

An Overview of Applied Math Activities at Argonne

Paul Hovland

Mathematics and Computer Science Division

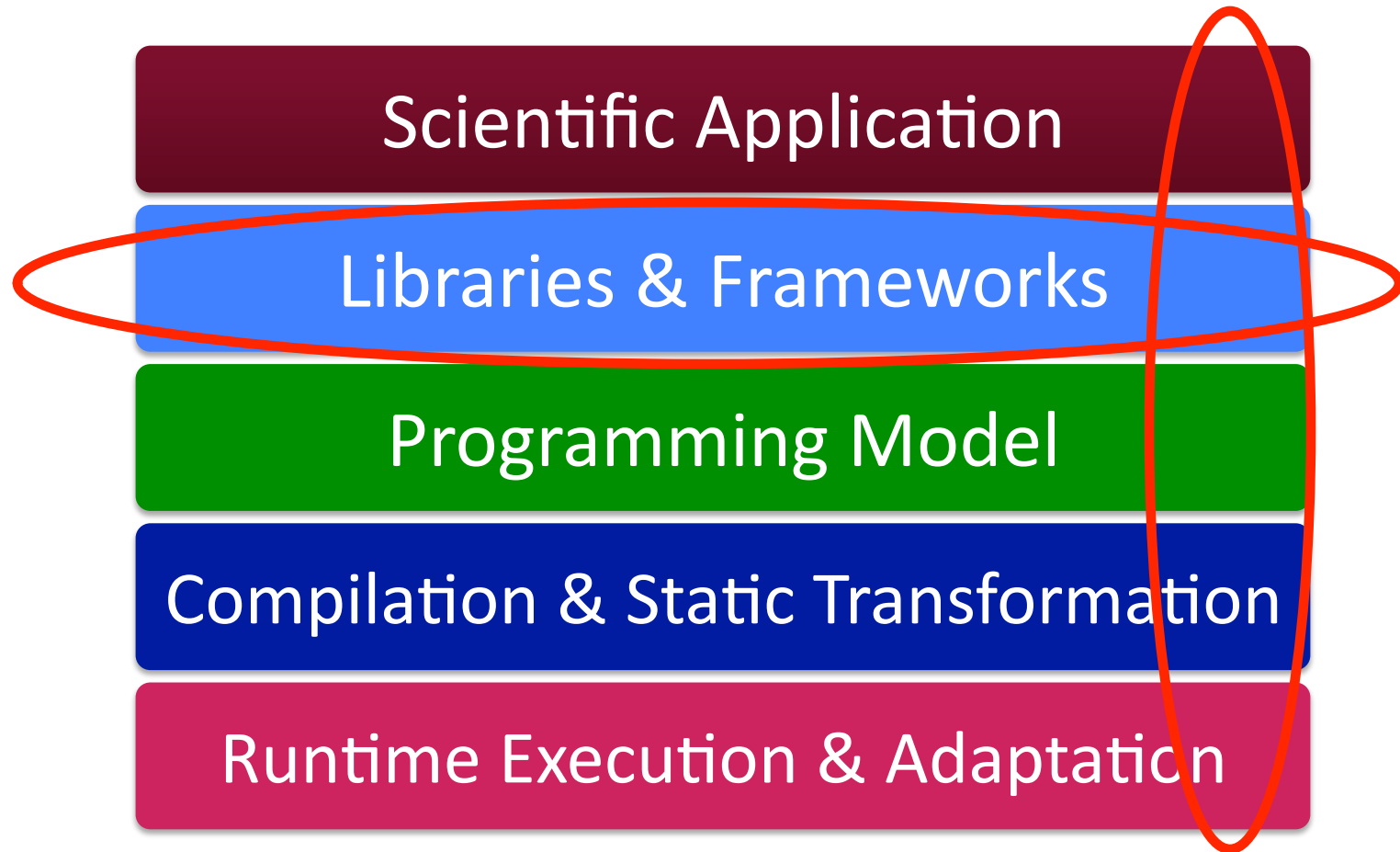
Argonne National Laboratory

UIUC-INRIA Workshop

University of Illinois at Urbana-Champaign

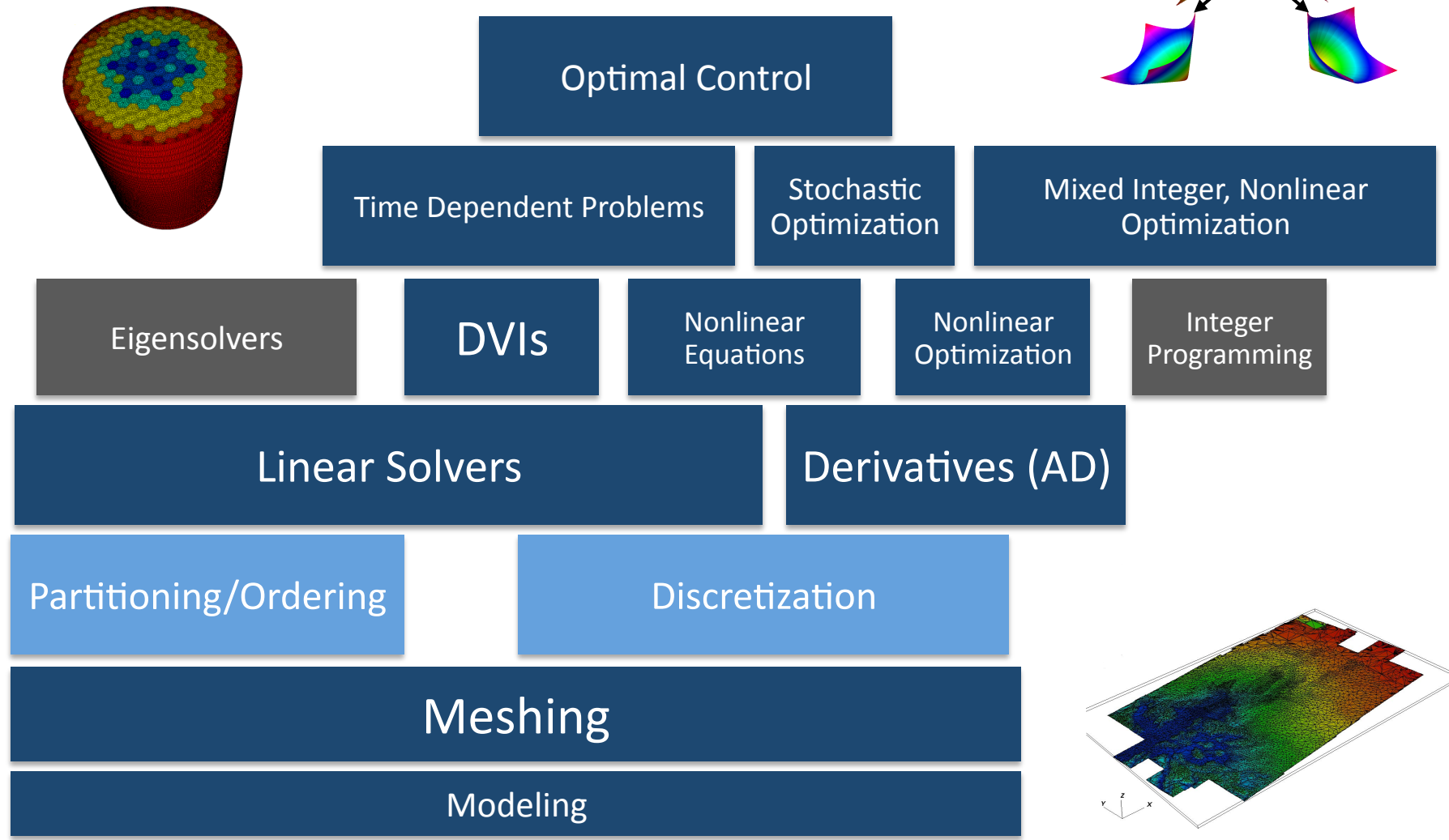
November 21, 2011

Ecosystem for Computational Science Software



Libraries & Frameworks

Many Layers of Mathematical Abstraction



Argonne National Laboratory ---- Mathematics and Computer Science Division



11/21/11

Our Philosophy: A Strategy for Progress

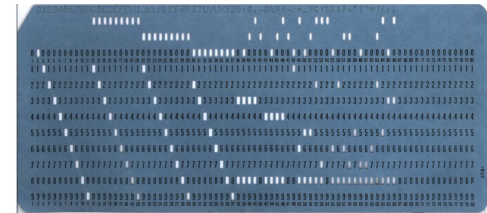


- Raise the mathematical level of abstraction
 - Do the analysis to develop the theory
 - Develop algorithms and software
 - Help applications scientists identify appropriate abstraction
- Develop new, more efficient algorithms
 - Pursuing higher fidelity and multiphysics demands scaling to more cores
 - Constantly striving to reduce data movement
- Change the programming model
 - to increase productivity
 - to better map to the target architecture(s)
- Increase the feedback between and among the layers

An example from the past

- Old: LINPACK

- Dense linear algebra
- Implemented using a sequential, imperative programming model

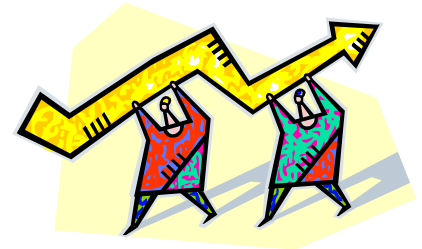


- New: TAO

- Toolkit for advanced optimization
- Scalable nonlinear optimization algorithms
- Implemented using a distributed memory, object-oriented programming model
- Built on top of other libraries (including PETSc) and can use derivatives computed using automatic differentiation

