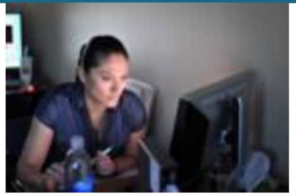
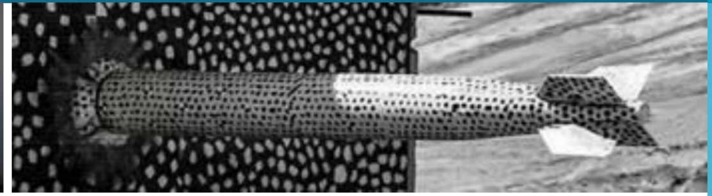
SAND2019-4818PE

Brief Gemma Overview

PRESENTED BY

William L. Langston

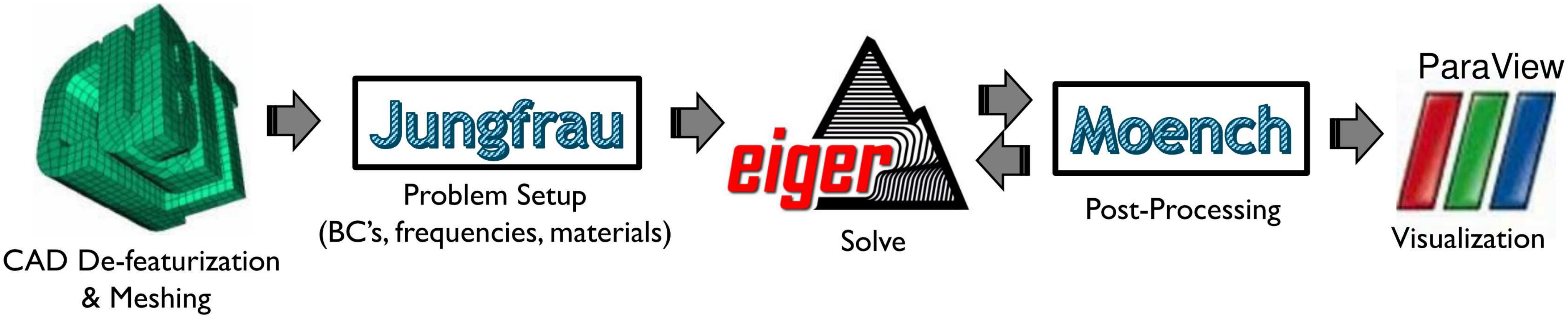
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

EIGER Overview

- **E**lectromagnetics **I**nteractions **G**en**ER**alized – **EIGER**
- Initially developed collaboratively by SNL & University of Houston (Bill Johnson, Roy Jorgenson, Don Wilton)
- Massively parallel (MPI), three-dimensional
- Frequency-domain, boundary-element, Method of Moments
- Custom feature set for Sandia use cases
  - Ability to rigorously model very high-quality-factor (Q) cavities
  - Ability to handle non-ideal (dissipative) materials
  - Wire and slot subcell models for high-fidelity coupling calculations

EM Coupling Simulations Using EIGER—Workflow

- Pre-processing:
  - Significant level of effort required in preparing the model and mesh
  - Mechanical joints are key features for calculating EM leakage into a system
  - The leakage is often dominated by the electrical contacts at the mechanical joint
  - *Resolving the mesh to capture the geometrical details and electrical lengths is critical*



