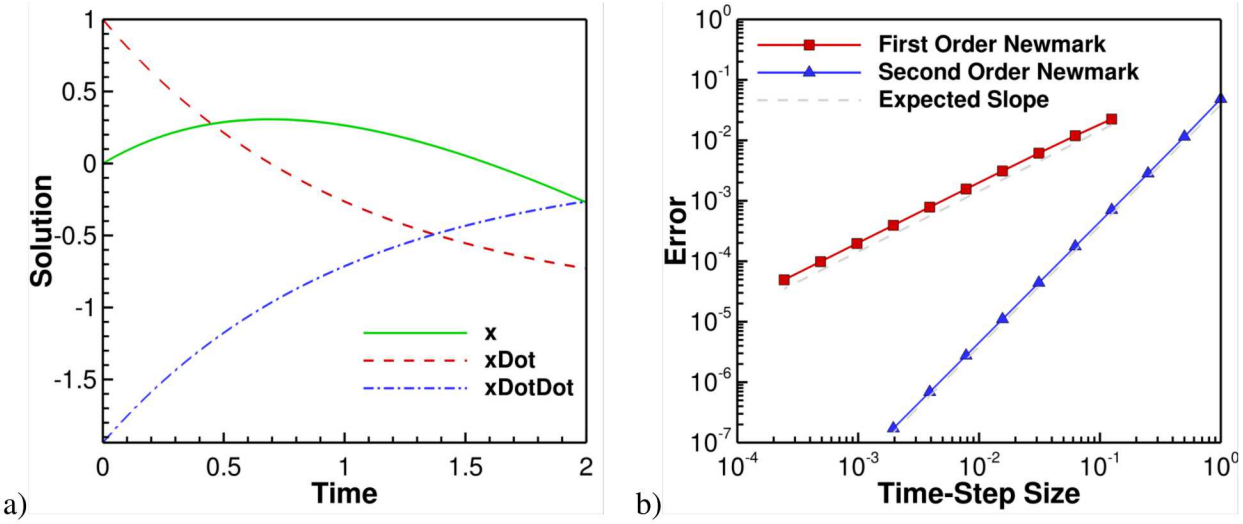


Figure 3.5: Order of convergence for the Newmark stepper with the Harmonic Oscillator test.

3.3.1.5 Operator-Split Stepper

This stepper is a stepper of steppers and allows applications to create a sequence steppers and their associated ModelEvaluators, which will be executed in-order to obtain a first-order splitting. This can be written as a composition

$$\mathbf{x}^n = (f \circ g \circ h \circ \ldots)(\mathbf{x}^{n-1}) \tag{3.4}$$

where for the functions f, g, and h, the following sequence of evaluations would be applied

$$\mathbf{x}^* = h(\mathbf{x}^{n-1})$$
$$\mathbf{x}^{**} = g(\mathbf{x}^*)$$
$$\mathbf{x}^n = f(\mathbf{x}^{**})$$

In EMPIRE, there are two PIC steppers (explicit and implicit), an electrostatic stepper, and an electromagnetic stepper. The PIC steppers and an electrostatic stepper are utilizing their EMPIRE implementations. The electromagnetic stepper can use any of the Tempus steppers available, and are demonstrated in the following section.

3.3.2 EMPIRE Results

Currently, Tempus reproduces the same results on almost all the EMPIRE tests within the regression tolerances, using Tempus' Backward Euler stepper. Four of these tests have been selected to maintain coverage for Tempus integration

- LangmuirWave Test - sets up a standing Langmuir wave by perturbing the velocity at $t = 0$. This problem exercises particles using explicit time stepping with electrostatics.
- ImplicitSlab Test - sets up an expanding slab of plasma integrated in time using an electrostatic finite element extension of the Energy Conserving Semi-Implicit Method from Lapenta, G. "Exactly Energy Conserving Implicit Moment Particle in Cell Formulation." JCP 2016. This problem exercises particles using implicit time stepping with electrostatics.
- OscillatingEField1D Test - more details below.
- EMWaveInPlasma3D - this test is of a TEM microwave propagating in a plasma. The derivation of this test was taken from section 4.12 of "Introduction to Plasma Physics and Controlled Fusion", 3rd edition, by Francis F. Chen. This problem exercises particles with electromagnetics.

3.3.2.1 OscillatingEField1D Test

From the test description, this test exercises basic electromagnetic behavior without particles on a (quasi) 1D domain. It simulates a single cycle of a half wavelength of a sinusoidal wave. The characteristics of the wave are

- wavelength: 2 meters
- velocity: 299792458.0 meters / second (speed of light, c)
- period: 6.67128190396304e-09 seconds

The general solution for the E-field and B-field is

$$\begin{aligned} E_0(x,y,t) &= 0.1 \times 10^{10} \\ E_1(x,y,t) &= 1.0 \times 10^{10} \sin(\pi x) cos(c\pi t) \\ B(x,y,t) &= -1.0 \times 10^{10} \cos(\pi x) sin(c\pi t)/c \end{aligned}$$

The E_0 component is set to a non-zero constant to decrease numerical noise for the purposes of the test.

In Figure 3.6a, the final solution is shown for the OscillatingEField1D Test, which is easily compared against the initial conditions since the solution is periodic. In Figure 3.6b, the order of accuracy is shown for a variety of Tempus steppers, ranging from first through fifth order.

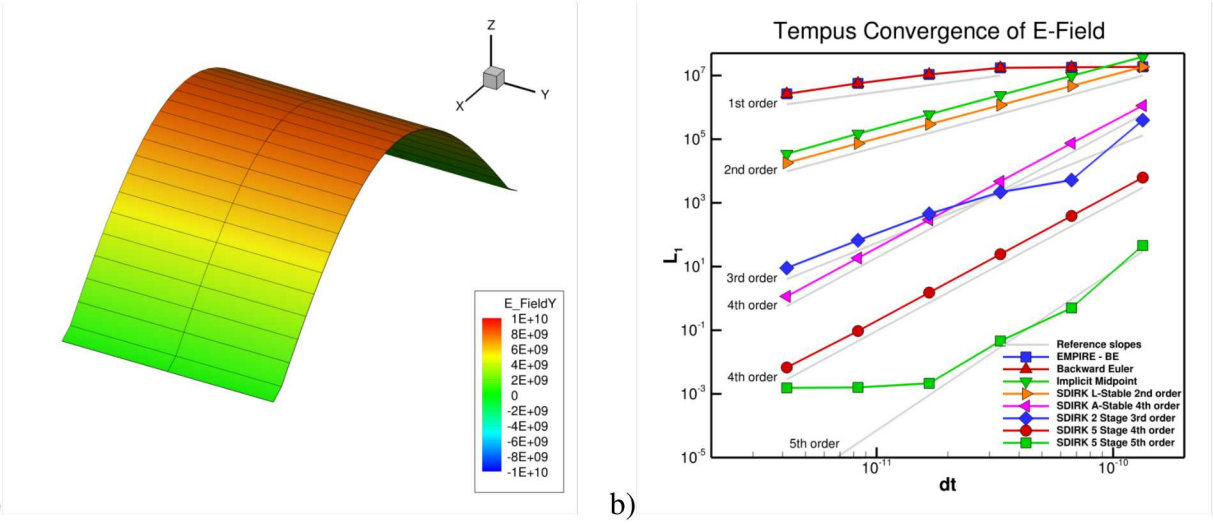


Figure 3.6: a) Solution to OscillatingEField1D test, and b) the order of accuracy for various steppers.

The Tempus Backward Euler results match the EMPIRE Backward Euler to four digits, including the non-asymptotic region at large Δt. The convergence plateaus for the fifth order method. It is believed that this is due to numerical precision for this problem.

It should be noted that the higher-order methods are currently only available on a feature branch. It is expected that this branch will be merged before the end of FY2018. This delay is primarily due to the rapid development of EMPIRE and the changing interfaces to the time loop and ModelEvaluators.

3.4 Tempus Features Available for EMPIRE

There are a couple features that are available for future use by EMPIRE: embedded Runge-Kutta methods for time-step control, and sensitivity analysis.

3.4.1 Explicit Runge-Kutta Embedded Pairs

Explicit Runge-Kutta Embedded Pairs have been developed and implemented in Tempus by Sidafa Conde. These methods share their stages and typically differ in order. Therefore, let us assume that the vectors b and $\tilde{b}$ correspond to order p and $\tilde{p}$, respectively. In general the assumption is $\tilde{p} \leq p$. However, in practice typically we assume that $\tilde{p} = p - 1$. The Butcher tableau of an ERK method and its embedded pair is given in Table 3.1. Some well-known embedded RK methods can be found in [7].

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_s	a_{s1}	a_{s2}	$\dots$	$a_{s,s-1}$	
	b_1^T	b_2^T	$\dots$	b_{s-1}^T	b_s^T
	$\tilde{b}_1^T$	$\tilde{b}_2^T$	$\dots$	$\tilde{b}_{s-1}^T$	$\tilde{b}_s^T$

Table 3.1: The Butcher tableau of an ERK method and its embedded pair.

Embedded RK methods are crucial for automatic step size control, i.e. the method automatically chooses the step size in each step. Let us consider the numerical solutions u_{n+1} and $\tilde{u}_{n+1}$ with order p and $p-1$, respectively. Taking into account ERK methods and their embedded pairs we have an approximation for the global error vector which is denoted by $\tilde{e}_{n+1}$. It can be calculated as

$$\tilde{e}_{n+1} = u_{n+1} - \tilde{u}_{n+1} = \Delta t \sum_{j=1}^{s} \left(b_j - \tilde{b}_j\right) F(y_j).$$

The above formula shows that the Butcher form is efficient from the programming point of view.

The global error of the order $p-1$ method can be approximated by the term $M(\Delta t)^p$, where M is an appropriate constant. On the other hand, from the local truncation errors we can conclude that $|\tilde{e}_{n+1}| \approx M(\Delta t)^p$. Therefore, if the relation $|\tilde{e}_{n+1}| < \varepsilon$ holds in case of a given tolerance $\varepsilon > 0$, then we accept the numerical solution $\tilde{u}_{n+1}$. Otherwise, we have to choose a new step size Δt_{new}. In this case we have the relation $\tilde{e}_{n+1} \approx M(\Delta t_{\text{new}})^p < \varepsilon$. Since $|\tilde{e}_{n+1}| \approx M(\Delta t)^p$, it implies that

$$\frac{M(\Delta t_{\text{new}})^p}{M(\Delta t)^p} < \frac{\varepsilon}{\tilde{e}_{n+1}}.$$

Hence, it requires the condition

$$\Delta t_{\text{new}} := \Delta t \left(\frac{\varepsilon}{\tilde{e}_{n+1}}\right)^{\frac{1}{p}}.$$

In Figure 3.7(top), the solution to the van der Pol problem is shown with error-controlled time-step selection. As the solution varies, the time-step is appropriately changed to maintain an error tolerance. In Figure 3.7(bottom), the time-step initially increases due to the near steady state of the solution. As the solution varies, the time-step decreases to maintain the error tolerance, and then again increases with the near steady state.