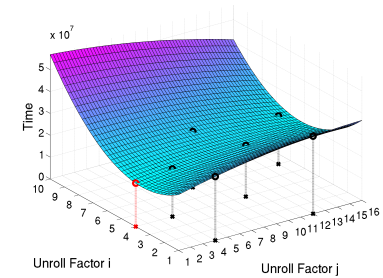


Preparing for the future



- Raise the mathematical level of abstraction
 - We anticipate a growing emphasis on analysis and decision support
 - Developing theory, algorithms, software for analysis, optimization, UQ
- Develop new, more efficient algorithms
 - Full approximation scheme to reduce memory bandwidth requirements
 - Trade recomputation for checkpoints in adjoints & analysis
- Change the programming model
 - Domain specific languages targeting library developers
 - Generated code to exploit high levels of on-chip concurrency
- Increase the feedback between and among the layers
 - Use advances in derivative free optimization to improve autotuning

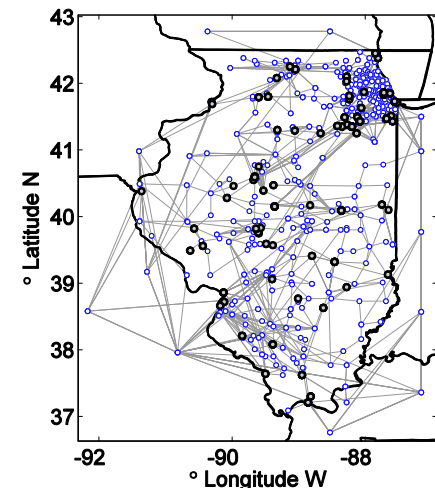
CACHE Institute



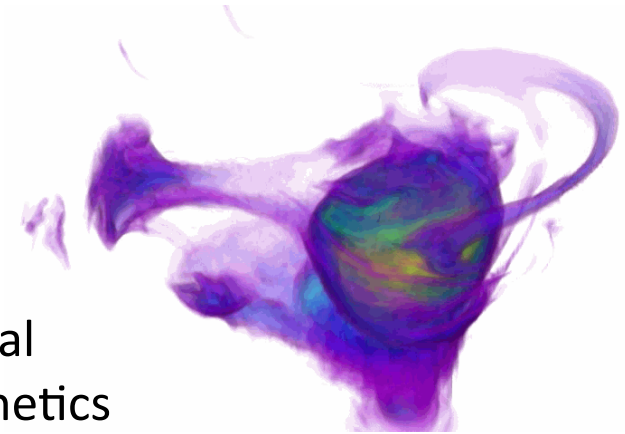
- Goal: Simplify the task of developing new numerical algorithms that perform well on high-performance computer architectures
 - PETSc contains a large amount of hand-optimized code; rewriting this code for new architectures and new programming models is a daunting task
 - PETSc contains specialized routines such as VecAXPBYPCZ because of limitations of the programming model
- Approach: Combine a domain specific language (DSL) specifically aimed at library developers with autotuning
- Observation #1: DSLs targeted at library developers can be very different from DSLs targeted at application scientists—goal is to convey high level semantic information, not to keep code written to a bare minimum
- Observation #2: Autotuning can be modeled as a mathematical optimization problem and the advances in derivative free optimization brought to bear on the search problem

Stochastic Optimization

- Goal: Develop scalable methods for solving stochastic optimization problems on multi-core architectures
- Impact: Real-time decision making under uncertainty for complex energy dispatch with unprecedented resolution.
 - Stochastic Unit Commitment with wind power integration: 189M variables (US Midwest)
 - Unit Commitment with transmission constraints: 2.16B vars (Illinois – 48hr window)
- Approach: use interior-point optimization with scenario-based parallelization of linear algebra
- Software: PIPS, based on MPI+OpenMP parallelization
- Performance: 96% strong scaling on BG/P (131k cores)
- Future:
 - Better resolution of uncertainty
 - Faster dynamics
 - Longer horizon



