Introduction to Numerical Software

Presented to
ATPESC 2022 Participants

Ulrike Meier Yang
Lawrence Livermore National Laboratory

Date 08/09/2022
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
  – 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/
Setup instructions
• Today’s agenda
• VIP talks
• Getting help
• Session Selection Survey
• Panel question submission
• SME speed dating selections
## Agenda


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

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

https://xsdk-project.github.io/MathPackagesTraining2022/
## Agenda

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

- **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

- **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

- **Designed for integration into existing codes**
  - xSDK package; installs with Slack
  - Permissive license enables integration with open and closed-source codes

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

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

Source Code: github.com/SCOREC/core
User Guide: scorec.rpi.edu/pumi/PUMI.pdf
Parallel management of unstructured meshes with particles.
Framework for GPU accelerated particle-in-cell applications using unstructured meshes.

### PUMIPic
Parallel Unstructured Mesh Infrastructure for Particle-in-Cell

#### 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/SCOREC/pumi-pic](https://github.com/SCOREC/pumi-pic)
- **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](https://extremecomputingtraining.anl.gov/agenda-2022/#Track-5)

<table>
<thead>
<tr>
<th>Time</th>
<th>Room?</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 – 9:30</td>
<td>Introduction to Numerical Software – Ulrike Yang</td>
</tr>
<tr>
<td>9:30 – 10:45</td>
<td>Structured Discretization (AMReX) – Ann Almgren, Erik Palmer</td>
</tr>
<tr>
<td></td>
<td>Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith</td>
</tr>
<tr>
<td>10:45 – 11:15</td>
<td>Break</td>
</tr>
<tr>
<td>11:15 – 12:30</td>
<td>Iterative Solvers &amp; Algebraic Multigrid (Trilinos/Belos/MueLU) – Christian Glusa, Graham Harper</td>
</tr>
<tr>
<td></td>
<td>Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels</td>
</tr>
<tr>
<td>12:30 – 1:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>1:30 – 2:45</td>
<td>Nonlinear Solvers (PETSc) – Richard Mills</td>
</tr>
<tr>
<td></td>
<td>Time Integration (SUNDIALS) – Dan Reynolds, David Gardener</td>
</tr>
<tr>
<td>2:45 – 3:15</td>
<td>Break</td>
</tr>
<tr>
<td>3:15 – 4:30</td>
<td>Optimization (TAO) – Todd Munson, Richard Mills</td>
</tr>
<tr>
<td></td>
<td>Iterative Solvers &amp; Algebraic Multigrid (hypre) – Sarah Osborn, Ulrike Yang</td>
</tr>
<tr>
<td>4:30 – 5:30</td>
<td>Wrap-up / Panel</td>
</tr>
<tr>
<td>6:30 – 7:30</td>
<td>Dinner</td>
</tr>
<tr>
<td>7:30 – 9:30</td>
<td>SME Speed Dating</td>
</tr>
</tbody>
</table>
Ability to solve single or sequence of linear systems
- Simultaneously solved systems w/ multiple-RHS: \( AX = B \)
- Sequentially solved systems w/ multiple-RHS: \( AX_i = B_i, \ i=1,\ldots,t \)
- Sequences of multiple-RHS systems: \( A_iX_i = B_i, \ i=1,\ldots,t \)

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

Iterative Krylov-based solvers. Templated C++ allows for generic scalar, ordinal, and compute node types.
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, ℓ₁ 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

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

[Image of Z machine diagram]

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


- **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

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](https://portal.nersc.gov/project/sparse/superlu)
STRUMPACK
Structured Matrix Package

- **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](https://github.com/pghysels/STRUMPACK)