3.4.1.1 Time-Step Control Strategies

There exists a variety of error control algorithms. Tempus has the following error control strategies:

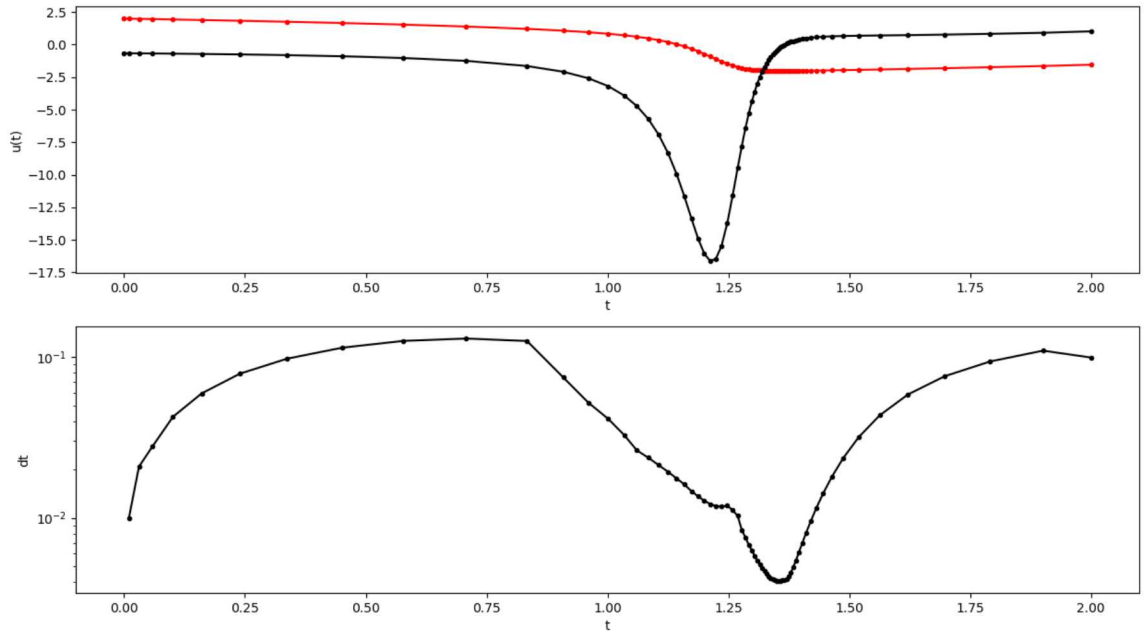


Figure 3.7: Solution to the Van der Pol problem (top) with error-controlled time-step size selection (bottom).

Proportional-Integral-Derivative Controller (PID)

$$(\Delta t)_{n+1} = (\Delta t)_n \left(e_n^{-k_1/p} e_{n-1}^{k_2/p} e_{n-2}^{-k_3/p} \right) \tag{3.5}$$

Proportional-Integral Controller (PI)

$$(\Delta t)_{n+1} = (\Delta t)_n \left(e_n^{-k_1/p} e_{n-1}^{k_2/p} \right) \tag{3.6}$$

Integral Controller (I)

$$(\Delta t)_{n+1} = (\Delta t)_n \left(e_n^{-k_1/p} \right) \tag{3.7}$$

In Figure 3.8, the work-precision diagram is shown. Using the PID controller, the accuracy is very close to the prescribed tolerance (top figure). As the tolerance changes, the number of time steps (and the work) also changes (bottom figure), and thus the controller is behaving as expected.

3.4.2 Transient Adjoint Sensitivity Capabilities

Transient adjoint sensitivity capabilities have been developed in Tempus by Eric Phipps for embedded optimization and uncertainty quanification (UQ), and provides analytic, transient adjoint sensitivity capabilities to all Trilinos users through the Tempus time integration package.

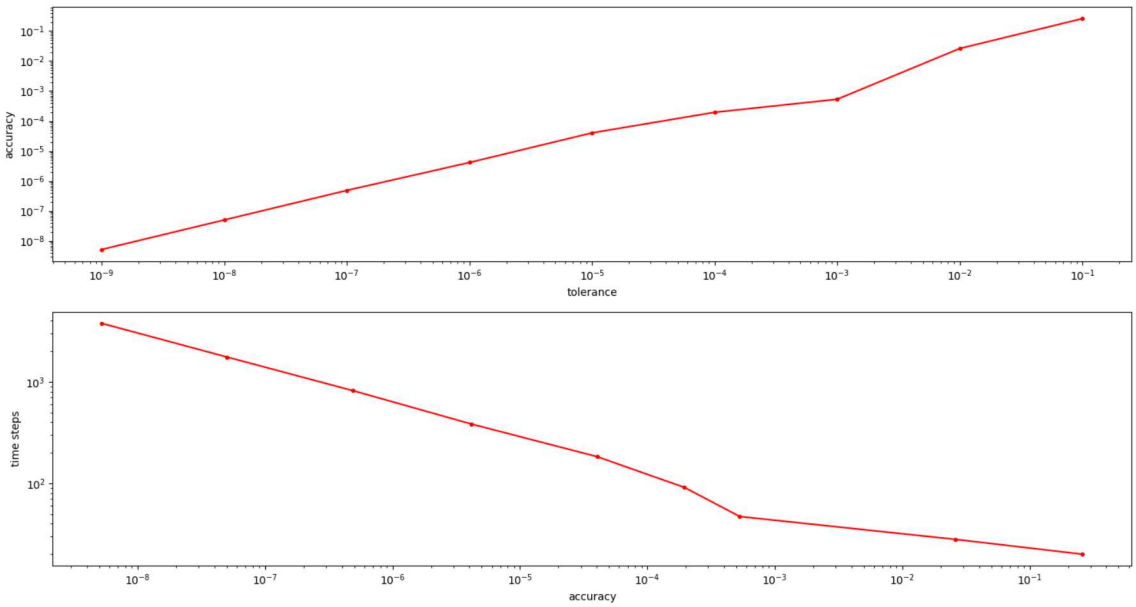


Figure 3.8: Work-Precision Diagram. Accuracy of the solution as a function of tolerance (top figure), and the number of time-steps as a function of accuracy (bottom). Both trends are indications of a working time-step controller.

Both forward and adjoint sensitivities are available. For large-scale optimization problems, adjoint sensitivities enable efficient large-scale optimization, UQ, and derivative computation.

Forward, adjoint, and pseudo-transient (forward/adjoint) sensitivities are supported in Tempus for most time integrators. Additionally, ROL reduced-space interface leveraging these sensitivities is available, which enables transient embedded optimization. Lastly, a ROL full-space interface nearly complete.

In Figure 3.9, forward and adjoint convergence is shown BDF2 forward transient solution and adjoint sensitivity applied to an analytic test problem. The expected second-order convergence is demonstrated. Coverage for these capabilities is supported through multiple regression tests.

3.5 Discussion

Incorporating Tempus into EMPIRE provides access to a variety of time-integration capabilities, ranging from standard time integrators (e.g., Backward Euler) to high-order Runge-Kutta methods to operator-split and IMEX methods. Additionally, Tempus is able to integrate second-order PDEs with several methods. For Explicit Runge-Kutta methods, Tempus has embedded algorithms for error control, which include several time-step control strategies. Tempus can also calculate forward, adjoint and pseudo-transient senstivities.

During the 2018 fiscal year, Tempus was incorporated into EMPIRE, and was able to reproduce

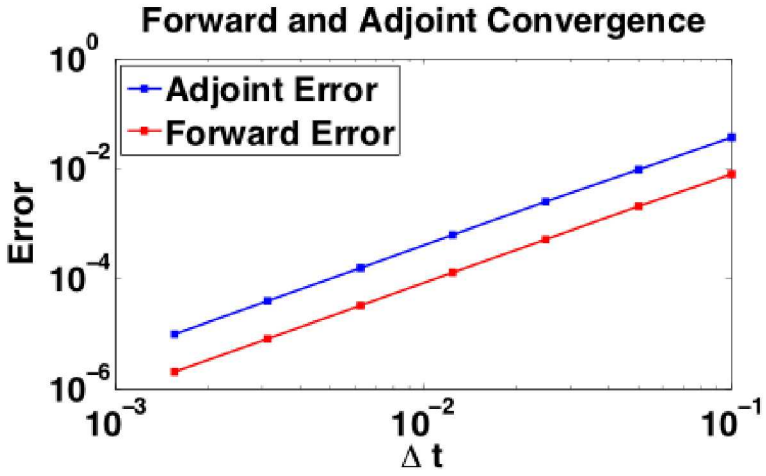


Figure 3.9: Forward and adjoint convergence for transient sensitivity.

EMPIRE's Backward Euler capabilities. Only a few tests are not currently passing and require investigation to determine the cause. Four of EMPIRE's tests that span the PIC and E-field solves are currently providing coverage for Tempus capabilities.

Higher-order time integration has also been demonstrated in EMPIRE with Tempus, but still requires merging into the head of the repository. It is expected that this will occur before the end of the fiscal year.

Follow on work will include integrating Tempus into the EMPIRE-Fluids and EMPIRE-Hybrid codes. As mentioned earlier the partitioned IMEX schemes are targeting the EMPIRE-Fluids to evolve the hydrodynamics explicitly while the second time derivative steppers (HHT-α and Newmark-β) are targeting research on alternate formulations in the EMPIRE-PIC code. We note that the second time derivative stepper development is already being leveraged by a separate ASC funded code (Albany/LCM), used for solid mechanics research.

Chapter 4

Verification

The overall verification effort for EMPIRE is an integral part of maturing the codebase with regards to *code verification* (finding bugs) and *solution verification* (building confidence in the implementation). Throughout this chapter, when we discuss *verification* we are referring to *solution verification*. Oberkampf and Roy [15] state: "*Solution verification* addresses the question of whether a given simulation (i.e., numerical approximation) of a mathematical model is sufficiently accurate for its intended use." This chapter does not discuss the efforts of developers to develop unit tests that cover the whole codebase, although that is a crucial part of the verification process.

4.1 Verification Testing Methodologies

In the EMPIRE test suite, we have three general classes of verification tests:

Regression test A test with an accepted 'gold' solution that is used to detect when code behavior changes. These often have very tight tolerances (generally to less than one part per million) to detect minute changes in behavior.

Analytic test A test with a known analytic solution that ensures that the code is reproducing known physical behavior to acceptable levels of accuracy. These tests often have looser tolerances than regression tests to allow them to run in a reasonable amount of time.

Convergence test A set of tests that incrementally refine the simulation in space and/or time to determine the actual convergence rate to an analytic solution for a given set of physics.

Some verification tests are able to span all three categories by having one criteria that detects if the behavior has changed compared to a known solution within the acceptable level of accuracy (a regression and analytic test), and then for a longer test, the simulation can be run two or three times and a convergence rate calculated and compared with the theoretical value.

The regression tests are generally run using the `ctest` utility and are curated to be run quickly and in a development environment. The longer analytic and convergence tests are generally run using the `vvtest` utility, which was developed specifically for verification and validation of scientific computing codes. By using `vvtest`, the analysis of the simulation can be more in-depth

and nuanced than is typically done with regression tests and the metrics can be more concrete. Some examples of more concrete metrics include: deviations from particle paths, particle velocity distributions, and the frequency or amplitude profile of EM waves.

4.2 Case Study: TEM Wave in Plasma

As a case study of one of the verification tests currently in the EMPIRE test suite, we will look at an infinite, planar TEM wave traveling through an infinite neutral plasma. This problem was chosen because it is an electromagnetic example problem that has the electromagnetic field interacting with the plasma and, with a few assumptions, has an analytic solution.

