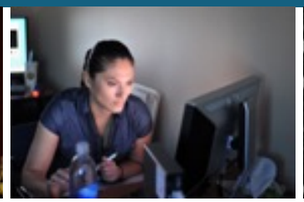
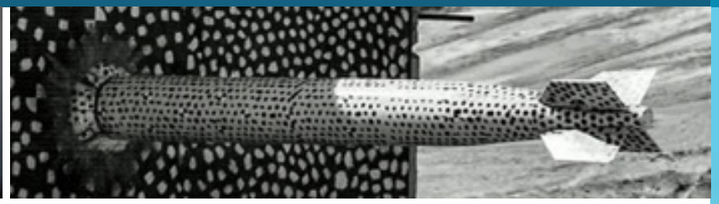
Enabling Scalable Predictive Circuit Simulation using Trilinos





PRESENTED BY

Heidi K. Thornquist, Sandia National Laboratories

Trilinos User & Developer Group Meeting

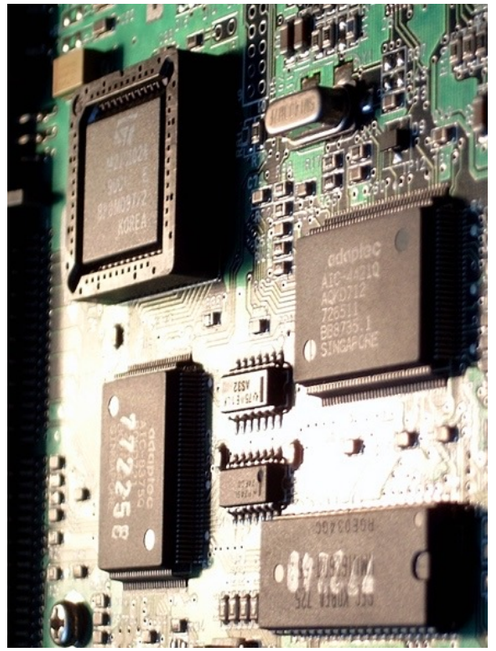
March 18-19, 2026

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.

SAND2026-18545C

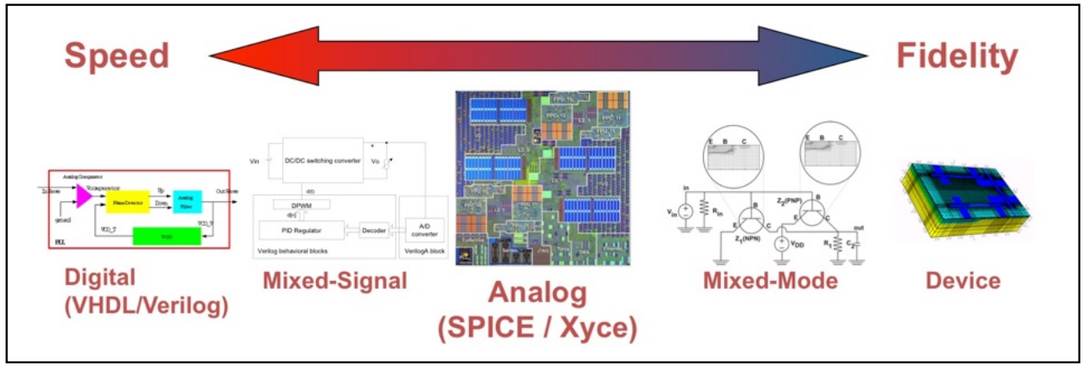
Outline

- What is circuit simulation?
- Xyce software architecture
  - Trilinos solver stack
  - Tpetra transition effort
- Highlights of capabilities and performance
  - Linear solver development
  - Recent solver improvements



What is circuit simulation?

Definition: A technique for checking and verifying the design of electrical and electronic circuits and systems prior to manufacture and deployment[1]



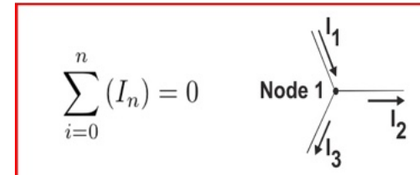
Analog circuit simulation provides tradeoff between fidelity and speed

- Models network(s) of devices coupled via Kirchoff's current and voltage laws

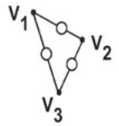
$$f(x(t)) + \frac{dq(x(t))}{dt} = b(t)$$