CAD De-featurization & Meshing

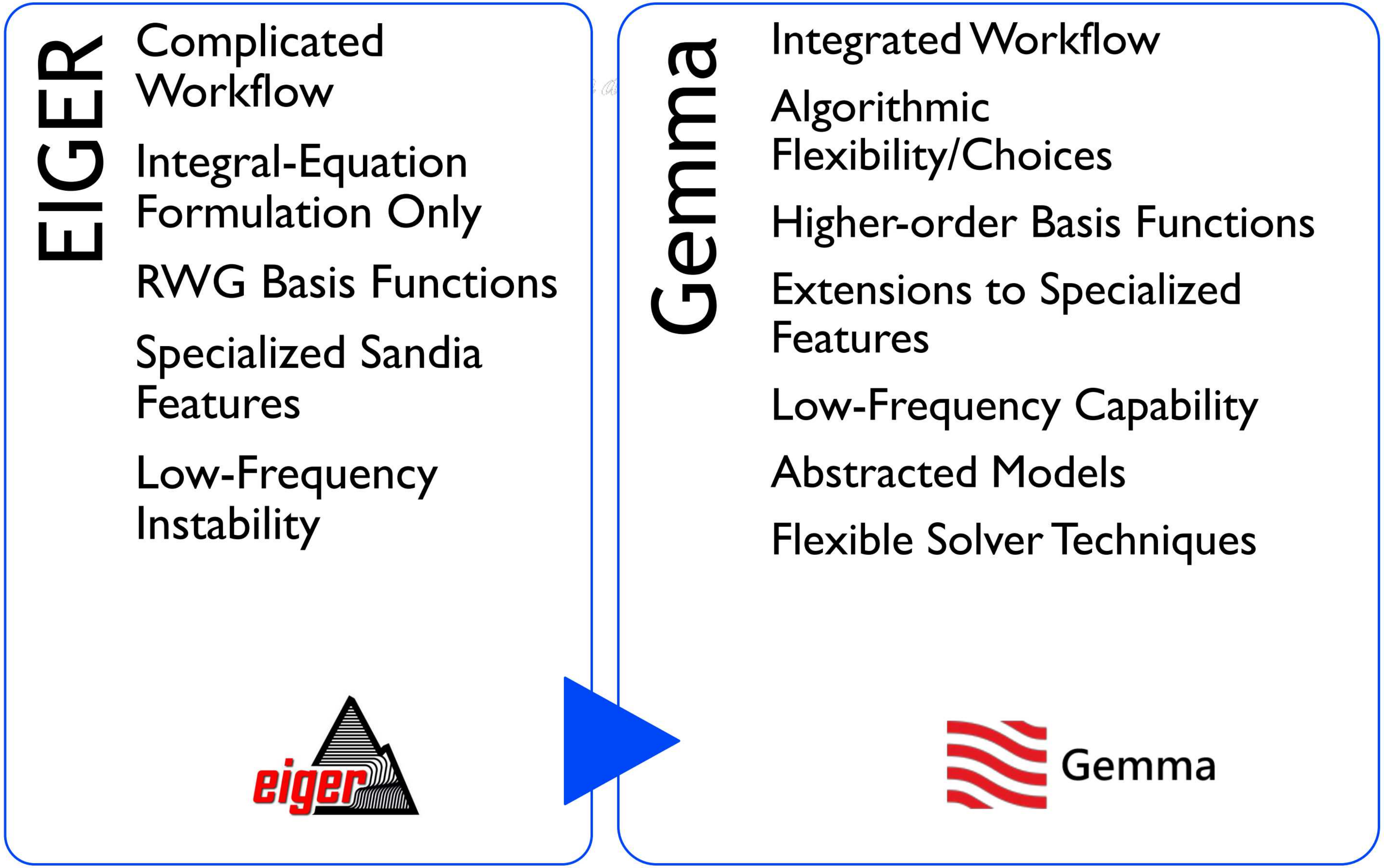
Problem Setup (BC's, frequencies, materials)