4.2.1 Analytical Solution

The following equations of the analytic solution come from Chen [16], where the differences between a TEM wave in a vacuum and in a plasma, holding the wave vector constant, are derived.

For this problem, the controlling parameters are the number density of the plasma $n_0 = 10^{15}\text{m}^{-3}$, the maximum magnitude of the electric field $E_{\text{mag}} = 100$ V/m (typical airport radar at ~ 3 m), and the vacuum frequency of the EM wave $f_v \approx 1.420$ GHz with $\omega_v = 2\pi f_v$ (the microwave hydrogen line).

For this problem, two assumptions were made: first, that the EM wave is of such high frequency that the ions are assumed stationary, and second, that the $J \times B$ forces on particles are negligible. This means that the electrons are assumed to only oscillate linearly in the plane of the electric field (in reality, they trace elongated ovals in that plane).

The plasma frequency is

$$\omega_p = \sqrt{\frac{n_0 q^2}{m_e \varepsilon_0}} \approx 1.784 \times 10^9 \text{ rad/sec} \tag{4.1}$$

with q being the elementary charge, m_e is the electron mass, and ε_0 the vacuum permittivity. This result can be used to find the actual frequency of the wave in the plasma

$$f = \frac{\omega}{2\pi} = \frac{1}{2\pi}\sqrt{\omega_p^2 + \omega_v^2} \approx 1.448 \text{ GHz} \tag{4.2}$$

which equates to a 1.9% frequency shift from the wave in a vacuum.

Because the wave is an infinite, steady wave, we can calculate the constant phase velocity

$$v_p = \sqrt{\frac{f}{f_v}}c \approx 1.02c > c \tag{4.3}$$

which is always greater the speed of light in a vacuum and corresponds nicely to the 2% frequency shift from above.

The maximum initial electron velocity is

$$v_e = \frac{qE_{\text{mag}}}{m_e \omega} \approx 1,932.5 \text{ m/s} \tag{4.4}$$

and is initialized in phase with the electric field.

The maximum magnitude of the magnetic field is defined to be congruous with the magnitude of the electric field

$$B_{\text{mag}} = \frac{\lambda}{2\pi} \frac{E_{\text{mag}}}{c^2} \left(\frac{n_0 q^2}{m_e \varepsilon_0 \omega} + \omega \right) \approx 3.53 \times 10^{-7} \text{ T}. \tag{4.5}$$

4.2.2 Computational Description

Based on the analytic derivation above, the computational description is straightforward. The problem is set up on a 3D domain with periodic boundary conditions in each direction, essentially giving an infinite spatial extent for the planar wave. The EM wave is traveling aligned to the Z-axis of the computational mesh, with most of the computational elements spanning the Z-direction. The lateral directions (X- and Y-directions) each have a constant 4 elements, regardless of refinement level. The elements are defined to be cubes, which implies that the computational domain contracts with refinement. The time step is fixed throughout a simulation (i.e., there is no time step adaptation), but it does decrease with increasing refinement levels.

The CFL number of the simulation decreases as the simulation is refined because the algorithms presently implemented in EMPIRE are theoretically second-order accurate in space and first-order accurate in time. The CFL number can be calculated for a given refinement level r by

$$CFL = \frac{c\Delta t}{\Delta x} = \frac{cTN_{e0}}{\lambda N_{t0}} \frac{\sqrt{2}^r}{2^r} \approx \frac{0.49}{\sqrt{2}^r} \tag{4.6}$$

where c is the speed of light, T is the period of the wave, N_{e0} is the unrefined number of elements in the longitudinal direction, N_{t0} is the unrefined number of time steps for a single period, and λ is the wavelength of the wave in the plasma. If the spatial and temporal convergence orders are equal, the CFL number would then be constant upon refinement.

Because the ions are assumed stationary, they are forced to be immovable in the simulation as a way to speed up the computations. The initial electron velocity is confined to the transverse direction in the plane of the electric field.

4.2.3 Discussion of Results

The primary system response quantity (SRQ) for this simulation is the L_{inf} norm of the E-field error after one full oscillation. The error is primarily due to discrepancies in the E_y component (the

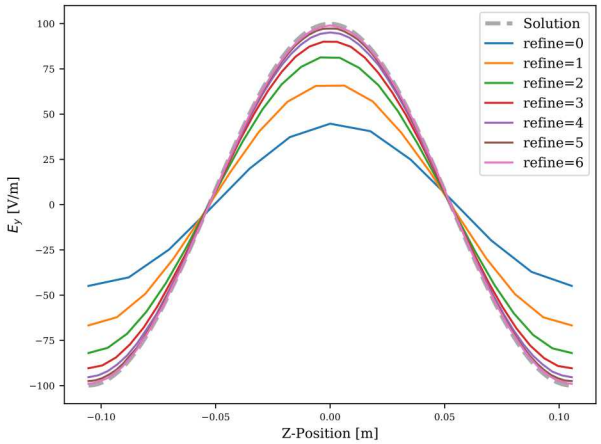


Figure 4.1: Convergence of the E_y field after one full oscillation upon spatial and temporal refinement. These simulations were run with 320 particles per cell. This demonstrates visually that the simulation is converging to the correct solution.

E_x and E_z components are zero by comparison). Figure 4.1 shows the convergence of the final E_y fields for several simulations.

To address the assumption about electron motion, the particle velocities at the final time step were analyzed (see Figure 4.2) for induced motion perpendicular to the E-field direction. The maximum velocity in the E-field direction was found to be 1,946 m/s, while the maximum velocities in the other directions were 20.3 m/s and 13.6 m/s, a difference of about 1%.

To address the assumption about immobile ions, another simulation was run where the ions were allowed to move. No detectible velocity was present at the final time step.

4.2.4 Convergence

The convergence analysis was performed according to the refinement schedule found in Oberkampf and Roy [15] for analyzing spatial and temporal order verification using only two numerical solutions. The benefit of their refinement schedule is that it allows analysis of both space and time convergence simultaneously against a theoretical convergence rate.

The TEM wave in a neutral plasma problem has been run seven times at different refinement levels with the L_{inf} norm of the E-field error being the SRQ of interest. As the convergence methodology does not consider the number of computational particles used in PIC codes, this convergence process was repeated five times a different number of particles per cell in each run. The results of this analysis can be found in Figure 4.3. The most refined simulation took about half an hour on 16 cores.

As the number of particles per cell increases, the convergence rate approaches the theoretical

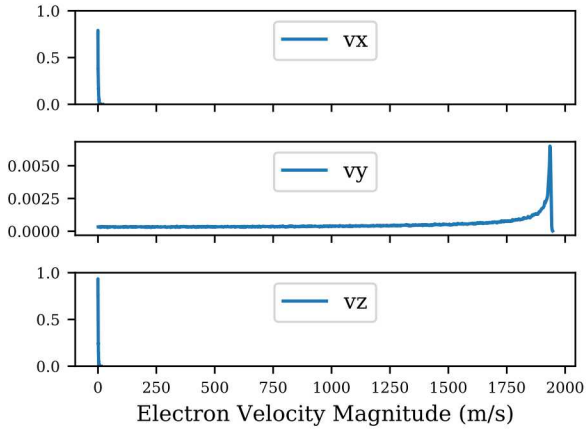


Figure 4.2: A plot showing the PDFs for the magnitude of the electron velocity for each velocity component. It demonstrates that the assumption that the electrons move very little out of the E-field plane is valid.

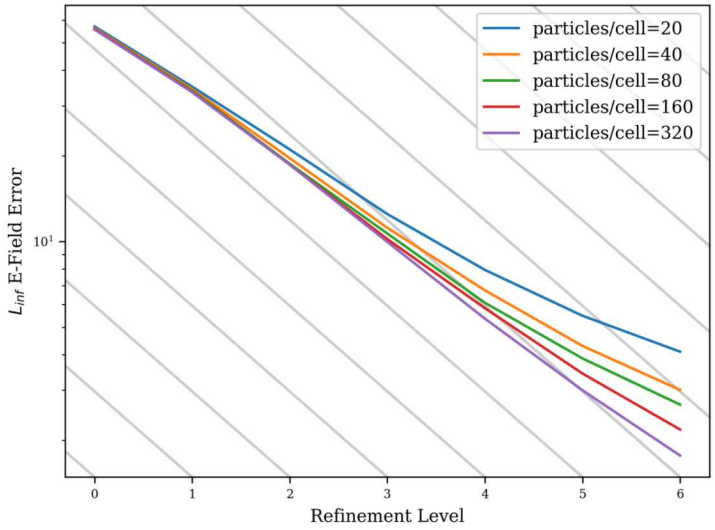


Figure 4.3: A convergence plot for the TEM wave in plasma test depicting the L_{inf} norm of the E-field error versus the refinement level (analytic values for E_y vary up to 100V/m). The gray lines represent the expected slope of the error for second-order space and first-order time. Several curves are presented, each with a unique constant number of particles per cell. Note how the convergence behavior approaches the theoretical rate as the number of particles per cell is increased.

values of second order in space and first order in time that we would expect from EMPIRE. It is interesting to see how the convergence rate levels off for the simulation series with lower particle per cell values. This effect demonstrates the accuracy limits for this simulation as a function of discretization error in the particle field.

4.3 Case Study: Langmuir waves and Landau Damping

Langmuir waves are longitudinal waves that can be simulated both electrostatically and electromagnetically. For the example in the report the waves are simulated electrostatically. The damping behavior of the waves depends on wavenumber: small wavenumbers (large wavelengths) are not damped, but large wavenumbers (short wavelengths) are collsionalessly damped. The convergence of both will be shown in this report. Landau damping is a kinetic effect that depends on the shape of the distribution, especially at the phase velocity, $v_p = \omega/k$, of the longitudinal wave. Landau damping occurs when there are more particles just below the phase velocity than just above the phase velocity. As the wave speeds-up more particles than it slows-down, the wave transfers energy to the particles. The theory that EMPIRE is compared to is based on small amplitude waves; for a wave with a larger amplitude, the volume in phase space is increased, which includes more particles. It will be shown that as the amplitude is decreased EMPIRE does converge to the small amplitude theoretical limit, but the convergence is less expensive numerically to run in a larger amplitude regime. Nonlinear Landau damping is still an open research question, see Mouhot and Villani [17] or Herr [18].