- SPICE[2] is the defacto industry standard for analog circuit simulation

$$\sum_{i=0}^{n} (I_n) = 0$$



$$\sum_{i=0}^{n} (\Delta V) = 0$$



○ = Device
• = Node

$$G = \frac{\delta I}{\delta V} = \text{conductance} = \text{Jacobian term}$$

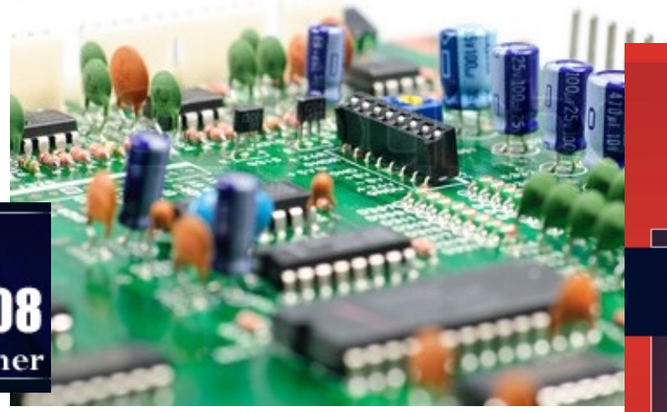
Ohm's Law: $I = GV = V/R$

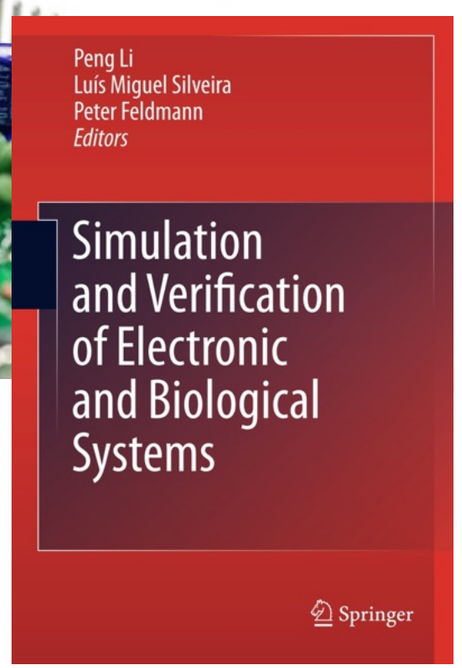
[1] Najm, F. N. Circuit Simulation. John Wiley & Sons, Inc., 2010.
[2] Nagel, Laurence W. SPICE2: A Computer Program to Simulate Semiconductor Circuits, California Univ Berkeley Electronics Research Lab, 1975.

Motivation for Xyce

Xyce provides a capability to analyze general network systems and has been integral to modeling and simulating Sandia's Complementary Metal-Oxide-Semiconductor (CMOS) circuit designs, as well as:

- Biological networks
- Neural networks
- Power grids





Single code base that efficiently and seamlessly simulates systems that can have a wide range of dynamics, topological complexity, and size (10-10^7)

- Provides support for both serial and distributed-memory parallel builds
- Open source, GPLv3, see https://xyce.sandia.gov or https://github.com/xyce

Xyce™ PARALLEL ELECTRONIC SIMULATOR

A high-level view of the Xyce simulation flow

Parsing

- Convert netlist file syntax to equivalent devices and network/circuit connectivity
- Distribute devices over multiple processors
- Determine global ordering and communication

Device Evaluation

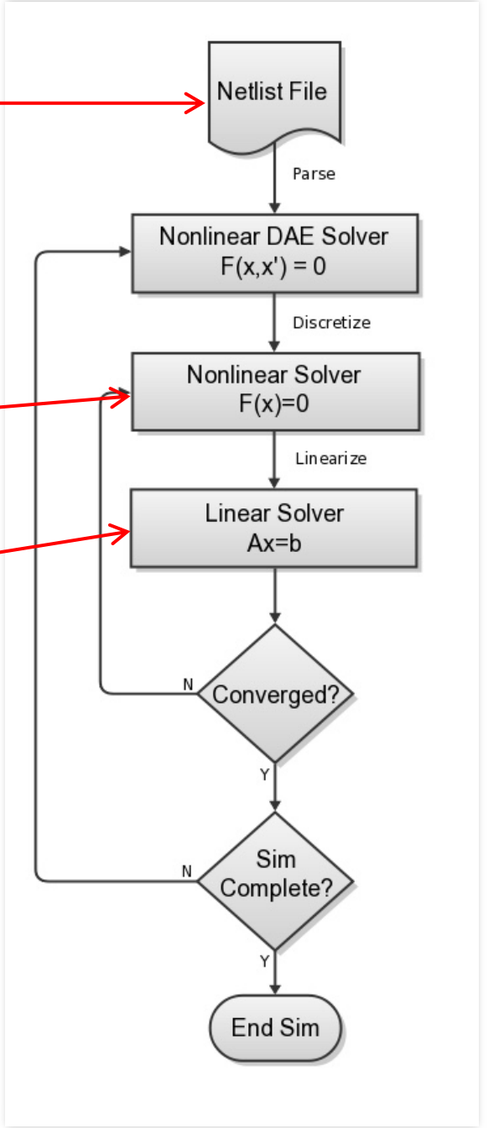
- Loop through all devices for state evaluation and matrix loading

Linear Solve

- Sparse linear algebra and solvers used to solve linearized system

Advanced Analysis Methods

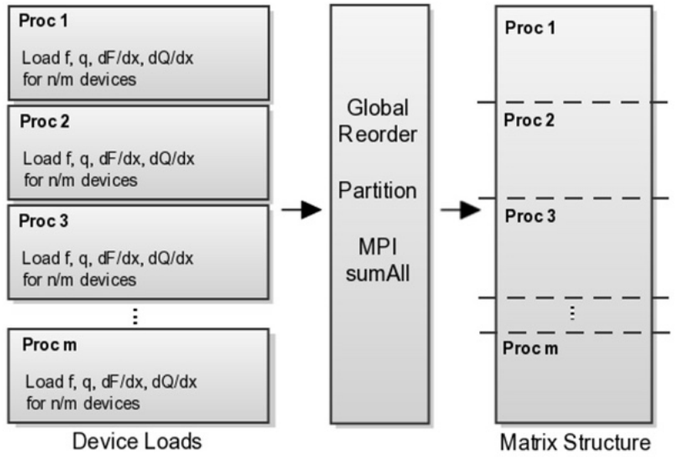
- Sampling: Monte Carlo / LHS (DAKOTA),
- Sensitivity: PCE (Stokhos)
- Optimization: ROL