Solve

Post-Processing

Visualization

Gemma: A Next-Generation EM Code



EIGER

Complicated Workflow

Integral-Equation Formulation Only

RWG Basis Functions

Specialized Sandia Features

Low-Frequency Instability

Gemma

Integrated Workflow

Algorithmic Flexibility/Choices

Higher-order Basis Functions

Extensions to Specialized Features

Low-Frequency Capability

Abstracted Models

Flexible Solver Techniques

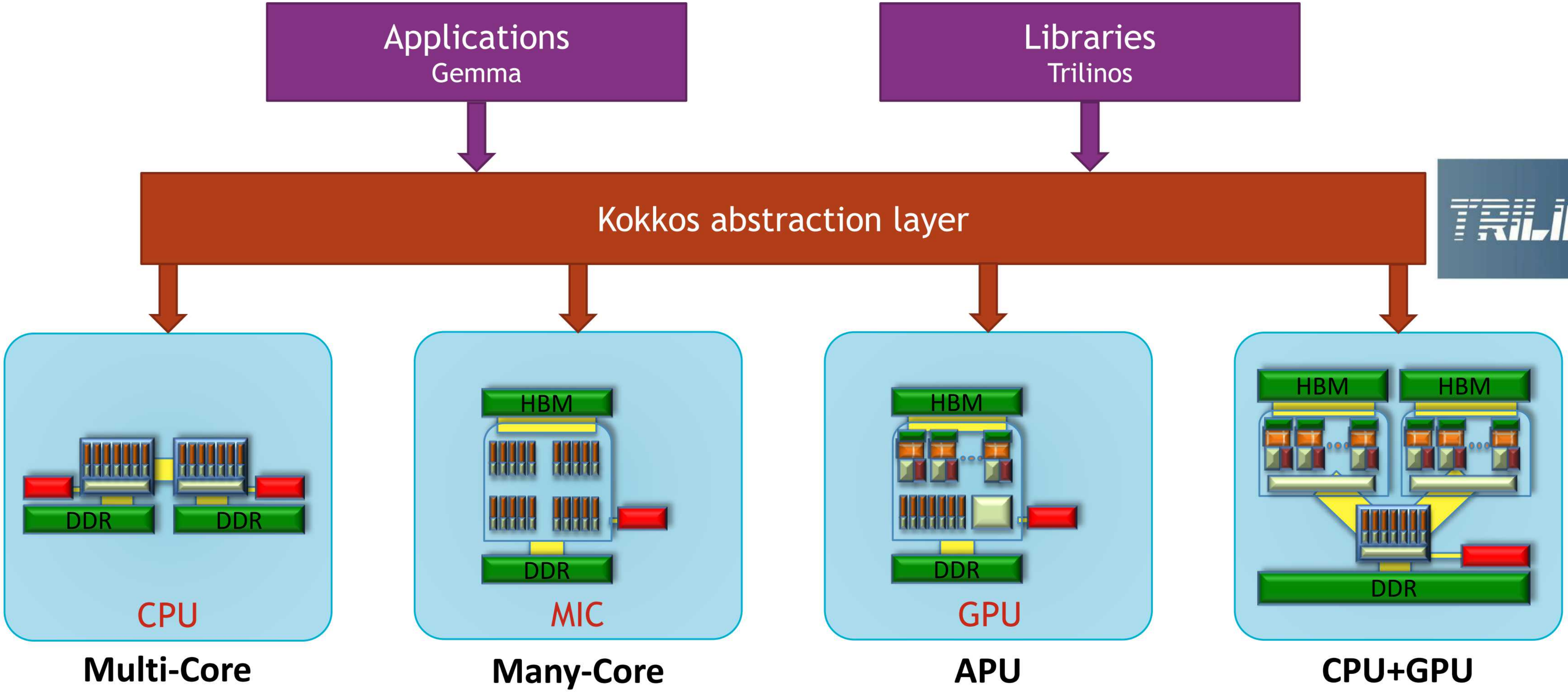
Gemma Goals

Gemma

- **Modern development environment**
  - C++ 11 (driving toward newer C++ standards)
  - git + cmake environment supporting multiple target compilers and architectures
  - Templated implementation for code readability and maintenance
  - Scrum process enabling increased communication and transparency among team members

- **Integrated, flexible, extensible physics**
  - Frequency-domain, full-wave EM method-of-moments simulation code (core EM functionality)
  - Alternative methods for flexibility and extensibility
  - Capability to couple EM to other domains, such as mechanical, thermal, and circuits

Kokkos: Code Development for Heterogeneous Compute Environments



Kokkos is the cornerstone for performance portability across next generation HPC architectures at multiple DOE laboratories and other organizations.

Gemma: Capability and Performance on Next-Generation Hardware

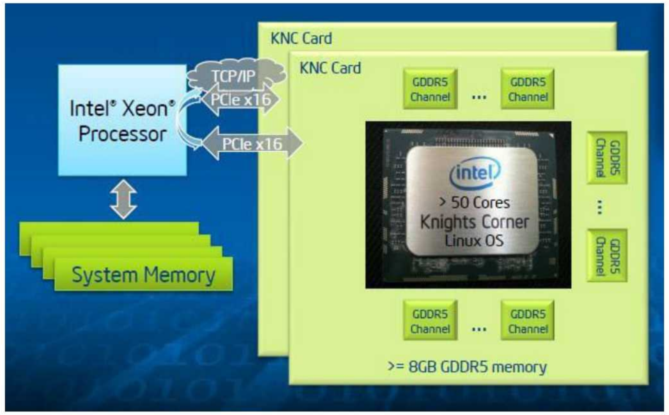
FORTRAN
MPI Paradigm
(EIGER)

- MPI inter- and intranode parallelism
- High processor clock speed
- High memory per processor

Heterogenous
Paradigm
(Gemma)