4.3.1 Analytical Solution

The details of the calculation of the longitudinal dispersion relation is in textbooks such as Chen [16], Nicholson [19], and Bittencourt [20], so will not be reproduced here. The dispersion relation can be written as:

$$k^2 + 1 + i\sqrt{\pi} C \, \mathrm{e}^{-C^2} (1 + \mathrm{erf}(iC)) = 0, \tag{4.7}$$

where k is the wavenumber normalized to the Debye length, $k\lambda_d$, C is the normalized phase velocity, $C = \omega/(k\sqrt{2})$, erf is the error function, and ω is the frequency normalized to the plasma frequency, ω/ω_p. When C is small the dispersion relation can be approximation as:

$$\omega^2 = 1 + 3k^2 \tag{4.8}$$

$$\omega_i = \frac{\sqrt{\pi/8}}{k^3} \exp\left(-\frac{1}{2k^2} - \frac{3}{2}\right) \tag{4.9}$$

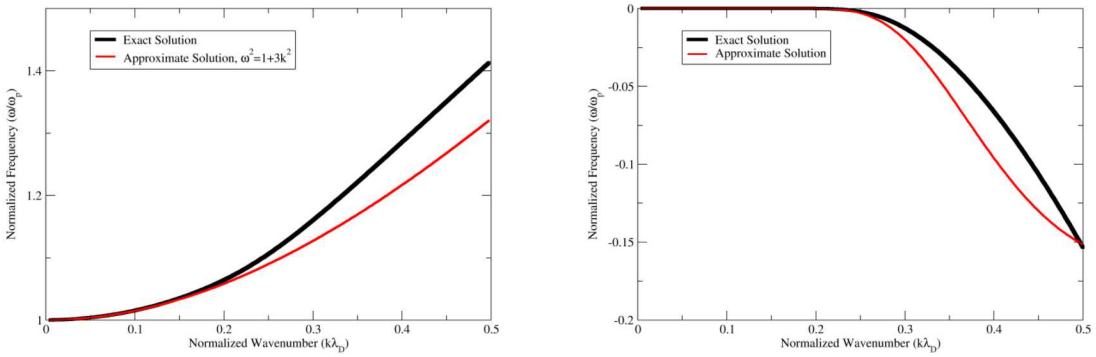
The dispersion relation is shown in Fig. 4.4. It should be noted that (4.9) is accurate enough for the Langmuir wave, but not accurate enough for the Landau damping example that is at a larger wavenumber.

Table 4.1: Simulation parameters for the Langmuir wave and Landau damping cases.

Parameters	Langmuir Wave Case	Landau Damping Case
Electron Density	1×10^{14} m^{-3}	
Wavelength/Length of the system	5 cm	
Plasma Frequency (f_p)	89.786628 MHz	
Plasma Period	11.13751591 ns	
Electron Temperature	2 eV	10 eV
Debye Length	0.1051315096 cm	0.2350784418 cm

Table 4.2: Wave parameters for the Langmuir wave and Landau damping cases.

Parameters	Langmuir Wave Case	Landau Damping Case
$k\lambda_d$	0.1321121513	0.2954082823
$Re(\omega/\omega_p)$	1.0268090997	1.154502855
$Img(\omega/\omega_p)$	-2.06944×10^{-11}	-0.01119157
$Re(C)$	0.0959218	0.241159
Wave Kinetic Energy	$v_{perturbation}/v_t$	
0.1 eV	0.316229	0.141423
0.01 eV	0.1	0.044722
0.001 eV	0.0316229	0.0141423



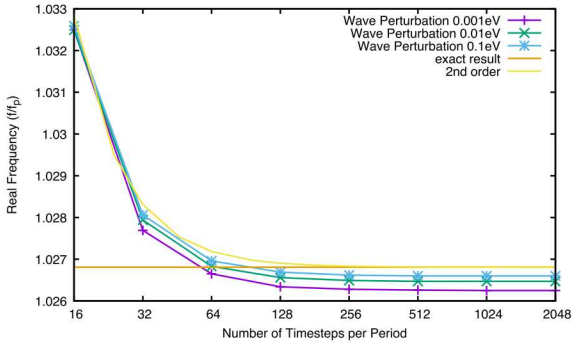
(a) Real part of the dispersion relation showing the exact solution and an approximation, $\omega^2 = 1 + 3k^2$.

(b) The imaginary part of the dispersion relation showing the exact solution using the Dawson integral and a common approximation, $\omega_i = \sqrt{\pi/8}/k^3 \exp(-1/(2k^2) - 3/2)$. Since the imaginary term is negative in ω it leads to temporal decay of the wave amplitude.

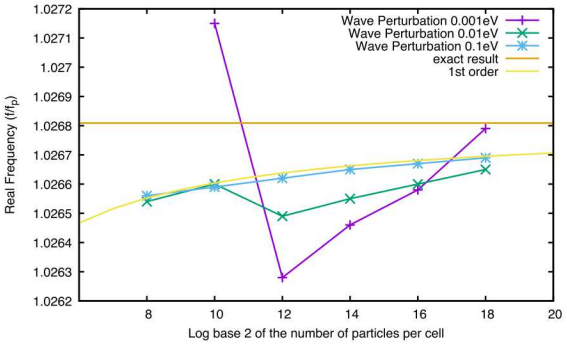
Figure 4.4: Real and imaginary parts of the dispersion relationship. The approximation is more accurate for small k values. The convergence analysis is done for two values of k: one near $k = 0.13$, where the damping of the wave is small (negligible), and the other near $k = 0.3$, where the damping of the wave is large.

4.3.2 Convergence

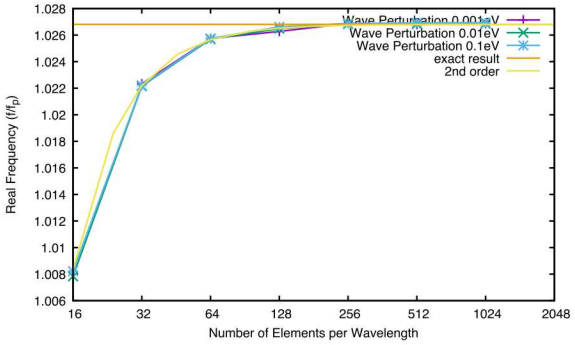
For the undamped Langmuir wave, shown in Figures 4.5 and 4.6, the real frequency converges (Fig. 4.5) at second order as the mesh is refined, at first order as the number of particles is increased, and at second order as the time-step is decreased, as is expected in standard PIC[1]. In each case, since the other two quantities are fixed, convergences is not to the exact result, but is limited by the discretization error in the other two numerical quantities. Also shown in these plots is the convergence rate as the size of the wave perturbation is decreased. As the perturbation decreases it becomes harder to resolve the simulations, specifically in Fig. 4.5(b), where the number of particles needed for the simulation to be in the basin of convergence is increased as the perturbation magnitude is decreased. As for the imaginary part of the the frequency, shown in Fig. 4.6, the plot still show convergence to the theoretical value, in this case near zero. In this case, the error is dominated by the number of particles used. The damping term converges at first order in time and with number of particles, and at second order in space.



(a) Convergence of the wave frequency as a function of Δt while keeping 2^{12} particles per cell and the number of of cells at 128 per wavelength.

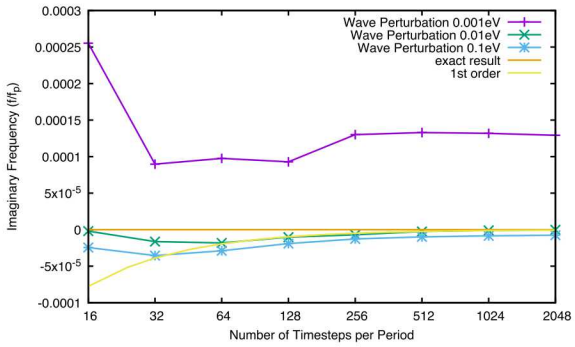


(b) Convergence of the wave frequency as a function of number of particles per cell while keeping $\Delta t = 1/(256 f_p)$ and the number of cells at 128 per wavelength

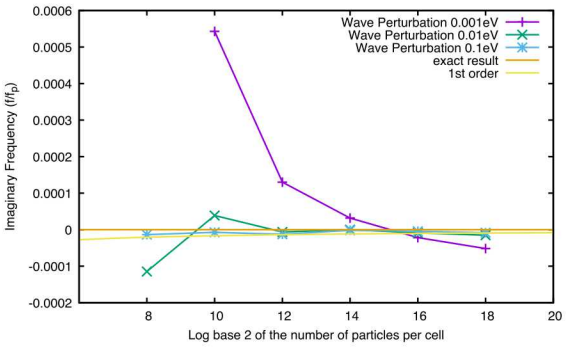


(c) Convergence of the wave frequency as a function of number of cells per wavelength while keeping 2^{12} particles per cell and $\Delta t = 1/(256 f_p)$

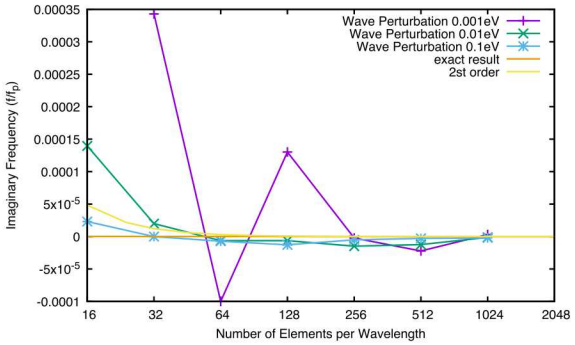
Figure 4.5: Convergence of the real part of the frequency of the Langmuir wave as a function of Δt, number of cells, and number of particles per cell.



(a) Convergence of the imaginary part of wave frequency as a function of Δt while keeping 2^{12} particles per cell and the number of cells at 128 per wavelength.



(b) Convergence of the imaginary part of wave frequency as a function of number of particles per cell while keeping $\Delta t = 1/(256 f_p)$ and the number of cells at 128 per wavelength



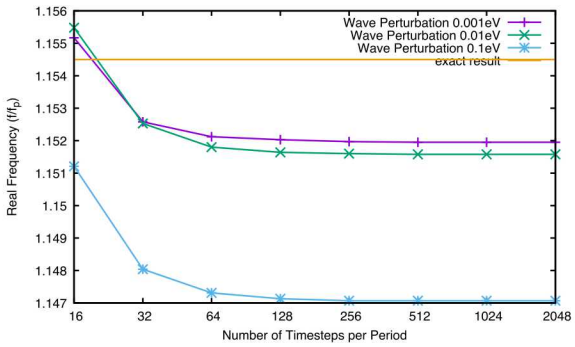
(c) Convergence of the imaginary part of wave frequency as a function of number of cells per wavelength while keeping 2^{12} particles per cell and $\Delta t = 1/(256 f_p)$

Figure 4.6: Convergence of the imaginary part of the frequency (growth or decay) of the Langmuir wave as a function of Δt, number of cells, and number of particles per cell. The imaginary part should be near zero. For these plots a positive number is a decay in the wave amplitude.

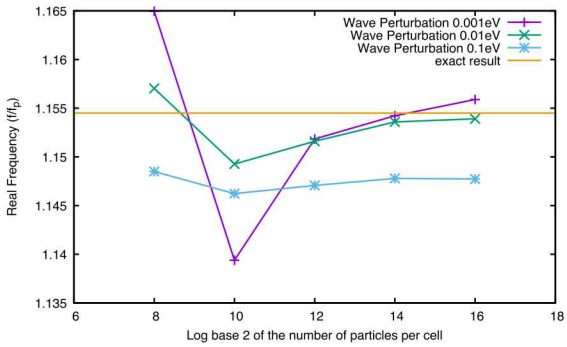
The Landau damping case, shown in Figures 4.7 and 4.8, show similar trends to the Langmuir wave convergence. The real part of the frequency, Fig. 4.7, shows convergence with time-step size, number of particles and mesh resolution; however, the solution is not as converged as it is for the Langmuir wave. It is unclear why the Landau damping case needs more resolution for the same accuracy. The imaginary part of the frequency is even harder to resolve. The simulations show a weak convergence to the theoretical results. Part of this is because the larger amplitude waves are showing convergence to a damping value near the theoretical value, getting closer to the theoretical results going from a perturbation of 0.1 eV to 0.01 eV. For the 0.001 eV case, the imaginary part of the frequency is converging in Fig. 4.8(a) to the same value as in the 2^{12} particles per cell and 128 cell per wave length cases seen in subfigures (b) and (c), but because the amplitude is smaller it is more difficult to resolve the wave.

