

Introduction to Numerical Software

Presented to
ATPESC 2022 Participants

Ulrike Meier Yang
Lawrence Livermore National Laboratory

Date 08/09/2022

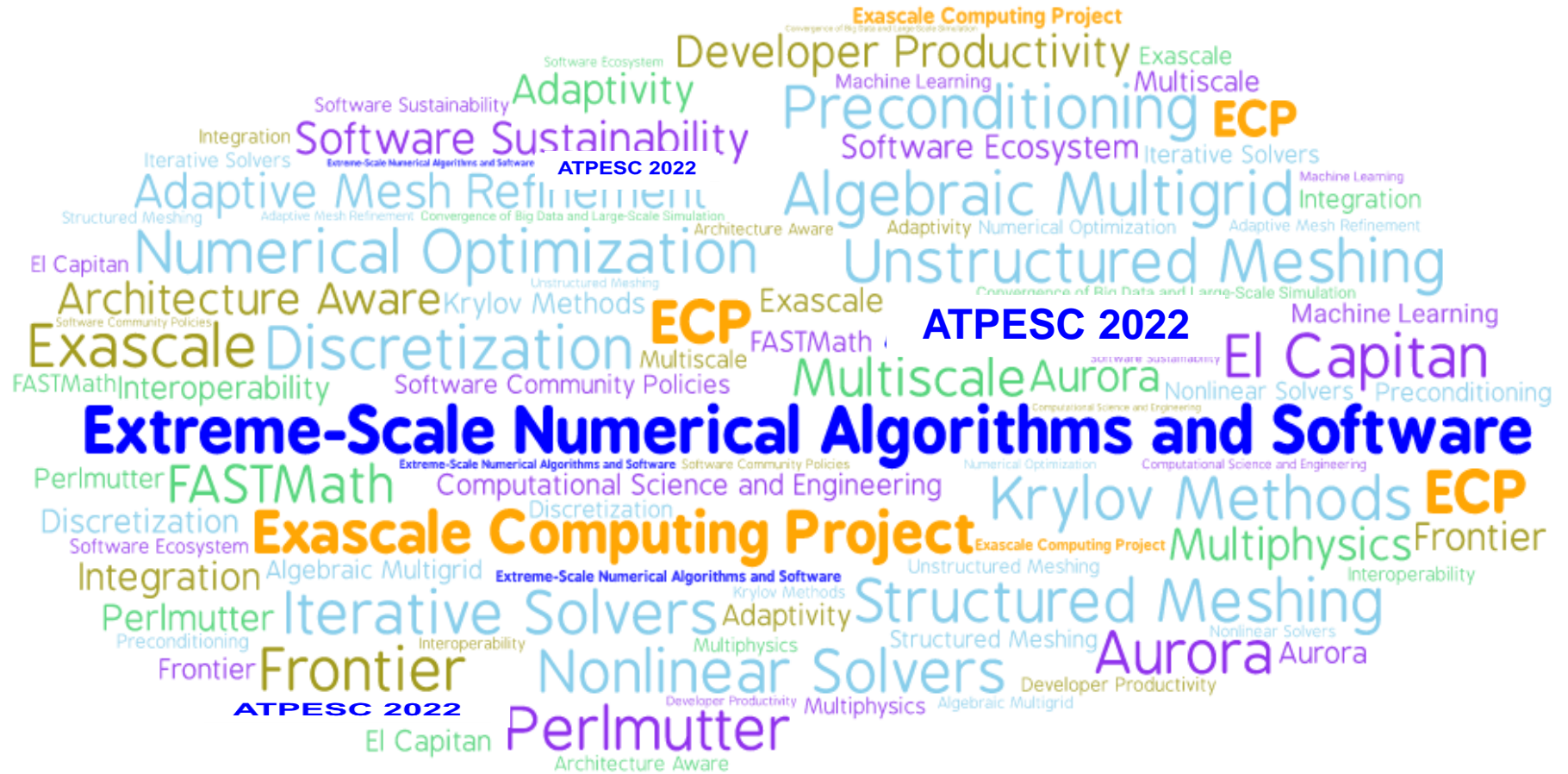


ATPESC Numerical Software Track



Outline

- Logistics for the day
- Intro to numerical algorithms and software for extreme-scale science

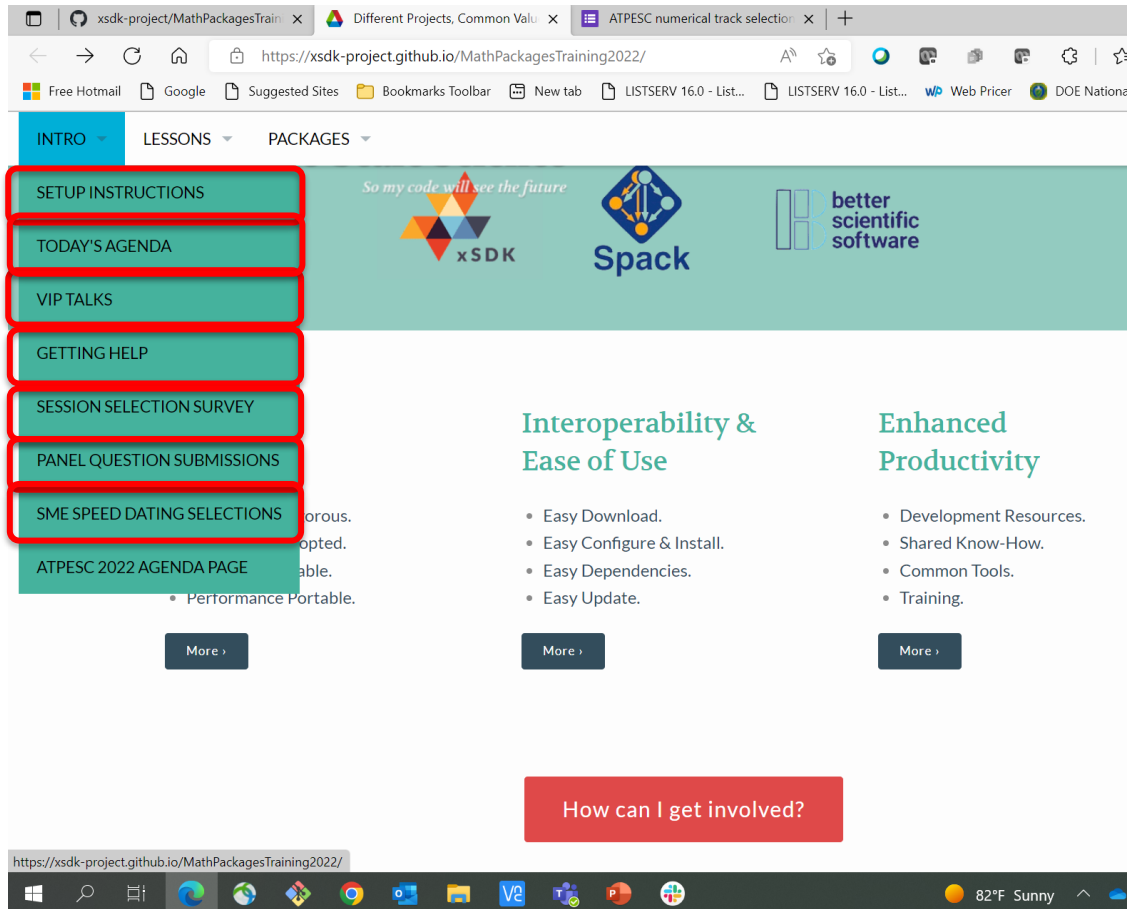


Your home bases for the day: ATPESC Track 5

Numerical Algorithms and Software for Extreme-Scale Science

- Main ATPESC Agenda
 - <https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>
 - slides (pdf) and presenter bios
- Math Packages Training Site
 - session abstracts, links to parallel breakout rooms, hands-on lessons, more
 - <https://xsdk-project.github.io/MathPackagesTraining2022/agenda/>

<https://xsdk-project.github.io/MathPackagesTraining2022/>



- Setup instructions
- Today's agenda
- VIP talks
- Getting help
- Session Selection Survey
- Panel question submission
- SME speed dating selections

Agenda

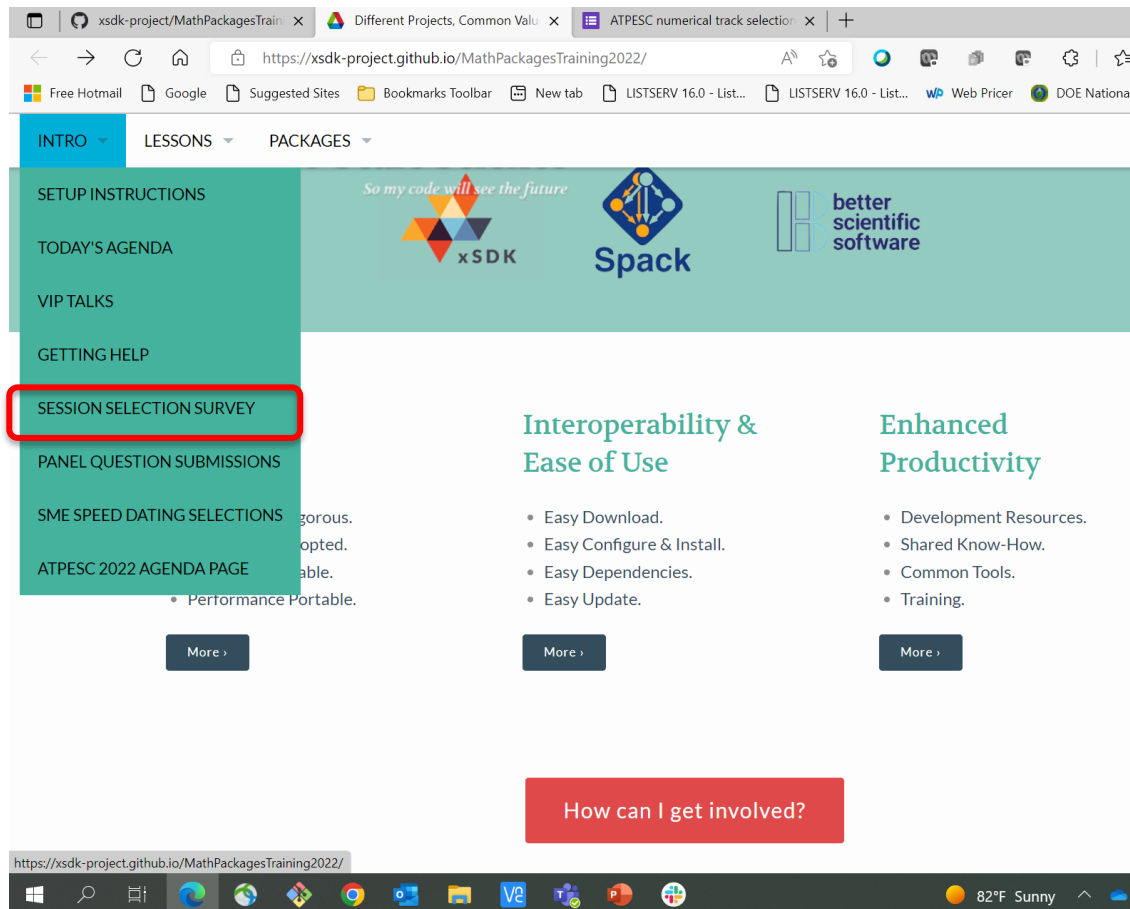
<https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break, Subject Matter Expert (SME) Selections, Panel Questions	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch, SME Selections, Panel Questions	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break, SME Selections, Panel Questions Due	
3:15 – 4:30	Optimization (TAO) – Todd Munson, Richard Mills	Iterative Solvers & Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up / Panel: Extreme-Scale Numerical Algorithms and Software	
5:30 – 6:30	Unstructured Time: SME Selections Due , Informal Discussion, Continue Hands-on	
6:30 – 7:30	Dinner	
7:30 – 9:30	Optional Activity: SME Speed-dating	

Choose which lecture you want to attend!

- Access: <https://forms.gle/axrawtNsTgbjDTJP8>

<https://xsdk-project.github.io/MathPackagesTraining2022/>

A screenshot of a Google Forms interface titled "ATPESC numerical track selections form". Below the title is a subtitle: "Participants, please denote sessions you plan to attend so that needed room size can be determined". A user is logged in as "yang11@llnl.gov" with a "Switch account" link. A red asterisk indicates a required field. The first required field is "Email *", with a placeholder text "Your email". Below this is a section titled "Parallel Session One *" with three radio button options: "Structured Meshes (with AMReX)", "Unstructured Meshes (with MFEM/PUMI)", and "Undecided (happy to be placed in either)". The next section is titled "Parallel Session Two *" with two radio button options: "Krylov Solvers & Multigrid Preconditioning (with Belos and MueLu)" and "Direct Solvers (with SuperLU/STRUMPACK)". A small edit icon is visible in the bottom right corner of the form area.

Agenda

<https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break	
3:15 – 4:30	Optimization (TAO) – Todd Munson	Iterative Solvers & Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up / Panel	
6:30 – 7:30	Dinner	
7:30 – 9:30	SME Speed Dating	



Block-structured adaptive mesh refinement framework. Scalable support for hierarchical mesh and particle data, with embedded boundaries.

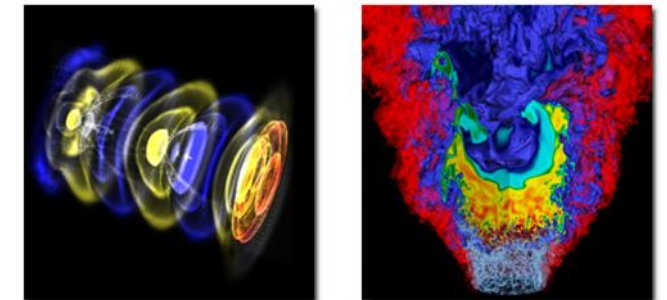
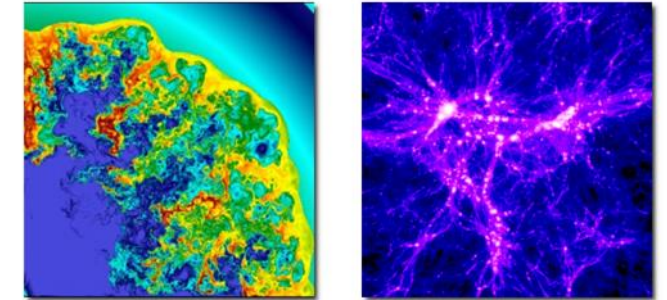
■ Capabilities

- Support for PDEs on a hierarchical adaptive mesh with particles and embedded boundary representations of complex geometry
- Support for multiple modes of time integration
- Support for explicit and implicit single-level and multilevel mesh operations, multilevel synchronization, particle, particle-mesh and particle-particle operations
- Hierarchical parallelism –
 - hybrid MPI + OpenMP with logical tiling on multicore architectures
 - hybrid MPI + GPU support for hybrid CPU/GPU systems (NVIDIA CUDA, AMD HIP, Intel SYCL)
- Native multilevel geometric multigrid solvers for cell-centered and nodal data
- Highly efficient parallel I/O for checkpoint/restart and for visualization – native format supported by Visit, Paraview, yt

■ Open source software

- Used for diverse apps, including accelerator modeling, astrophysics, combustion, cosmology, multiphase flow, phase field modeling, atmospheric modeling and more
- Source code and development hosted on github with rigorous testing framework
- Extensive documentation, examples and tutorials

Examples of AMReX applications

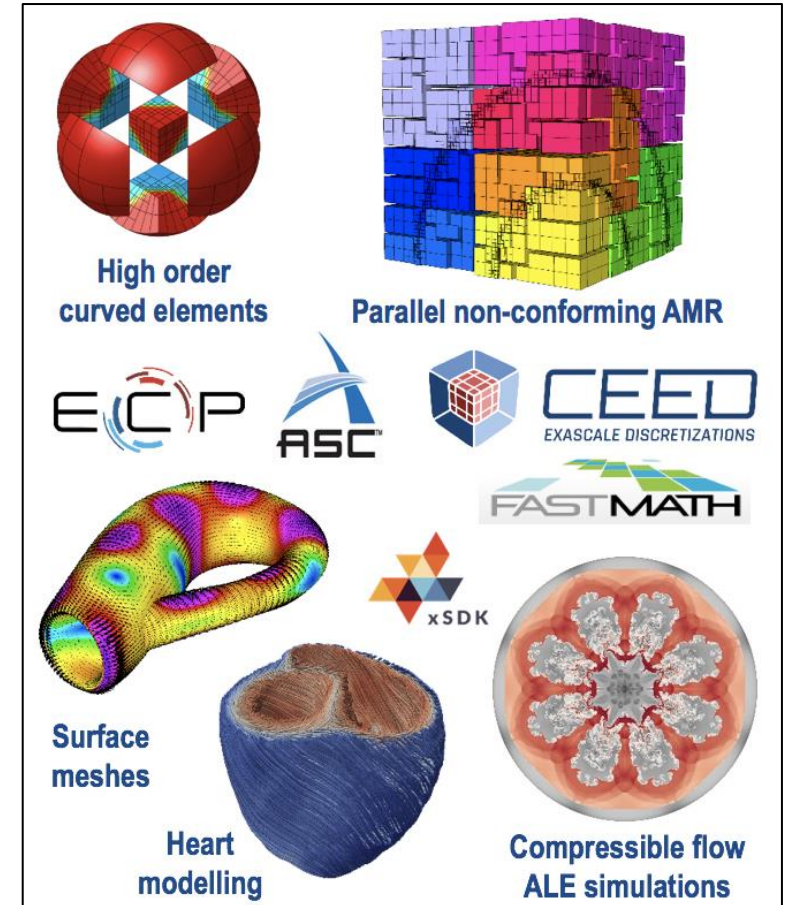


<https://www.github.com/AMReX-Codes/amrex>



Free, lightweight, scalable C++ library for finite element methods. Supports arbitrary high order discretizations and meshes for wide variety of applications.

- **Flexible discretizations on unstructured grids**
 - Triangular, quadrilateral, tetrahedral and hexahedral meshes.
 - Local conforming and non-conforming refinement.
 - Bilinear/linear forms for variety of methods: Galerkin, DG, DPG, ...
- **High-order and scalable**
 - Arbitrary-order H1, H(curl), H(div)- and L2 elements. Arbitrary order curvilinear meshes.
 - MPI scalable to millions of cores and includes initial GPU implementation. Enables application development on wide variety of platforms: from laptops to exascale machines.
- **Built-in solvers and visualization**
 - Integrated with: HYPRE, SUNDIALS, PETSc, SUPERLU, ...
 - Accurate and flexible visualization with VisIt and GLVis
- **Open source software**
 - LGPL-2.1 with thousands of downloads/year worldwide.
 - Available on GitHub, also via OpenHPC, Spack. Part of ECP's CEED co-design center.



<http://mfem.org>

Parallel Unstructured Mesh Infrastructure

Parallel management and adaptation of unstructured meshes.
Interoperable components to support the
development of unstructured mesh simulation workflows

■ Core functionality

- Distributed, conformant mesh with entity migration, remote read only copies, fields and their operations
- Link to the geometry and attributes
- Mesh adaptation (straight and curved), mesh motion
- Multi-criteria partition improvement
- Distributed mesh support for Particle In Cell methods

■ Designed for integration into existing codes

- xSDK package; installs with Slack
- Permissive license enables integration with open and closed-source codes

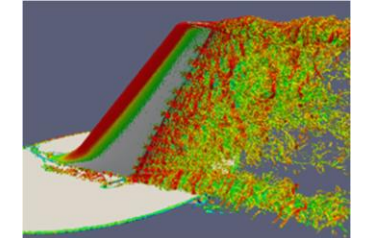
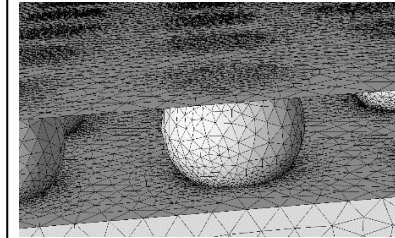
■ In-memory integrations developed

- MFEM: High order FE framework
- PetraM: Adaptive RF fusion
- PHASTA: FE for turbulent flows
- FUN3D: FV CFD
- Proteus: Multiphase FE
- ACE3P: High order FE for EM
- M3D-C1: FE based MHD
- Nektar++: High order FE for flow
- Albany/Trilinos: Multi-physics FE

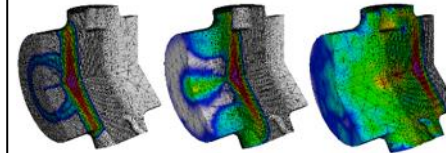
PUMi



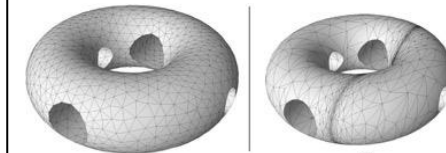
Rensselaer



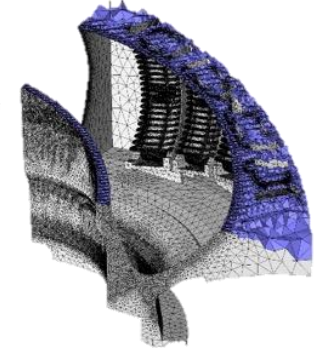
Applications with billions of elements: flip-chip (L), flow control (R)



Mesh adaptation for evolving features



Anisotropic adaptation for curved meshes



RF antenna and plasma surface in vessel.