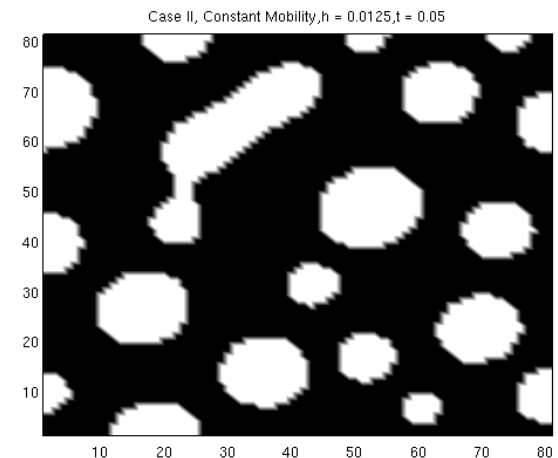
Nek5000 & NekCEM



- Goal: provide scalable solvers for computational fluid dynamics and computational electromagnetics
- Approach: use high order spectral element discretizations to provide high flop to byte ratio; make advances in solver algorithms and implementations to scale to the largest possible problem sizes and processor counts
- Nek5000 is a mature, but constantly evolving, open source code for solving the Navier-Stokes equations
- NekCEM is adapted from Nek5000 and uses a spectral element discontinuous Galerkin method to solve Maxwell's equations
- As the target platform for Nek has progressed from small Intel hypercubes to today's leadership computers with $O(10^6)$ cores, achieving scalability has required revolutionary advances in the coarse grid solver, improved communication kernels and a change from one Posix file to true parallel I/O

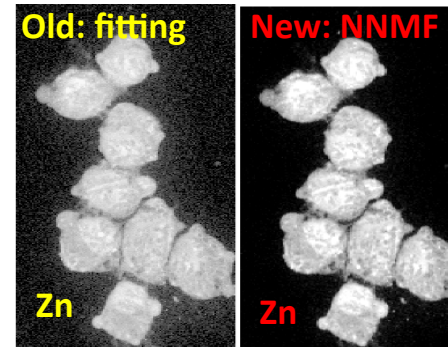
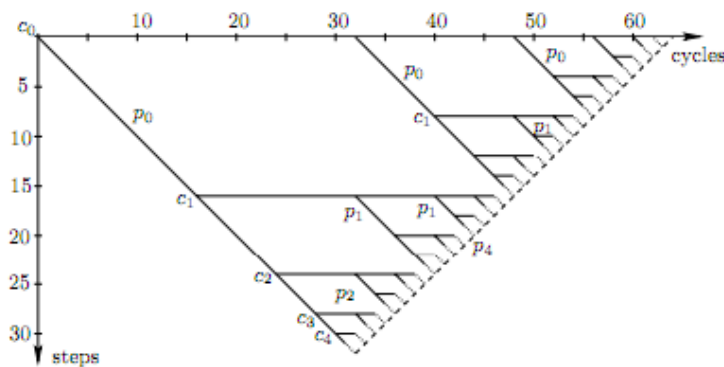
PETSc-DVI

- Goal: solve material science phase field problems with high accuracy and low cost
- Approach: replace traditional PDE-based formulation using smoothed potentials with a differential variational inequality (DVI) formulation
- Strategy: work with material scientists to develop new model; leverage algorithms for solving complementarity problems from PATH software and scalable linear algebra and data structures in PETSc to develop scalable DVI solver
- Outcome #1: accurate solutions without numerical artifacts or excessive stiffness
- Outcome #2: many material scientists eager to work with PETSc-DVI team



More examples / grand ambitions

- Task-oriented programming models for use with stochastic optimization, mixed integer nonlinear optimization, and other scenario-based computations
- Integration of checkpointing and recomputation for adjoint computation, fault tolerance, and analysis, with appropriate hints to parallel I/O system
- Online, model-based data reduction techniques for use with high-bandwidth data sources (experimental facilities and simulations)



Another historical example...

- ~1989: Jorge Moré challenges the lunch table to produce a function whose gradient is expensive to compute relative to the function.
- Andreas Griewank attempts to identify such a function and in the process rediscovers the reverse mode of automatic differentiation and proof that no such function can exist.
- ~1990: Christian Bischof works on parallel algorithms for automatic differentiation and recognizes that a source transformation tool based on a robust compiler infrastructure offers the greatest promise for efficient derivatives; John Dennis and Ken Kennedy at Rice University provide introduction to Alan Carle; ADIFOR is born.
- 1991: Paul Hovland joins ADIFOR research team; describes mechanism to apply reverse mode at the statement level.
- 1997: Hovland completes Ph.D. thesis at University of Illinois Urbana-Champaign on automatic differentiation of parallel programs.
- Today: Continued development of AD tools for Fortran 9x and C++; collaboration with RWTH Aachen and INRIA Sophia-Antipolis; revisiting Bischof

Some collaboration opportunities

- Algebraic multigrid: for preconditioning and for solving combinatorial problems
- Automatic differentiation: already collaborating with INRIA; significant compiler, PL, and system software elements
- Co-design: design architectures for the algorithms of tomorrow, not today
- Domain specific languages: targeting numerical library developers
- Meshing: generation, management, interfaces
- Numerical libraries: new algorithms, new platforms
- Numerical optimization: a historical strength; new applications?
- Other emerging opportunities: optimal control, time stepping
- Resilient algorithms: especially for numerical optimization
- Uncertainty quantification: our emphasis has been on intrusive approaches
- TBD: other opportunities?