



Software Needs, Opportunities and Challenges in Polynomial Chaos Modeling for Large-Scale Applications

Eric Phipps

etphipp@sandia.gov

Optimization & Uncertainty Quantification Department
Sandia National Laboratories
Albuquerque, NM USA



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.





Bridging the Gap Between Algorithms Research and Applications is the Challenge

- The challenges for UQ software tools stem from a single tension:
 - The software users, are not the same as the simulation software developers, are not the same as the UQ algorithm developers
- Users want software tools that allow them to get their jobs done
 - Pick software tools that are “good enough”
 - Prefer tools that are easier to use and control rather than the most efficient or accurate
- Simulation software developers
 - Are typically experts on physics/simulation, not UQ
 - Make design choices based on available tools and knowledge
- UQ algorithm developers
 - Needs software tools to develop algorithms that can impact applications
 - Need realistic applications to develop and test those ideas and algorithms



These Challenges Are Not Unique to UQ

- Virtually all areas of advanced analysis suffer from these challenges
- Algorithms require information application codes don't normally provide, in order to do the research needed to impact the problems the applications are trying to solve
- Progress in these areas has been made by identifying the software “hooks” and tools that makes algorithm research in production applications feasible
 - Optimization -- Derivatives
 - Error Estimation -- Adjoint & element residuals
 - Stability Analysis -- Access to linear algebra
- To make progress with UQ & PCE, we must identify these software hooks and tools needed to impact hard applications



Software Needs, Opportunities and Challenges (Discussed in this talk)

- Computing and representing stochastic inputs (need)
 - PC, KL & Fourier expansions of random fields
 - Representing random fields, boundary conditions, and geometries in software
- Intrusive PC propagation in individual applications (opportunity)
 - Dealing with nonlinearities
 - Adaptive and local methods
- PC propagation in coupled systems (challenge)
 - Simulation tools for coupled systems
 - Stochastically coupling system components



Computing Stochastic Representations



Computing Stochastic Representations

- First challenge in stochastic modeling is computing appropriate stochastic input representations
 - Spectral density (a.k.a. PSD)
 - Commonly supplied by engineering codes (e.g., turbulent fluid flow)
 - Karhunen-Loeve
 - Polynomial Chaos
- Few general tools exist to compute these representations for input data
 - Critical for real science applications
 - Overall UQ modeling is only as good as the input representation



Three Software Needs

- Tools for computing stochastic representations from data, e.g.,
 - Maximum likelihood, Bayesian update, least-squares, maximum entropy, Kalman filter
 - What are the software tools needed to implement these algorithms for general use?
- Tools that compute representations from other representations
 - PSD to PCE
 - A code may generate a PSD as input to another code
 - FFTs may be used to generate realizations, how about PCEs?
 - PCE to KL
 - Model reduction based on coarse grid solution (Doostan *et al*, 2007)
 - Dimension reduction between coupled system components
 - Need tools for large (sparse?) eigenvalue problems. Anything else?
 - PECOS
 - New library by Mike Eldred, initially focused on random field realizations from PSD's.



Three Software Needs ... Continued

- Libraries to incorporate these representations in application codes
 - Necessary for both intrusive AND non-intrusive
 - Can be done externally through scripts for non-intrusive, but this is not well integrated (remember users want code that is easy to use!)
- Are existing geometry/discretization libraries sufficient?
 - Exodus, NetCDF, HDF5, ???
 - Random material properties, geometries, boundary conditions, ???
- Experience has shown changing the library specifications is very difficult
 - Is it possible to build UQ wrappers around these basic libraries
- We must have such tools for UQ & PCE to be widely used



Intrusive PC Propagation in Individual Applications



Intrusive Galerkin PC Approximation

- Assuming we have a suitable stochastic representation of model inputs (PCE or K-L, for example)
- Discretized the deterministic part of the problem (e.g., finite-element)

Find $\mathbf{u}(\xi)$ such that $F(\mathbf{u}; \xi) = 0$, $\xi : \Omega \rightarrow \Gamma \subset \mathbb{R}^m$

$$\hat{\mathbf{u}}(\xi) = \sum_{i=0}^P \mathbf{u}_i \psi_i(\xi) \rightarrow F_i(\mathbf{u}_0, \dots, \mathbf{u}_P) = \int_{\Omega} F(\hat{\mathbf{u}}(\xi); \xi) \psi_i(\xi) d\mu = 0$$