Source Code: github.com/SCOREC/core
User Guide: scorec.rpi.edu/pumi/PUMI.pdf
Paper: scorec.rpi.edu/REPORTS/2014-9.pdf

PUMIPic Parallel Unstructured Mesh Infrastructure for Particle-in-Cell

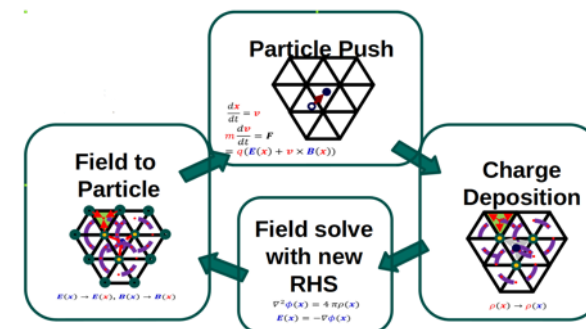
Parallel management of unstructured meshes with particles.
Framework for GPU accelerated particle-in-cell applications using unstructured meshes.

Core functionality

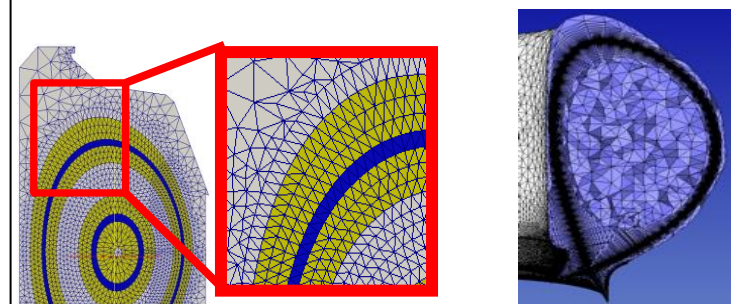
- Unstructured mesh-based approach
 - Particles accessed through mesh
 - Particle search through mesh adjacencies
 - Effective coupling to PDE solvers
 - Partitioning using bounding flux surfaces, graph, or geometric methods
 - PICpart: owned elements (defined by partition) + copied elements from topologically or spatially neighboring processes
 - Stored on GPU using Omega_h library: github.com/SNLComputation/omega_h
- Particles
 - Supports multiple species each with distinct combinations of 'Plain Old Data' per particle
 - Group particles by the mesh element that they are spatially located within
 - Multiple choices for particle storage using abstraction layer: Sell-C-Sigma [Kreutzer 2014], COPA Cabana, and CSR.
- Parallel kernel launch function abstracts underlying particle and mesh storage

Applications Supported

- GITRm: impurity transport
- XGCm: core+edge fusion plasma physics
- Weak scaling on up to 24,000 GPUs of Summit with 1.15 trillion particles running push, particle-to-mesh, and mesh-to-particle operations with an XGCm tokamak mesh and domain decomposition



Stages of a PIC application supported by PUMIPic



(Left) Two PICparts defined as sets of flux faces in XGCm mesh. (Center) The blue face is the 'core' and the yellow faces are its 'buffers'. (Right) 3D GITRm mesh for impurity transport simulation.

Source Code: github.com/SCOREC/pumi-pic
Paper: scorec.rpi.edu/REPORTS/2020-2.pdf



Agenda

<https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break	
3:15 – 4:30	Optimization (TAO) – Todd Munson, Richard Mills	Iterative Solvers & Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up / Panel	
6:30 – 7:30	Dinner	
7:30 – 9:30	SME Speed Dating	

Trilinos/Belos

Iterative Krylov-based solvers. Templated C++ allows for generic scalar, ordinal, and compute node types.

- **Ability to solve single or sequence of linear systems**

- Simultaneously solved systems w/ multiple-RHS: $AX = B$
- Sequentially solved systems w/ multiple-RHS: $AX_i = B_i, i=1,...,t$
- Sequences of multiple-RHS systems: $A_iX_i = B_i, i=1,...,t$

- **Standard methods**

- Conjugate Gradients (CG), GMRES
- TFQMR, BiCGStab, MINRES, fixed-point

- **Advanced methods**

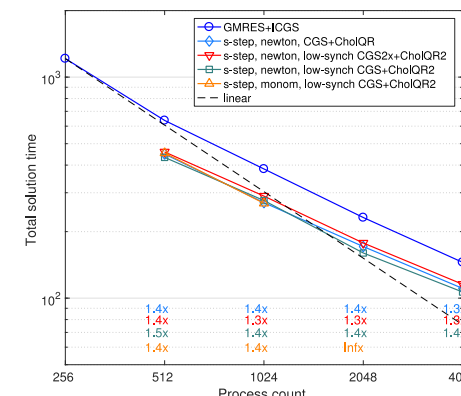
- Block GMRES, block CG/BICG
- Hybrid GMRES, CGRODR (block recycling GMRES)
- TSQR (tall skinny QR), LSQR
- Pipelined and s-step methods
- Stable polynomial preconditioning

- **Performance portability via Kokkos (CPUs, NVIDIA/Intel/AMD GPUs, Phi)**

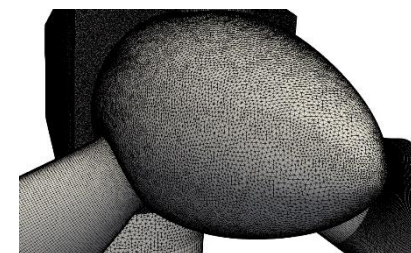
- **Ongoing research**

- Communication avoiding methods

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.



Speed-ups of various s-step Krylov methods within low Mach CFD wind-energy code Nalu-Wind.



Thomas et al., "High-fidelity simulation of wind- turbine incompressible flows", SISC, 2019.



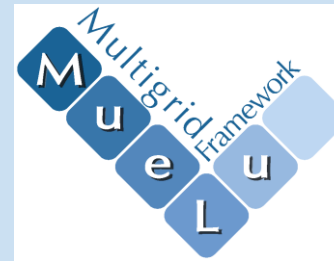
<https://trilinos.github.io/belos.html>

Trilinos/MueLu

Structured and unstructured aggregation-based algebraic multigrid (AMG) preconditioners

- **Robust, scalable, portable AMG preconditioning critical for many large-scale simulations**

- Multifluid plasma simulations
- Shock physics
- Magneto-hydrodynamics (MHD)
- Low Mach computational fluid dynamics (CFD)



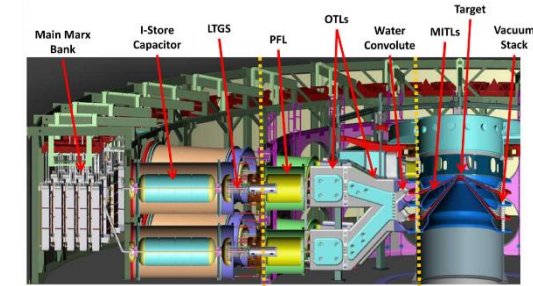
- **Capabilities**

- Aggregation-based coarsening
- **Smoothers**: Jacobi, GS, *l* GS, polynomial, ILU, sparse direct
- **Load-balancing** for good parallel performance
- Structured coarsening, geometric multigrid
- Setup and solve phases can run on GPUs.
- Performance portability via Kokkos (CPUs, NVIDIA/Intel/AMD GPUs, Xeon Phi)

- **Research Areas**

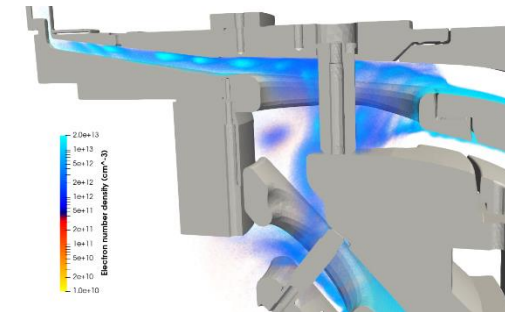
- AMG for multiphysics
- Multigrid for coupled structured/unstructured meshes
- Algorithm selection via machine learning

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.



Z machine diagram, from "Redesign of a High Voltage Test Bed for Marxes on Z", W.M. White et al., 2018.

AMG preconditioning for H(curl) systems is key enabling technology in Z machine simulations for determining power from Marx banks to Target.



Plasma density in Z machine Target simulation, courtesy of D. Sirajuddin (SNL).



<https://trilinos.github.io/muelu.html>

SuperLU



Supernodal Sparse LU Direct Solver. Flexible, user-friendly interfaces.
Examples show various use scenarios. Testing code for unit-test. BSD license.

Capabilities

- Serial (thread-safe), shared-memory (SuperLU_MT, OpenMP or Pthreads), distributed-memory (SuperLU_DIST, hybrid MPI+ OpenM + CUDA/HIP).
 - Written in C, with Fortran interface
- Sparse LU decomposition (can be nonsymmetric sparsity pattern), triangular solution with multiple right-hand sides
- Incomplete LU (ILUTP) preconditioner in serial SuperLU
- Sparsity-preserving ordering: minimum degree or graph partitioning applied to $A^T A$ or $A^T + A$
- User-controllable pivoting: partial pivoting, threshold pivoting, static pivoting
- Condition number estimation, iterative refinement, componentwise error bounds

Exascale early systems GPU-readiness

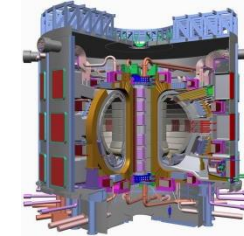
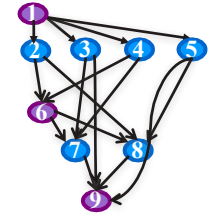
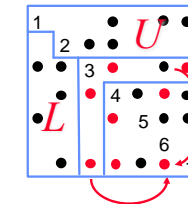
- Available: Nvidia GPU (CUDA), AMD GPU (HIP)
- In progress: Intel GPU (DPC++ planned)

Parallel Scalability

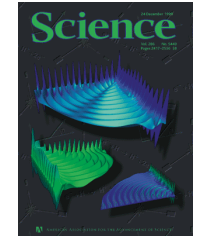
- Factorization strong scales to 32,000 cores (IPDPS'18, PARCO'19)
- Triangular solve strong scales to 4000 cores (SIAM CSC'18, SIAM PP'20)

Open-source software

- Used in a vast range of applications, can be used through PETSc and Trilinos, available on github



ITER tokamak



quantum mechanics

Widely used in commercial software, including
AMD (circuit simulation), Boeing (aircraft design),
Chevron, ExxonMobile (geology), Cray's LibSci,
FEMLAB, HP's MathLib, IMSL, NAG, SciPy,
OptimaNumerics, Walt Disney Animation.



<https://portal.nersc.gov/project/sparse/superlu>

STRUMPACK

Structured Matrix Package



Hierarchical solvers for dense rank-structured matrices and fast algebraic sparse solver and robust and scalable preconditioners.



■ Dense Matrix Solvers using Hierarchical Approximations

- Hierarchical partitioning, low-rank approximations
- Hierarchically Semi-Separable (HSS), Hierarchically Off-Diagonal Low-Rank (HODLR), Hierarchically Off-Diagonal Butterfly (HODBF), Block Low-Rank (BLR), Butterfly
- C++ Interface to ButterflyPACK (Fortran)
- Applications: BEM, Cauchy, Toeplitz, kernel & covariance matrices, ...
- Asymptotic complexity much lower than LAPACK/ScaLAPACK routines

■ Sparse Direct Solver

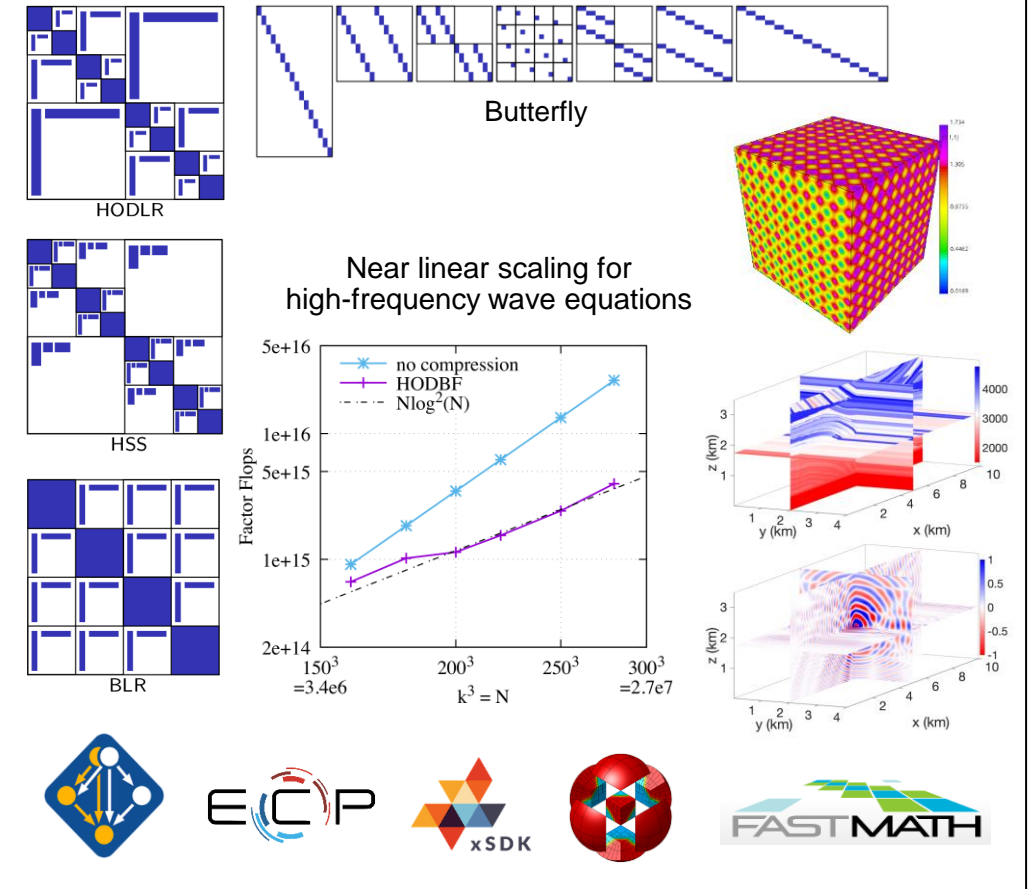
- Algebraic sparse direct solver
- GPU: CUDA, HIP/ROCm, DPC++ (in progress)
- Orderings: (Par)METIS, (PT)Scotch, RCM

■ Preconditioners

- Approximate sparse factorization, using hierarchical matrix approximations
- Scalable and robust, aimed at PDE discretizations, indefinite systems, ...
- Iterative solvers: GMRES, BiCGStab, iterative refinement

■ Software

- BSD license
- Interfaces from PETSc, MFEM, Trilinos, available in Spack



github.com/pghysels/STRUMPACK

Agenda

<https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break	
3:15 – 4:30	Optimization (TAO) – Todd Munson, Richard Mills	Iterative Solvers & Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up / Panel	
6:30 – 7:30	Dinner	
7:30 – 9:30	SME Speed Dating	



Adaptive time integrators for ODEs and DAEs and efficient nonlinear solvers
Used in a variety of applications. Freely available. Encapsulated solvers & parallelism.

■ ODE and DAE time integrators:

- *CVODE*: adaptive order and step BDF (stiff) & Adams (non-stiff) methods for ODEs
- *ARKODE*: adaptive step implicit, explicit, IMEX, and multirate Runge-Kutta methods for ODEs
- *IDA*: adaptive order and step BDF methods for DAEs
- *CVODES* and *IDAS*: provide forward and adjoint sensitivity analysis capabilities

■ Nonlinear Solvers: *KINSOL* – Newton-Krylov; accelerated Picard and fixed point

■ Modular Design: Easily incorporated into existing codes; Users can supply their own data structures and solvers or use SUNDIALS provided modules

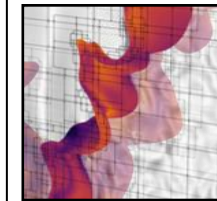
■ Support on NVIDIA, AMD, and Intel GPUs:

- Vectors: CUDA, HIP, OpenMP Offload, RAJA, SYCL (DPC++)
- Linear solvers: cuSOLVER, MAGMA, matrix-free Krylov methods

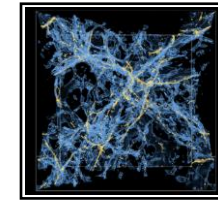
■ Open Source: BSD License; Download from LLNL site, GitHub, or Spack

- Supported by extensive documentation; user email list with an active community
- Available through MFEM, AMReX, deal.II, and PETSc

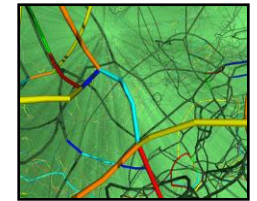
SUNDIALS is used worldwide in applications throughout research and industry



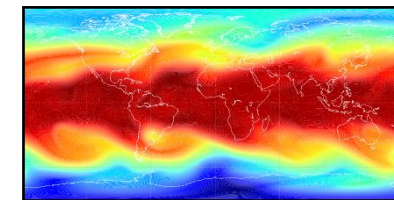
Combustion
(Pele)



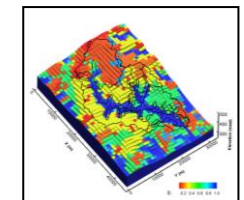
Cosmology
(Nyx)



Dislocation dynamics
(ParaDiS)



Atmospheric Dynamics
(Tempest)

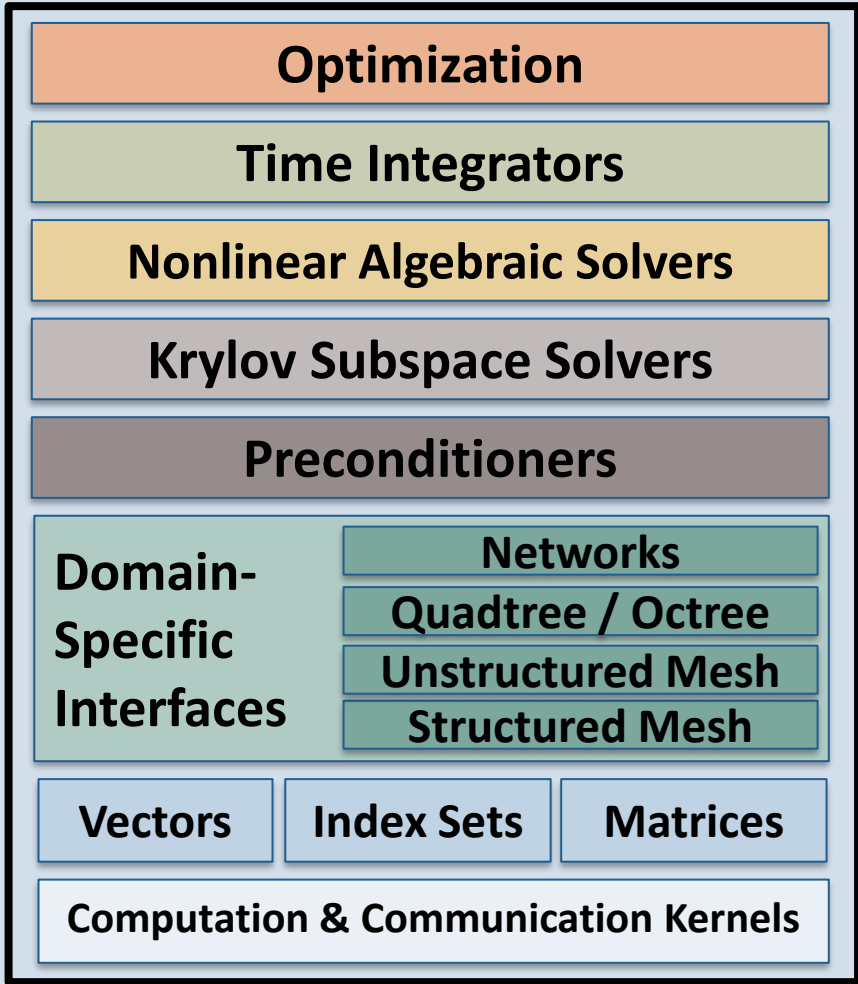


Subsurface flow
(ParFlow)



<http://www.llnl.gov/casc/sundials>

Scalable algebraic solvers for PDEs. Encapsulate parallelism in high-level objects. Active & supported user community. Full API from Fortran, C/C++, Python.

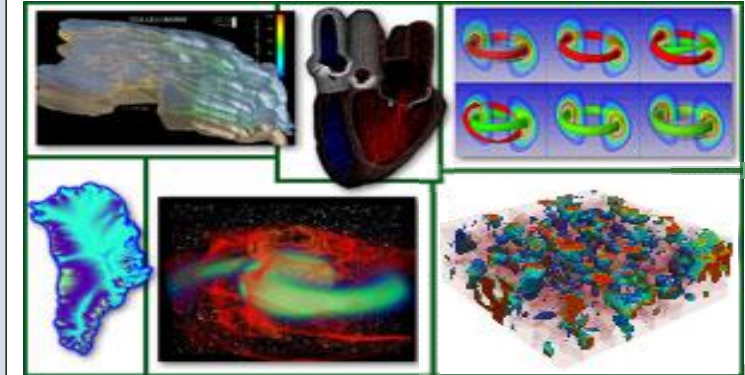


■ **Easy customization and composability of solvers at runtime**

- Enables optimality via flexible combinations of physics, algorithmics, architectures
- Try new algorithms by composing new/existing algorithms (multilevel, domain decomposition, splitting, etc.)