Note that for computational expediency, it was chosen to perform these convergence studies in each parameter with the other two parameters not at their most refined value. This is adequate to determine convergence rates, but not for determining the converged solution value that EMPIRE is predicting.

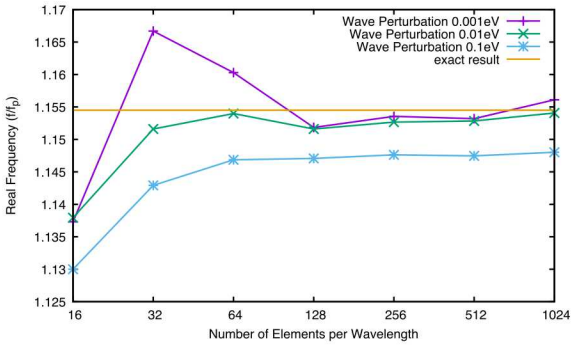
Figures 4.5-4.8 show (with some uncertainty) that the convergence is second order accurate in time, second order accurate in space, and first order accurate in the number of particles. With this you can construct a refinement schedule expanding on Oberkampf and Roy [15] analyzing for spatial and temporal order verification using only two numerical solutions to three numerical paramters. The benefit of this refinement schedule is that it allows analysis of both space, time, and particle convergence simultaneously against a theoretical convergence rate. This is show in Fig. /refrefinement. The refinement plot for the real part of Langmuir wave (a) is close to the theoretical value for for all values of the wave amplitude. The refinement plot imaginary part of the Langmuir wave (b) shows some systematic departure from the theoretical values. For the larger wave amplitude is looks like the convergence has stalled for the last refinement value. For the smallest wave amplitude the imaginary values is still converging; however, from this plot it is unclear if there is a convergence floor or that the two larger amplitudes have stoped converging. The refinement plot for the real part of the Landau damping case (c) the larger amplitude wave is converging to a value near the small amplitude limit. The smaller amplitude wave do show convergence for all the refinement levels. For the $0.01eV$ perturbation the error value has switched sign between 3 and 4 refinement level. For the imaginary part of the frequency for the Landau damping (d), the convergence is not as expected. For the largest amplitude, $0.1eV$, the simulation have converged to a nearby values. For the $0.01eV$ perturbation the error value has switched sign between 0 and 1 refinement level and then is converging to a value near the small amplitude theoretical value. The $0.01eV$ amplitude is converging to a value closer to the theoretical small amplitude value than the $0.1eV$ amplitude wave value. The smallest wave amplitude simulated is still converging, but the convergence rates is about half of that which is expected. There are still some open question in how EMPIRE is converging for the Landau damping case and this will be further investigated next year.



(a) Convergence of the real part of the wave frequency as a function of Δt while keeping 2^{12} particles per cell and the number of cells at 128 per wavelength.

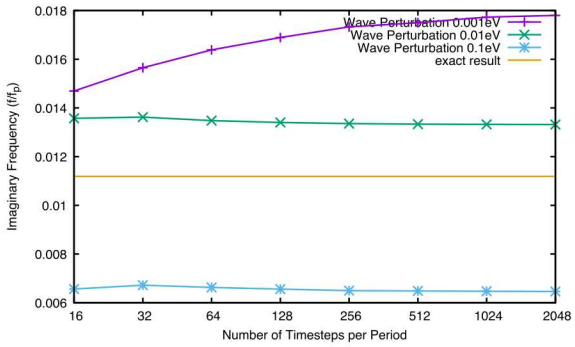


(b) Convergence of the real part of the wave frequency as a function of number of particles per cell while keeping $\Delta t = 1/(256 f_p)$ and the number of cells at 128 per wavelength

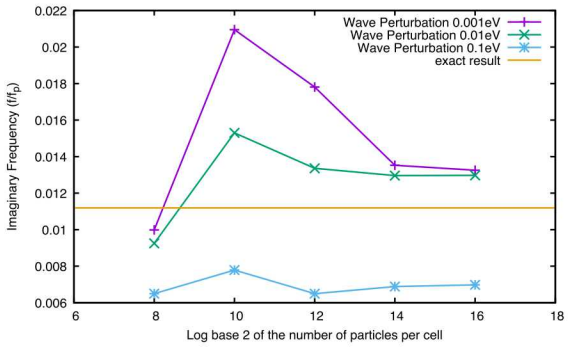


(c) Convergence of the real part of the wave frequency as a function of number of cells per wavelength while keeping 2^{12} particles per cell and $\Delta t = 1/(256 f_p)$

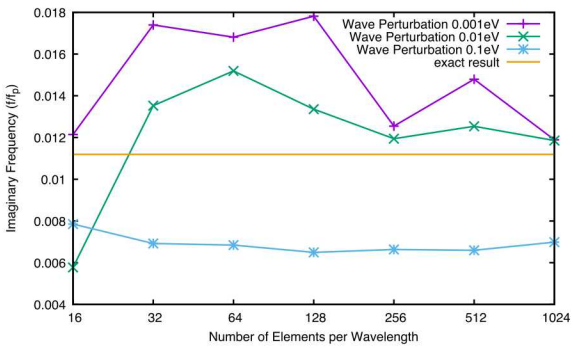
Figure 4.7: Convergence of the Langmuir wave frequency in the Landau damping case as a function of Δt, number of cells, and number of particles per cell.



(a) Convergence of the imaginary part of the wave frequency as a function of Δt while keeping 2^{12} particles per cell and the number of cells at 128 per wavelength.

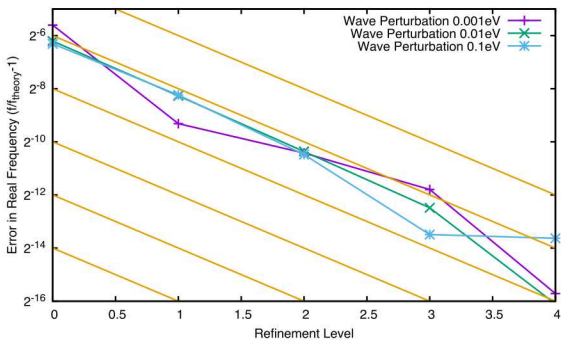


(b) Convergence of the imaginary part of the wave frequency as a function of number of particles per cell while keeping $\Delta t = 1/(256 f_p)$ and the number of cells at 128 per wavelength

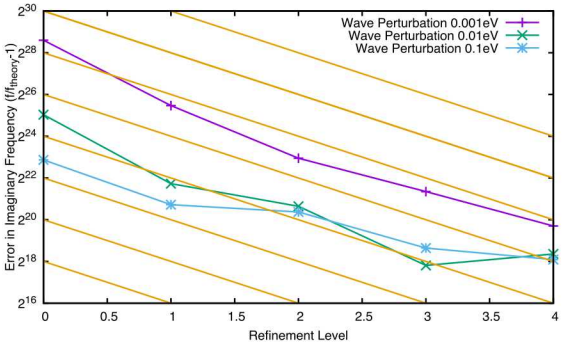


(c) Convergence of the imaginary part of the wave frequency as a function of number of cells per wavelength while keeping 2^{12} particles per cell and $\Delta t = 1/(256 f_p)$

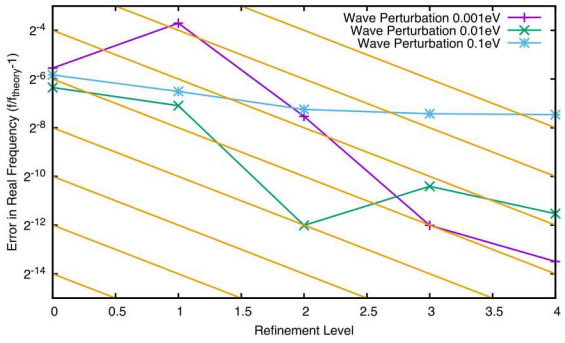
Figure 4.8: Convergence of the Langmuir wave damping frequency in the Landau damping case as a function of Δt, number of cells, and number of particles per cell. For these plots a positive number is a decay in the wave amplitude.



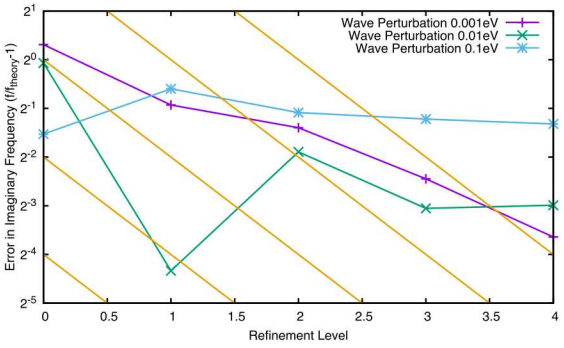
(a) Convergence of the real part of the wave frequency for the Langmuir wave case.



(b) Convergence of the imaginary part of the wave frequency for the Langmuir wave case.



(c) Convergence of the real part of the wave frequency for the Landau damping case.



(d) Convergence of the imaginary part of the wave frequency for the Landau damping case.

Figure 4.9: Refinement plots for a Langmuir wave and and Landau damping.

4.4 Verification Conclusion

The two examples shown above shows how EMPIRE converges to theoretical solutions in two different regimes; first for a cold fluid EM wave and then for a warm electrostatic wave with and without kinetic effects. These are representative of the range of problems that EMPIRE can solve. The results show the general trend that for electrostatic problems the time stepping and the spatial discretization is second-order accurate and the convergence in number of macro particles is first-order accurate, consistent with the theoretical rates of the numerical algorithms used. For electromagnetics, the spatial discretization is second-order accurate and the time stepping and convergence in number of macro particles is first-order accurate. A second-order electromagnetic time integrator will be implemented in EMPIRE soon.

Chapter 5

I/O

HPC application performance is often hindered by the platform's I/O subsystem because the hardware simply cannot store data as fast as the application can produce it. Historically users have not had many options to fix I/O bound problems other than reducing the amount of data that is made persistent. Fortunately, new platforms such as ATS-1 and ATS-2 feature Burst Buffer [21] devices that provide sizable amounts of nonvolatile memory (NVM) for temporary storage. This NVM provides a way for the platform to rapidly resolve periodic I/O spikes that are common in bulk synchronous parallel applications. The I/O community is actively working with application teams to help determine the best way to take advantage of this technology.

This section summarizes our experiences in adapting EMPIRE to leverage the Cray DataWarp [22] Burst Buffer available on the ATS-1 platform. In addition to evaluating I/O for mesh and particle result files, a new checkpoint restart capability was developed for EMPIRE using the new ATDM-developed FAODEL software. Initial testing at small scale on Mutrino indicates that the Burst Buffers dramatically improve I/O performance, and that FAODEL can offer additional performance benefits.