- Software needs/challenges
 - Computing PC residuals (i.e., intrusive propagation)
 - Data structures for forming nonlinear PC system and its linearization via Newton's method
 - Solvers for solving the resulting linear and nonlinear systems



Computing PC Residuals is a Significant Challenge in Nonlinear Applications

- Transforming simulation code to compute PC residuals is a significant hurdle for adopting/researching method in complex applications
- Need methods that can transform existing codes into stochastic codes easily
- Need libraries that make it easy to build new stochastic codes
- Several approaches
 - Compute projections in an operation by operation fashion
 - Manual code transformation (most commonly used method)
 - Automation of this through an automatic differentiation-like facility
 - Automation through symbolic finite elements (Sundance -- Kevin Long)
 - Element residual/Jacobian quadrature



What is Automatic Differentiation (AD)?

- Technique to compute analytic derivatives without hand-coding the derivative computation
- How does it work -- freshman calculus
 - Computations are composition of simple operations (+, *, sin(), etc...) with known derivatives
 - Derivatives computed line-by-line, combined via chain rule
- Derivatives accurate as original computation
 - No finite-difference truncation errors
- Provides analytic derivatives without the time and effort of hand-coding them

$$y = \sin(e^x + x \log x), \quad x = 2$$

	x	$\frac{d}{dx}$
$x \leftarrow 2$ $\frac{dx}{dx} \leftarrow 1$	2.000	1.000
$t \leftarrow e^x$ $\frac{dt}{dx} \leftarrow t \frac{dx}{dx}$	7.389	7.389
$u \leftarrow \log x$ $\frac{du}{dx} \leftarrow \frac{1}{x} \frac{dx}{dx}$	0.301	0.500
$v \leftarrow xu$ $\frac{dv}{dx} \leftarrow u \frac{dx}{dx} + x \frac{du}{dx}$	0.602	1.301
$w \leftarrow t + v$ $\frac{dw}{dx} \leftarrow \frac{dt}{dx} + \frac{dv}{dx}$	7.991	8.690
$y \leftarrow \sin w$ $\frac{dy}{dx} \leftarrow \cos(w) \frac{dw}{dx}$	0.991	-1.188



Sacado: AD Tools for C++ Applications

- AD via operator overloading and C++ templating
 - Expression templates for OO efficiency
 - Application code templating for easy incorporation
- Designed for use in large-scale C++ codes
 - Apply AD at “element-level”
 - Manually integrate derivatives into parallel data structures and solvers
 - Sacado : : FEApp example demonstrates approach
- Very successful in enabling analytic sensitivity calculations in large-scale codes
 - Charon, Aria, Xyce, Alegria, LAMMPS
 - Jacobians, parameter sensitivities, 2nd derivatives



Computing PC Residuals Operation by Operation via AD

- AD relies on known derivative formulas for all intrinsic operations plus chain rule
- AD infrastructure provides deep interface into application code
 - Access to entire computational graph
- Similar approach can be used for any computation that can be done in an operation by operation manner
 - Assume inductively that PC expansions for two intermediate variables a and b have been computed, and we wish to compute a third c

$$\text{Given } a = \sum_{i=0}^P a_i \psi_i(\xi), \quad b = \sum_{i=0}^P b_i \psi_i(\xi),$$

$$\text{Find } c = \sum_{i=0}^P c_i \psi_i(\xi) \text{ such that } \int_{\Omega} (c - \varphi(a, b)) \psi_i d\mu = 0, \quad i = 0, \dots, P$$



Projections of Arithmetic Intermediate Operations

$$\langle fg \rangle \equiv \int_{\Omega} f(\xi)g(\xi)d\mu, \quad \{\psi_k\}_{k=0}^P \text{ orthogonal}$$

- Addition/subtraction

$$c = a \pm b \Rightarrow c_i = a_i \pm b_i$$

- Multiplication

$$c = a \times b \Rightarrow \sum_i c_i \psi_i = \sum_i \sum_j a_i b_j \psi_i \psi_j \rightarrow c_k = \sum_i \sum_j a_i b_j \frac{\langle \psi_i \psi_j \psi_k \rangle}{\langle \psi_k^2 \rangle}$$