■ **Portability & performance**

- Largest DOE machines, also clusters, laptops; NVIDIA, AMD, and Intel GPUs
- Thousands of users worldwide



PETSc provides the backbone of diverse scientific applications.
 clockwise from upper left: hydrology, cardiology, fusion, multiphase steel, relativistic matter, ice sheet modeling



<https://www.mcs.anl.gov/petsc>

Agenda

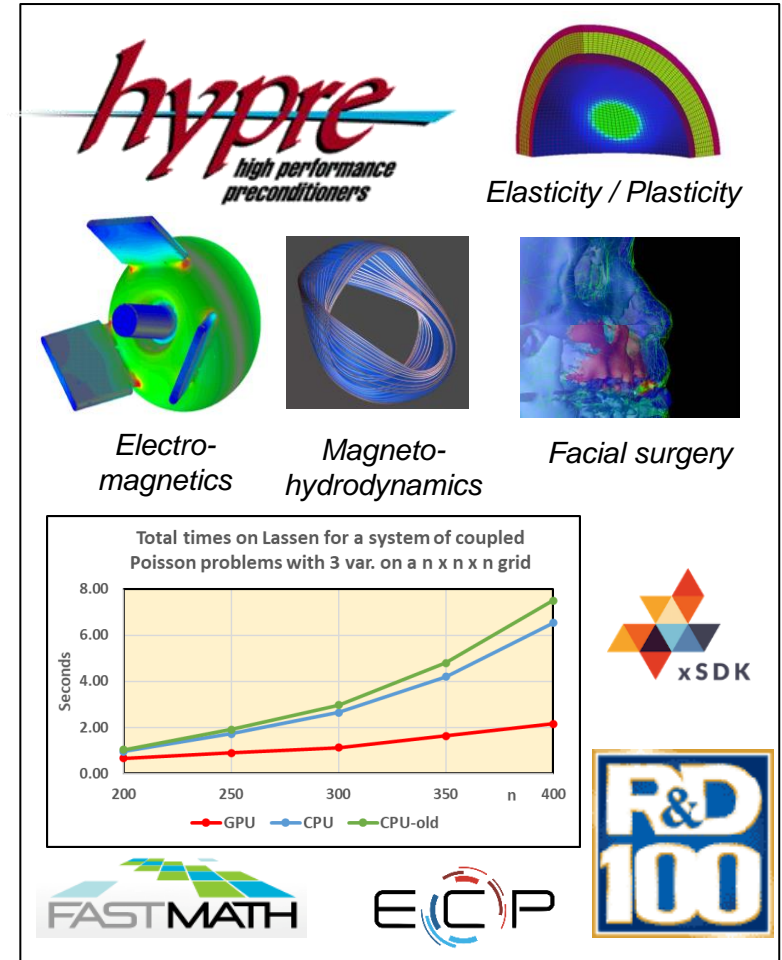
<https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break	
3:15 – 4:30	Optimization (TAO) – Todd Munson, Richard Mills	Iterative Solvers & Algebraic Multigrid (hype) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up / Panel	
6:30 – 7:30	Dinner	
7:30 – 9:30	SME Speed Dating	



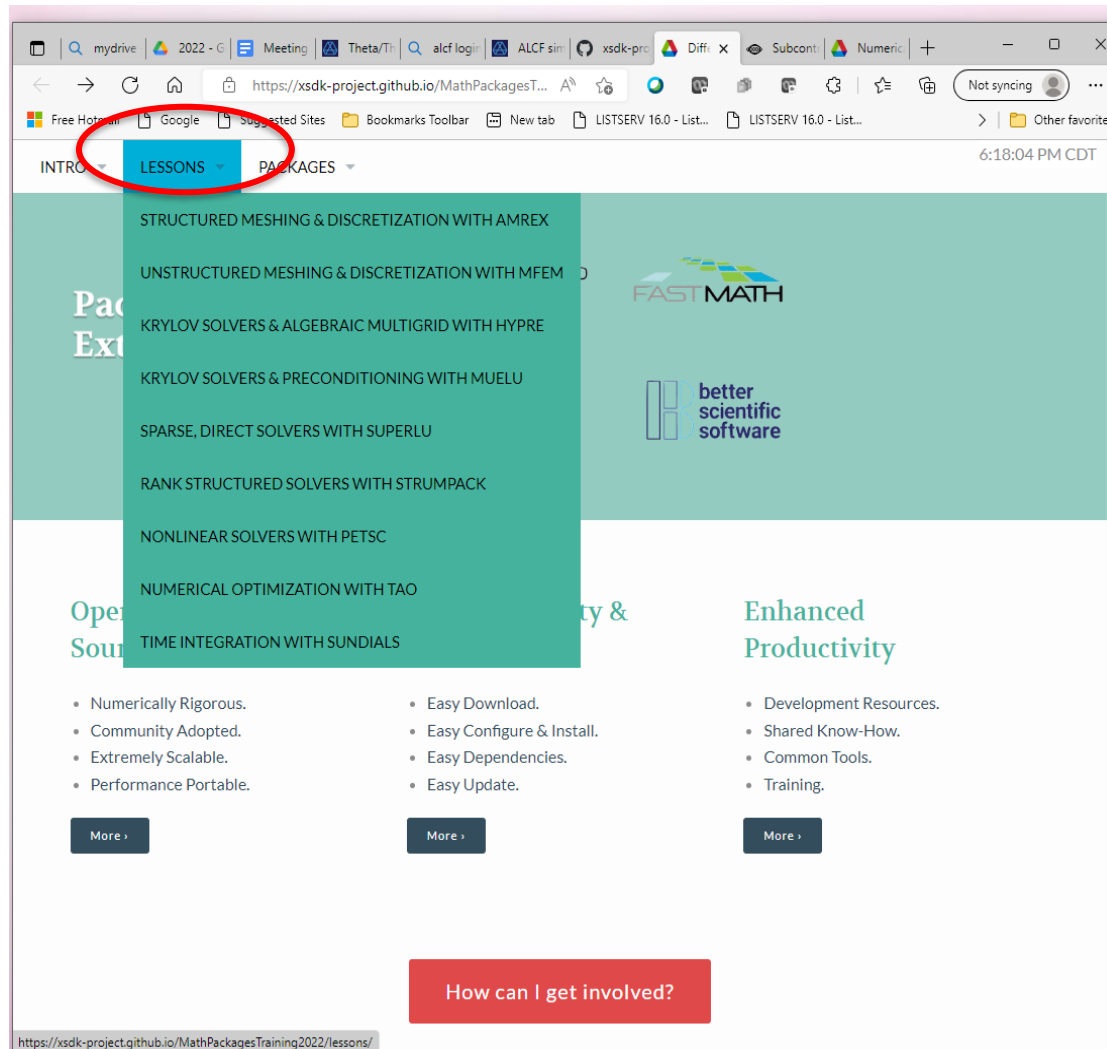
Highly scalable multilevel solvers and preconditioners. Unique user-friendly interfaces. Flexible software design. Used in a variety of applications. Freely available.

- **Conceptual interfaces**
 - Structured, semi-structured, finite elements, linear algebraic interfaces
 - Provide natural “views” of the linear system
 - Provide for efficient (scalable) linear solvers through effective data storage schemes
- **Scalable preconditioners and solvers**
 - Structured and unstructured algebraic multigrid solvers
 - Maxwell solvers, H-div solvers
 - Multigrid solvers for nonsymmetric systems: pAIR, MGR
 - Matrix-free Krylov solvers
- **Exascale early systems GPU-readiness**
 - Available: Nvidia GPU (CUDA), AMD GPU (HIP)
 - In progress: Intel GPU (SYCL)
- **Open-source software**
 - Used worldwide in a vast range of applications
 - Can be used through PETSc and Trilinos
 - Provide CPU and GPU support
 - Available on github: <https://www.github.com/hypre-space/hypre>



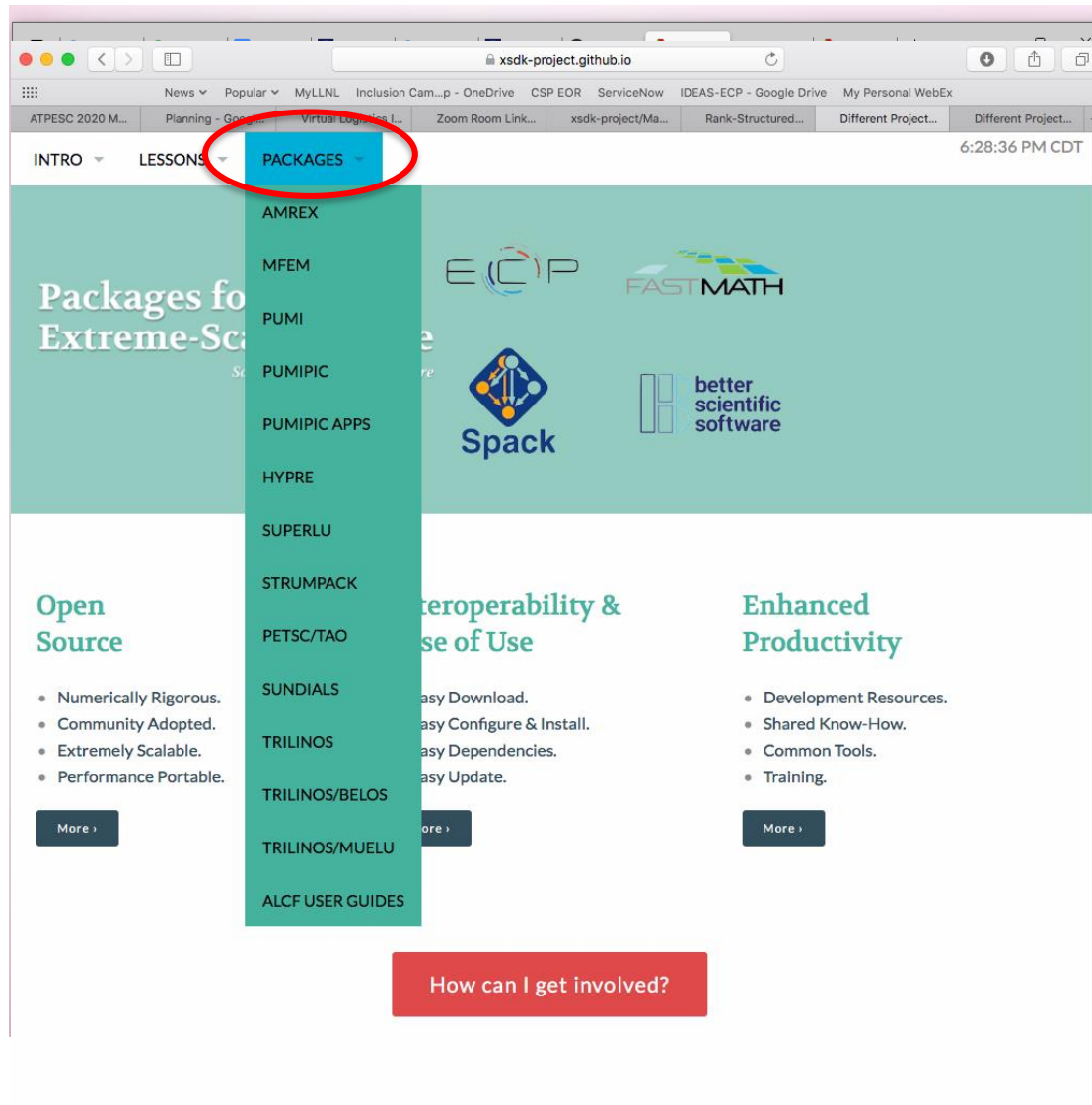
<http://www.llnl.gov/CASC/hypre>

<https://xsdk-project.github.io/MathPackagesTraining2022/>



- Hands-on Lessons

<https://xsdk-project.github.io/MathPackagesTraining2022/>



- Hands-on Lessons
- Packages

Agenda

<https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5>

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break, Subject Matter Expert (SME) Selections, Panel Questions	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch, SME Selections, Panel Questions	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break, SME Selections, Panel Questions Due	
3:15 – 4:30	Optimization (TAO) – Todd Munson, Richard Mills	Iterative Solvers & Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up (Ann Almgren) / Panel: Extreme-Scale Numerical Algorithms and Software	
5:30 – 6:30	Unstructured Time: SME Selections Due , Informal Discussion, Continue Hands-on	
6:30 – 7:30	Dinner	
7:30 – 9:30	Optional Activity: SME Speed-dating	

Next steps: <https://xsdk-project.github.io/MathPackagesTraining2022/agenda>

- **WrapUp (Ann Almgren) @ 4:30pm**

Panel: Main Room @ 4:45 pm

- **SME Speed Dating: @7:30pm**

- **During breaks and lunch**

- **Provide Panel Questions**

Due: 3:15 pm

- **Sign up for discussions with numerical software developers (optional)**

- **Your email address**

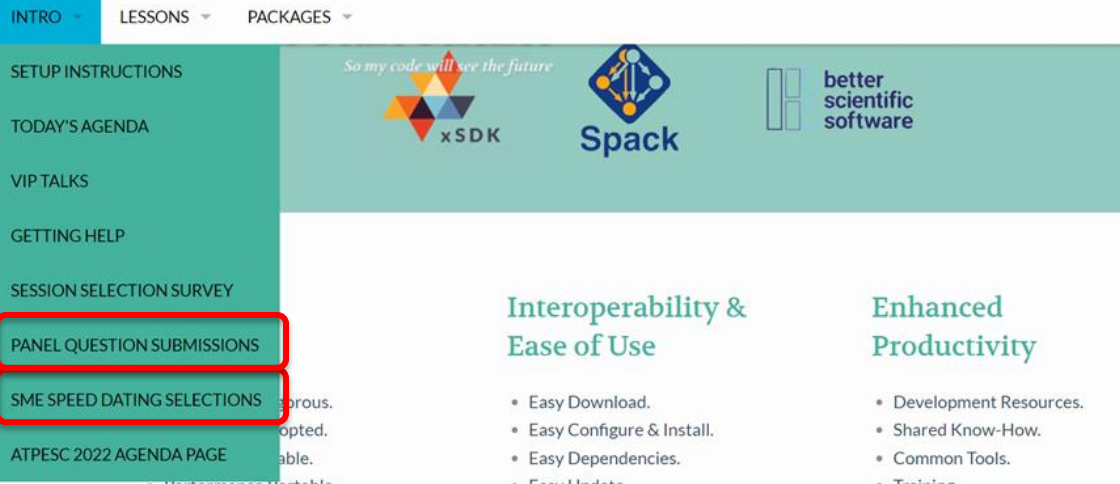
Due 6:30 pm



Subject Matter Expert (SME) 2-on-1 interviews

This is an optional activity. It is a great opportunity to spend some time chatting with various subject matter experts (SMEs).

In the form below, you may enter your first, second and third priorities for up to three, 20 minute, two-on-one discussions with various SMEs during the evening session.



The screenshot shows the xSDK website. The navigation menu on the left includes: INTRO, LESSONS, PACKAGES, SETUP INSTRUCTIONS, TODAY'S AGENDA, VIP TALKS, GETTING HELP, SESSION SELECTION SURVEY, **PANEL QUESTION SUBMISSIONS**, **SME SPEED DATING SELECTIONS**, and ATPESC 2022 AGENDA PAGE. The header features the slogan "So my code will see the future" and logos for xSDK, Spack, and better scientific software. The main content area highlights "Interoperability & Ease of Use" and "Enhanced Productivity" with lists of benefits.

Interoperability & Ease of Use

- Easy Download.
- Easy Configure & Install.
- Easy Dependencies.
- Easy Update.

Enhanced Productivity

- Development Resources.
- Shared Know-How.
- Common Tools.
- Training.

Panel: Extreme-Scale Numerical Algorithms and Software

- **Q&A Session:** ATPESC learners ask questions about working with numerical packages and the community of numerical package developers
 - Questions in **#numerical** slack channel and via Google form

- Panelists



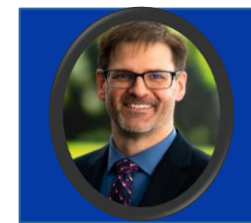
Pieter Ghysels, LBL



Sarah Osborn, LLNL



Dan Reynolds, SMU

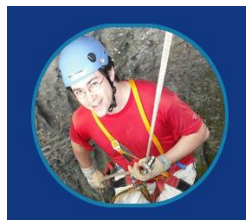


Todd Munson, ANL



Graham Harper, SNL

- Moderator



Richard Mills, ANL

Panel Question Submission Form

Please enter here a question you would like to ask our panelists during the 45 minute panel session.

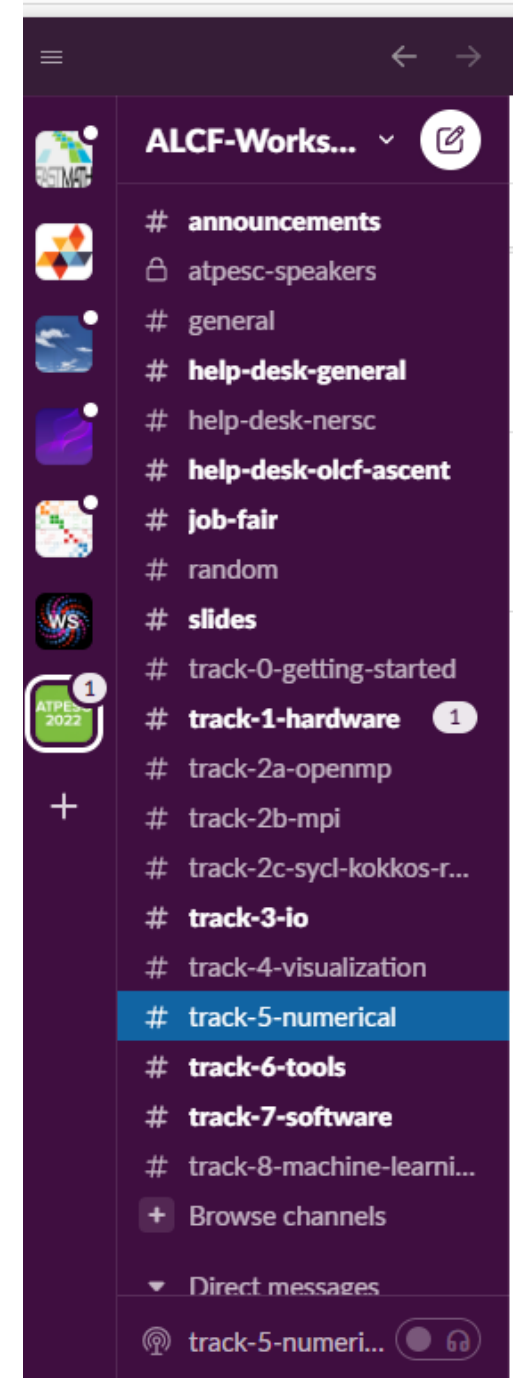
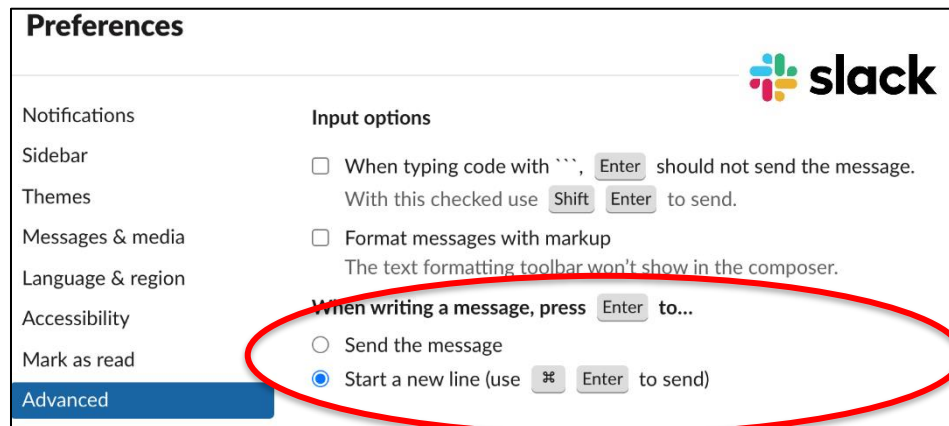
We ask that you please include your name in case we may need to call upon you to clarify your question.

Using Slack



- Recommend using the desktop app, but browser ok too
- **#track-5-numerical** channel
 - For all chat during presentations
 - For all chat outside any specific parallel session
 - For general help
 - Recommend using the thread option to help keep track of discussions on subtopics

Tip: Consider setting Preferences to customize when to send



Track 5: Numerical Algorithms and Software: Tutorial Goals

1.

Provide a basic understanding of a variety of applied mathematics algorithms for scalable linear, nonlinear, and ODE solvers, as well as discretization technologies (e.g., adaptive mesh refinement for structured and unstructured grids) and numerical optimization

2.

Provide an overview of software tools available to perform these tasks on HPC architectures ... including where to go for more info

3.

Practice using one or more of these software tools on basic demonstration problems

This presentation provides a high-level introduction to HPC numerical software

- How HPC numerical software addresses challenges in computational science and engineering (CSE)
- Toward extreme-scale scientific software ecosystems
- Using and contributing: Where to go for more info

Why is this important for you?

- Libraries enable users to focus on their primary interests
 - Reuse algorithms and data structures developed by experts
 - Customize and extend to exploit application-specific knowledge
 - Cope with complexity and changes over time
- More efficient, robust, reliable, scalable, sustainable scientific software
- Better science, broader impact of your work

The ATPESC Team 2022

Extreme-scale numerical algorithms and software
Integrated lectures and hands-on examples, panel session, individual discussions ... and more!