Hierarchical solvers for dense rank-structured matrices and fast algebraic sparse solver and robust and scalable preconditioners.
<table>
<thead>
<tr>
<th>Time</th>
<th>Room?</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 – 9:30</td>
<td>Introduction to Numerical Software – Ulrike Yang</td>
</tr>
<tr>
<td>9:30 – 10:45</td>
<td>Structured Discretization (AMReX) – Ann Almgren, Erik Palmer</td>
</tr>
<tr>
<td>10:45 – 11:15</td>
<td>Break</td>
</tr>
<tr>
<td>11:15 – 12:30</td>
<td>Iterative Solvers &amp; Algebraic Multigrid (Trilinos/Belos/MueLU) – Christian Glusa, Graham Harper</td>
</tr>
<tr>
<td>12:30 – 1:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>1:30 – 2:45</td>
<td>Nonlinear Solvers (PETSc) – Richard Mills</td>
</tr>
<tr>
<td>2:45 – 3:15</td>
<td>Break</td>
</tr>
<tr>
<td>3:15 – 4:30</td>
<td>Optimization (TAO) – Todd Munson, Richard Mills</td>
</tr>
<tr>
<td>4:30 – 5:30</td>
<td>Wrap-up / Panel</td>
</tr>
<tr>
<td>6:30 – 7:30</td>
<td>Dinner</td>
</tr>
<tr>
<td>7:30 – 9:30</td>
<td>SME Speed Dating</td>
</tr>
</tbody>
</table>
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
PETSc provides the backbone of diverse scientific applications.
clockwise from upper left: hydrology, cardiology, fusion, multiphase steel,
relativistic matter, ice sheet modeling

PETSc is 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

https://www.mcs.anl.gov/petsc
<table>
<thead>
<tr>
<th>Time</th>
<th>Room?</th>
<th>Room?</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 – 9:30</td>
<td>Introduction to Numerical Software – Ulrike Yang</td>
<td></td>
</tr>
<tr>
<td>9:30 – 10:45</td>
<td>Structured Discretization (AMReX) – Ann Almgren, Erik Palmer</td>
<td>Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith</td>
</tr>
<tr>
<td>10:45 – 11:15</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>11:15 – 12:30</td>
<td>Iterative Solvers &amp; Algebraic Multigrid (Trilinos/ Belos/MueLU) – Christian Glusa, Graham Harper</td>
<td>Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels</td>
</tr>
<tr>
<td>12:30 – 1:30</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>1:30 – 2:45</td>
<td>Nonlinear Solvers (PETSc) – Richard Mills</td>
<td>Time Integration (SUNDIALS) – Dan Reynolds, David Gardener</td>
</tr>
<tr>
<td>2:45 – 3:15</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>4:30 – 5:30</td>
<td>Wrap-up / Panel</td>
<td></td>
</tr>
<tr>
<td>6:30 – 7:30</td>
<td>Dinner</td>
<td></td>
</tr>
<tr>
<td>7:30 – 9:30</td>
<td>SME Speed Dating</td>
<td></td>
</tr>
</tbody>
</table>
**hypre**