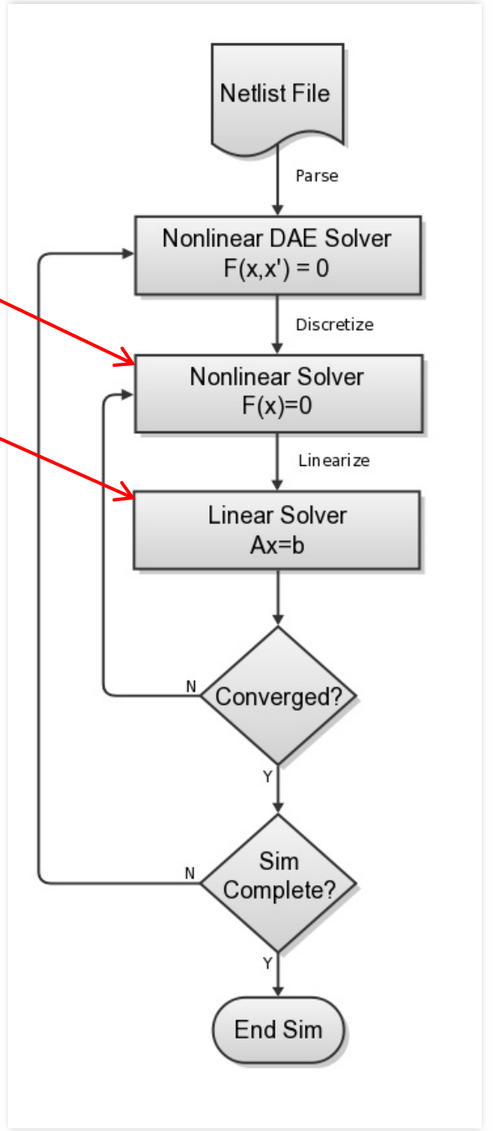


Challenges for the Xyce simulation flow

Computational load balancing

- For **smaller** circuits, performing the device evaluations dominates the total simulation time
- For **larger** circuits, solving the linear system of equations dominates the total simulation time
- A redistribution layer is employed for parallel execution to enable separate load balancing strategies

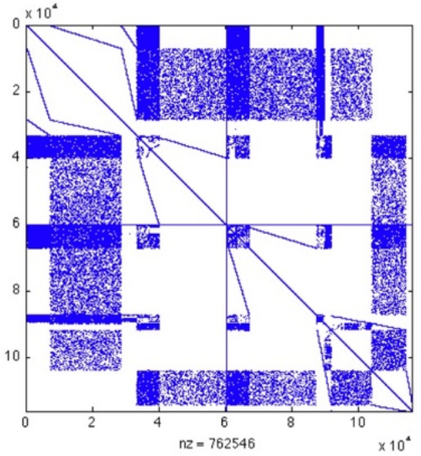


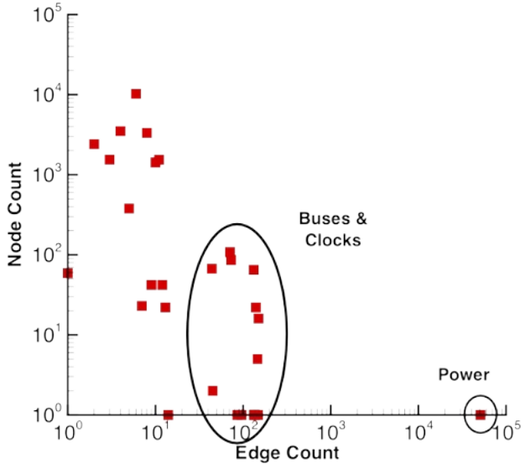


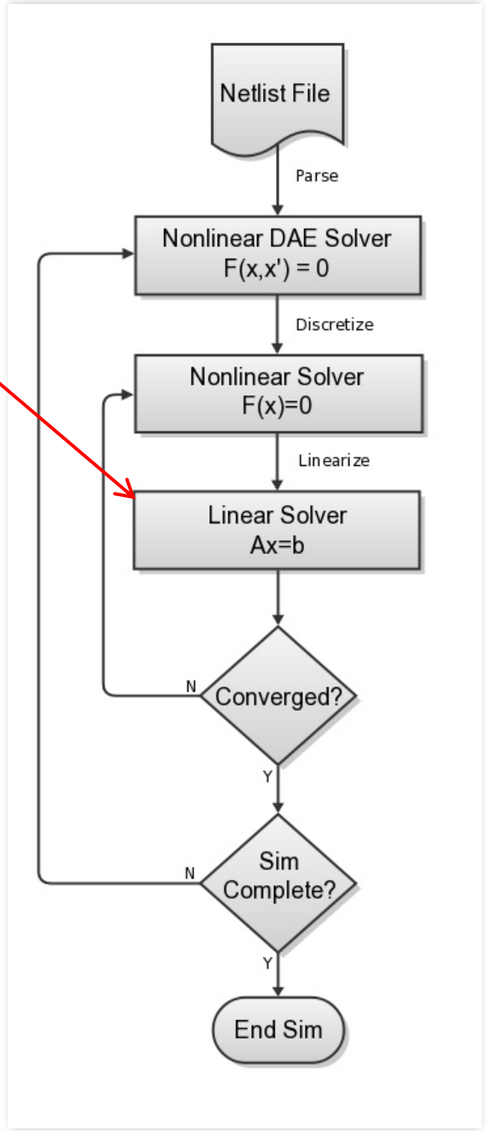
Challenges for the Xyce simulation flow

Hierarchical connectivity:

- Analog circuit simulation often generates graph structures that are hierarchical when compared to the spatial topology of PDEs
- High-connectivity nodes result in high communication in parallel execution, so removing them is beneficial
- Linear systems are predominately non-Hermitian and very sparse. Iterative solvers provide scalability, but direct solvers provide robustness

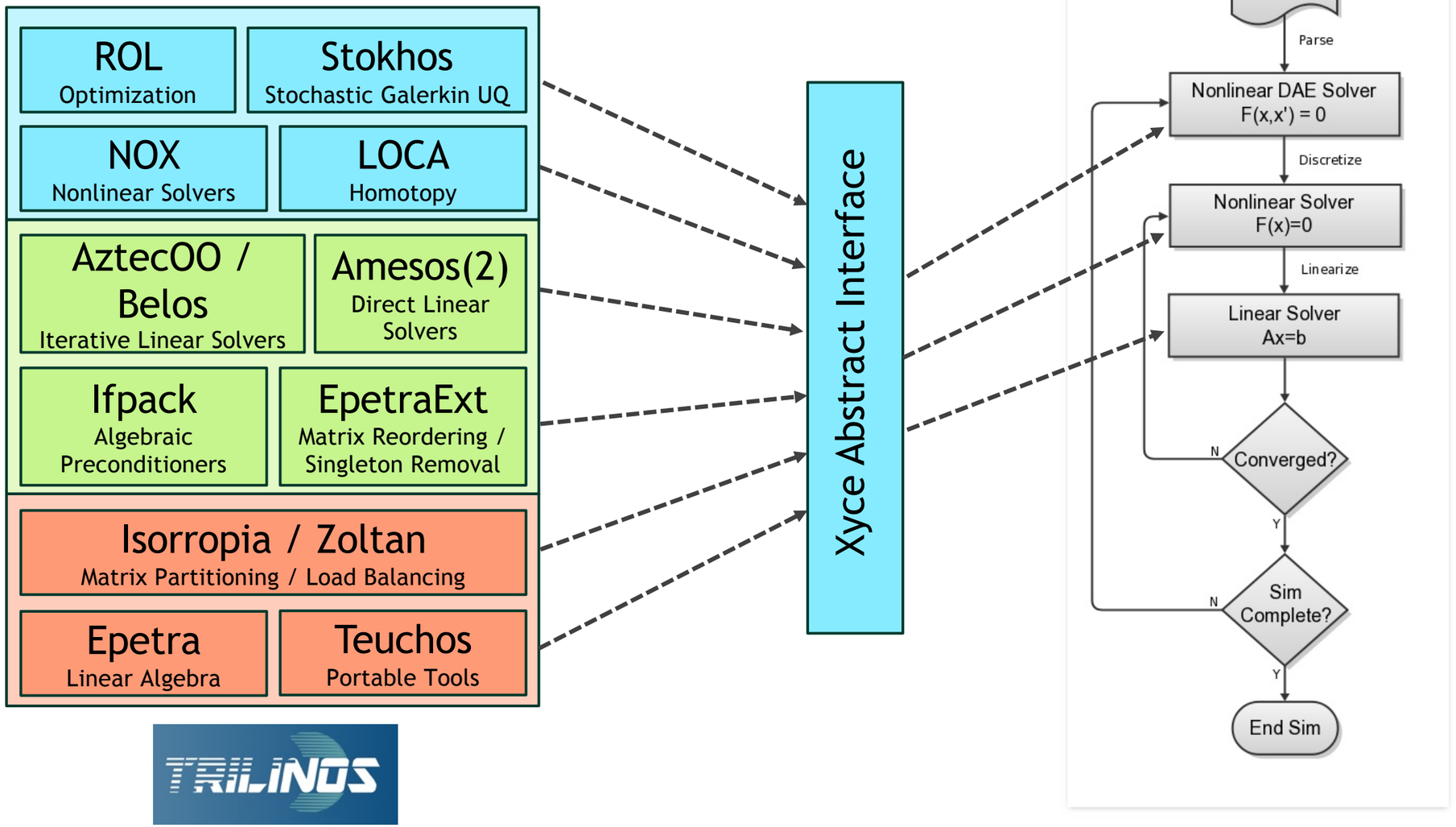




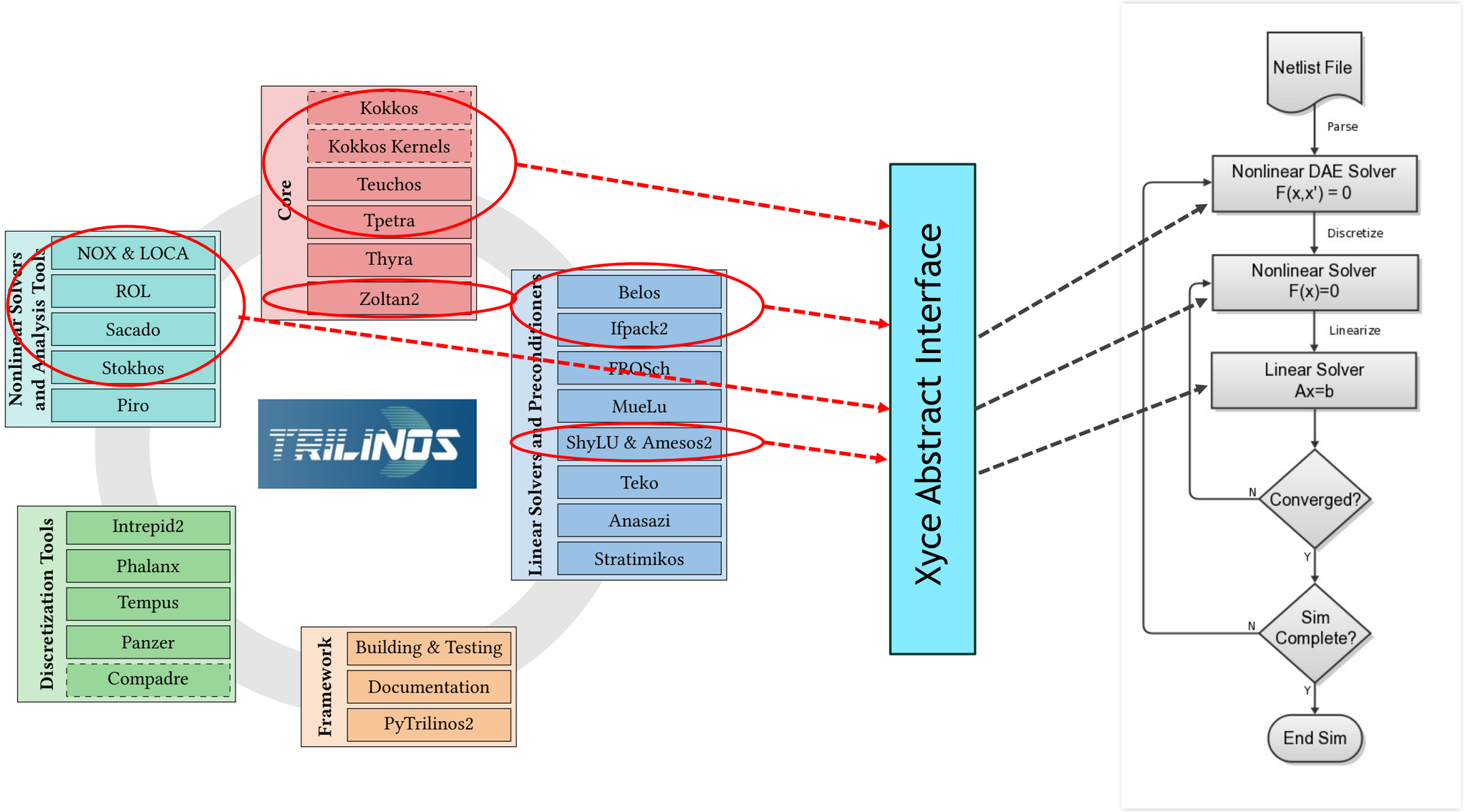


Xyce Simulation Flow + Trilinos Solver Stack

[Epetra based]



Xyce Simulation Flow + Trilinos Solver Stack
[Tpetra based]



Trilinos image courtesy of "Trilinos: Enabling Scientific Computing Across Diverse Hardware Architectures at Scale," Mayr, et al., https://arxiv.org/abs/2503.08126

Tpetra Transition Effort

Initially (circa 1999), Xyce was one of the first application codes at Sandia to fully integrate the Trilinos solver stack, which influenced its design:

- In general, Xyce abstract interfaces look a lot like Epetra
- Rampant use of raw pointers, followed by an inelegant use of smart pointers
- Direct access to matrix/vector memory in device models
- Ubiquitous use of copy semantics, instead of view, for map/graph objects, for example:

```
epetraGraph_ = Teuchos::rcp( new Epetra_CrsGraph( Copy, *epetraMap, numIndicesPerRow.data() ) );
```

- Unnecessarily leverages dynamic creation of graph/matrix objects
- Relies on supporting tools provided by EpetraExt
  - Singleton filtering
  - Partitioning interfaces
  - Remapping utilities

Tpetra Transition Effort

As Xyce was transitioning to Tpetra there were some helpful things:

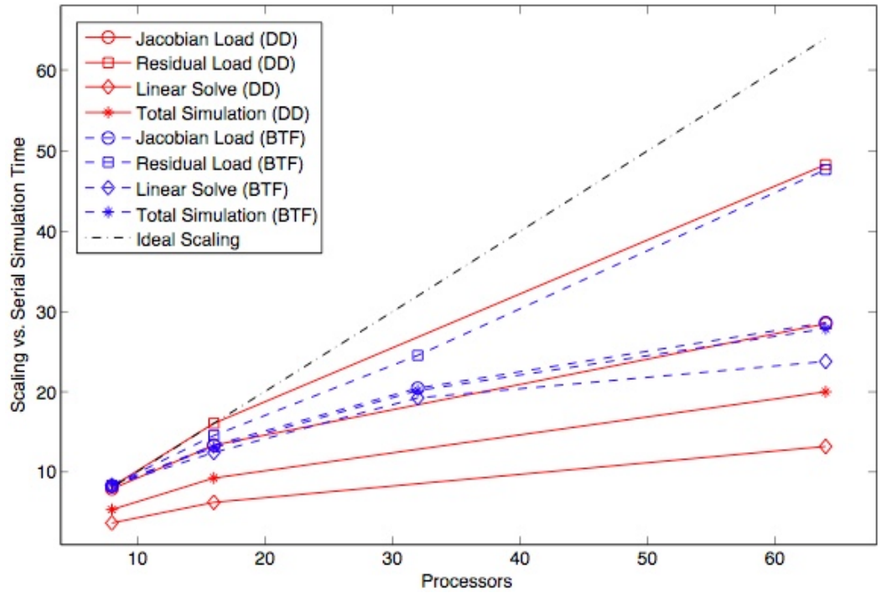
- Many of the linear algebra classes have included Epetra-esque interfaces.