Ann Almgren, LBL



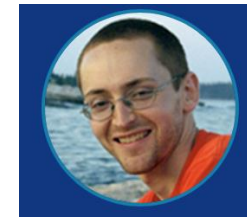
Erik Palmer, LBL



Aaron Fisher, LLNL



Mark Shephard, RPI



Cameron Smith, RPI



Ulrike Yang, LLNL



Christian Glusa, SNL



Graham Harper, SNL



Sherry Li, LBL



Pieter Ghysels, LBL



Satish Balay, ANL



Sarah Osborn, LLNL



Dan Reynolds, SMU



David Gardner, LLNL



Richard Mills, ANL

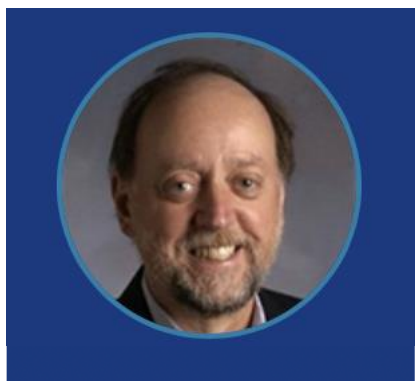


Todd Munson, ANL

VIPs of ATPESC Extreme-Scale Numerical Software Track



- **Jim Demmel, UC Berkeley** [[bio](#)]
 - Communication-Avoiding Algorithms for Linear Algebra, Machine Learning, and Beyond
 - ATPESC 2019 [[slides](#), [video](#)]
 - ENLA Seminar, June 2020 [[video](#)]



- **Jack Dongarra, Univ of Tennessee** [[bio](#)]
 - **An Accidental Benchmark, ATPESC 2022, tomorrow, 8/10, 7:30pm, ATPESC 2021** [[slides](#)]
 - Adaptive Linear Solvers and Eigensolvers, ATPESC 2019 [[slides](#), [video](#)]



- **David Keyes, KAUST** [[bio](#)]
 - Adaptive Nonlinear Preconditioning for PDEs with Error Bounds on Output Functionals, University of Manchester, 2021 [[slides](#), [video](#)]
 - Data-sparse Linear Algebra for Large-scale Applications on Emerging Architectures, ENLA Seminar, September 2020 [[slides](#), [video](#)]

This work is founded on decades of experience and concerted team efforts to advance numerical software ...



- Exascale Computing Project
- FASTMath SciDAC Institute
- Developers of xSDK packages

... While improving software productivity & sustainability as key aspects of advancing overall scientific productivity



- IDEAS Software Productivity Project
- Better Scientific Software Community

See also Track 7:
Software Productivity and Sustainability (Aug 11)

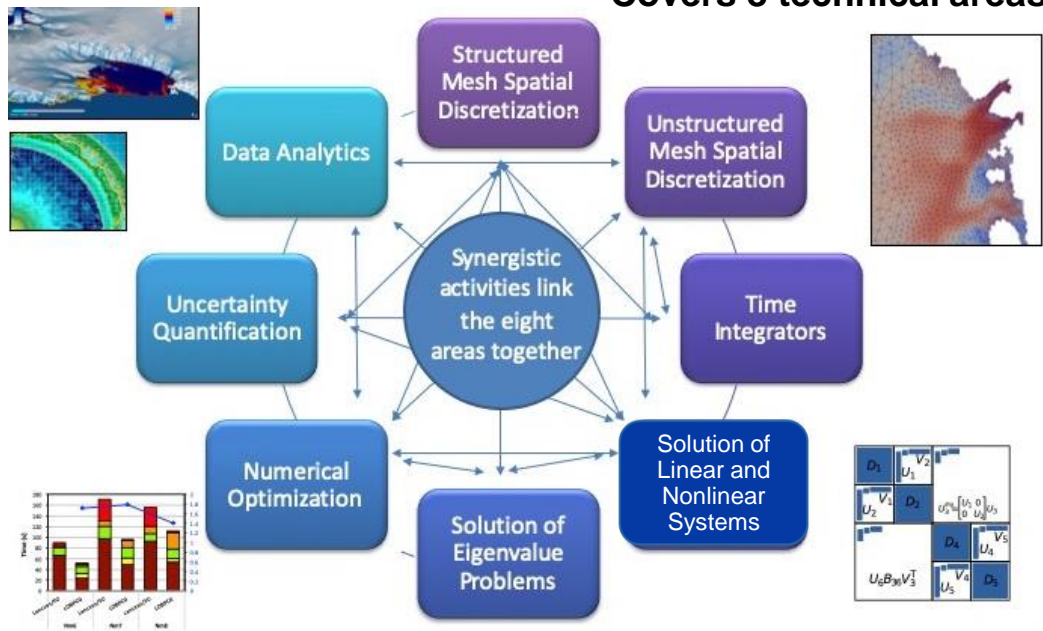
Community efforts:
Join us!



FASTMath: Frameworks, Algorithms & Scalable Technologies for Mathematics

<https://scidac5-fastmath.lbl.gov/>

Covers 8 technical areas



- FASTMath Goals:**
- Develop advanced numerical techniques for DOE applications
 - Deploy high-performance software on DOE supercomputers
 - Demonstrate basic research technologies from applied mathematics
 - Engage and support of the computational science community

100's of person years of experience building math software

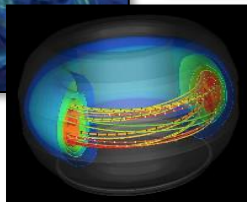
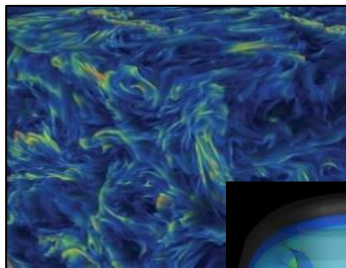
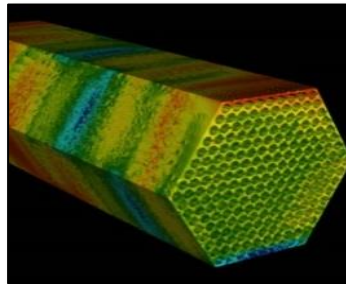
50+ researchers from 5 DOE labs and 5 universities



ECP's holistic approach uses co-design and integration to achieve exascale computing

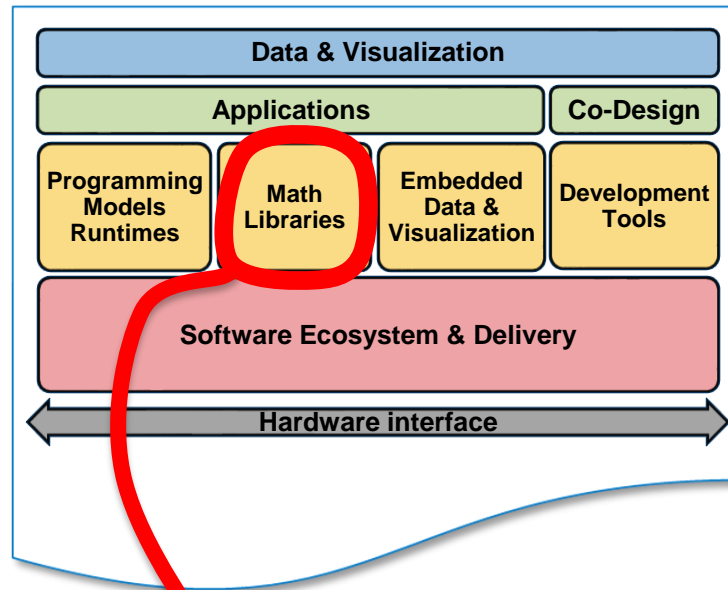
Application Development

Science and mission applications



Software Technology

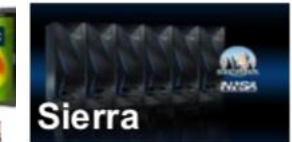
Scalable software stack



Emphasis for this presentation

Hardware and Integration

Relationships: facilities with AD/ST, with vendors

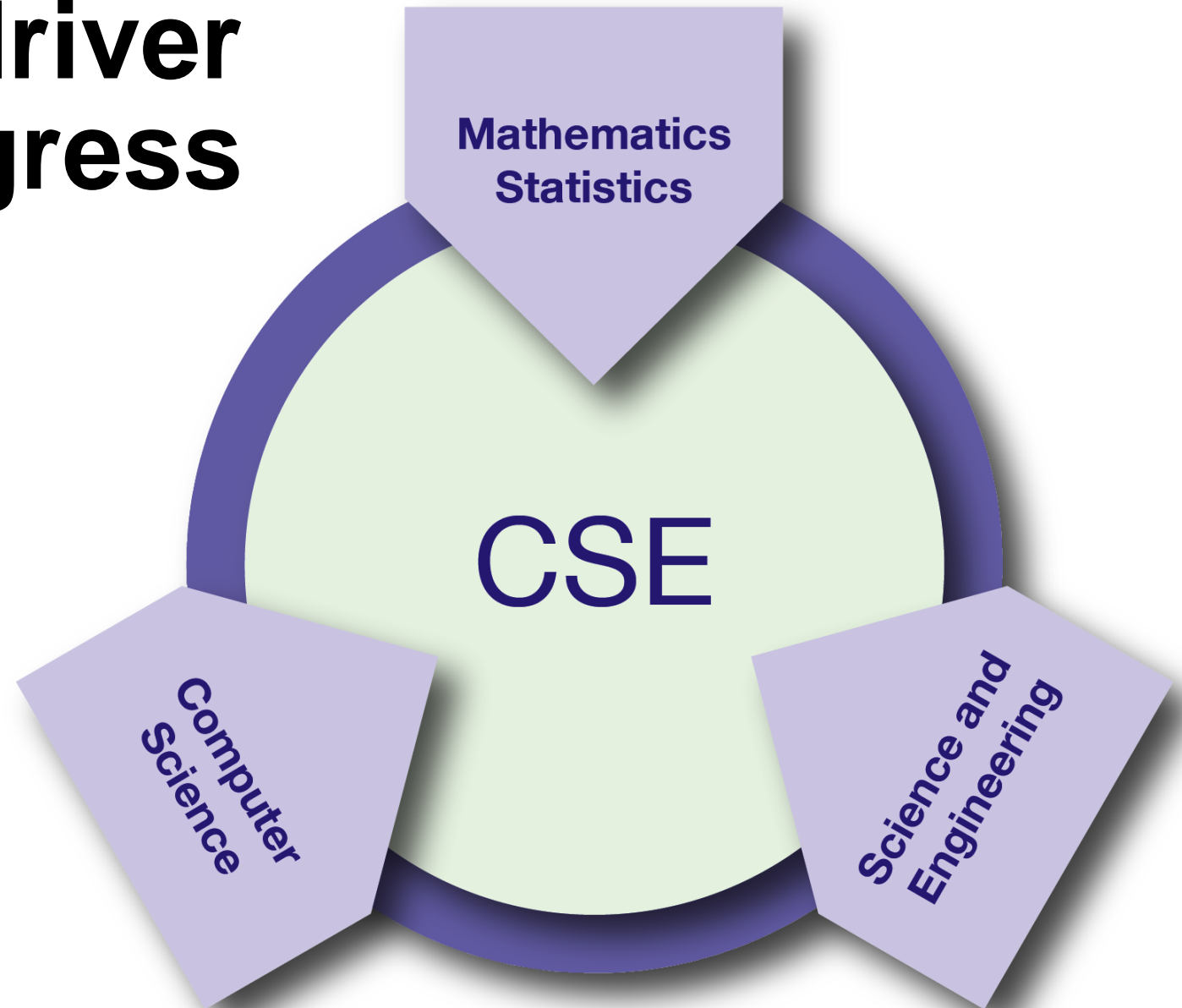


CSE: Essential driver of scientific progress

CSE = Computational Science & Engineering

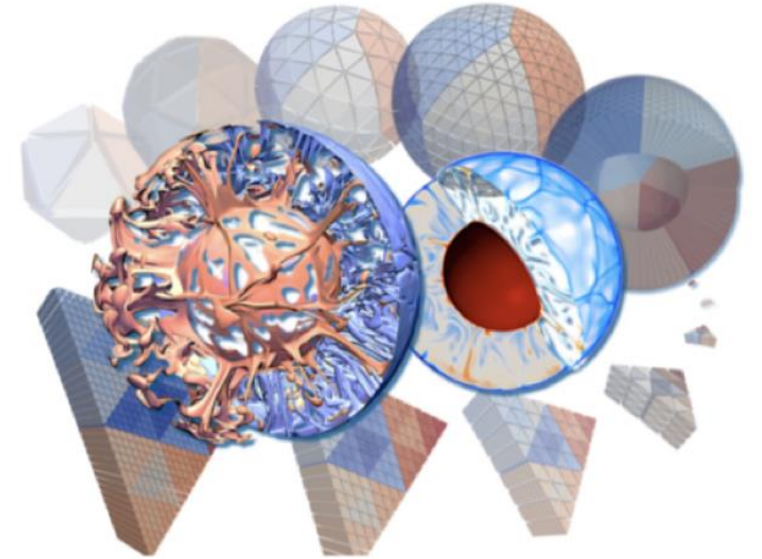
Development and use of computational methods for scientific discovery

- all branches of the sciences
- engineering and technology
- support of decision-making across a spectrum of societally important applications



Rapidly expanding role of CSE: New directions toward predictive science

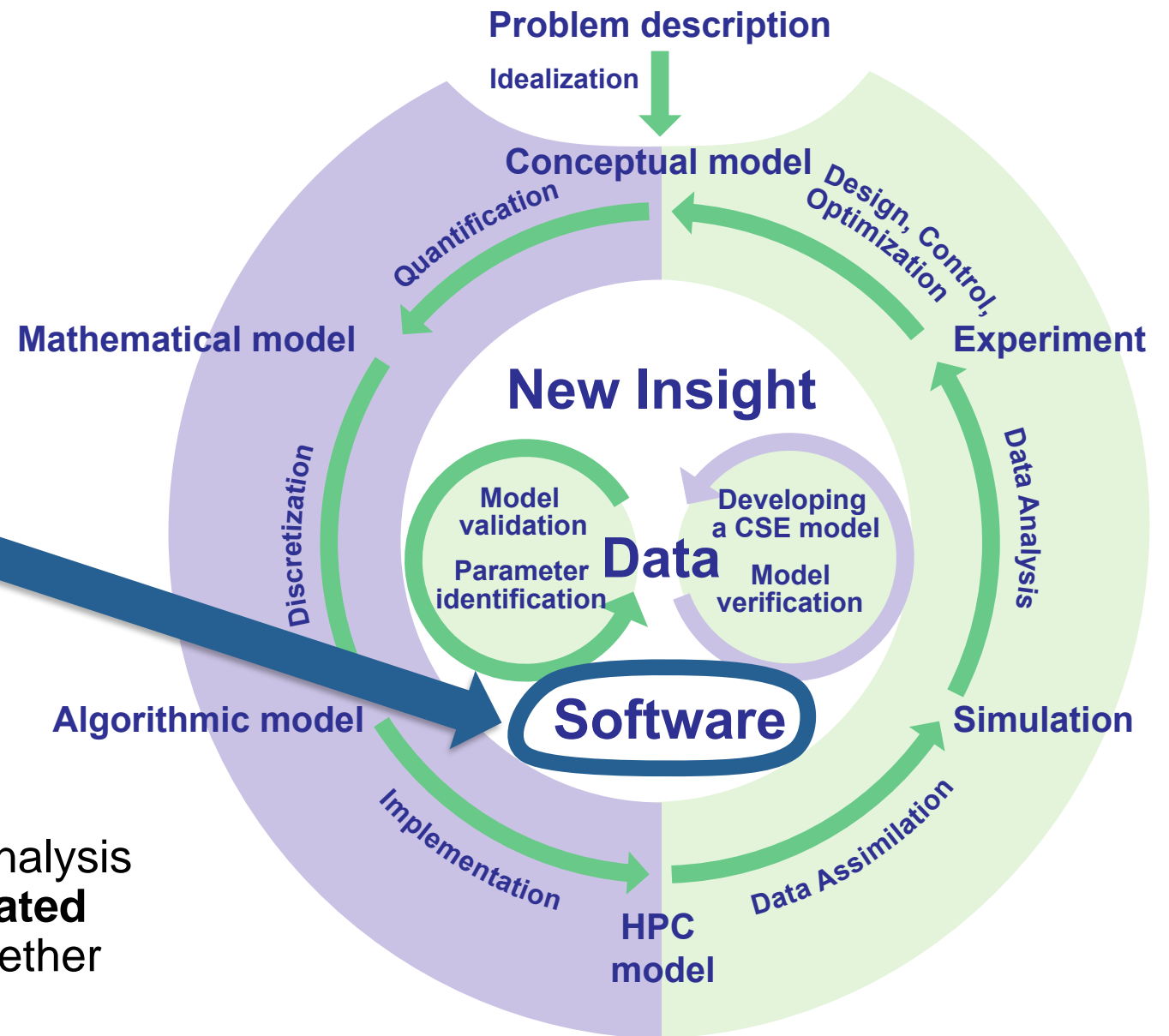
- Mathematical methods and algorithms
- CSE and HPC: Ubiquitous parallelism
- CSE and the data revolution
- CSE software
- CSE education & workforce development



Research and Education in Computational Science & Engineering

U. Rüde, K. Willcox, L.C. McInnes, H. De Sterck, G. Biros, H. Bungartz, J. Coronas, E. Cramer, J. Crowley, O. Ghattas, M. Gunzburger, M. Hanke, R. Harrison, M. Heroux, J. Hesthaven, P. Jimack, C. Johnson, K. Jordan, D. Keyes, R. Krause, V. Kumar, S. Mayer, J. Meza, K.M. Mørken, J.T. Oden, L. Petzold, P. Raghavan, S. Shontz, A. Trefethen, P. Turner, V. Voevodin, B. Wohlmuth, C.S. Woodward, **SIAM Review**, 60(3), Aug 2018, <https://doi.org/10.1137/16M1096840>.

Software is the foundation of sustained CSE collaboration and scientific progress.



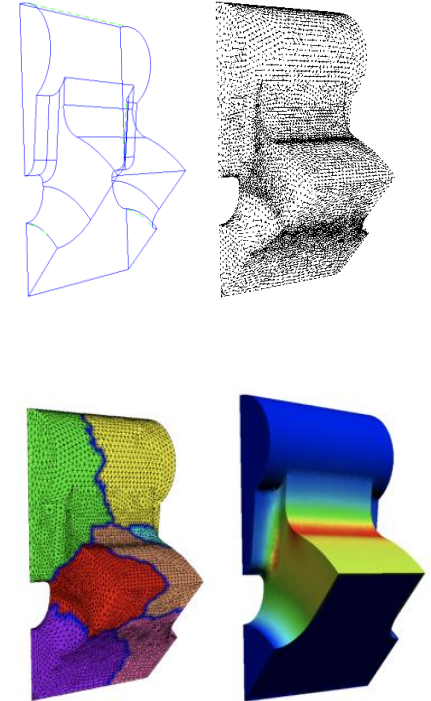
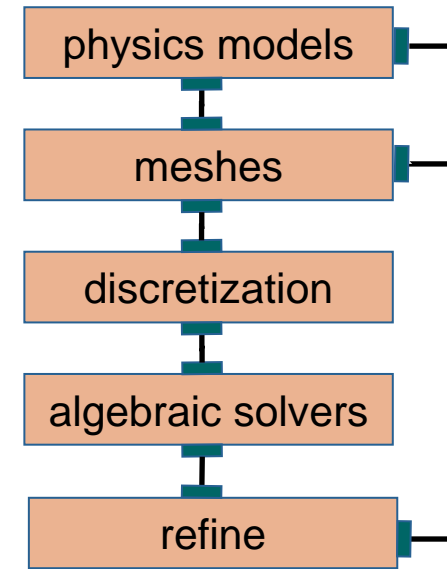
CSE cycle: Modeling, simulation, and analysis

- **Software: independent but interrelated elements** for various phases that together enable CSE

CSE simulation starts with a forward simulation that captures the physical phenomenon of interest

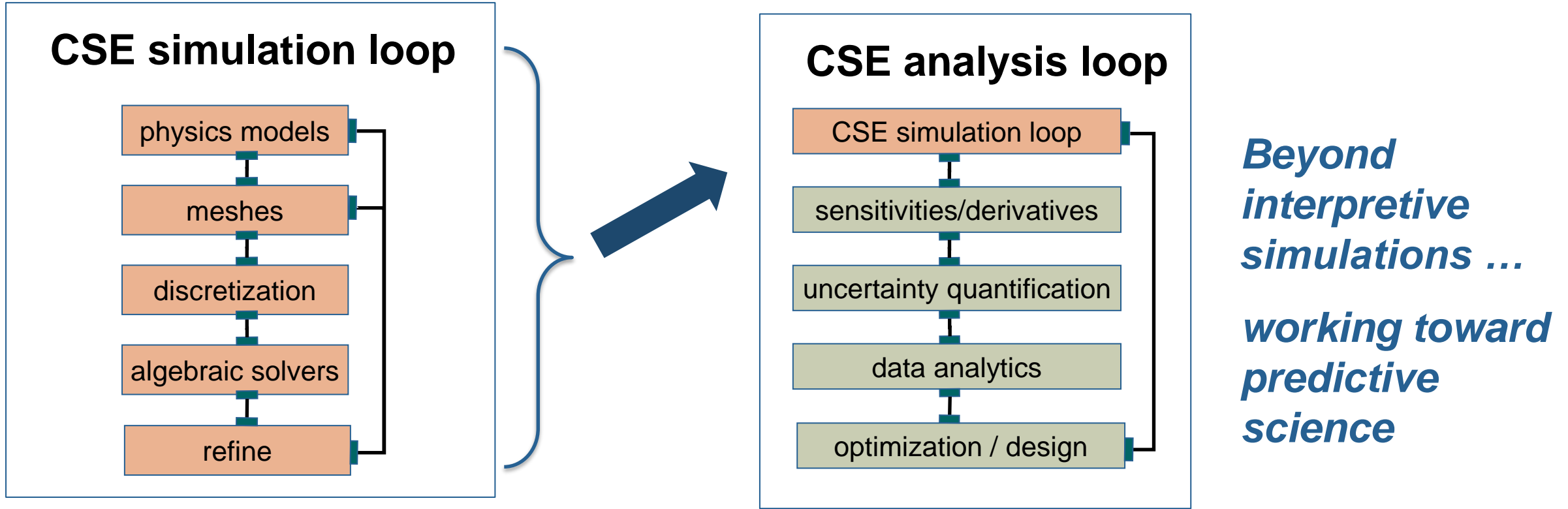
- Develop a mathematical model of the phenomenon of interest
- Approximate the model using a discrete representation
- Solve the discrete representation
- Adapt and refine the mesh or model
- Incorporate different physics, scales

CSE simulation loop



Requires: mesh generation, partitioning, load balancing, high-order discretization, time integration, linear & nonlinear solvers, eigensolvers, mesh refinement, multiscale/multiphysics coupling, etc.