- Division

$$c = a/b \Rightarrow \sum_i \sum_j c_i b_j \psi_i \psi_j = \sum_i a_i \psi_i \rightarrow \sum_i \sum_j c_i b_j \langle \psi_i \psi_j \psi_k \rangle = a_k \langle \psi_k^2 \rangle$$



Projections of Transcendental Operations More Challenging

- Debusschere *et al* (2004) proposed two approaches
 - Taylor series approximation
 - Use arithmetic rules repeatedly until error tolerance achieved
 - Time integration
 - Transcendental operations satisfy simple differential equations
 - By picking integration path, starting and ending points can compute coefficients using standard time integrator
 - Both approaches encapsulated into the UQLib library by Najm, Debusschere, Ghanem, and Knio
- Third approach also based on differential equations
 - Differentiate w.r.t. polynomial argument leads to linear systems for coefficients of degree > 0 , up to scaling
 - Degree 0 & scaling computed by summing series at origin
 - Not readily extensible to multivariate problems



Another Approach is Quadrature for Element-Based Codes

$$F(u, p) = \sum_{k=0}^N Q_k^T f_k(P_k u(p); p)$$

- Evaluate via quadrature for globally assembled residual

$$F_i = \int_{\Gamma} F(\hat{u}(y); y) \psi_i(y) \rho(y) dy$$

- Requires parallel quadrature routines but only interface to global residual

- Apply quadrature for each element residual then assemble

$$F_i = \sum_{k=0}^N Q_k^T \int_{\Gamma} f_k(P_k \hat{u}(y); y) \psi_i(y) \rho(y) dy$$

- Requires only serial quadrature routines but needs element-level interface

- Boundary conditions add complexity

- Jacobian decomposes similarly

- In either case, application interface is significantly simpler than op-by-op

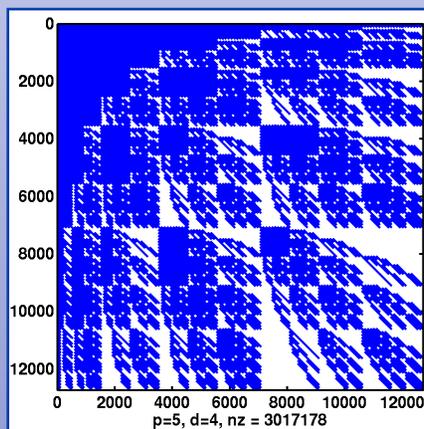
Intrusive PCE Requires Much More Than PC Propagation

- Data structures for forming block PC nonlinear system

$$\bar{F}(\bar{u}) = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_P \end{bmatrix}, \quad \bar{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_P \end{bmatrix}$$

- Linear solver/preconditioner methods for solving block PC linear systems (after linearization)

$$\frac{\partial F_i}{\partial u_j} = \sum_{k=0}^P J_k \langle \psi_i \psi_j \psi_k \rangle, \quad J_k = \int_{\Gamma} \frac{\partial F}{\partial u}(\hat{u}(y); y) \psi_k(y) \rho(y) dy$$



- Trilinos provides nice opportunity for developing these capabilities



The Trilinos Project

- <http://trilinos.sandia.gov>
- Algorithms and enabling technologies
- Large-scale scientific and engineering applications
- Object oriented framework
- “String of Pearls”
- *Focus on packages*
 - Over 30 packages in 8.0 release
 - Over 40 in development
 - Growing exponentially

The screenshot shows the homepage of the Trilinos Project. At the top, it features the project title "The Trilinos Project" and the Sandia National Laboratories logo. A navigation menu includes links for Home, About, Resources, Packages, and Download. The main content area is divided into several sections:

- Welcome to the Trilinos Project Home Page:** A paragraph describing the project's goal to develop algorithms and enabling technologies within an object-oriented software framework for large-scale, complex multi-physics engineering and scientific problems. It highlights the focus on packages.
- Trilinos Packages:** A section explaining that each package is a self-contained, independent piece of software with its own set of requirements, its own development team and group of users. It notes that Trilinos is designed to respect the autonomy of packages and offers various ways for a particular package to interact with other Trilinos packages. It also mentions tools for assisting package developers with builds across multiple platforms, generating documentation, and regression testing.
- Trilinos Release 8.0 Now Available:** A prominent announcement stating that Trilinos 8.0 is now available for download. It lists new features across most packages, including Belos (next-generation iterative solvers), Sacado (automatic differentiation), and TrilinosCouplings (select Trilinos package interfaces). It also provides a link to release notes for more information.
- Trilinos User Group 2007:** A section reporting that the Trilinos User Group Meeting was held from November 6-8 (Tuesday - Thursday) in Albuquerque.
- User presentations:** A note that user presentations focused on the capabilities released as part of Trilinos 8.0 in August 2007, and those scheduled for release as part of Trilinos 9.0 in September 2008, are now available at the provided link.

Logos for ASC, TOPS (Towards Optimal Postcure Simulations), and R&D 100 are also visible on the left side of the page. A "Trilinos" logo is displayed in a blue box on the right side.



Trilinos Packages

	Objective	Package(s)
Discretizations	Meshing & Spatial Discretizations	phdMesh, Intrepid
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Core	Linear algebra objects	Epetra, Jpetra, Tpetra
	Abstract interfaces	Thyra, Stratimikos, RTOp
	Load Balancing	Zoltan, Isorropia
	“Skins”	PyTrilinos, WebTrilinos, Star-P, <i>ForTrilinos</i>
	C++ utilities, (some) I/O	Teuchos, EpetraExt, Kokkos, Triutils
Solvers	Iterative (Krylov) linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi
	ILU-type preconditioners	AztecOO, IFPACK
	Multilevel preconditioners	ML, CLAPS
	Block preconditioners	Meros
	Nonlinear system solvers	NOX, LOCA
	Optimization (SAND)	MOOCHO, Aristos



Trilinos Tools Useful for Intrusive PCE

- Epetra -- MPI-based vector/matrix data structures & operator interfaces
 - Used by application codes to form FE residuals, Jacobians
- Thyra -- Abstract vector, operator, and nonlinear interfaces
 - Allows abstraction of solver algorithms away from application data structures
 - Product vectors for representing block PCE solution/residual vectors

$$\bar{F}(\bar{u}) = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_P \end{bmatrix}, \quad \bar{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_P \end{bmatrix} \quad (\bar{J}\bar{v})_i = \sum_{j,k=0}^P \langle \psi_i \psi_j \psi_k \rangle J_k v_j$$

- Operators implementing PCE matrix-vector-product in “matrix-free” fashion
- Nonlinear interface transforming deterministic Thyra interface into PCE
- Ifpack, ML, Amesos -- Preconditioners and direct solvers
 - Approximate inverse of mean (degree 0) PCE block
- AztecOO, Belos -- Iterative linear solvers
 - Advanced solvers for block PCE linear systems
- NOX & Rythmos -- Abstract nonlinear solver & time integration algorithms
 - Use nonlinear Thyra PCE interface to solve steady & transient PCE problems



Trilinos Package Stokhos

- These ideas form the basis for a new Trilinos package called Stokhos
 - Collaborative effort among the PCE community to develop tools for large-scale codes
- Initial thoughts are Stokhos will provide
 - PCE vector/operator interfaces
 - Nonlinear PCE application interface
 - Solver/preconditioner algorithms
 - Intrusive propagation methods
- Currently it only has
 - General facilities for intrusive propagation
 - Wrappers around UQLib library of Najm, Debusschere, Ghanem & Knio
 - Implementation of the linear-solve AD approach
 - Sacado wraps these for AD PCE
- Sacado::FEApp
 - Example 1D finite element code demonstrating AD
 - Initial implementation of PCE solvers/interfaces using Epetra

