As an HPC user and multi-disciplinary researcher, I find resilient computing fascinating. The fact that we have created systems so complex that their computing and reliability characteristics have spawned entire fields of study is incredible. More generally, I view problems such as resilient computing at extreme-scale, as requiring a multi-disciplinary approach. For example, silent, transient data corruption (SDC) can cause numerical simulations to produce either correct and incorrect solutions. A pure hardware treatment of the problem can replicate physical components to make the hardware appear more reliable, or a pure systems software approach may replicate software on unreliable hardware to make the software appear more reliable. Both approaches are valid, but neither considers the properties of the algorithms used. For example, numerical methods typically have proven error-dampening properties. For certain algorithms, I believe we can theoretically establish that numerical analysis can handle some of the fault tolerance burden, while exploiting software and hardware approaches to tolerate what mathematics cannot.

My dissertation work combines reliability modeling, systems programming, and numerical analysis. I consider silent, transient faults that do not crash the application or give any indication an error has occurred. Instead, these faults may corrupt arithmetic or storage, and can cause the application to behave abnormally. My approach to coping with SDC is drastically different from all prior work, and it realizes my vision to couple multiple disciplines to create a response that is theoretically sound and more efficient than current standard approaches. Rather than attempt to detect and correct all SDC, I exploit numerical analysis to derive invariants that link the inputs provided by the user to the algorithm. I also identify existing numerical properties that can be easily checked to determine if the algorithm is in a theoretically unallowable state. This coupling of numerical analysis and fault tolerance allows me to enforce bounds on key values within the algorithm. My bounds are cheap to evaluate (a single if statement), and some may be determined entirely before running the algorithm. To demonstrate my findings I derived a bound on the Arnoldi process, which is the basis for many eigensolvers as well as the GMRES iterative solver [1]. By using an interdisciplinary approach, I am able to filter “bad” errors, while showing that the remaining errors are easily tolerated by GMRES. I call my approach Skeptical Programming.

Showing that algorithm-based fault tolerance for GMRES can be cheap and scalable is significant, but most researchers solving linear equations employ preconditioners. No prior work has attempted to address SDC in arbitrary preconditioners. In recent work, which is under review [2], I show that Skeptical Programming is scalable and extensible to arbitrary preconditioners. This is paramount, as it demonstrates that by coupling domain knowledge to fault tolerance, we are able to harden complex “black-box” numerical codes against SDC. I show that my skeptical checks were able to detect faults arising from two classes of preconditioners: Additive Schwarz and Algebraic Multigrid, while not modifying or

instrumenting the preconditioner's codebase. Evaluating my approach at scale requires thousands of core hours per run, and I anticipate using over a million node-hours to complete my dissertation.

In general, my dissertation asks the question “Given a linear solver and the data being operated on, how can this information be exploited to ensure that SDC does not decimate the solver”. Deriving algorithm invariants has proven extremely useful, but there is more that can be achieved by exploiting the representation of floating point data. Assuming a bit is perturbed in IEEE-754 floating-point data; I have shown that the likelihood of this bit flip introducing a large absolute error is not uniformly distributed [3]. Due to the floating point specification, for multiplication and addition, values less than or equal to one have a probability of 1/64 of introducing large absolute error, while the remaining 63 bit positions will all produce an error less than or equal to one. If we perform dot products using normalized vectors then we can ensure that a single bit flip will introduce either a small error (less than one) or a very large error (typically on the order of 2^{150}). I extend this result to matrices using matrix equilibration, and demonstrate the usefulness of this finding by relating it to the Arnoldi process in used by GMRES.

GMRES attempts to solve a linear system by constructing a small orthonormal basis of the linear operator. An orthonormal basis is simply a set of normalized vectors that are orthogonal (the mathematical sense) to each other. The heart of GMRES requires performing dot products using these normalized vectors as well as matrix-vector multiplies against each of these normalized vectors. This produces a perfect scenario for demonstrating the benefit of my scaling result. GMRES always constructs an orthonormal basis regardless of whether the input matrix is scaled, which means that given the two vectors required to perform a dot product, one is always normalized. If the input matrix is equilibrated, then both vectors used in the dot product will be close to unit length, and the likelihood of seeing a large error due to a bit flip in the matrix or vector will be less than one 91% of the time and large only 8% of the time. Coupling this finding with my skeptical programming results allows me to show that even with faulty arithmetic, I can force errors to be bounded. This allows numerical analysis to easily dampen the errors introduced, and ensures that the solver does not “explode”, allowing us to roll-through errors with minimal overhead.

1. **James Elliott**, M. Hoemmen, and F. Mueller, “Evaluating the Impact of Silent Data Corruption on the GMRES Iterative Solver,” In *Proceedings of the **28th IEEE International Parallel and Distributed Processing Symposium** (IPDPS) 2014*, Phoenix, AZ, May 2014.
2. **James Elliott**, M. Hoemmen, and F. Mueller, “Tolerating Silent Data Corruption in Opaque Preconditioners,” Sandia Technical Report.
3. **James Elliott**, M. Hoemmen, and F. Mueller, “Exploiting Data Representation for Fault Tolerance,” In *Proceedings of the **Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems** (ScalA'14).*