CSE analysis builds on the CSE simulation loop ... and relies on even more numerical algorithms and software



Requires: adjoints, sensitivities, algorithmic differentiation, sampling, ensembles, data analytics, uncertainty quantification, optimization (derivative free & derivative based), inverse problems, etc.

First consider a very simple example

- 1D rod with one end in a hot water bath, the other in a cold water bath
- Mathematical model

$$\nabla^2 T = 0 \in \Omega$$
$$T(0) = 180^\circ \quad T(1) = 0^\circ$$

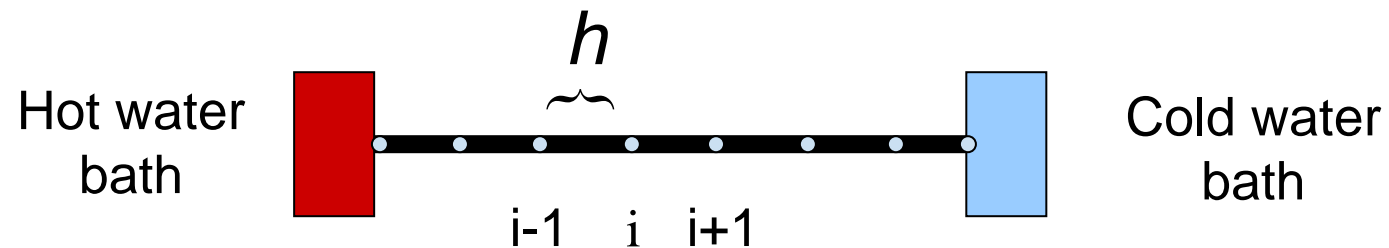


The first step is to discretize the equations

- Approximate the derivatives of the continuous equations with a discrete representation that is easier to solve
- One approach: Finite differences

$$\nabla^2 T \approx (T_{i+1} - 2T_i + T_{i-1})/h^2 = 0$$

$$T_0 = 180^\circ \quad T_n = 0^\circ$$

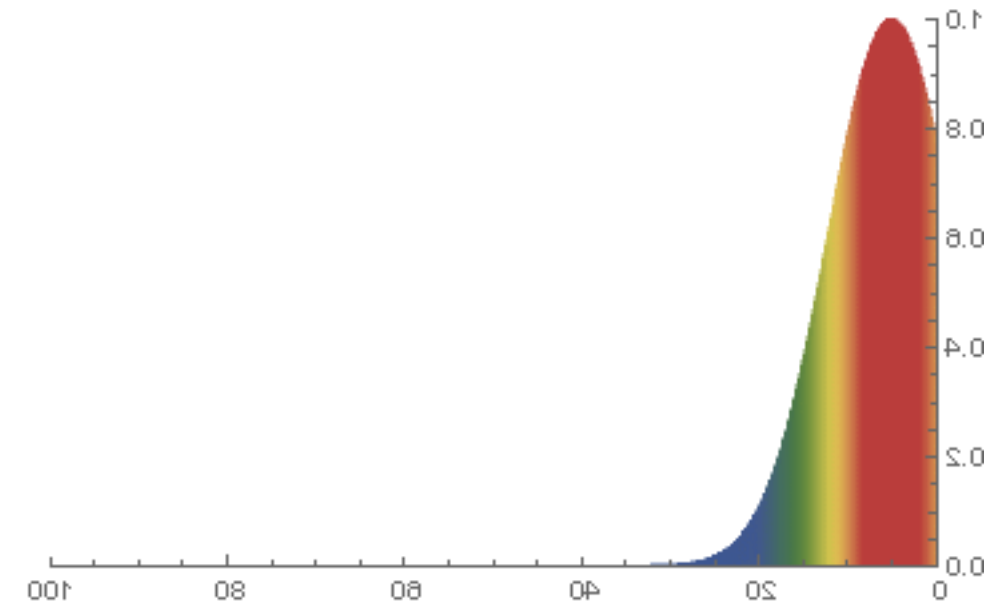


Then you can solve for the unknowns T_i

- Set up a matrix of the unknown coefficients
 - include the known boundary conditions
- Solve the linear system for T_i

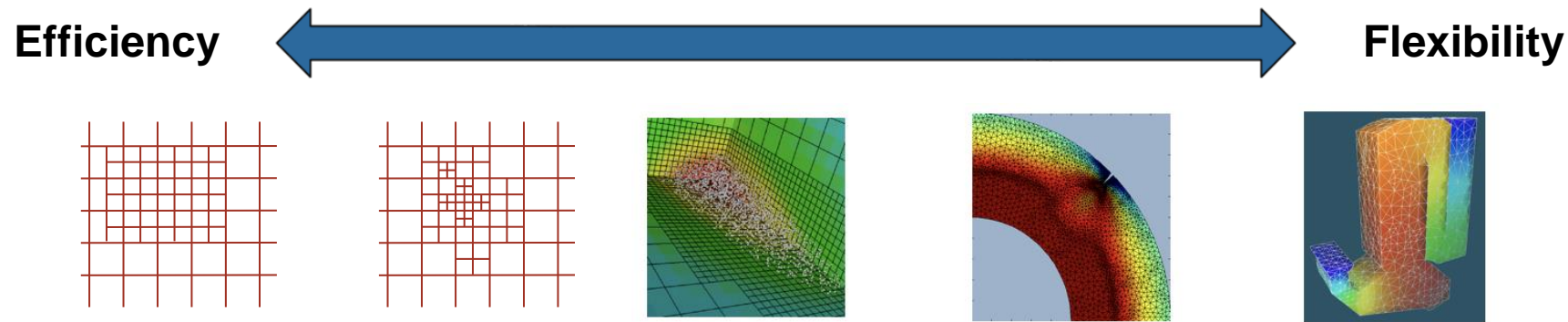
$$\begin{pmatrix} 2 & -1 & 0 & \dots\dots\dots 0 \\ -1 & 2 & -1 & 0 & \dots\dots\dots 0 \\ 0 & -1 & 2 & -1 & 0 & \dots\dots 0 \\ & & & \dots\dots\dots & & \\ 0 & \dots\dots\dots 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{n-1} \end{pmatrix} = \begin{pmatrix} 180 h^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- Visualize and analyze the results



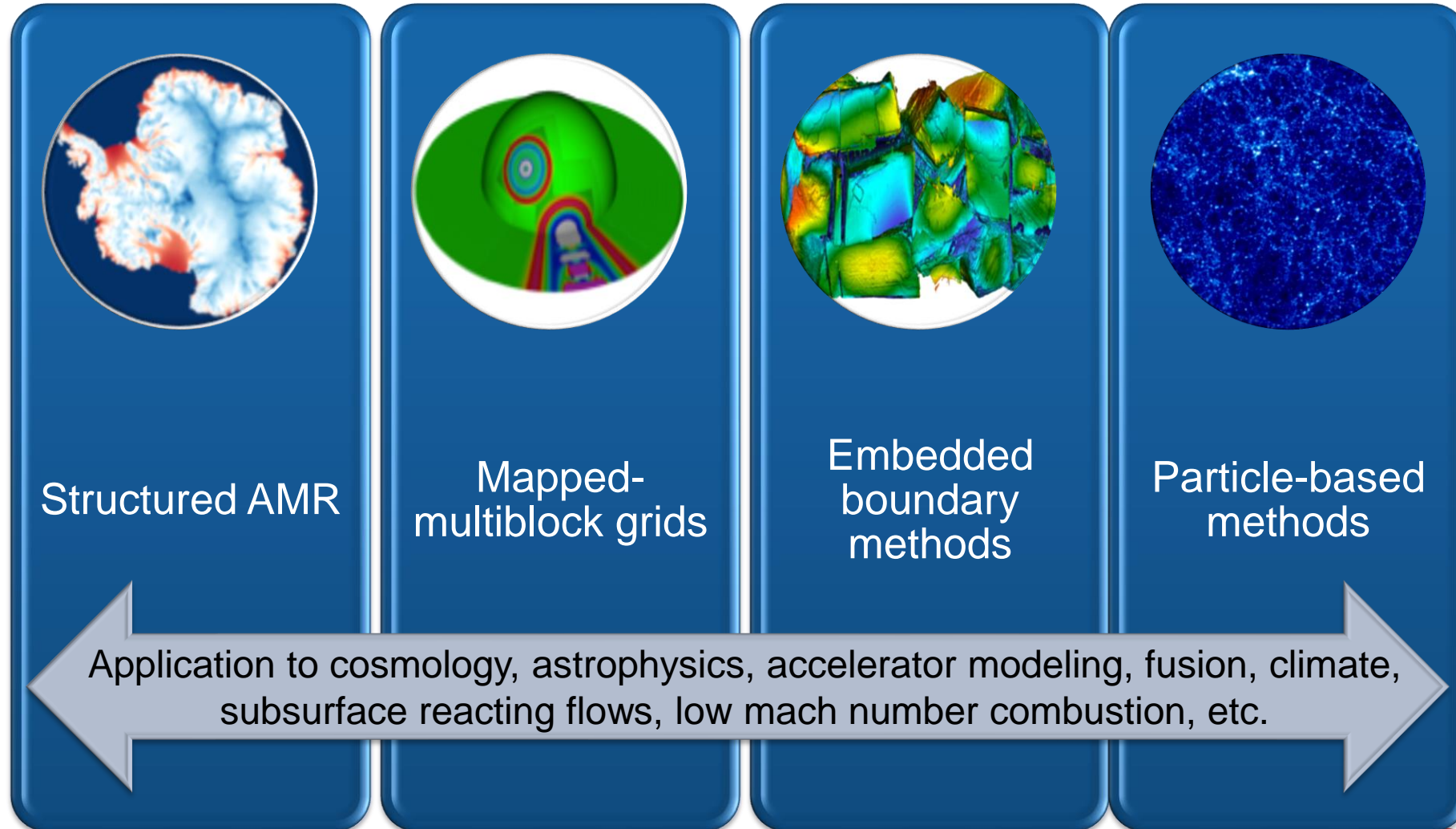
As problems get more complicated, so do the steps in the process

- Different discretization strategies exist for differing needs

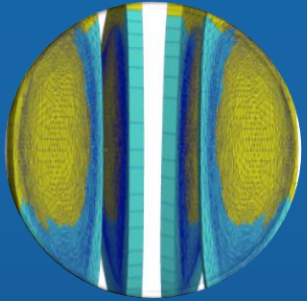


- Most problems are time dependent and nonlinear
 - Need higher algorithmic levels than linear solvers
- Increasingly combining multiple physical processes
 - Interactions require careful handling
- Goal-oriented problem solving requires optimization, uncertainty quantification

Structured grid efforts focus on high-order, mapped grids, embedded boundaries, AMR, and particles



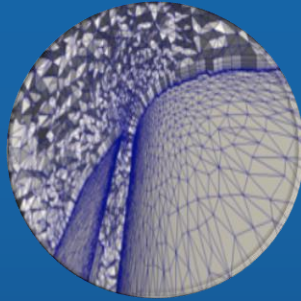
Unstructured grid capabilities focus on adaptivity, high-order, and the tools needed for extreme scaling



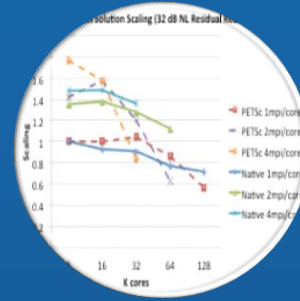
Parallel mesh infrastructures



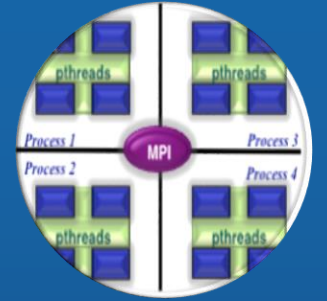
Dynamic load balancing



Mesh adaptation and quality control



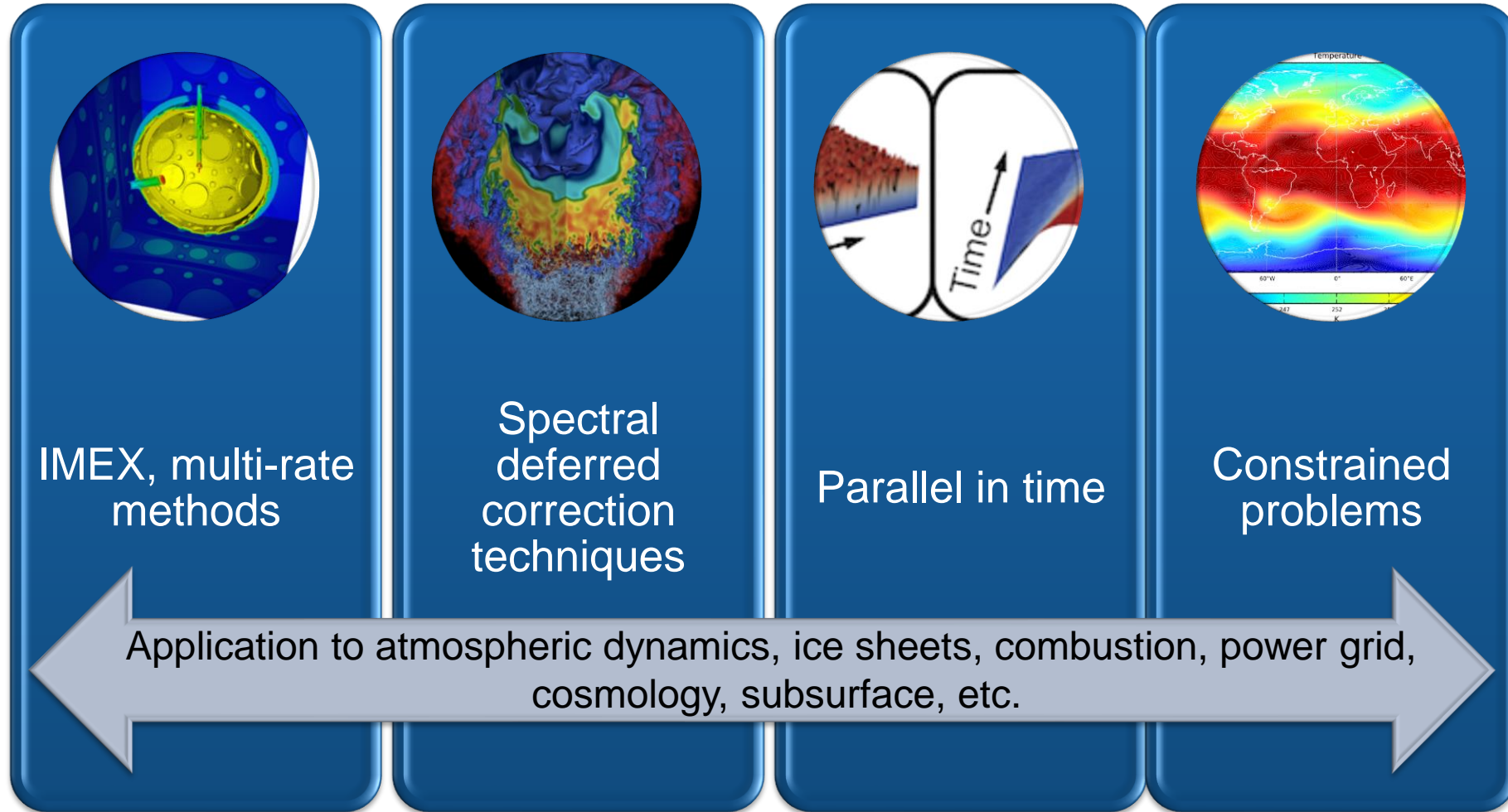
Parallel performance on unstructured meshes



Architecture aware implementations

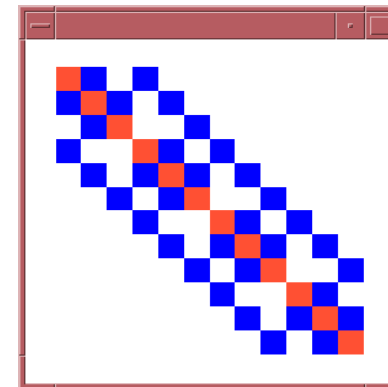
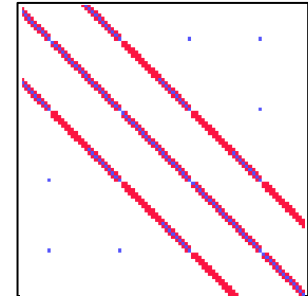
Application to fusion, climate, accelerator modeling, NNSA applications, nuclear energy, manufacturing processes, etc.

Time discretization methods provide efficient and robust techniques for stiff implicit, explicit and multi-rate systems

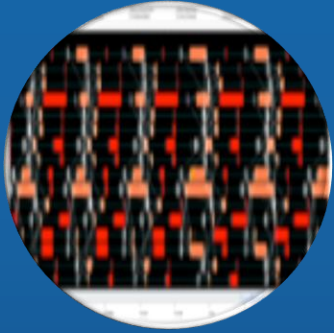


As problems grow in size, so do corresponding discrete systems

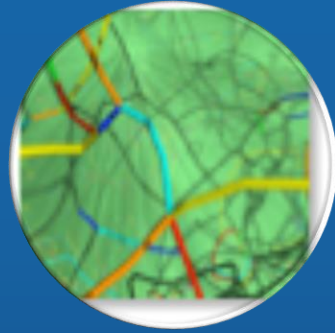
- Targeting applications with billions grid points and unknowns
- Most linear systems resulting from these techniques are LARGE and sparse
- Often most expensive solution step
- Solvers:
 - Direct methods (e.g., Gaussian Elimination)
 - Iterative methods (e.g., Krylov Methods)
 - Preconditioning is typically critical
 - Mesh quality affects convergence rate
- Many software tools deliver this functionality as numerical libraries
 - hypre, PETSc, SuperLU, Trilinos, etc.