**Lawrence Livermore National Laboratory**

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](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
<table>
<thead>
<tr>
<th>Time</th>
<th>Room?</th>
<th>Room?</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 – 9:30</td>
<td>Introduction to Numerical Software – Ulrike Yang</td>
<td></td>
</tr>
<tr>
<td>9:30 – 10:45</td>
<td>Structured Discretization (AMReX) – Ann Almgren, Erik Palmer</td>
<td>Unstructured Discretization (MFEM/PUMI) – Aaron Fisher, Mark Shephard, Cameron Smith</td>
</tr>
<tr>
<td>10:45 – 11:15</td>
<td>Break, Subject Matter Expert (SME) Selections, Panel Questions</td>
<td></td>
</tr>
<tr>
<td>11:15 – 12:30</td>
<td>Iterative Solvers &amp; Algebraic Multigrid (Trilinos/Belos/MueLU) – Christian Glusa, Graham Harper</td>
<td>Direct Solvers (SuperLU, STRUMPACK) – Sherry Li, Pieter Ghysels</td>
</tr>
<tr>
<td>12:30 – 1:30</td>
<td>Lunch, SME Selections, Panel Questions</td>
<td></td>
</tr>
<tr>
<td>1:30 – 2:45</td>
<td>Nonlinear Solvers (PETSc) – Richard Mills</td>
<td>Time Integration (SUNDIALS) – Dan Reynolds, David Gardener</td>
</tr>
<tr>
<td>2:45 – 3:15</td>
<td>Break, SME Selections, Panel Questions Due</td>
<td></td>
</tr>
<tr>
<td>4:30 – 5:30</td>
<td>Wrap-up (Ann Almgren) / Panel: Extreme-Scale Numerical Algorithms and Software</td>
<td></td>
</tr>
<tr>
<td>5:30 – 6:30</td>
<td>Unstructured Time: SME Selections Due, Informal Discussion, Continue Hands-on</td>
<td></td>
</tr>
<tr>
<td>6:30 – 7:30</td>
<td>Dinner</td>
<td></td>
</tr>
<tr>
<td>7:30 – 9:30</td>
<td>Optional Activity: SME Speed-dating</td>
<td></td>
</tr>
</tbody>
</table>
Next steps: [https://xsdk-project.github.io/MathPackagesTraining2022/agenda](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
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
Christian Glusa, SNL
Mark Shephard, RPI
Cameron Smith, RPI
Ulrike Yang, LLNL
Erik Palmer, LBL
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 Benchmarker, 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)
FASTMath: Frameworks, Algorithms & Scalable Technologies for Mathematics

Covers 8 technical areas

- Data Analytics
- Structured Mesh Spatial Discretization
- Unstructured Mesh Spatial Discretization
- Synergistic activities link the eight areas together
- Time Integrators
- Uncertainty Quantification
- Numerical Optimization
- Solution of Linear and Nonlinear Systems
- Solution of Eigenvalue Problems

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

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

100’s of person years of experience building math software

50+ researchers from 5 DOE labs and 5 universities

Esmond G. Ng, FASTMath Institute Director (EGNg@lbl.gov)
ECP’s holistic approach uses co-design and integration to achieve exascale computing.

**Application Development**
Science and mission applications

**Software Technology**
Scalable software stack

**Hardware and Integration**
Relationships: facilities with AD/ST, with vendors

Emphasis for this presentation
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

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

**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

**CSE simulation loop**
- physics models
- meshes
- discretization
- algebraic solvers
- refine

**CSE analysis loop**
- CSE simulation loop
- sensitivities/derivatives
- uncertainty quantification
- data analytics
- optimization / design

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

Beyond interpretive simulations … working toward predictive science
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

\[ \nabla^2 T \approx \frac{(T_{i+1} - 2T_i + T_{i-1})}{h^2} = 0 \]

\[ T_0 = 180^\circ \quad T_n = 0^\circ \]

Hot water bath \[ h \] \[ i-1 \quad i \quad i+1 \]

Cold water bath
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 $T_i$

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

- 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.

Application to cosmology, astrophysics, accelerator modeling, fusion, climate, subsurface reacting flows, low mach number combustion, etc.
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.

- IMEX, multi-rate methods
- Spectral deferred correction techniques
- Parallel in time
- Constrained problems

Application to atmospheric dynamics, ice sheets, combustion, power grid, cosmology, subsurface, etc.
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

Multiphysics simulations: Challenges and opportunities

David E Keyes1,2, Lois C McInnes3, Carol Woodward4, William Gropp1, Eric Myra5, Michael Pernice1, John Bell6, Jed Brown7, Alain Clo1, Jeffrey Connors, Emil Constantinescu1, Don Estep8, Kate Evans9, Charbel Farhat10, Ammar Hakim11, Glenn Hammon12, Glen Hansen14, Judith Hill10, Tobin Isaac13, Xiangmin Jiao13, Kirke Jordan13, Dinesh Kaulshik13, Efthimios Kaxiras13, Alice Koniges13, Kilwan Lee13, Aaron Lott13, Qiming Lu10, John Magerlein14, Reed Maxwell11, Michael McCourt12, Miriam Meh13, Roger Pawlowksi14, Amanda P Randles11, Daniel Reynolds14, Beatrice Riviere13, Ulrich Rüde15, Tim Scheibe16, John Shadid15, Brendan Sheehan16, Mark Shephard12, Andrew Siegel1, Barry Smith1, Xianzhu Tang17, Cian Wilson12 and Barbara Wohlmut13