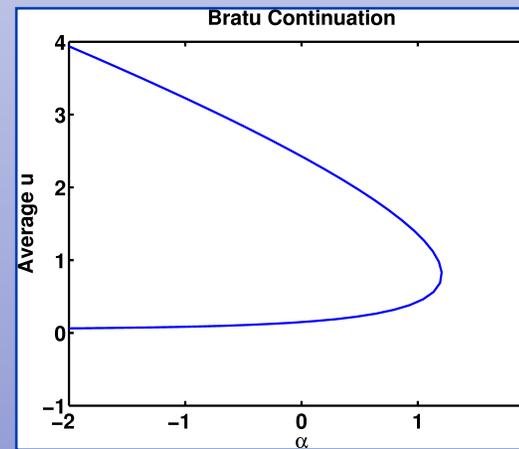
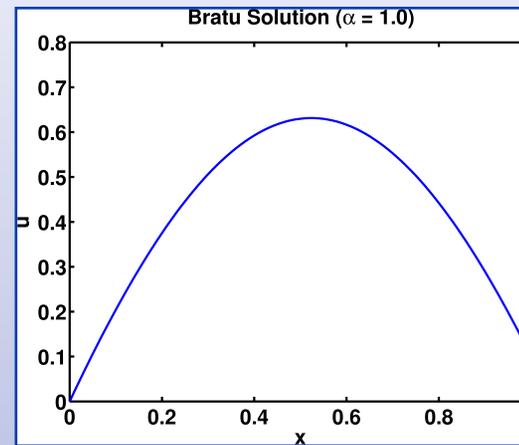


Sacado::FEApp Demonstration of Intrusive PCE using Stokhos & Sacado

- 1-D Bratu problem:

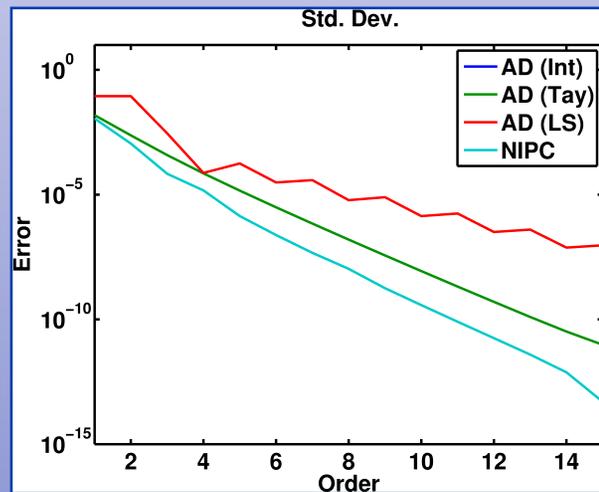
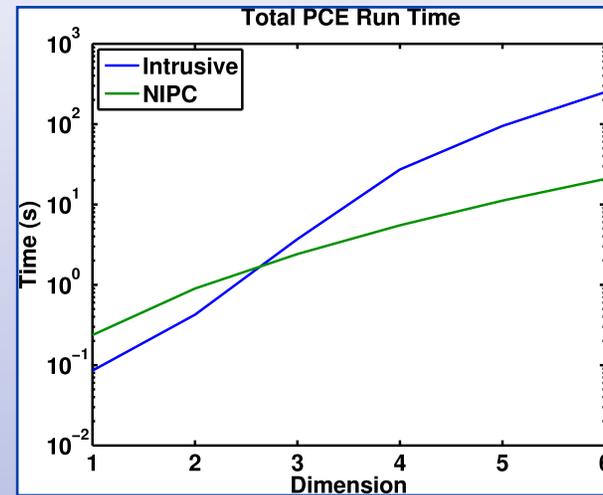
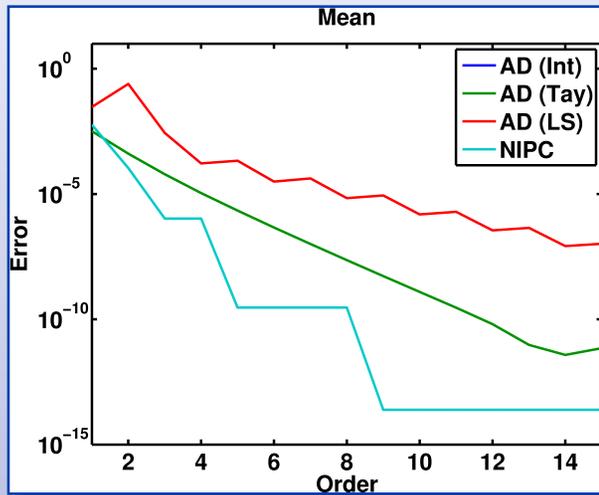
$$\frac{\partial^2 u}{\partial x^2} + e^\alpha e^u = 0, \quad 0 \leq x \leq 1$$