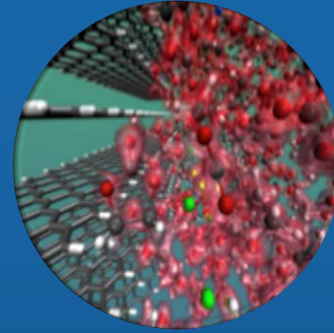
Research on algebraic systems provides key solution technologies to applications



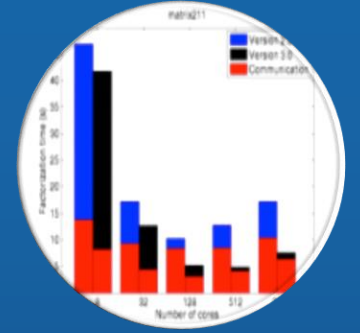
Linear system
solution using direct
and iterative solvers



Nonlinear system
solution using
acceleration
techniques and
globalized Newton
methods



Eigensolvers using
iterative techniques
and optimization



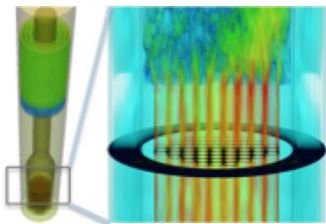
Architecture aware
implementations

Application to fusion, nuclear structure calculation, quantum chemistry,
accelerator modeling, climate, dislocation dynamics etc,

Multiphysics: A primary motivator for exascale

Multiphysics: greater than 1 component governed by its own principle(s) for evolution or equilibrium

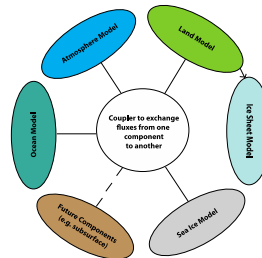
- Also: broad class of coarsely partitioned problems possess similarities



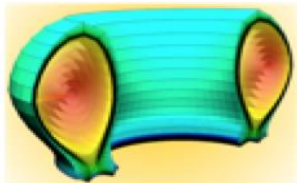
nuclear reactors
A. Siegel, ANL



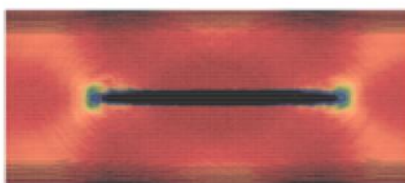
particle accelerators
K. Lee, SLAC



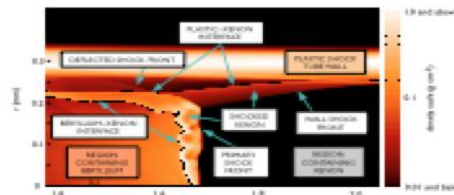
climate
K. Evans, ORNL



fusion
A. Hakim, PPPL



crack propagation
E. Kaxiras, Harvard



radiation hydrodynamics
E. Myra, Univ. of Michigan

IJHPCA, Feb 2013
Vol 27, Issue 1, pp. 4-83



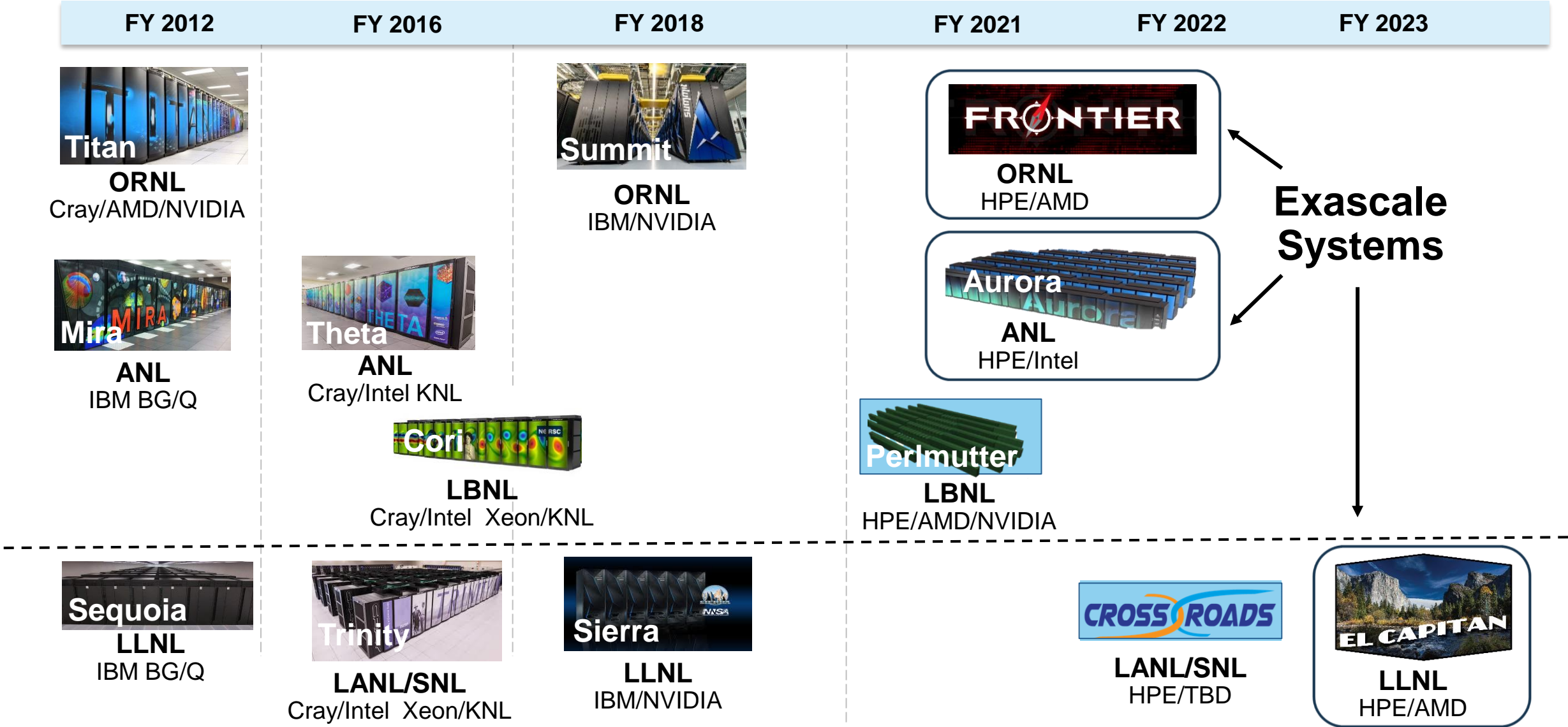
The International Journal of High Performance Computing Applications
27(1) 4-83
© The Author(s) 2012
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342012468181
hpc.sagepub.com
SAGE

Multiphysics simulations: Challenges and opportunities

David E Keyes^{1,2}, Lois C McInnes³, Carol Woodward⁴, William Gropp⁵, Eric Myra⁶, Michael Pernice⁷, John Bell⁸, Jed Brown³, Alain Clo¹, Jeffrey Connors⁴, Emil Constantinescu³, Don Estep⁹, Kate Evans¹⁰, Charbel Farhat¹¹, Ammar Hakim¹², Glenn Hammond¹³, Glen Hansen¹⁴, Judith Hill¹⁰, Tobin Isaac¹⁵, Xiangmin Jiao¹⁶, Kirk Jordan¹⁷, Dinesh Kaushik³, Efthimios Kaxiras¹⁸, Alice Koniges⁸, Kihwan Lee¹⁹, Aaron Lott⁴, Qiming Lu²⁰, John Magerlein¹⁷, Reed Maxwell²¹, Michael McCourt²², Miriam Mehl²³, Roger Pawlowski¹⁴, Amanda P Randles¹⁸, Daniel Reynolds²⁴, Beatrice Riviere²⁵, Ulrich Rüde²⁶, Tim Scheibe¹³, John Shadid¹⁴, Brendan Sheehan⁹, Mark Shephard²⁷, Andrew Siegel³, Barry Smith³, Xianzhu Tang²⁸, Cian Wilson² and Barbara Wohlmuth²³

doi:10.1177/1094342012468181

DOE HPC Roadmap to Exascale Systems



Disruptive changes in HPC architectures

- **Extreme levels of concurrency**

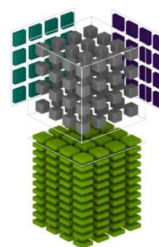
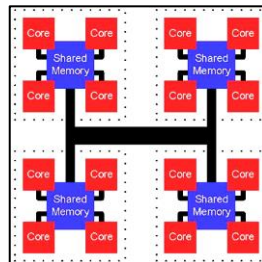
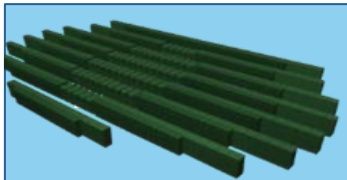
- Increasingly deep memory hierarchies
- Very high node and core counts

- **Additional complexities**

- Hybrid architectures
- GPUs, multithreading, manycore
- Relatively poor memory latency and bandwidth
- Challenges with fault resilience
- Must conserve power – limit data movement
- New (not yet stabilized) programming models
- Etc.

- **Research advances: On-node and inter-node capabilities**

- Reduce communication and synchronization
- Increase concurrency
- Address memory footprint
- Enable large communication/computation overlap
- Use GPUs and multithreading
- Compare task and data parallelism
- Low-level kernels for vector operations that support hybrid programming models
- Mixed precision (leverage compute power available in low-precision tensor cores)
- Etc.



$$D = \begin{matrix} \text{FP16 or FP32} & \text{FP16} & \text{FP16} & \text{FP16 or FP32} \end{matrix} + \begin{matrix} \text{FP16} & \text{FP16} & \text{FP16} & \text{FP16 or FP32} \end{matrix}$$

Software libraries facilitate progress in computational science and engineering

- **Software library:** a high-quality, encapsulated, documented, tested, and multiuse software collection that provides functionality commonly needed by application developers
 - Organized for the purpose of being reused by independent (sub)programs
 - User needs to know only
 - Library interface (not internal details)
 - When and how to use library functionality appropriately
- **Key advantages** of software libraries
 - Contain complexity
 - Leverage library developer expertise
 - Reduce application coding effort
 - Encourage sharing of code, ease distribution of code
- **References:**
 - [https://en.wikipedia.org/wiki/Library_\(computing\)](https://en.wikipedia.org/wiki/Library_(computing))
 - [What are Interoperable Software Libraries? Introducing the xSDK](#)

Broad range of HPC numerical software

Some packages with general-purpose, reusable algorithmic infrastructure in support of high-performance CSE:

- ★ • **AMReX** – <https://github.com/AMReX-codes/amrex>
- **Chombo** - <https://commons.lbl.gov/display/chombo>
- **Clawpack** - <http://www.clawpack.org>
- **Deal.II** - <https://www.dealii.org>
- **FEniCS** - <https://fenicsproject.org>
- ★ • **hypr** - <http://www.llnl.gov/CASC/hypr>
- **libMesh** - <https://libmesh.github.io>
- **MAGMA** - <http://icl.cs.utk.edu/magma>
- ★ • **MFEM** - <http://mfem.org/>
- ★ • **PETSc/TAO** – <http://www.mcs.anl.gov/petsc>
- ★ • **PUMI** - <http://github.com/SCOREC/core>
- ★ • **SUNDIALS** - <http://computation.llnl.gov/casc/sundials>
- ★ • **SuperLU** - <http://crd-legacy.lbl.gov/~xiaoye/SuperLU>
- ★ • **Trilinos** - <https://trilinos.github.io/>
- **Uintah** - <http://www.uintah.utah.edu>
- **waLBerla** - <http://www.walberla.net>

See info about scope, performance, usage, and design, including:

- tutorials
- demos
- examples
- how to contribute

★ Discussed today

... and many, many more ... Explore, use, contribute!

ECP applications need sustainable coordination among math libraries

ECP AD Teams

Combustion-Pele, EXAALT, ExaAM, ExaFEL, ExaSGD, ExaSky, ExaStar, ExaWind, GAMESS, MFIX-Exa, NWChemEx, Subsurface, WarpX, WDMApp, WarpX, ExaAM, ATDM (LANL, LLNL, SNL) apps, AMReX, CEED, CODAR, CoPA, ExaLearn

Examples:

- **ExaAM:** DTK, hypre, PETSc, Sundials, Tasmanian, Trilinos, FFT, etc.
- **ExaWind:** hypre, KokkosKernels, SuperLU, Trilinos, FFT, etc.
- **WDMApp:** PETSc, hypre, SuperLU, STRUMPACK, FFT, etc.
- **CEED:** MFEM, MAGMA, hypre, PETSc, SuperLU, Sundials, etc.
- And many more ...

ECP Math Libraries



Software libraries are not enough

Apps need to use software packages **in combination**

**“The way you get
programmer productivity
is by eliminating lines of
code you have to write.”**

– Steve Jobs, Apple World Wide
Developers Conference, Closing Keynote, 1997

- **Need consistency** of compiler (+version, options), 3rd-party packages, etc.
- **Namespace and version conflicts** make simultaneous build/link of packages difficult
- **Multilayer interoperability** requires careful design and sustainable coordination

Need software ecosystem perspective

Ecosystem: A group of independent but interrelated elements comprising a unified whole

Ecosystems are challenging!

“We often think that when we have completed our study of one we know all about two, because ‘two’ is ‘one and one.’ We forget that we still have to make a study of ‘and.’ ”



– Sir Arthur Stanley Eddington (1892–1944), British astrophysicist

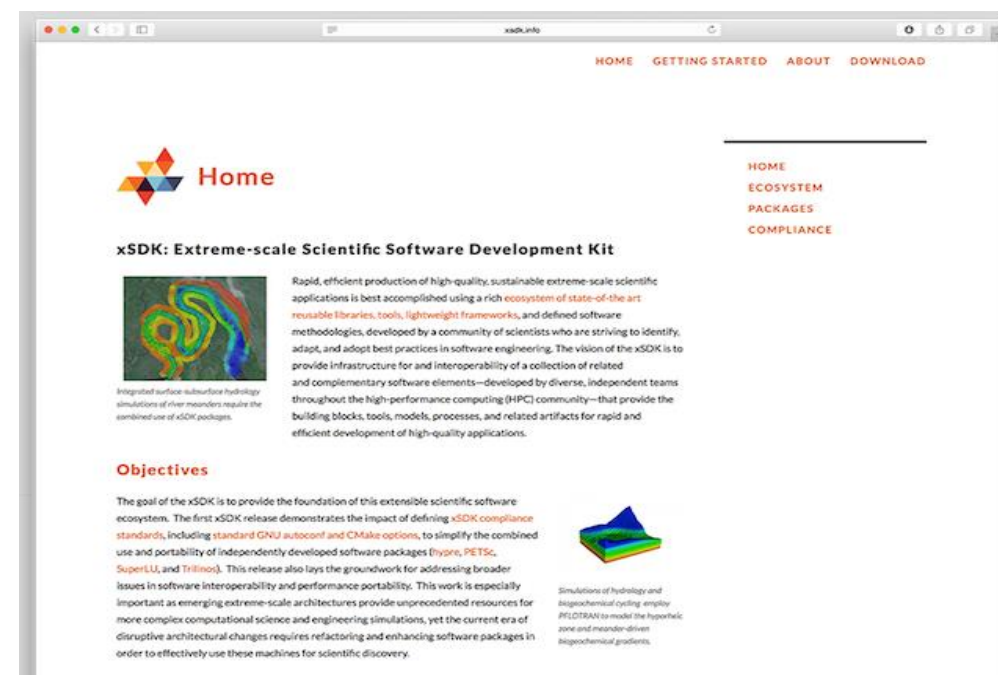


Building the foundation of a highly effective extreme-scale scientific software ecosystem

Focus: Increasing the functionality, quality, and interoperability of important scientific libraries, domain components, and development tools

Impact:

- Improved code quality, usability, access, sustainability
- Inform potential users that an xSDK member package can be easily used with other xSDK packages
- Foundation for work on performance portability, deeper levels of package interoperability

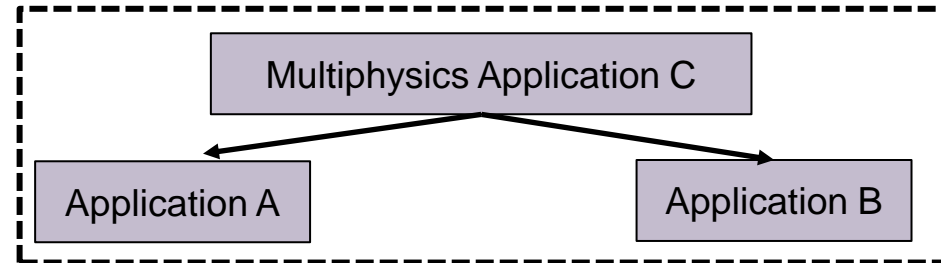


website: xSDK.info



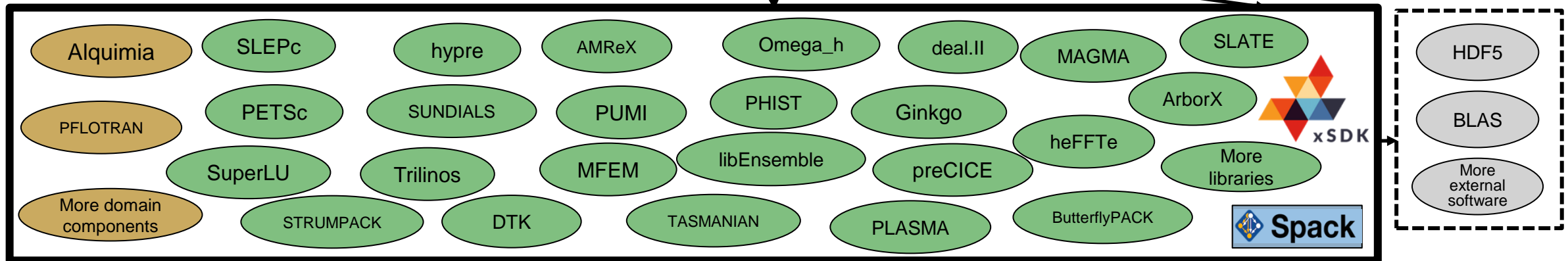
<https://xsdk.info>

Each xSDK member package uses or can be used with one or more xSDK packages, and the connecting interface is regularly tested for regressions.



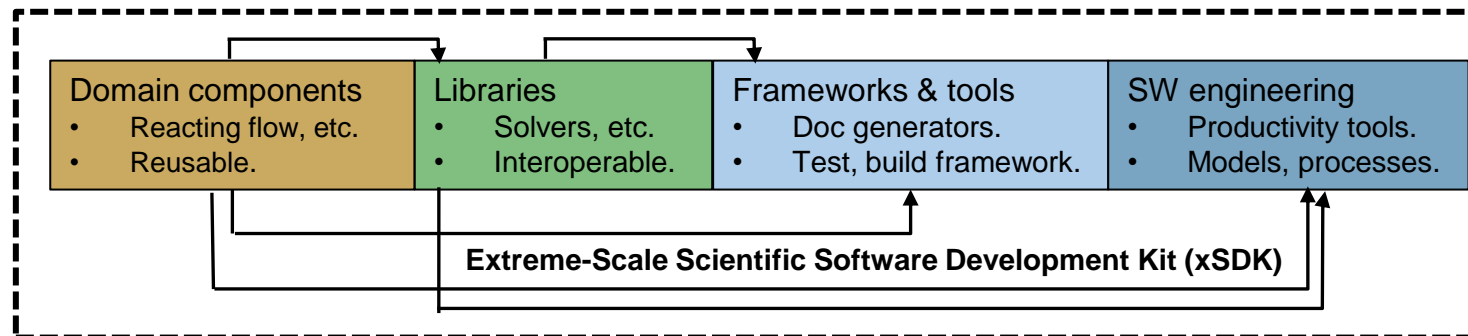
xSDK functionality, Nov 2021

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X



November 2021

- 24 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer



Impact: Improved code quality, usability, access, sustainability

Foundation for work on performance portability, deeper levels of package interoperability

xSDK collaborators



xSDK Release 0.7.0, Nov 2021

- **xSDK release lead:** Satish Balay, SNL
- **xSDK planning**
 - Ulrike Meier Yang (LLNL)
- **Leads for xSDK testing**
 - Satish Balay, ANL: ALCF testing
 - Piotr Luszczek, UTK: OLCF testing
 - Cody Balos, LLNL: general testing
 - Veselin Dobrev, LLNL: general testing
 - Keita Teranishi, SNL: general testing
- **Spack liaison:** Todd Gamblin, LLNL

and many more ...

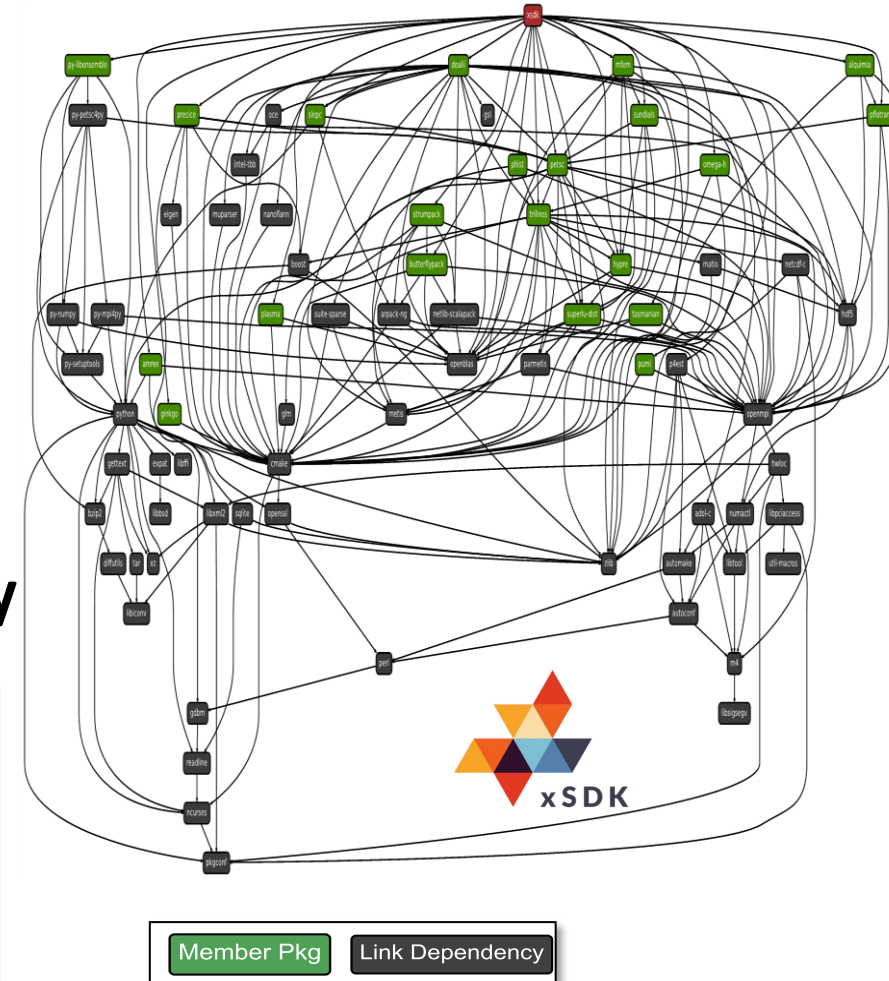
Package compatibility with xSDK community policies:

- **ArborX:** Daniel Arndt (ORNL)
- **AMReX:** Ann Almgren, Michele Rosso (LBNL)
- **DTK:** Stuart Slattery, Bruno Turcksin (ORNL)
- **deal.II:** Wolfgang Bangerth (Colorado State University)
- **Ginkgo:** Hartwig Anzt (Karlsruhe Institute of Technology)
- **hypre:** Ulrike Meier Yang, Sarah Osborn, Rob Falgout (LLNL)
- **libEnsemble:** Stefan Wild, Steve Hudson (ANL)
- **MAGMA, PLASMA, heFFTe, SLATE:** Piotr Luszczek, Stan Tomov, Mark Gates (UTK)
- **MFEM:** Veselin Dobrev, Tzanio Kolev (LLNL)
- **Omega_h:** Dan Ibanez (SNL)
- **PETSc/TAO:** Satish Balay, Richard Mills (ANL)
- **preCICE:** Frederic Simonis (Technical University Munich)
- **PUMI:** Cameron Smith (RPI)
- **SUNDIALS:** Cody Balos, David Gardner, Carol Woodward (LLNL)
- **SuperLU, STRUMPACK, ButterflyPACK:** Sherry Li, Pieter Ghysels, Yang Liu (LBNL)
- **TASMANIAN:** Miroslav Stoyanov, Damien Lebrun Grandie (ORNL)
- **Trilinos:** Keita Teranishi, Jim Willenbring (SNL)
- **PHIST:** Jonas Thies (DLR, German Aerospace Center)
- **SLEPc:** José Roman (Universitat Politècnica de València)
- **Alquimia:** Sergi Mollins (LBNL)
- **PFLOTRAN:** Glenn Hammond (SNL)



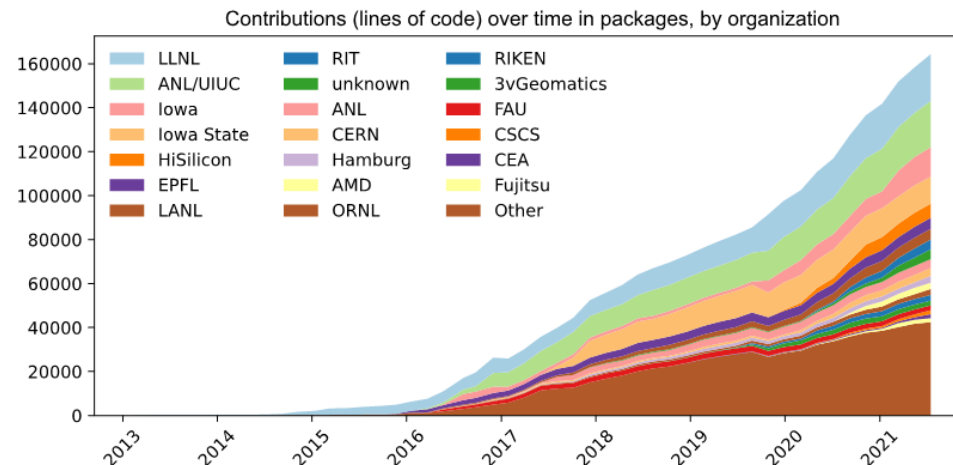
The xSDK is using Spack to deploy its software

- The xSDK packages depend on a number of open-source libraries
- Spack is a flexible package manager for HPC
- Spack allows the xSDK to be deployed with a single command
 - User can optionally choose compilers, build options, etc.
 - Will soon support combinatorial test dashboards for xSDK packages

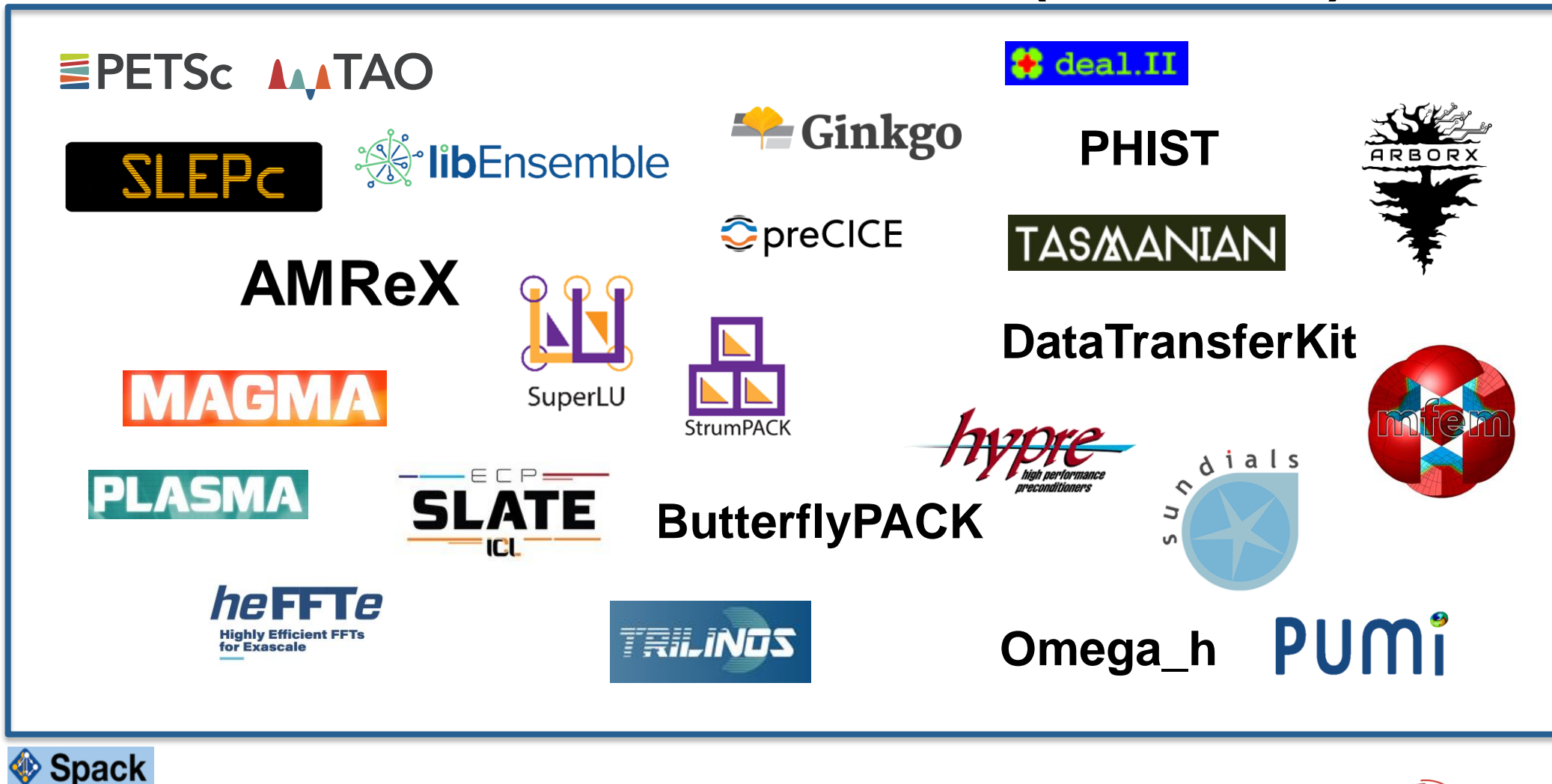


Spack has grown into a thriving open-source community

- Over 840 contributors
- Over 5,700 software packages
- Used world-wide
- Key component of ECP strategy for software deployment



24 diverse individually developed math libraries in xSDK version 0.7.0 (Nov 2021)





xSDK: <https://xsdk.info>

Building the foundation of an extreme-scale scientific software ecosystem

xSDK community policies: Help address challenges in interoperability and sustainability of software developed by diverse groups at different institutions

<https://github.com/xsdk-project/xsdk-community-policies>

xSDK compatible package: must satisfy the mandatory xSDK policies (M1, ..., M16)

Topics include configuring, installing, testing, MPI usage, portability, contact and version information, open-source licensing, namespacing, and repository access

Also specify **recommended policies**, which currently are encouraged but not required (R1, ..., R8)

Topics include public repository access, error handling, freeing system resources, and library dependencies, [documentation quality](#)

xSDK member package:

- (1) Must be an xSDK-compatible package, *and*
- (2) it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

xSDK policies 0.6.0: Oct 2020

- Facilitate combined use of independently developed packages

Impact:

- Improved code quality, usability, access, sustainability
- Foundation for work on deeper levels of interoperability and performance portability

We encourage feedback and contributions!

xSDK community policies

<https://github.com/xsdk-project/xsdk-community-policies>

<https://xsdk.info/policies>



Version 0.6.0,
October 2020

Mandatory xSDK policies: must be satisfied

- M1.** Support portable installation through Spack (includes xSDK Spack variant guidelines)
- M2.** Provide a comprehensive test suite.
- M3.** Employ user-provided MPI communicator.
- M4.** Give best effort at portability to key architectures.
- M5.** Provide a documented, reliable way to contact the development team.
- M6.** Respect system resources and settings made by other previously called packages.
- M7.** Come with an open-source license.
- M8.** Provide a runtime API to return the current version number of the software.
- M9.** Use a limited and well-defined symbol, macro, library, and include file name space.
- M10.** Provide an accessible repository (not necessarily publicly available).
- M11.** Have no hardwired print or IO statements.
- M12.** Allow installing, building, and linking against an outside copy of external software.
- M13.** Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
- M14.** Be buildable using 64-bit pointers. 32 bit is optional.
- M15.** All xSDK compatibility changes should be sustainable.
- M16.** Have a debug build option.

Recommended xSDK policies: currently encouraged, but not required

- R1.** Have a public repository.
- R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3.** Adopt and document consistent system for error conditions/exceptions.
- R4.** Free all system resources it has acquired as soon as they are no longer needed.
- R5.** Provide a mechanism to export ordered list of library dependencies.
- R6.** Provide versions of dependencies.
- R7.** Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.
- R8.** Provide sufficient documentation to support use and further development.

xSDK member package: Must be an xSDK-compatible package, *and* it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

**We welcome feedback.
What policies make
sense for your software?**



xSDK community policies

<https://github.com/xsdk-project/xsdk-community-policies>

<https://xsdk.info/policies>



Version 0.6.0,
October 2020

Mandatory xSDK policies: must be satisfied

- M1.** Support portable installation through Spack (includes xSDK Spack variant guidelines)
- M2.** **Provide a comprehensive test suite.**
- M3.** Employ user-provided MPI communicator.
- M4.** Give best effort at portability to key architectures.
- M5.** Provide a documented, reliable way to contact the development team.
- M6.** Respect system resources and settings made by other previously called packages.
- M7.** Come with an open-source license.
- M8.** Provide a runtime API to return the current version number of the software.
- M9.** Use a limited and well-defined symbol, macro, library, and include file name space.
- M10.** Provide an accessible repository (not necessarily publicly available).
- M11.** Have no hardwired print or IO statements.
- M12.** Allow installing, building, and linking against an outside copy of external software.
- M13.** Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
- M14.** Be buildable using 64-bit pointers. 32 bit is optional.
- M15.** All xSDK compatibility changes should be sustainable.
- M16.** Have a debug build option.

Recommended xSDK policies: currently encouraged, but not required

- R1.** Have a public repository.
- R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3.** Adopt and document consistent system for error conditions/exceptions.
- R4.** Free all system resources it has acquired as soon as they are no longer needed.
- R5.** Provide a mechanism to export ordered list of library dependencies.
- R6.** Provide versions of dependencies.
- R7.** Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.
- R8.** Provide sufficient documentation to support use and further development.

xSDK member package: Must be an xSDK-compatible package, *and* it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

**We welcome feedback.
What policies make
sense for your software?**



xSDK community policies

<https://github.com/xsdk-project/xsdk-community-policies>

<https://xsdk.info/policies>



Version 0.6.0,
October 2020

Mandatory xSDK policies: must be satisfied

- M1.** Support portable installation through Spack (includes xSDK Spack variant guidelines)
- M2.** Provide a comprehensive test suite.
- M3.** Employ user-provided MPI communicator.
- M4.** **Give best effort at portability to key architectures.**
- M5.** Provide a documented, reliable way to contact the development team.
- M6.** Respect system resources and settings made by other previously called packages.
- M7.** Come with an open-source license.
- M8.** Provide a runtime API to return the current version number of the software.
- M9.** Use a limited and well-defined symbol, macro, library, and include file name space.
- M10.** Provide an accessible repository (not necessarily publicly available).
- M11.** Have no hardwired print or IO statements.
- M12.** Allow installing, building, and linking against an outside copy of external software.
- M13.** Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
- M14.** Be buildable using 64-bit pointers. 32 bit is optional.
- M15.** All xSDK compatibility changes should be sustainable.
- M16.** Have a debug build option.

Recommended xSDK policies: currently encouraged, but not required

- R1.** Have a public repository.
- R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3.** Adopt and document consistent system for error conditions/exceptions.
- R4.** Free all system resources it has acquired as soon as they are no longer needed.
- R5.** Provide a mechanism to export ordered list of library dependencies.
- R6.** Provide versions of dependencies.
- R7.** Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.
- R8.** Provide sufficient documentation to support use and further development.

xSDK member package: Must be an xSDK-compatible package, *and* it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

**We welcome feedback.
What policies make
sense for your software?**



What is performance portability?

- Discussion at 2016 DOE Center of Excellence meeting, Phoenix, AZ, USA (<https://performanceportability.org/perfport/definition/>)
 - Attendees included scientists, engineers for DOE's Office of Science and National Nuclear Security Agency, as well as vendors (Intel, NVIDIA, IBM, etc)
- "For the purposes of this meeting, it is the ability to run an application with acceptable performance across KNL and GPU-based systems with a single version of source code." (Rob Neely)
- "An application is performance portable if it achieves a consistent level of performance (e.g., defined by execution time or other figure of merit (not percentage of peak flops across platforms)) relative to the best-known implementation on each platform." (John Pennycook, Intel)
- "Hard portability = no code changes and no tuning. Software portability = simple code mods with no algorithmic changes. Non-portable = algorithmic changes" (Adrian Pope, Vitali Morozov)
- (Performance portability means) the same source code will run productively on a variety of different architectures" (Larkin)
- "Code is performance portable when the application team says its performance portable!" (Richards)

What is performance portability? (continued)

- Conclusion: There is currently no universally accepted definition of performance portability

An application is **performance portable** if it achieves a **consistent ratio** of the actual time to solution to either the best-known or the theoretical best time to solution **on each platform with minimal platform specific code required**.

BSSw, Anshu Dubey:

An application has portable performance if in addition to running on diverse platforms it exhibits **similar accuracy, stability, and reliability across these platforms for a given configuration**.

Moreover, the time to solution should reflect **efficient utilization of available computational resources on each platform**.

Portability Strategies of xSDK Libraries

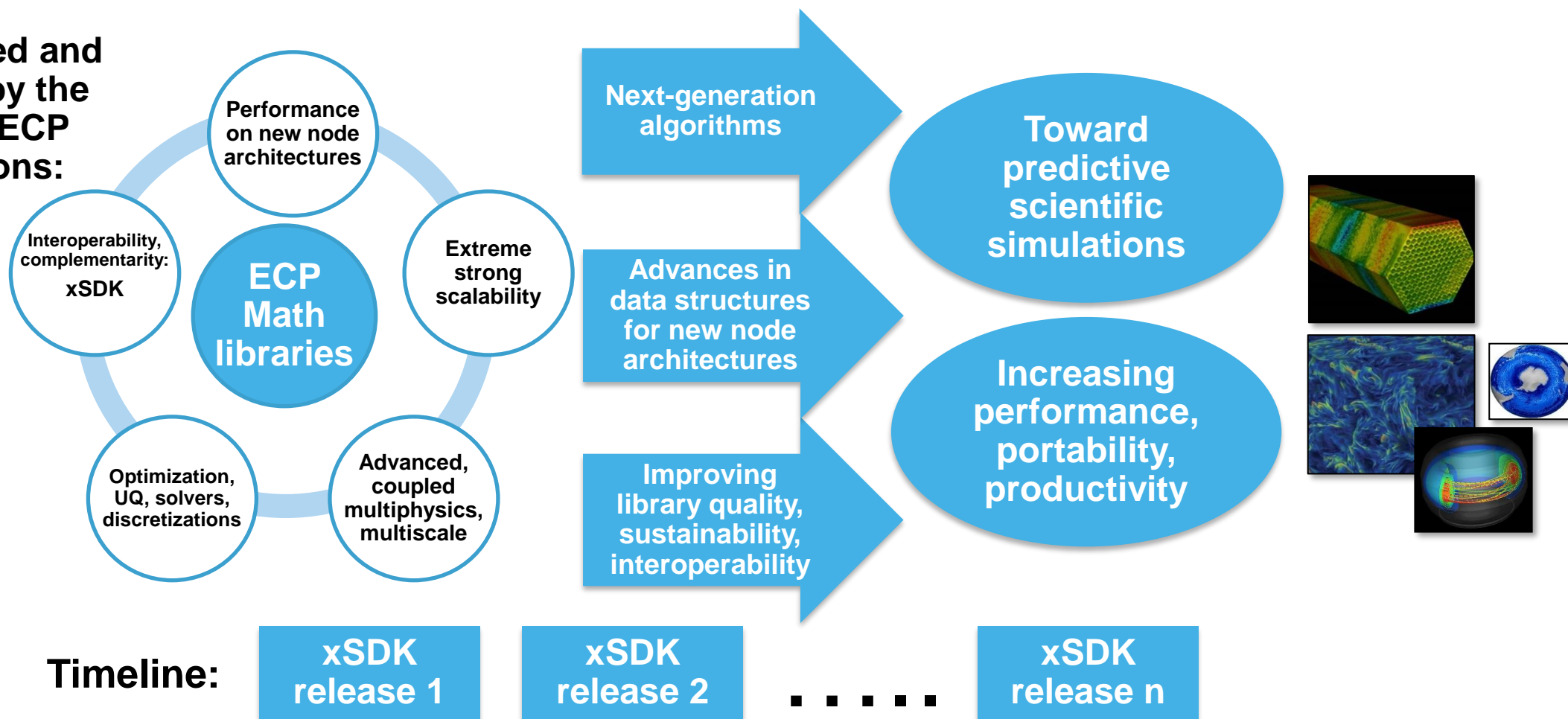
- Use of portable programming models that provide abstractions
- Use of abstraction to limit code that interacts with devices
- Use of fast kernel libraries designed for individual architectures
- Write own CUDA kernels, and use vendor provided tools to port kernels
- Develop new algorithms more suitable for GPUs
(most challenging, but possibly best results!)



cuBLAS, cuSPARSE
rocBLAS, rocSPARSE
MKL

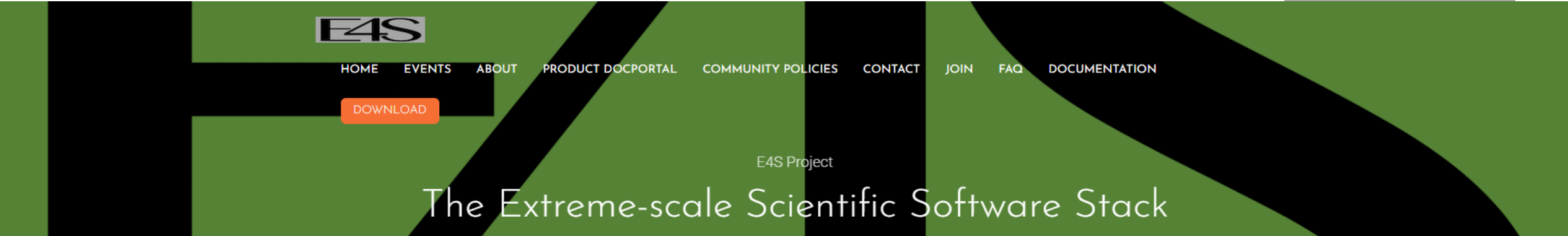
xSDK: Primary delivery mechanism for ECP math libraries' continual advancements toward predictive science

As motivated and validated by the needs of ECP applications:



Extreme-scale Scientific Software Stack (E4S)

<https://e4s.io>



- E4S is a community effort to provide open-source software packages for developing, deploying, and running scientific applications on HPC platforms.
- E4S provides containers and turn-key, from-source builds of 100+ popular HPC software packages:
- E4S containers support Docker, Singularity, Shifter and CharlieCloud container runtimes.
- E4S Spack build cache has over 80,000 binaries.
- Platforms: x86_64, ppc64le, and aarch64. GPUs runtimes: NVIDIA (CUDA) and AMD (ROCm).
- E4S DocPortal provide a single online location for *accurate* product descriptions for software products.
- E4S helps applications reduce the burden to install dependencies:

E4S Build Cache for Spack 0.18.0

To add this mirror to your Spack:

```
$> spack mirror add e4s https://cache.e4s.io
$> spack buildcache keys -lt
```