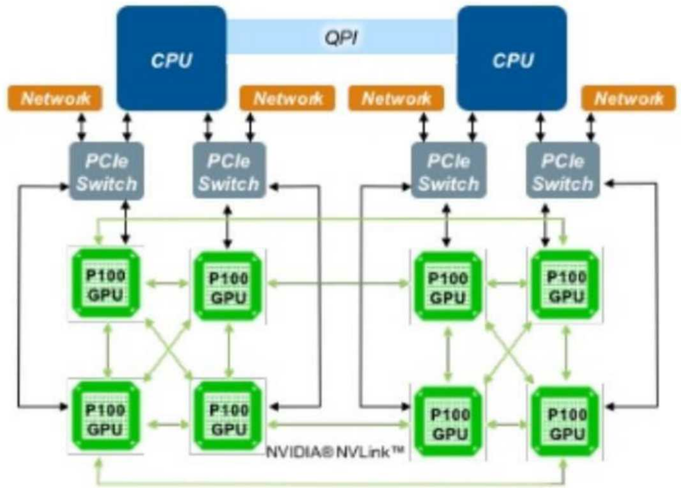
- MPI internode parallelism
- Threading intranode parallelism
- Low processor clock speed
- Low memory per processor

A Mix of CPU & MIC nodes



- ❑ Ideal Development: Gemma
  - Writing architecture independent source code
  - Using multicore technology efficiently

CPU & GPU on a single node



Gemma: Early Activities (FY18)

Gemma

FY18

- Integral-Equation (IE) Implementation
- CPU/GPU Performance
- Team Study and OSU Collaboration on Alternative Algorithms
- Multiphysics Coupling: Circuits, Mechanical, etc.
- Embedded Model Abstractions

Motivation:

- Retain EIGER core capability
- Extend algorithmic capability for flexibility
- Allow for breadth in analysis needs (quick to highly rigorous sims)

Gemma: Present Activities (FY19)

Gemma

FY19

- Identify Plan Forward on Gemma UI/UX
- MPI Implementation and Demo on CPU/GPU/MIC
- First Demo of Representative Coupling Simulation
- Implement time history of unit test coverage/regression test timing
- Iterative Physical Optics (IPO) Implementation and Demo

Motivation:

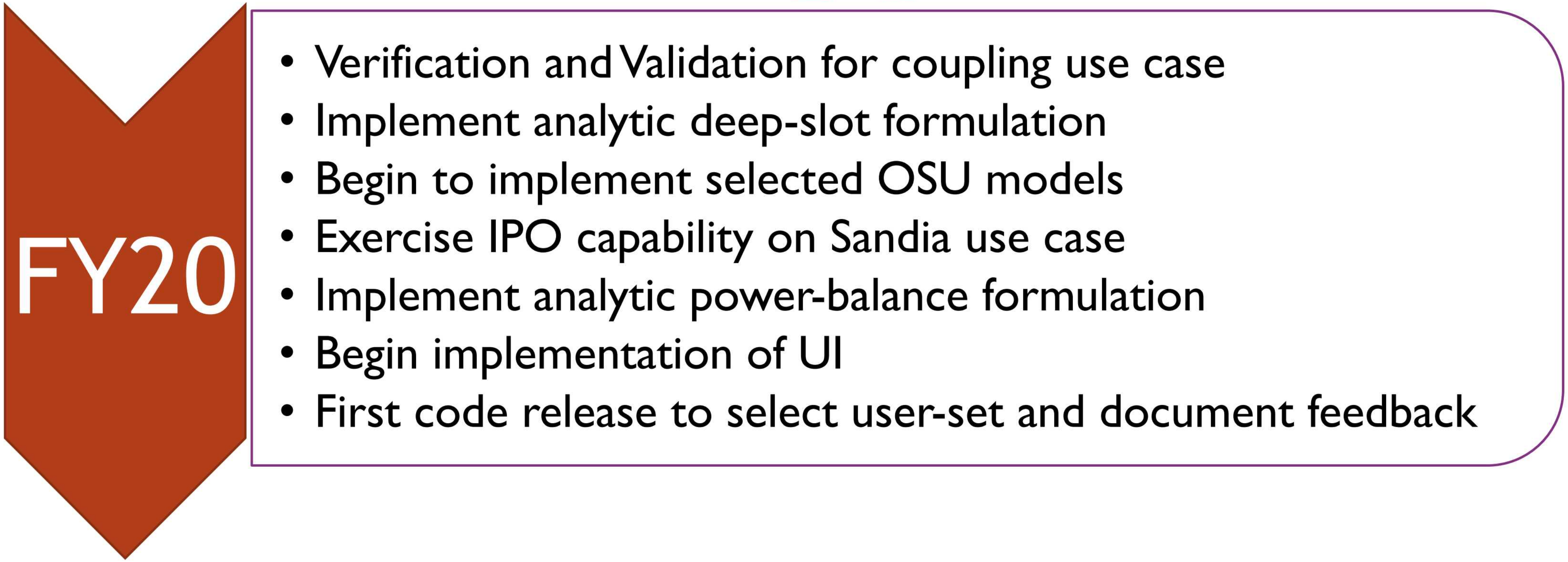
- Establish physics capabilities for Sandia use cases
- Establish V&V early in code lifecycle
- Enable parallel simulations across multiple nodes/GPUs
- Demonstrate candidate alternative modeling methods