- Exponential parameter dependence just for fun!
- Linear finite element discretization, 100 elements
- Uniform random variables for nonlinear factor over $[-1, 1]$, PCE using Legendre polynomials (to avoid bifurcation!)
- Sacado wrapping Stokhos wrapping UQLib library, using Taylor series approach for exponential
- Matrix free-PCE Jacobian using mean as preconditioner, Itpack RILU(0) preconditioner
- Solution mean used as quantity of interest





PCE for 1-D Bratu Problem



- NIPC using sparse-grid quadrature (Dakota)
- Intrusive run time is dominated by residual and Jacobian fill
- Solve scaling roughly linearly
- Suggests quadrature approach may be more efficient



Beyond Global PCE

- To impact real applications we must deal with lack of smoothness in parameter spaces
 - Bifurcations
 - Discontinuities
- Local/adaptive methods
 - Wavelet-type approximations (Le Maître *et al*)
 - Finite element-type approximations (Babuska *et al*)
- What are the tools needed to implement these in complex codes
 - In principle, the software ideas for global PCE still apply
 - Are product-vector structures the best way to store coefficients?
 - We must exploit parallelism over local parameter domains



PC Propagation in Coupled Systems



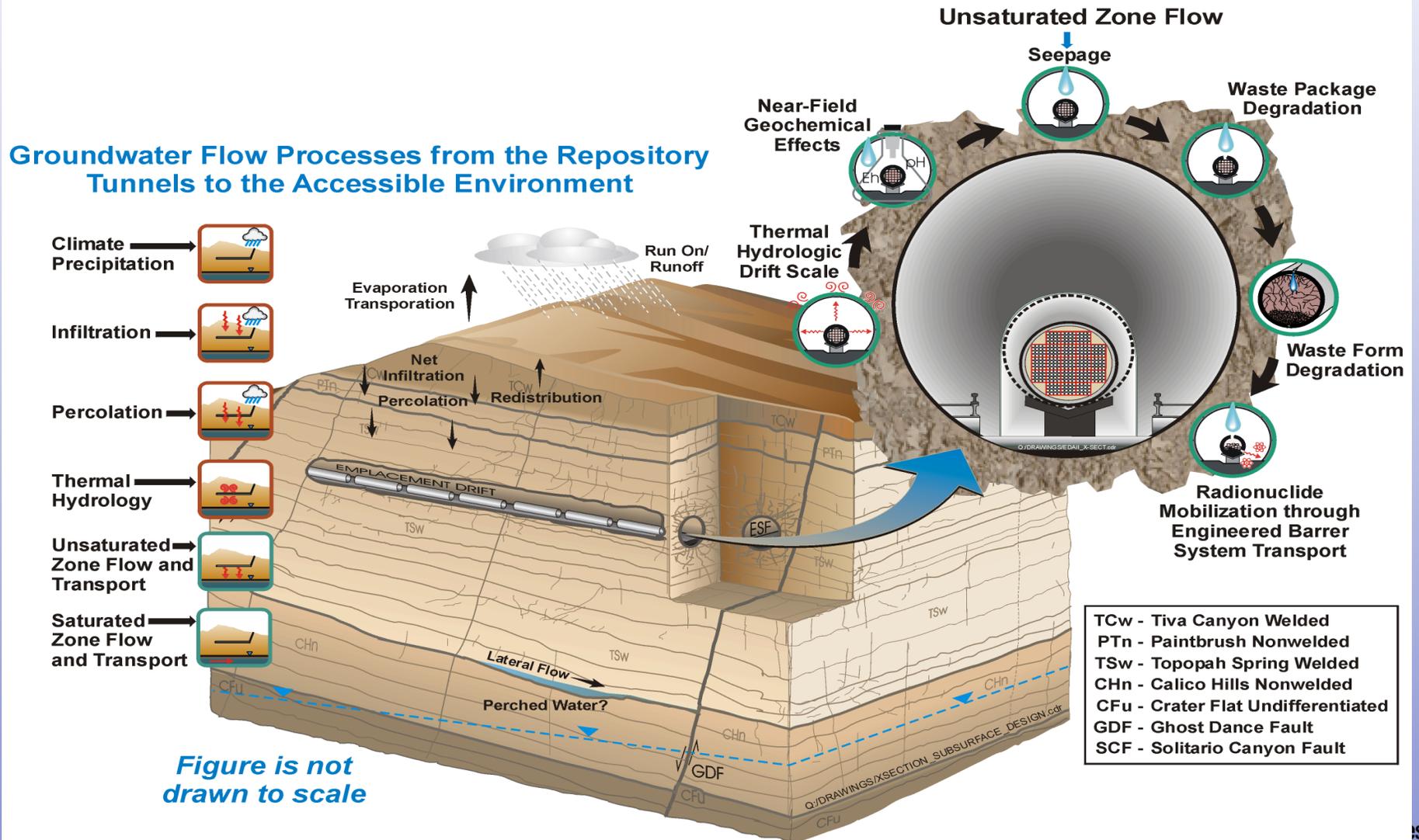
Coupled Systems Present New Opportunities