5.1 EMPIRE I/O Environment

In terms of I/O, EMPIRE executes in a manner that is similar to other HPC applications: after generating initial data or loading it from disk, the simulation stores a timestep's essential variables to result files that can be inspected at a later point in time. Result files are primarily used for visualization purposes, and require that the simulation data is converted into a file format (e.g., Exodus) that ensures compatibility with other tools. Given the scale at which EMPIRE is designed to run, it is important that EMPIRE's I/O facilities also provide an efficient mechanism for supporting checkpoint/restart. This mechanism allows users to run jobs that are longer than can be run in a single job allocation window, and mitigates system reliability issues that naturally occur on large-scale platforms. EMPIRE has two types of data that must be managed by the I/O software:

Mesh Field Data: EMPIRE uses a mesh to represent multiple variables in physical space. This field data can be exported to Exodus result files for visualization purposes. However, some of EMPIRE's internal field variables cannot be represented correctly in Exodus (e.g., multiple edge values) and must be interpolated to produce valid Exodus variables. This interpolation

adds overhead and prevents the Exodus result files from being used for restarts. As such, checkpoint data must be stored through separate mechanisms.

Particle Data: The bulk of EMPIRE's simulation data is particle data. An individual particle currently requires 80 bytes of state, but it is expected that an additional 16 bytes of data will be required to house random number generator state. Given the sheer size of the particle data, EMPIRE simulations typically store the data infrequently (e.g., never, end of simulation, or at hourly checkpoints). EMPIRE writes result files using the H5Hut [23] library, which produces HDF5 files that are readable in other applications.

Table 5.1 lists the amount of data that is expected to be generated for different outputs. The Exodus result file is approximately three times larger than the checkpoint size due to the generation of extra variables that are necessary for visualization.

Table 5.1: Estimated Output Sizes for Field (left) and Particle (right) data

# of Elements	Exodus Timestep Size	Checkpoint Size
337K	25MB	8MB
2.68M	202MB	67MB
20.7M	1.5GB	520MB
166M	12GB	4GB
1.332B	100GB	33GB

# of Particles	Checkpoint Size
16M	1.3GB
128M	10.7GB
1B	85GB
8.2B	687GB
65.6B	5.5TB

5.2 FAODEL

EMPIRE I/O work for FY18 focused on developing checkpoint/restart mechanisms that would be capable of managing both the particle and field data through a single library. As a demonstration of ATDM-developed technologies, the checkpoint/restart code was implemented using ATDM's FAODEL (Flexible, Asynchronous, Object Data-Exchange Libraries). This work was performed in such a way that the core checkpoint/restart software could be adapted to other libraries if FAODEL was deemed insufficient or inappropriate for EMPIRE. FAODEL is a collection of communication libraries for HPC platforms that make it easier for I/O developers to write custom data management services for applications. FAODEL provides a portable collection of low-level libraries I/O developers often need: RDMA-based messaging, network-memory management, a naming service for distributed resources, in-application RESTful services for introspection, and a key-blob service for migrating objects between applications, distributed memory, and storage. I/O developers can use FAODEL to rapidly migrate data from their MPI application to a separate set of caching nodes that can disseminate the data to other applications or offload to storage.

The goal of the FY18 work was to implement a basic checkpoint/restart service for EMPIRE that simply serializes EMPIRE data into FAODEL constructs that could then be written out in the most straightforward manner by the nodes. This approach reduces the complexity for the EMPIRE

team to use this work in real runs and provides a starting point in FY19 for experimenting with more sophisticated scenarios (e.g., using external nodes to aggregate data before I/O).

Checkpoint/restart units were developed to allow FAODEL to manage both particle and field data. The particle unit followed the same approach as the H5Hut unit and manually packs particles into an outgoing buffer, particle by particle. The mesh unit was written in a more efficient manner that copies data straight from the underlying Kokkos Views into an outgoing buffer. This approach greatly improves serialization time and represents an important opportunity for Sandia codes to use Kokkos data structures all the way through an application.

5.3 I/O Performance

Performance experiments were conducted on Mutrino to measure the amount of overhead created by I/O. These experiments adjusted parameters in EMPIRE's input decks to vary the size of the data, the target storage system, and the number of compute nodes used in the simulation. Each experiment generated a uniform distribution of particles in a mesh and then ran for ten timesteps. Simulations generated full checkpoints and result files at each timestep in order to maximize I/O load. EMPIRE was configured to use Kokkos' OpenMP backend to perform data-parallel computations efficiently on all of a node's cores. The experiments primarily focused on using a single rank per node.

Two storage targets were tested in these experiments: a Lustre-based scratch file system and the DataWarp Burst Buffer. Experiments targeting the Burst Buffer requested a striped-scratch topology with 2TB of capacity. Mutrino provisioned this storage using fragments on all six of its Burst Buffer nodes, resulting in a theoretical maximum write performance of 34GB/s. IOR performance tests of the allocation achieved 30GB/s of performance using an ideal I/O workload (i.e., multiple ranks on 90 nodes writing 512KB blocks of data to independent files).

For performance reasons, the experiments focused more on checkpoint operations than restart. Of the two, checkpoint performance is more important because checkpoints occur periodically in a simulation while restarts occur once per run. The experiments also focused more on particle data than field data due to the characteristics of the data. In addition to being larger in size, particles migrate between the ranks in the simulation and cause load balance issues for the I/O subsystem. In this section we examine the particle and field data independently in order to better report on their characteristics.

5.3.1 Particle Data

The first performance experiment for particle checkpointing used 64 compute nodes and varied the number of particles from 128M to 512M. As illustrated in Figure 5.1, the overhead to the application for checkpointing data grows linearly with the number of particles. The H5Hut method adds a sizable amount of overhead to the timestep when writing to Lustre. Part of this overhead comes

from the fact that a single output file is produced and ranks need to coordinate their appends. The Burst Buffer improves H5Hut performance significantly. However, FAODEL is more streamlined and achieves better performance. Haswell processors performed better than Knights Landing in all cases.

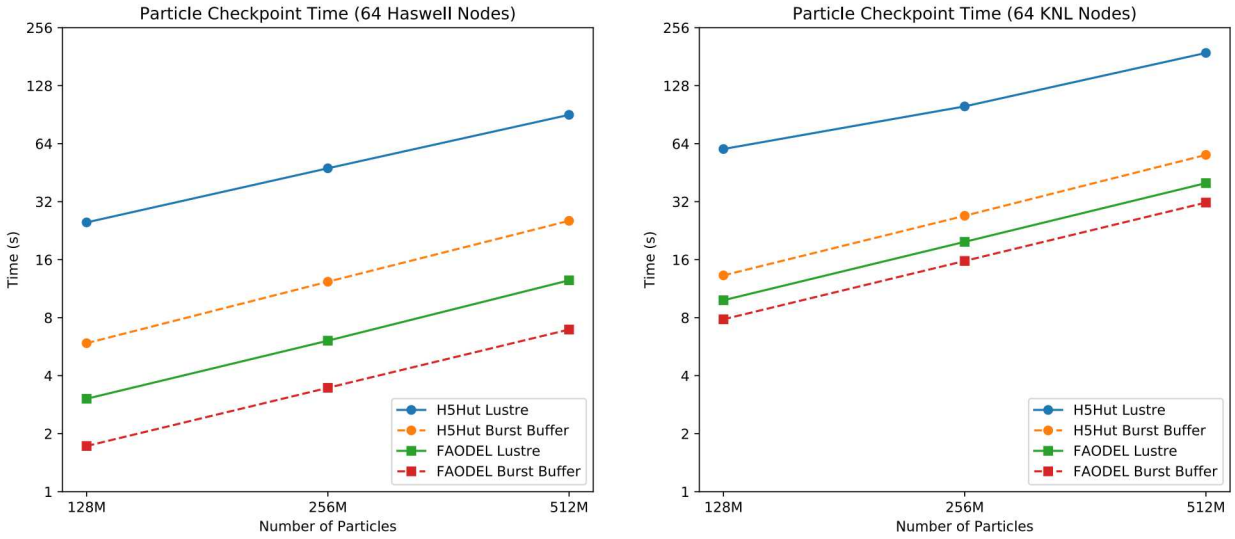


Figure 5.1: Particle Checkpoint Overheads for 64 Nodes on Haswell (left) and KNL (right)

The second performance experiment for particle checkpointing varied the number of compute nodes used to solve a fixed-sized problem. This experiment used two particle sizes (16M and 128M) and two corresponding ranges of nodes. As depicted in Figure 5.2, increasing the node count improves performance because the I/O workload is distributed across a larger number of ranks. Additional runs confirmed that these improvements taper out as the number of particles per rank diminishes and per-rank I/O startup costs dominate the operation.

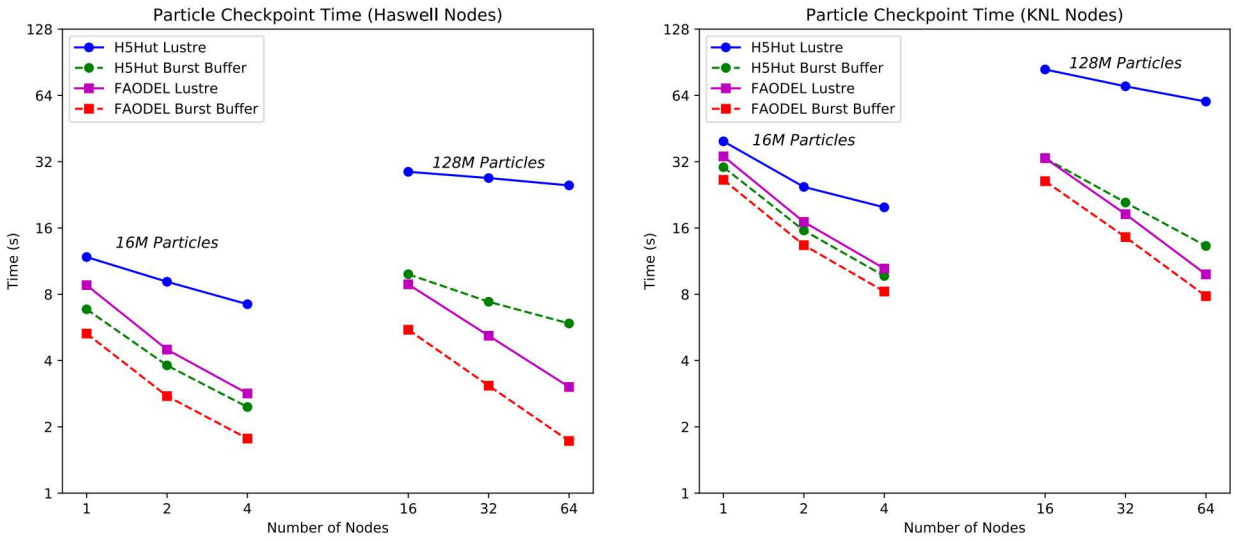


Figure 5.2: Impact of Node Scaling for Particle Checkpoints on Haswell (left) and KNL (right)

An important observation of the checkpoint overhead data is that performance is much lower than what the Burst Buffer can deliver. For example, checkpointing 512M particles through FAODEL to the Burst Buffer takes 6.9s. The resulting transfer rate of 6.2GB is much lower than the theoretical 34GB/s of performance that Mutrino's Burst Buffer offers. This loss is primarily due to data serialization overheads and is further explored in Section 5.3.4.