doi:10.1177/1094342012468181
## DOE HPC Roadmap to Exascale Systems

<table>
<thead>
<tr>
<th>FY 2012</th>
<th>FY 2016</th>
<th>FY 2018</th>
<th>FY 2021</th>
<th>FY 2022</th>
<th>FY 2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan (ORNL) Cray/AMD/NVIDIA</td>
<td>Mira (ANL) IBM BG/Q</td>
<td>Theta (ANL) Cray/Intel KNL</td>
<td>Summit (ORNL) IBM/NVIDIA</td>
<td>Frontier (ORNL) HPE/AMD</td>
<td>Perlmutter (LANL) HPE/Intel</td>
</tr>
<tr>
<td>Sequoia (LLNL) IBM BG/Q</td>
<td>Trinity (LANL/SNL) Cray/Intel Xeon/KNL</td>
<td>Sierra (LLNL) IBM/NVIDIA</td>
<td>Aurora (ANL) HPE/Intel</td>
<td>Perlmutter (LANL) HPE/AMD</td>
<td></td>
</tr>
<tr>
<td>Titan (ORNL) Cray/AMD/NVIDIA</td>
<td>Mira (ANL) IBM BG/Q</td>
<td>Theta (ANL) Cray/Intel KNL</td>
<td>Summit (ORNL) IBM/NVIDIA</td>
<td>Frontier (ORNL) HPE/AMD</td>
<td>Perlmutter (LANL) HPE/Intel</td>
</tr>
<tr>
<td>Sequoia (LLNL) IBM BG/Q</td>
<td>Trinity (LANL/SNL) Cray/Intel Xeon/KNL</td>
<td>Sierra (LLNL) IBM/NVIDIA</td>
<td>Titan (ORNL) Cray/AMD/NVIDIA</td>
<td>Frontier (ORNL) HPE/AMD</td>
<td>Perlmutter (LANL) HPE/Intel</td>
</tr>
</tbody>
</table>

**Exascale Systems**

- Summit (ORNL) IBM/NVIDIA
- Frontier (ORNL) HPE/AMD
- Perlmutter (LANL) HPE/Intel
- Aurora (ANL) HPE/Intel
- Perlmutter (LANL) HPE/AMD
- Summit (ORNL) IBM/NVIDIA
- Frontier (ORNL) HPE/AMD
- Perlmutter (LANL) HPE/Intel
- Aurora (ANL) HPE/Intel
- Perlmutter (LANL) HPE/AMD
- Summit (ORNL) IBM/NVIDIA
- Frontier (ORNL) HPE/AMD
- Perlmutter (LANL) HPE/Intel
- Aurora (ANL) HPE/Intel
- Perlmutter (LANL) HPE/Intel
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.
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**:
  • 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](https://github.com/AMReX-codes/amrex)
- **Chombo** - [https://commons.lbl.gov/display/chombo](https://commons.lbl.gov/display/chombo)
- **Clawpack** - [http://www.clawpack.org](http://www.clawpack.org)
- **Deal.II** - [https://www.dealii.org](https://www.dealii.org)
- **FEniCS** - [https://fenicsproject.org](https://fenicsproject.org)
- **hypre** - [http://www.llnl.gov/CASC/hypre](http://www.llnl.gov/CASC/hypre)
- **libMesh** - [https://libmesh.github.io](https://libmesh.github.io)
- **MAGMA** - [http://icl.cs.utk.edu/magma](http://icl.cs.utk.edu/magma)
- **MFEM** - [http://mfem.org/](http://mfem.org/)
- **PETSc/TAO** – [http://www.mcs.anl.gov/petsc](http://www.mcs.anl.gov/petsc)
- **PUMI** - [http://github.com/SCOREC/core](http://github.com/SCOREC/core)
- **SUNDIALS** - [http://computation.llnl.gov/casc/sundials](http://computation.llnl.gov/casc/sundials)
- **SuperLU** - [http://crd-legacy.lbl.gov/~xiaoye/SuperLU](http://crd-legacy.lbl.gov/~xiaoye/SuperLU)
- **Trilinos** - [https://trilinos.github.io/](https://trilinos.github.io/)
- **Uintah** - [http://www.uintah.utah.edu](http://www.uintah.utah.edu)
- **waLBerla** - [http://www.walberla.net](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