- Coupled systems
 - System of interacting components
 - Each component often simulated by separate code or module
 - Overall system driven by some solution process
- Coupled systems are where the funding is going
 - One kind of “complex system”?
- Coupled systems provide tremendous opportunities for PC modeling to impact science applications
 - Waste repository modeling
 - Nuclear reactor design & licensing
 - Fusion reactor design
 - Electrical systems

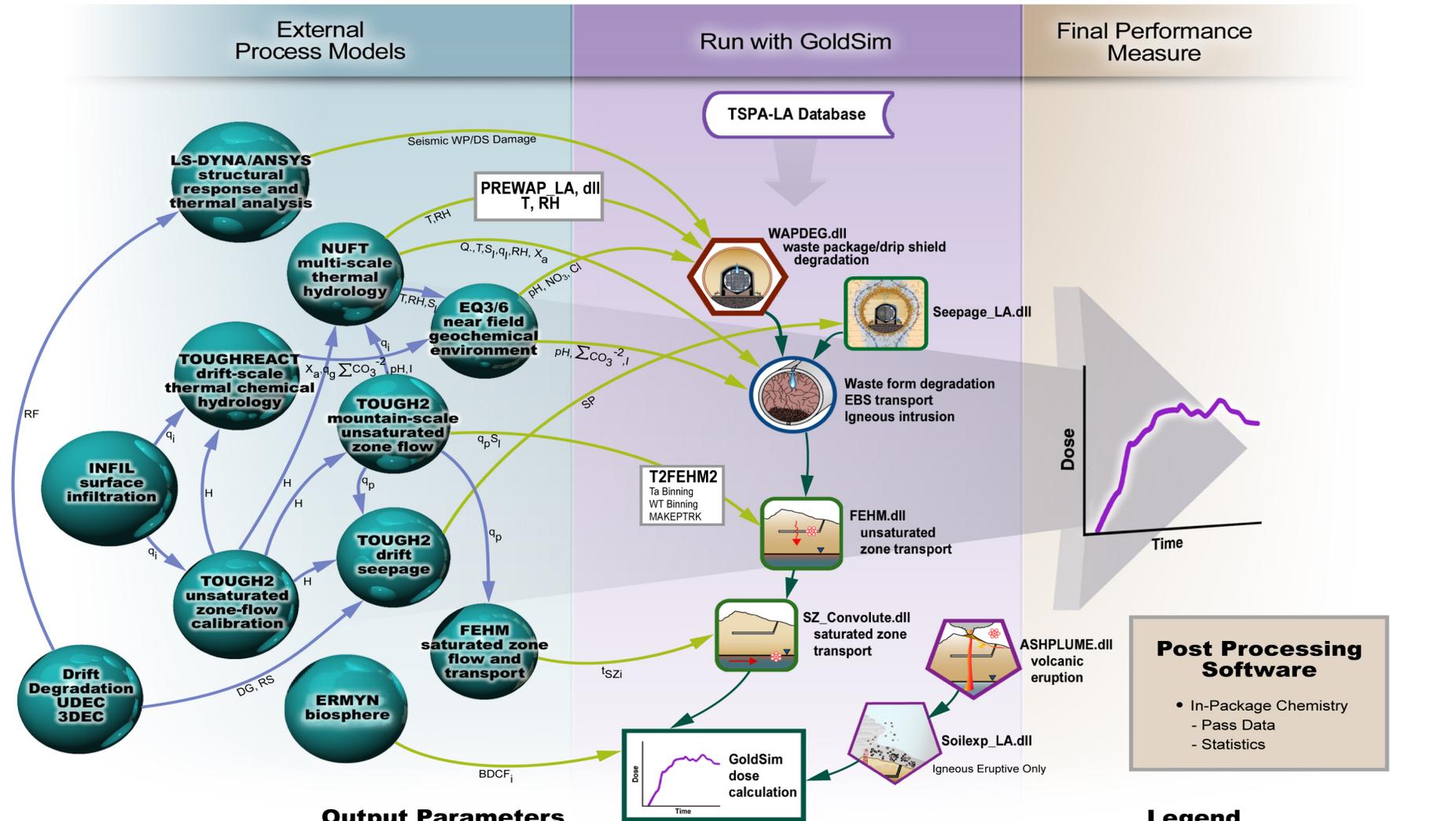


Yucca Mountain

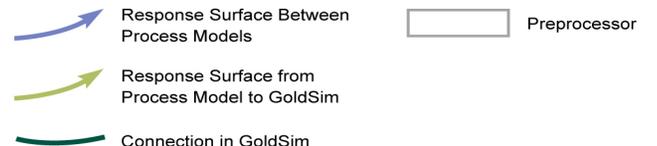
Groundwater Flow Processes from the Repository Tunnels to the Accessible Environment



TSPA-LA Software Architecture



f _S	Fraction of WPs with Seeps	T	Temperature	I	Ionic Strength
EBS	Engineered Barrier System	RH	Relative Humidity	t _{SZi}	Saturated Zone Transport Time
Q _S	Seep Flow Rate	S _l	Liquid Saturation	BDCF _i	Biosphere Dose Conversion Factor
Q _e	Evaporation Rate	X _a	Air Mass Fraction	q _g	Gas Flux
pH	pH	q _l	Liquid Flux	H	Hydrologic Properties
ΣCO ₃ ⁻²	Carbonate Concentration	q _i	Infiltration Flux	SP	Seepage Parameters
q _p	Percolation Flux	DG	Drift Geometry	RS	Rock Strength
NO ₃	Nitrate Concentration	Cl	Chloride Concentration	RF	Rock Fall Size and Number





Challenges for PCE Modeling of Coupled Systems

- Effective simulation tools don't exist!
 - Collection of distinct simulation codes coupled through scripts and user interaction
 - Loose nonlinear coupling that may not represent physics
 - Slow and/or inaccurate solution processes
 - Picard iteration
 - Operator splitting/lagging