5.3.2 Mesh Field Data

Similar experiments were performed to measure the amount of time required for the application to store mesh field data. These tests varied the number of elements in the mesh from 0.5M to 32M elements when using 64 nodes on Mutrino. As depicted in Figure 5.3, overheads for field data grow with the mesh size. While the Exodus result file writer takes more time than the FAODEL checkpoint writer, it is storing three times the number of variables and must perform interpolation.

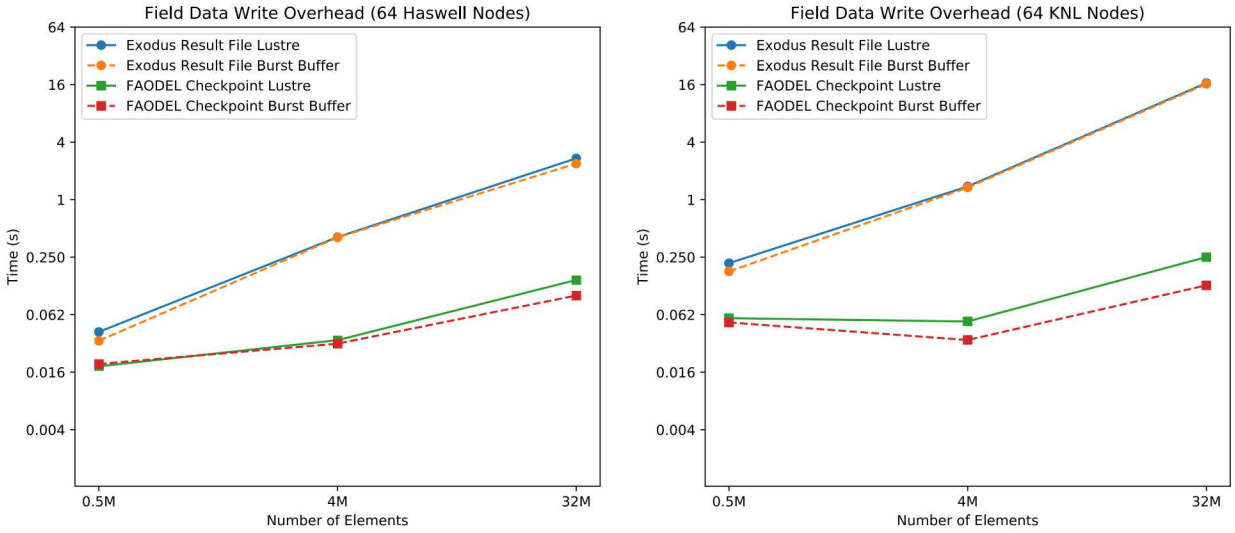


Figure 5.3: Mesh Checkpoint Overhead for 64 Nodes on Haswell (left) and KNL (right)

Additional experiments were conducted to observe how the number of nodes in the simulation affects performance. As illustrated in Figure 5.4, using more nodes decreases the checkpoint overhead because it distributes the workload. The FAODEL checkpoints demonstrated more variability than the Exodus operations. It is likely that caching effects are affecting the performance due to the small size of the data involved (e.g., only 12-48MB per node for 32M elements).

5.3.3 Load Balance Challenges

An important difference between the particle and field data checkpoints is that the distribution of particle data on the nodes changes over time while field data does not. Figure 5.5 depicts the amount of particle data that each node in a simulation produced during the first ten timesteps of

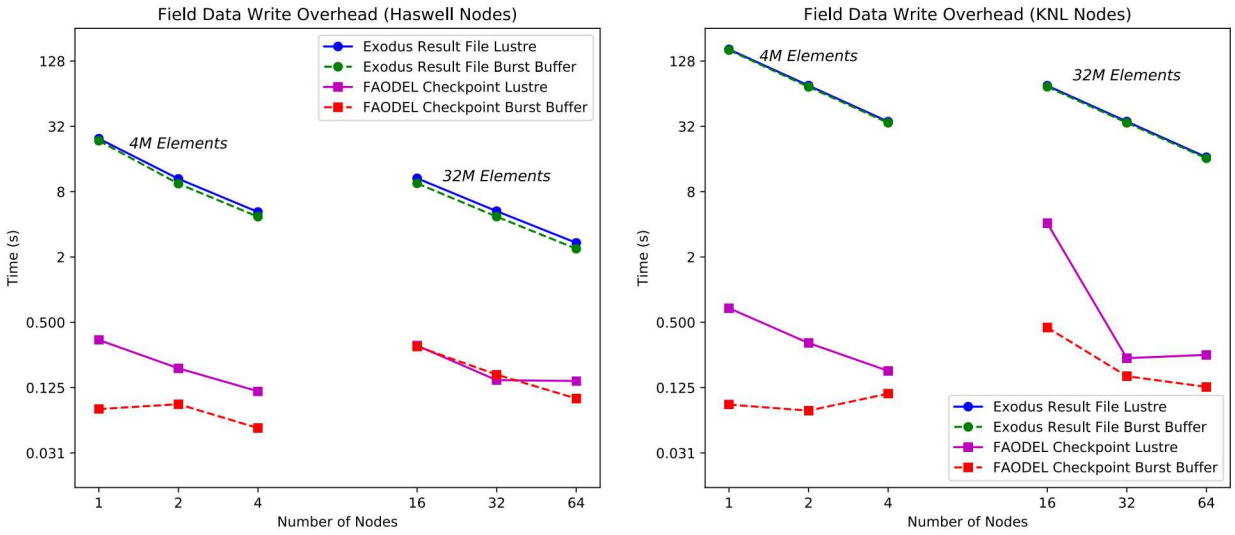


Figure 5.4: Impact of Node Scaling for Field Data Output on Haswell (left) and KNL (right)

the experiments. While the nodes start with equal amounts of data, the amount of data each node needs to checkpoint changes as the particles migrate through the mesh. Similar to other portions of EMPIRE, the checkpoint mechanisms will perform poorly when one rank has substantially more work to do than the other ranks.

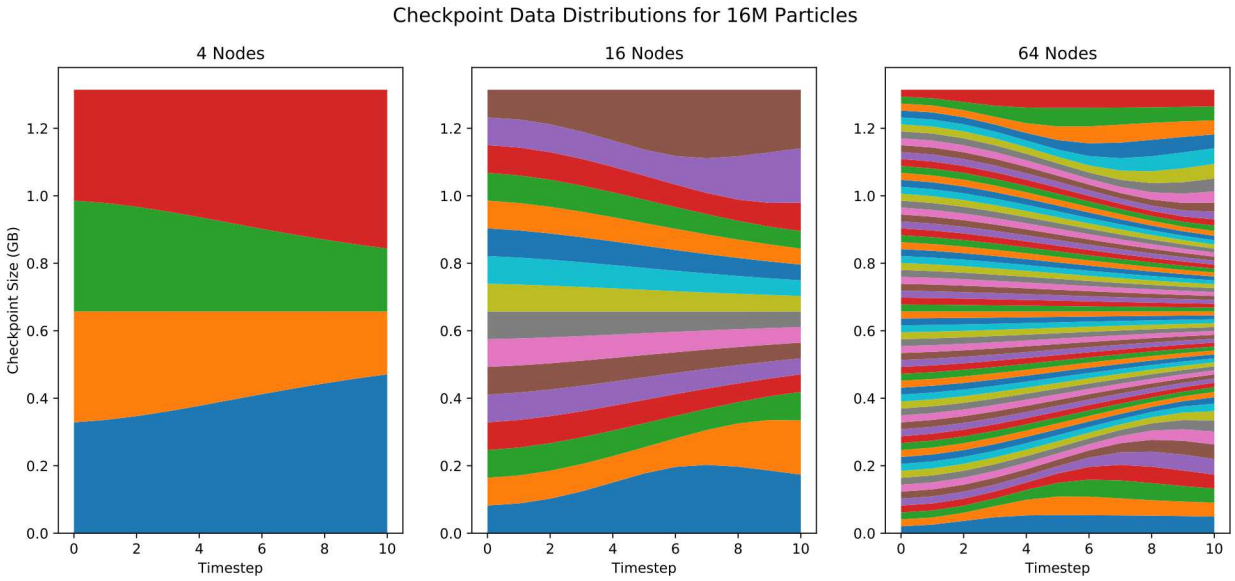


Figure 5.5: Particle Checkpoint Data Distribution for Initial Steps with 4, 16, and 64 Nodes

5.3.4 Serialization Challenges

The overhead numbers reported in the previous sections are from the application's perspective and include two phases of work: serializing the data and performing the actual write to storage. The particle checkpoint operations for FAODEL were instrumented to gain better insight into how much time is spent in each phase. The breakdowns for the particle and field data checkpoint overheads are presented in Figure 5.6 for jobs run with 64 nodes. In these scenarios 42GB of particle data and 805MB of field data is serialized and written to storage. For the particle data, the actual I/O write performance for the Burst Buffer and Lustre is respectively 26GB/s and 6GB/s. Serializing the particle data takes place at a rate of 8GB/s on Haswell nodes and 1.4GB/s on KNL. The field data rates are difficult to accurately measure due to the small amount of data that each node writes (12MB) and caching effects. The I/O rates at this size are approximately 10-15GB/s.

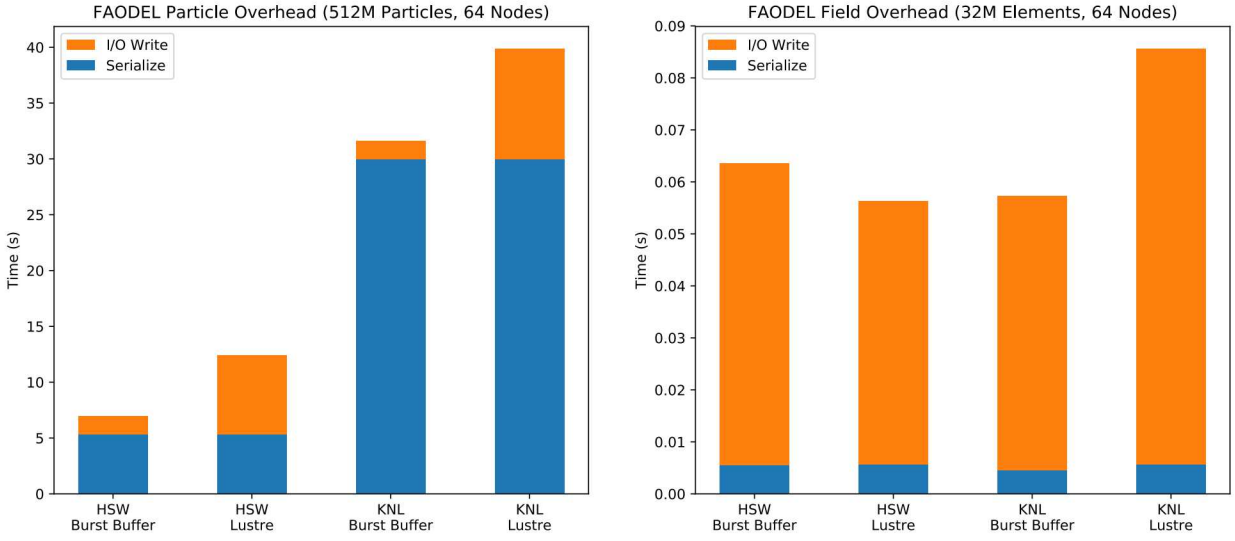


Figure 5.6: FAODEL Serialization and I/O Write Times for Particle (left) and Mesh (right) Checkpoints

In the particle checkpoint case, serialization is the dominant cost because the checkpoint software must assemble a large number of small items. The current implementation is not written in a parallel form and provides poor performance, especially on KNL nodes. Section 5.5 provides information about our strategy for mitigating this issue. The field data does not incur significant serialization overhead because the packing process involves copying a few, large arrays of data into an outgoing buffer. The current field data implementation is expected to be sufficient for larger scale operation.