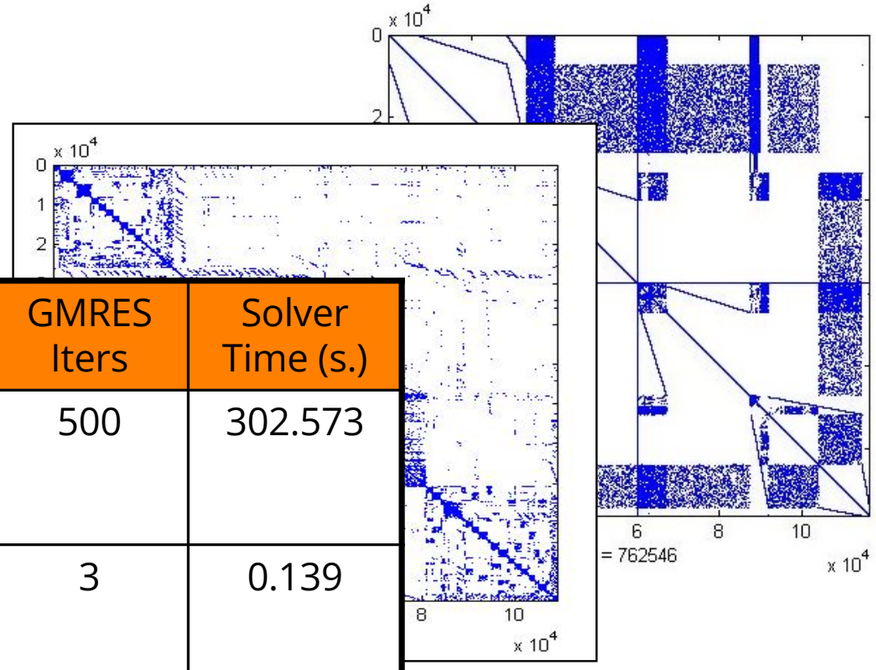
As well as some unexpected challenges:

- The need to integrate `resumeFill()` into matrix loading, particularly in cases when matrix loading is a multi-stage process
- Obtaining a pointer to underlying memory to pass to the device models required interacting with Kokkos objects, often disguised by a typedef
- Some capabilities required by block analysis methods may not be compatible with Tpetra design
- Non-member helper functions were introduced to provide “all-in-one” methods to create new objects by import/export data, but the results were mixed

Trilinos supports linear solver development for Xyce

- **Custom approaches for linear solvers have been necessary to achieve scalable performance for analog simulation**
- For small scale circuits, the Dulmage-Mendelsohn permutation (BTF) was leveraged in the KLU sparse direct solver [Tim Davis (UF, Texas A&M)]
- BTF structure was leveraged to create a new preconditioned iterative method
  - Great for (older) CMOS memory circuits
  - Circuits with parasitics are more challenging





Preconditioning Method	Residual	GMRES Iters	Solver Time (s.)
Local AMD ILUT ParMETIS	3.43e-01 (FAIL)	500	302.573
BTF Block Jacobi Hypergraph	3.47e-10	3	0.139

Linear Solver Improvements (2016)

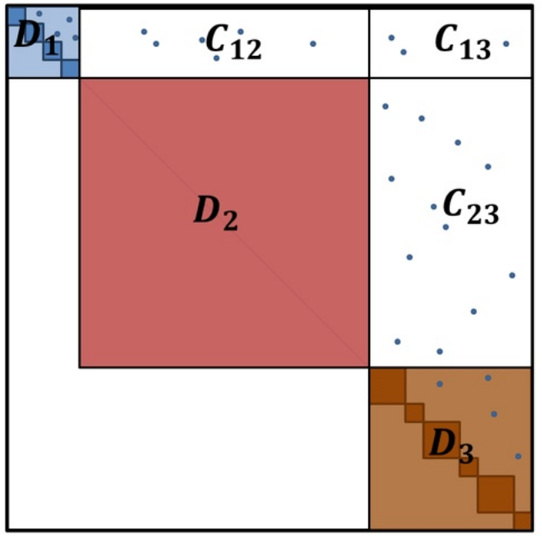
Trilinos-Amesos2: Basic sparse LU factorization with row-pivoting

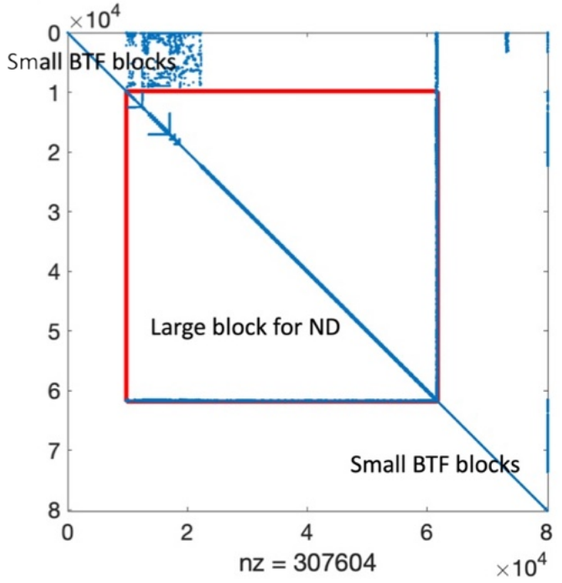
Templated on a scalar type, `BASKER_ScalarTraits<>`