ECP Math Libraries

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


• 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
xSDK Version 0.7.0: November 2021

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

xSDK Version 0.7.0: November 2021

https://xsdk.info

Multithreaded Application C

Application A

Application B

Extreme-Scale Scientific Software Development Kit (xSDK)

Domain components
- Reacting flow, etc.
- Reusable.

Libraries
- Solvers, etc.
- Interoperable.

Frameworks & tools
- Doc generators.
- Test, build framework.

SW engineering
- Productivity tools.
- Models, processes.

Impact: Improved code quality, usability, access, sustainability

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

November 2021
- 24 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer

More libraries

More external software

Alquimia
SLEPC
hypre
AMReX
Omega_h
deal II
MAGMA
SLATE
Spack
HDF5
BLAS

PFLOTRAN
PETSc
SUNDIALS
PUMI
MFEM
libEnsemble
PHIST
Ginkgo
heFFTe
ArborX

SuperLU
TRILINOS
MFEM
TASMANIAN
PLASMA
ButterflyPACK

STRUMPACK
DTK

MFEM
SUNDIALS
PUMI

More domain components

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

https://xsdk.info
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

---

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)
- **hype:** 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, Tzario 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)

and many more …
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**

**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](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, namespaces, 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

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?
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 if 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?
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!)
xSDK: Primary delivery mechanism for ECP math libraries’ continual advancements toward predictive science

As motivated and validated by the needs of ECP applications:

- Performance on new node architectures
- Extreme strong scalability
- Interoperability, complementarity: xSDK
- Optimization, UQ, solvers, discretizations
- Advanced, coupled multiphysics, multiscale

Timeline:

- xSDK release 1
- xSDK release 2
- . . . .
- xSDK release n

Next-generation algorithms

Advances in data structures for new node architectures

Improving library quality, sustainability, interoperability

Toward predictive scientific simulations

Increasing performance, portability, productivity
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

E4S Community Policies

E4S 2022.05 release
• 101 ECP ST products
• CUDA
• ROCm
• Tensorflow
• PyTorch

E4S build pipeline
• Cori, NERSC

https://e4s.io
https://spack.io
## E4S Summary

<table>
<thead>
<tr>
<th>What E4S is not</th>
<th>What E4S is</th>
</tr>
</thead>
<tbody>
<tr>
<td>A closed system taking contributions only from DOE software development teams.</td>
<td>Extensible, open architecture software ecosystem accepting contributions from US and international teams. Framework for collaborative open-source product integration.</td>
</tr>
<tr>
<td>A monolithic, take-it-or-leave-it software behemoth.</td>
<td>A full collection if compatible software capabilities and A manifest of a la carte selectable software capabilities.</td>
</tr>
<tr>
<td>A commercial product.</td>
<td>Vehicle for delivering high-quality reusable software products in collaboration with others.</td>
</tr>
<tr>
<td>A simple packaging of existing software.</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 …
HandsOn Lessons

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

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!

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

• Access https://forms.gle/axrawtNsTgbjDTJP8
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.