5.4 Discussion

Three technology choices were explored in the performance experiments: using the Burst Buffer instead of the parallel file system, using Haswell nodes instead of KNL, and using FAODEL

instead of existing file I/O libraries. In order to better examine the impact of each of these technologies individually, the particle performance numbers have been normalized to show speedup for each technology in Figure 5.7.

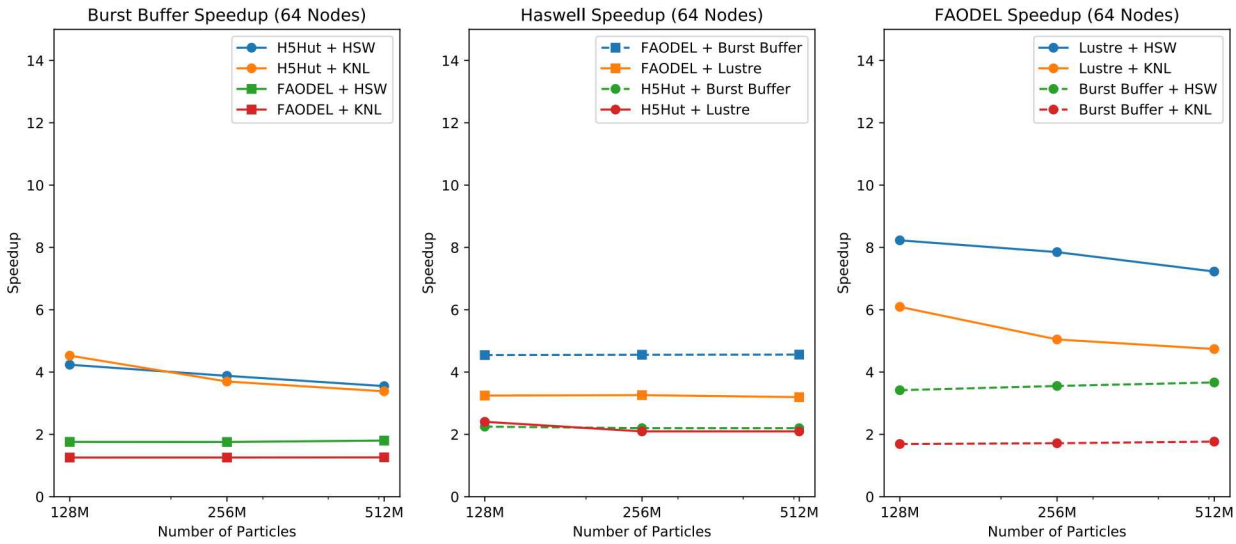


Figure 5.7: Impact of Burst Buffer (left), Haswell (center), and FAODEL (right) on Particle Checkpoint Performance

A number of observations can be made from these speedup measurements as well as the checkpoint experiments in general:

Burst Buffer Acceleration: The Burst Buffer provides a significant improvement in performance with little developer effort. A 4x improvement was observed for H5Hut, simply by changing the output location from a Lustre mountpoint to the Burst Buffer. FAODEL received a 1.5x speedup.

Haswell vs. Knights Landing: Haswell provided better performance than Knights Landing for I/O operations. This difference can be attributed to the fact that the data formatting operations in the I/O code is currently serial and will naturally run significantly faster on the Haswell.

FAODEL Improvement: The FAODEL writer is considerably faster (approximately 2-8x) than the H5Hut writer. While the amount of application data being saved is the same for both, HDF5 performs coordination between ranks to write a single output file and incurs overhead for formatting data for the file container.

Linear I/O Performance at Small Scale: Checkpoint overhead increased linearly with dataset size in these small scale tests. While the dataset sizes were large enough to remove I/O caching effects, the actual I/O operations were not intense enough to saturate the Burst Buffer.

Serialization Dominates Checkpoint Overhead: Serializing particles is currently the dominant expense in checkpointing EMPIRE's data. The current implementation uses a single core to serialize the particle data and accounts for approximately 70% of the Burst Buffer checkpoint overhead. It is expected that this overhead can easily be diminished by parallelizing the serializer or switching to a mechanism that simply uses the underlying Kokkos View as a native container for the data.

5.5 Current Limitations and Next Steps for I/O

The small-scale I/O testing has helped the I/O team identify potential risks to the FY20 performance goals and gain a better understanding of where effort should be placed next. The following is a prioritized list of limitations that will be addressed in the near future:

Limited Restart Capability: The current checkpoint/restart infrastructure has largely been tested with electrostatic simulations and will need revisions to better support electromagnetic simulations. The field data interface is not currently robust, as it relies on EMPIRE's meshing components to initialize the underlying structure. A better solution would be to make the checkpoint/restart code have greater awareness of the underlying mesh structure.

Lacking Random Number Generator (RNG) Storage: EMPIRE uses a large number of RNGs to control how particles progress in the simulation. The current implementation does not checkpoint the state of these RNGs, causing the restart code to create new RNGs during a restart. As such, restarts will not produce identical results. This capability is desirable and will need to be implemented in FY19. The work involves extracting the state value for each RNG and saving it, as well as modifying the particle insertion code to use a predefined RNG state.

Limited Per-Node Particle Size: The current serialization code attempts to pack a rank's entire list of particles into a single object, which FAODEL limits to 4GB in size. This constraint limits an individual rank to holding approximately 50 million particles, and creates unnecessary pressure on the memory system. While fixing this problem is straight forward, we deferred this problem to FY19 in order to focus on other aspects of the I/O problem. We expect that storing a rank's particles in multiple objects will have additional benefits due to pipelining in the I/O system.

Poor Serialization Performance: The initial implementation of the particle data is not parallel and has poor performance. We will investigate converting the operations to a parallel form, as well as determine whether the native Kokkos View representation can be leveraged to bypass serialization.

Limited Scale Testing: The experiments conducted this year focused on small-scale studies on Mutrino. While larger I/O runs have been performed on Trinity, additional tests will be necessary to determine if I/O performance will become an issue at scale. While it is expected

that the scaling of Trinity's Burst Buffer resources will be sufficient, the leading concern is that the sheer number of files generated by 100K ranks could impact performance. The mitigation for this issue would be to use FAODEL to aggregate the I/O.

Lack of M-to-N Restarts: The current implementation is only able to restart on the same number of ranks as the simulation that checkpointed the data. It is straightforward to adapt the particles data to an M-to-N restart (e.g., all ranks load all data with discard). Changes to the mesh will require more significant work.

Broken Job-to-Job Handoffs on Cray Platforms: A benefit of FAODEL is that it allows users to migrate I/O and analysis tasks into a separate job. Unfortunately there is a known bug in Cray's Dynamic RDMA Credentials (DRC) [24] service on NNSA's platforms that prevents the application from acquiring a DRC. Without a DRC, job-to-job communication is impossible. Once the administrators apply Cray's update for this problem, we will revisit the task of using FAODEL to export EMPIRE's data into a separate job.

References

[1] Charles K Birdsall and A Bruce Langdon. *Plasma physics via computer simulation*. CRC Press, 2005.

[2] Roger W. Hockney and James W. Eastwood. *Computer simulation using particles*. Taylor & Francis Group, 1988.

[3] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[4] A. Bossavit. Whitney forms: a class of finite elements for three-dimensional computations in electromagnetism. *IEE Proc.*, 135:493–500, 1988.

[5] Jason M. Gates Eric C. Cyr, Roger P. Pawlowski. Panzer package. [Online; https://trilinos.org/packages/panzer/].

[6] P. Bochev, J. Hu, C. Siefert, and R. Tuminaro. An algebraic multigrid approach based on a compatible gauge reformulation of Maxwell's equations. *SIAM Journal on Scientific Computing*, 31(1):557–583, 2008.

[7] Enrst Hairer, Syvert Paul Norsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Number 8 in Springer Series in Computational Mathematics. Springer-Verlag, New York, second revised edition edition, 2000.

[8] Enrst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff Problems*. Number 14 in Springer Series in Computational Mathematics. Springer-Verlag, New York, second revised edition edition, 2002.

[9] Shadid, Cyr, Pawlowski, Widley, Scovazzi, Zeng, Phillips, Conde, Chuadhry, Hensinger, Fischer, Robinson, Rider, Niederhaus, and Sanchez. Towards an IMEX Monolithic ALE Method with Integrated UQ for Multiphysics Shock-hydro. Technical Report SAND2016-11353, Sandia National Laboratories, 2016.

[10] E. C. Cyr. IMEX Lagrangian Methods. Technical Report SAND2015-3745C, Sandia National Laboratories, 2015.

[11] G. M. Hulbert and J. Chung. Explicit time integration algorithms for structural dynamics with optimal numerical dissipation. *Computer Methods in Applied Mechanics and Engineering*, 137:175–188, October 1996.

[12] H. M. Hilber, T. J. R. Hughes, and R. L. Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics*, 5:283–292, 1977.

[13] A. Mota, W. Klug, and M. Ortiz. Finite element simulation of firearm injury to the human cranium. *Computational Mechanics*, 31:115–121, 2003.

[14] N. M. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division ASCE*, 85(EM3):67–94, 1959.

[15] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2012.

[16] Francis F Chen. *Introduction to plasma physics and controlled fusion*. Springer, third edition, 2016.

[17] Clément Mouhot and Cédric Villani. On landau damping. *Acta mathematica*, 207(1):29–201, 2011.

[18] W. Herr. Introduction to landau damping. In *CAS-CERN Accelerator School: Advanced Accelerator Physics*, 2014.

[19] D. R. Nicholson. *Introduction to Plasma Theory*. Krieger Publishing Company, Malabar, Florida, 1992.

[20] J. A. Bittencourt. *Fundamentals of Plasma Physics*. Pergamon Press, New York, 1986.

[21] Ning Liu, Jason Cope, Philip Carns, Christopher Carothers, Robert Ross, Gary Grider, Adam Crume, and Carlos Maltzahn. On the role of burst buffers in leadership-class storage systems. In *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*, pages 1–11. IEEE, 2012.

[22] Dave Henseler, Benjamin Landsteiner, Doug Petesch, Cornell Wright, and Nicholas J Wright. Architecture and design of cray datawarp. *Cray User Group CUG*, 2016.

[23] Mark Howison, Andreas Adelmann, E Wes Bethel, Achim Gsell, Benedikt Oswald, et al. H5hut: A high-performance i/o library for particle-based simulations. In *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, pages 1–8. IEEE, 2010.

[24] James Shimek and James Swaro. Dynamic rdma credentials. *Cray User Group CUG*, 2016.

DISTRIBUTION:

1 MS 0899 Technical Library, 9536 (electronic copy)


Sandia National Laboratories