- Default / Copy constructor
- Arithmetic operators: +,-,*,/
- Relational/comparison operators: ==, !=, >, <
- Multiple linear algebra (Epetra, Tpetra) adapters exist in Amesos2

Multi-threaded version (ShyLU-Basker)[3]:

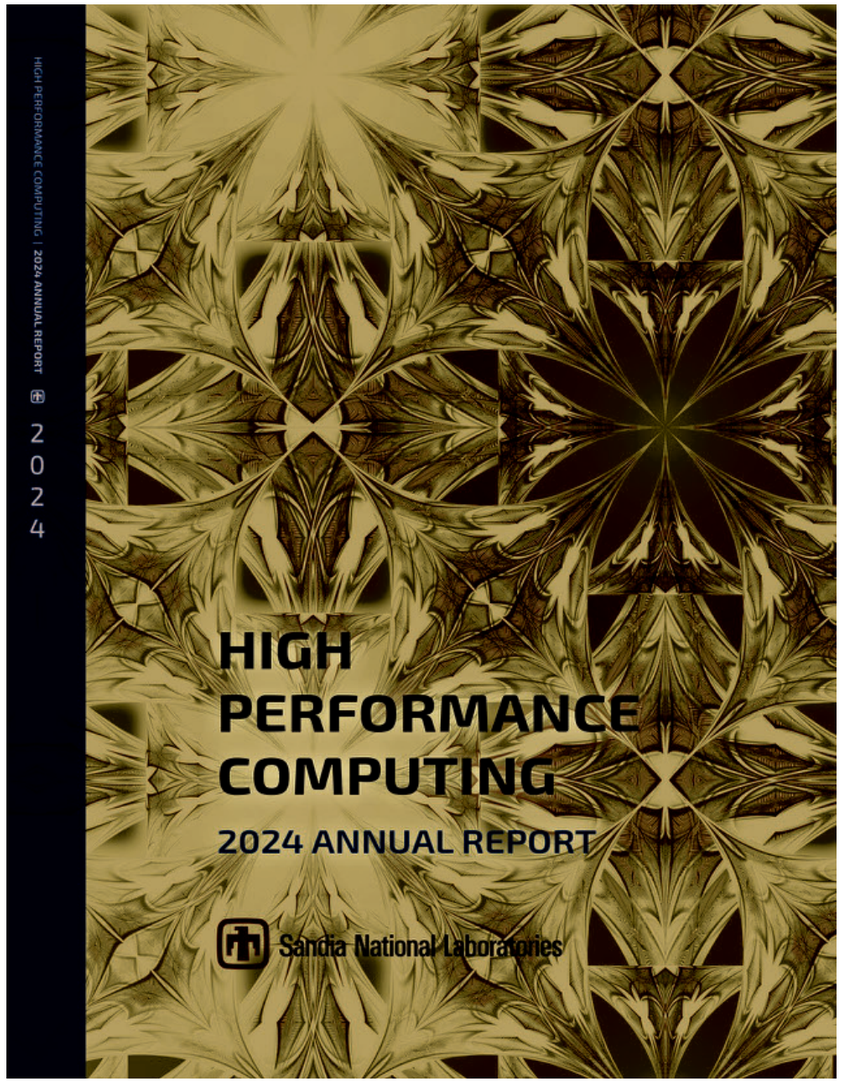
- Parallelizes the Gilbert-Peierls algorithm
- Uses hierarchical data layouts to expose fine-grain parallelism and map to memory structure of most-many core systems
  - Block triangular form (BTF) to compute coarse structure
  - Nested Dissection (ND) to expose parallelism for large diagonal blocks (ex. D_2)
- Improves transient performance in Xyce for small to medium sized circuits





[3] "Basker: A Threaded Sparse LU Factorization Utilizing Hierarchical Parallelism and Data Layouts," J. Booth, S. Rajamanickam, and H. Thornquist, 2016 IEEE IPDPSW, pp. 673-682.

Recent Solver Improvements (2024)



Radiation-aware Xyce simulations of memory circuits

Integrated circuit simulation for design

AUTHORS/TEAM
Eric Keiter, Heidi Thornquist, Ting Mei, Pranita Kerber, Biliana Paskaleva, John Teifel, Spencer Nelson, Clark Dohrmann, Ichitaro Yamazaki

CONTRIBUTING WRITER
Antonia (TJ) Cardella

the microelectronics community and form the basis for the electronic design automation industry.

One type of computational analysis is circuit simulation[1] and involves a

the Berkeley SPICE program,[2]

to constraints not present in analog designs, and these constraints can often be exploited to expedite computational analysis. Hence, the traditional

simulators, using standard cell models as input.