88,401 total packages
Last updated 2022-05-30 16:42 PDT

☒ All Arch
 ☐ PPC64LE
 ☐ X86_64

☒ All OS
 ☐ CentOS 7
 ☐ CentOS 8
 ☐ RHEL 7
 ☐ RHEL 8
 ☐ Ubuntu 18.04
 ☐ Ubuntu 20.04

adiak@0.1.1 adiak@0.2.1 adios2@2.5.0 adios2@2.6.0 adios2@2.7.0 adios2@2.7.1 adios2@2.8.0 adios@1.13.1 adlbx@0.9.2 adlbx@1.0.0 adolc@2.7.2 alquimia@1.0.9 alsa-lib@1.2.3.2 amg@1.2 amr@0.1.0
 amr-wind/ascent amr-wind/main amrex@20.07 amrex@20.09 amrex@20.10 amrex@20.11 amrex@20.12 amrex@21.01 amrex@21.02 amrex@21.03 amrex@21.04 amrex@21.05 amrex@21.06 amrex@21.07
 amrex@21.08 amrex@21.09 amrex@21.10 amrex@21.11 amrex@21.12 amrex@22.01 amrex@22.02 amrex@22.03 amrex@22.04 amrex@22.05 amr@1.10.0 amr@1.10.7 amr@1.10.7 amr@1.10.7 amr@1.10.7 amr@1.10.7 amr@1.10.7
 arborx@1.1 arborx@1.2 arborx@2.0.0 arborx@2.0.1 arborx@2.0.2 arborx@2.0.3 arborx@2.0.4 arborx@2.0.5 arborx@2.0.6 arborx@2.0.7 arborx@2.0.8 arborx@2.0.9 arborx@2.1.0 arborx@2.1.1
 ascent@panticon-ver asio@1.16.1 asio@1.18.2 asio@1.20.0 asio@1.21.0 asio@1.21.1 asio@1.21.2 asio@1.21.3 asio@1.21.4 asio@1.21.5 asio@1.21.6 asio@1.21.7 asio@1.21.8 asio@1.21.9 asio@1.22.0
 autoconf-archive@2019.01.06 autoconf-archive@2022.02.11 autoconf@2.69 autoconf@2.70 automake@1.15.1 automake@1.16.1 automake@1.16.2 automake@1.16.3 automake@1.16.4 automake@1.16.5 axi@0.1.1 axi@0.1.2 axi@0.1.3 axi@0.1.4 axi@0.1.5
 axom@0.3.3 axom@0.4.0 axom@0.5.0 axom@0.6.1 bacio@2.4.1 bash@5.0 bats@0.4.0 bdfpc@1.0.5 berkeley-db@18.1.40 berkeley-db@6.2.32 binutils@2.31.1 binutils@2.32 binutils@2.33 binutils@2.34
 binutils@2.36.1 binutils@2.37 binutils@2.38 bison@3.7.4 bison@3.7.5 bison@3.7.6 bison@3.7.7 bison@3.7.8 bison@3.7.9 bison@3.8.0 bison@3.8.1 bison@3.8.2 bison@3.8.3 bison@3.8.4 bison@3.8.5 bison@3.8.6
 bld@develop bld@develop bld@main bld@1.0 bld@1.0rc1 bld@1.0rc2 bld@1.0rc3 bld@1.0rc4 bld@1.0rc5 bld@1.0rc6 bld@1.0rc7 bld@1.0rc8 bld@1.0rc9 bld@1.1 bld@1.1rc1 bld@1.1rc2 bld@1.1rc3
 boost@1.79.0 bricks@0.1 bwa@1.1.5 butterfly@1.1.0 butterfly@1.1.1 butterfly@1.1.2 butterfly@1.1.3 butterfly@1.1.4 butterfly@1.1.5 butterfly@1.1.6 butterfly@1.1.7 butterfly@1.1.8 butterfly@1.1.9
 c-blosc@1.17.0 c-blosc@1.21.0 c-blosc@1.21.1 cabana@0.3.0 cabana@0.4.0 caio@1.1.6.0 caliper@2.0.1 caliper@2.2.0 caliper@2.3.0 caliper@2.4.0 caliper@2.5.0 caliper@2.6.0 caliper@2.7.0 camp@0.1.0 camp@0.2.2
 camtims@master catalyst@5.6.0 cdo@1.9.10 cereal@1.3.2 cern@4.2.0 chai@2.3.0 chai@2.4.0 charliecloud@0.22 charliecloud@0.23 charliecloud@0.24 charliecloud@0.25 charliecloud@0.26 cinch@develop cinch@master
 chl1@1.9.1 cmake@3.13.4 cmake@3.14.5 cmake@3.14.7 cmake@3.15.4 cmake@3.16.2 cmake@3.16.3 cmake@3.17.3 cmake@3.17.4 cmake@3.18.0 cmake@3.18.1 cmake@3.18.2 cmake@3.18.4 cmake@3.19.0
 ecpe4s/ubuntu18.04-e4s docker GitHub

E4S Community Policies

Version 1

The *Extreme-scale Scientific Software Stack* (E4S) and its related Software Development Kits (SDKs) are light-weight organizational elements intended to enhance collaboration and coordination across independently-developed products to provide an enhanced high-performance software stack.

E4S Community Policies are membership criteria for a product to become an E4S member package. This version of the policies was arrived at by the ECP SDK team, including representation from Programming Models and Runtimes, Development Tools, Math Libraries, Data and Vis, and Software Ecosystem and Delivery, and after considering multiple rounds of feedback from across ECP ST. The starting point for the majority of the policies was taken from the existing *sSDK Community Policies*.

E4S Membership Criteria

The policies below are criteria for E4S membership. To qualify for E4S membership, a package must demonstrate compatibility with each of these policies. Under special circumstances, a package may be granted an exception to a policy.

P1 *Spack-based Build and Installation* Each E4S member package supports a scriptable *Spack* build and production-quality installation in a way that is compatible with other E4S member packages in the same environment. When E4S build, test, or

E4S build pipeline

- Cori, NERSC

- ## E4S 2022.05 release
- 101 ECP ST products
 - CUDA
 - ROCm
 - Tensorflow
 - PyTorch



<https://e4s.io>



<https://spack.io>

E4S Summary



What E4S is not

A closed system taking contributions only from DOE software development teams.

A monolithic, take-it-or-leave-it software behemoth.

A commercial product.

A simple packaging of existing software.

What E4S is

Extensible, open architecture software ecosystem accepting contributions from US and international teams.
Framework for collaborative open-source product integration.

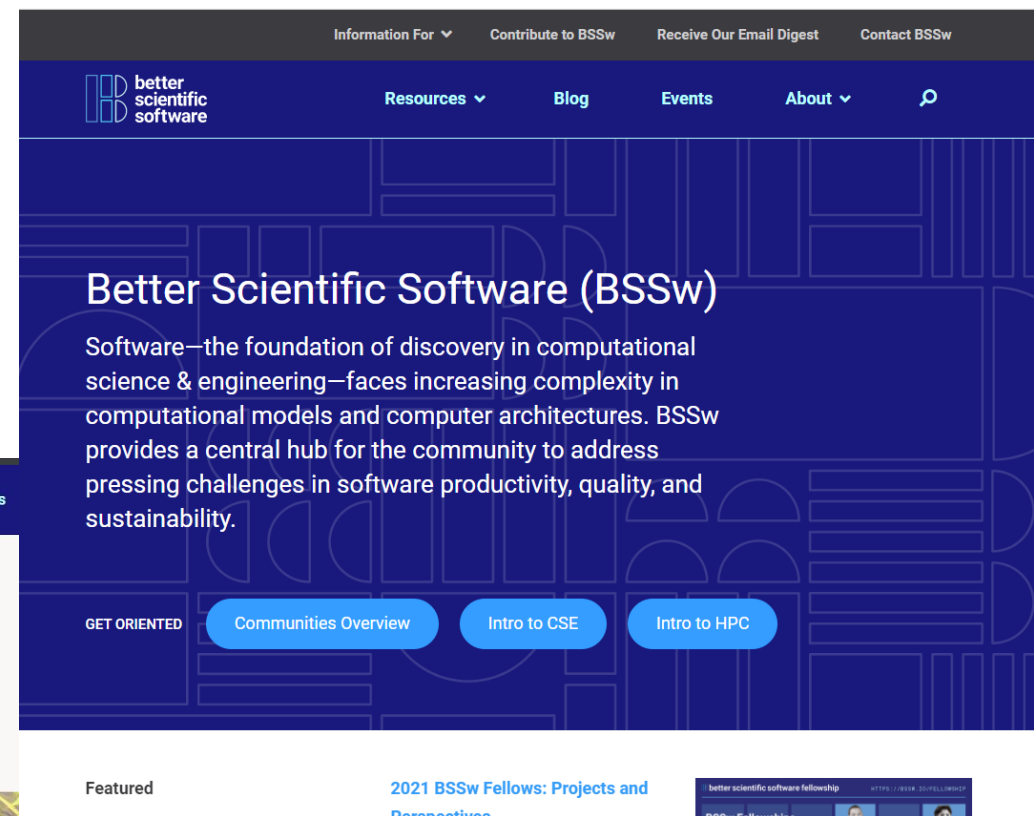
A full collection of compatible software capabilities **and**
A manifest of a la carte selectable software capabilities.

Vehicle for delivering high-quality reusable software products in collaboration with others.

The conduit for future leading edge HPC software targeting scalable next-generation computing platforms.
A hierarchical software framework to enhance (via SDKs) software interoperability and quality expectations.

Further reading


- [Building community through software policies](#), Piotr Luszczek and Ulrike Yang
- [SuperLU: How advances in software practices are increasing sustainability and collaboration](#), Sherry Li
- [Porting the Ginkgo package to AMD's HIP ecosystem](#), Hartwig Anzt
- [Scientific software packages and their communities](#), Rene Gassmoeller
- [Leading a scientific software project: It's all personal](#), Wolfgang Bangerth
- [The art of writing scientific software in an academic environment](#), Hartwig Anzt
- [Working Remotely: The Exascale Computing Project \(ECP\) panel series](#), Elaine Raybourn et al.
- [Better Scientific Software: 2021 highlights](#), Rinku Gupta
- And many more ...



See also Track 7:
Software Productivity &
Sustainability (Aug 11)

HandsOn Lessons

- Structured meshing & discretization
- Unstructured meshing & discretization
- Krylov solvers & preconditioners
- Sparse direct solvers
- Nonlinear solvers
- Time integration
- Numerical optimization



Packages for
Extreme-Scale Science
So my code will see the future

ATPESC 2022 Hands On Lessons

<u>Meshing and Discretization with AMReX</u>	A Block Structured Adaptive Mesh Refinement Framework
<u>Unstructured Meshing & Discretization with MFEM</u>	Finite Elements and Convergence
<u>Krylov Solvers and Algebraic Multigrid with hypre</u>	Demonstrate utility of multigrid
<u>Iterative Solvers & Algebraic Multigrid (with Trilinos, Belos & MueLu)</u>	Introduction to Krylov Solvers and Preconditioning, with emphasis on Multigrid
<u>Sparse, Direct Solvers with SuperLU</u>	Role and Use of Direct Solvers in Ill-Conditioned Problems
<u>Rank Structured Solvers with STRUMPACK</u>	Using STRUMPACK for dense and sparse linear systems

Github pages site:

<https://xsdk-project.github.io/MathPackagesTraining2022/lessons/>

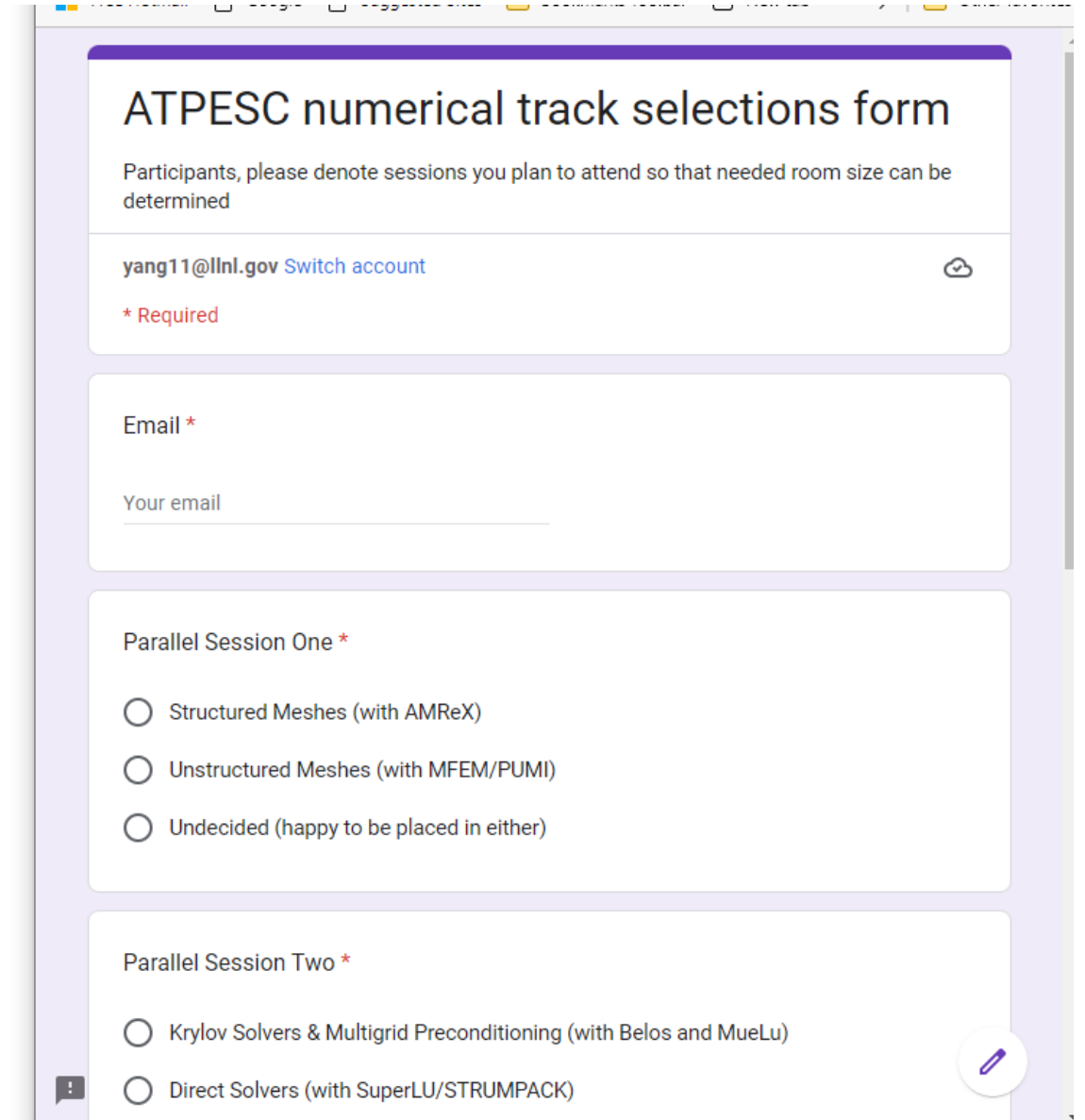
If you haven't yet done so, please choose which session you plan to attend!

Time	Room?	Room?
8:30 – 9:30	Introduction to Numerical Software – Ulrike Yang	
9:30 – 10:45	Structured Discretization (AMReX) – Ann Almgren, Erik Palmer	Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith
10:45 – 11:15	Break, Subject Matter Expert (SME) Selections, Panel Questions	
11:15 – 12:30	Iterative Solvers & Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper	Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels
12:30 – 1:30	Lunch, SME Selections, Panel Questions	
1:30 – 2:45	Nonlinear Solvers (PETSc) – Richard Mills	Time Integration (SUNDIALS) – Dan Reynolds, David Gardener
2:45 – 3:15	Break, SME Selections, Panel Questions Due	
3:15 – 4:30	Optimization (TAO) – Todd Munson, Richard Mills	Iterative Solvers & Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang
4:30 – 5:30	Wrap-up / Panel: Extreme-Scale Numerical Algorithms and Software	
5:30 – 6:30	Unstructured Time: SME Selections Due , Informal Discussion, Continue Hands-on	
6:30 – 7:30	Dinner	
7:30 – 9:30	Optional Activity: SME Speed-dating	

Choose which lecture you want to attend!

- Access

<https://forms.gle/axrawtNsTgbjDTJP8>



The screenshot shows a Google Forms interface titled "ATPESC numerical track selections form". Below the title is a subtitle: "Participants, please denote sessions you plan to attend so that needed room size can be determined". The form is associated with the user "yang11@llnl.gov" and includes a "Switch account" link. A red asterisk indicates required fields. The first required field is "Email *", with a placeholder text "Your email". Below this is a section for "Parallel Session One *" with three radio button options: "Structured Meshes (with AMReX)", "Unstructured Meshes (with MFEM/PUMI)", and "Undecided (happy to be placed in either)". The next section is "Parallel Session Two *" with two radio button options: "Krylov Solvers & Multigrid Preconditioning (with Belos and MueLu)" and "Direct Solvers (with SuperLU/STRUMPACK)". A small chat icon is visible in the bottom left corner, and a pencil icon is in the bottom right corner.

ATPESC numerical track selections form

Participants, please denote sessions you plan to attend so that needed room size can be determined

yang11@llnl.gov Switch account

* Required

Email *

Your email

Parallel Session One *

☐ Structured Meshes (with AMReX)

☐ Unstructured Meshes (with MFEM/PUMI)

☐ Undecided (happy to be placed in either)

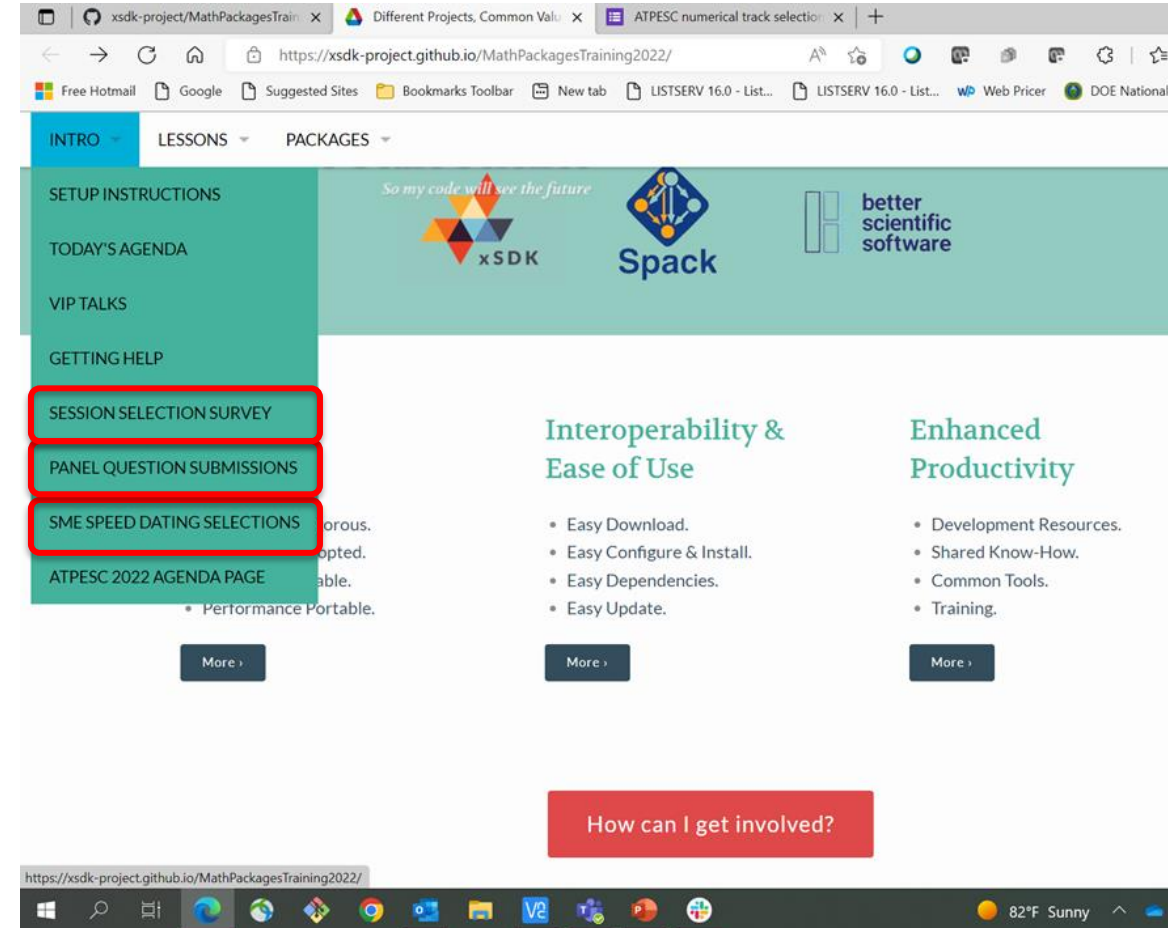
Parallel Session Two *

☐ Krylov Solvers & Multigrid Preconditioning (with Belos and MueLu)

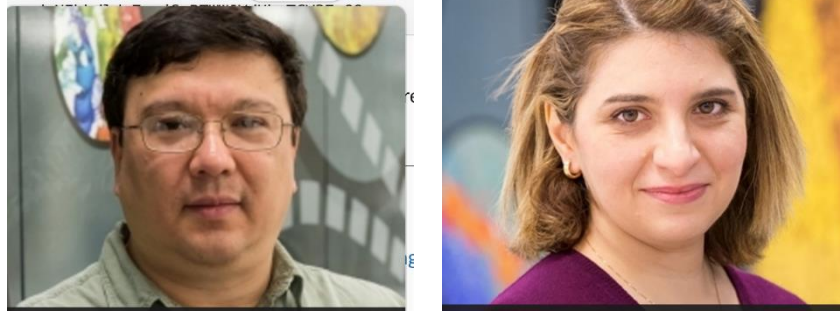
☐ Direct Solvers (with SuperLU/STRUMPACK)

Next steps

- If you haven't done so
 - Choose which session you will attend!
- During breaks and lunch
 - Submit questions for panelists (optional)
 - Sign up for discussions with numerical software developers (optional)
 - [Your email address](#)
 - Complete by 3:30 pm CDT



Thank you to all ATPESC staff



Special thanks to Ray Loy and Yasaman Ghadar

**For their outstanding work in running
the 2-week ATPESC program**

**And thank you to all ATPESC attendees for
engaging questions and discussions!**



This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-PRES-838470

This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), Scientific Discovery through Advanced Computing (SciDAC) program, and by the Exascale Computing Project, a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.