 - Best case scenario is a an automated system that can be driven by Monte Carlo
- Opportunities are ripe for PCE modeling
 - Exploit coupled system structure
 - Apply PCE to each component
 - Model reduction between components
 - Invert traditional layering of UQ on top of solution processes



Software Path Forward for PCE of Coupled Systems

- Foundational tools
 - Computing stochastic representations
 - PC propagation
- Software needs
 - Interfaces for coupling components through PCE
 - Nonlinear solver software for solving stochastically coupled system
 - Simulation tools to put these systems together
- From this we can begin to address these types of problems
 - Enable the real algorithm research that must occur



Major Challenges in Important Areas of PC Modeling

- Computing stochastic representations
 - Need tools for computing representations from data
 - Need tools for generating PCEs from KL's and PSD's, and vice versa
- PC propagation through nonlinear applications
 - AD approach is effective, but may not be the most efficient
 - Quadrature approach is simpler and may be faster
 - Trilinos provides tools to develop solvers optimized for PCE
- PC modeling of coupled systems exploiting structure
 - Good simulation tools don't exist
 - Opportunity to exploit structure by inverting UQ over solution layering



Much More To This Story

- Validation
 - Comparing to experimental data
 - Inverse problems
- Decision support
 - Weighing risk against performance
- Higher-order analysis
 - Optimization and uncertainty
 - Model reduction and uncertainty
 - Stability analysis and uncertainty
 - Error estimation and uncertainty