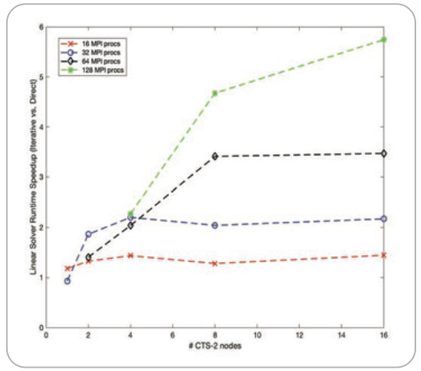


FIGURE 1

Runtime speedup of block subdomain preconditioned method (overlap = 1, Intel MKL Pardiso, 4 threads) compared to direct method (Intel MKL Pardiso, 16 threads) in simulating SRAM circuit on CTS-2. The MPI processors are varied from 16, 32, 64, and 128, while the number of CTS-2 nodes are varied from 1,2,4,8, and 16.

Trilinos-enabled frequency-domain simulation in Xyce

Harmonic Balance analysis is a frequency-domain simulation technique used to efficiently calculate the steady-state response of nonlinear RF and microwave circuits

$$\frac{dq(x)}{dt} + f(x) = b(t),$$

using Fourier series expansions for $x(t)$ and $b(t)$: $x(t) = \sum_{i=-\infty}^{\infty} X_i e^{ji\frac{2\pi}{T}t}, \quad b(t) = \sum_{i=-\infty}^{\infty} B_i e^{ji\frac{2\pi}{T}t}$

$$H_{HB} = \Omega Q(X) + F(X) - B = 0 \qquad \mathbf{\Omega} = \begin{pmatrix} -Mj\omega_0 & & \\ & \ddots & \\ & & Mj\omega_0 \end{pmatrix}, \omega_0 = \frac{2\pi}{T}$$

Expanding the frequency-domain equations using the discrete Fourier transform matrices, D and D^{-1}

$$\mathbf{H_{HB}} = \Omega D \mathbb{Q}(.) D^{-1} X + D \mathbb{F}(.) D^{-1} X - B$$

$$\mathbf{J_{HB}} = \Omega D \mathbb{C} D^{-1} + D \mathbb{G} D^{-1},$$

$$\mathbb{C} = \begin{pmatrix} C(t_0) & & \\ & \ddots & \\ & & C(t_{2M}) \end{pmatrix}, C(t_i) = \frac{dq}{dx}|_{x(t_i)} \qquad \mathbb{G} = \begin{pmatrix} G(t_0) & & \\ & \ddots & \\ & & G(t_{2M}) \end{pmatrix}, G(t_i) = \frac{df}{dx}|_{x(t_i)}$$

Linear Solver Improvements (2017)

Harmonic Balance linear solvers for the block-structured Jacobian matrix are challenging to develop

- Large matrix (n x N) with block structure, complex-valued, dense blocks
- Iterative methods are most often used because they avoid Jacobian matrix construction
  - For highly-nonlinear circuits effective preconditioners are difficult to construct
  - Preconditioners for linear to mildly-nonlinear circuits have been implemented in Xyce

A direct solver was developed using the templated Basker direct solver in Amesos2

- Enables the use of “block matrices” as a scalar type
- Motivated by the Berkeley Design Automation (BDA) direct solver[4]
- **New observation:** Dense blocks are created when voltage nodes are connected to nonlinear devices, all other nodes create diagonal blocks

Looking forward, this new solver provides a jumping off point

- Multi-threaded improvements to current Basker HB solver
- Use Basker HB solver on each node in parallel
- Algorithmic interpolation between Basker and block-Jacobi

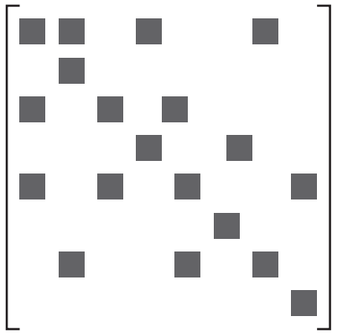
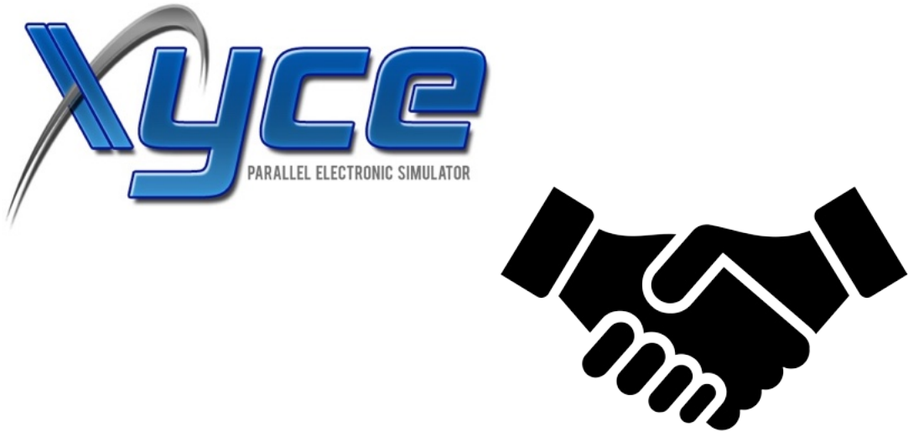


Figure 1: Sparsity structure of HB Jacobian

[4] “A robust and efficient harmonic balance (HB) using direct solution of HB Jacobian,” Mehrotra, Amit and Somani, Abhishek, DAC 2009.



Thanks to the Xyce and Trilinos teams!

Collaborators: Eric Keiter, Ting Mei, Siva Rajamanickam, Ichitaro Yamazaki, Nathan Ellingwood, David Day, Clark Dohrmann, ...