Gemma: FY19 Recent Progress

Gemma

- Initial implementations for multiple regions and dielectrics have been completed paving the way for initial coupling simulations.
- MPI version of Gemma being developed and close to a working version for CPU and MIC (Pliris direct solver does not support GPUs yet).
- Initial IPO code implementation is complete. It is now being tested to verify correct results and limitations.
- Integrated Workflow (IWF) is proceeding with EIGER testing for both serial and parallel simulations. This activity will inform the Gemma UI/UX development.

Gemma: FY20 Roadmap



FY20

- Verification and Validation for coupling use case
- Implement analytic deep-slot formulation
- Begin to implement selected OSU models
- Exercise IPO capability on Sandia use case
- Implement analytic power-balance formulation
- Begin implementation of UI
- First code release to select user-set and document feedback

Backup Slides

Maxwell's Equations in the Frequency Domain

$\exp(j\omega t)$ time harmonic convention

$$\text{Faraday}: \quad \nabla \times \mathbf{E} = -j\omega\mathbf{B}$$

$$\text{Ampere} - \text{Maxwell}: \quad \nabla \times \mathbf{H} = \mathbf{J} + j\omega\mathbf{D}$$

$$\text{Electric Gauss}: \quad \nabla \cdot \mathbf{D} = \rho$$

$$\text{Magnetic Gauss}: \quad \nabla \cdot \mathbf{B} = 0$$

$\mathbf{E}$: electric field [V/m]

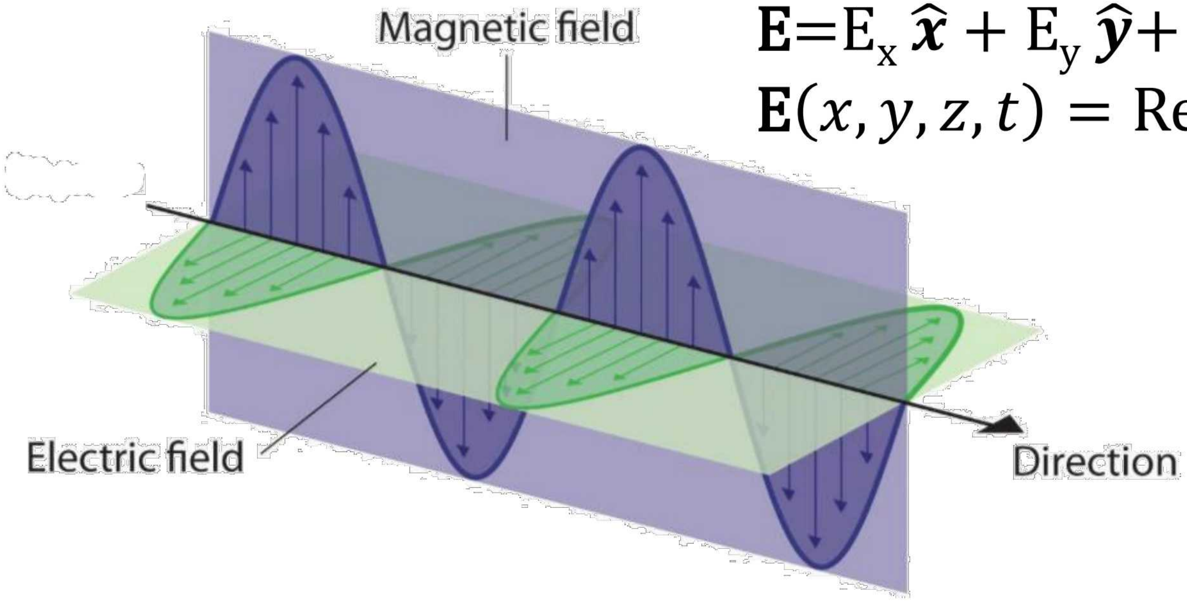
$\mathbf{H}$: magnetic field [A/m]

$\mathbf{D}$: electric flux density [C/m^2]

$\mathbf{B}$: magnetic flux density [T = Wb/m^2]

$\mathbf{J}$: electric current density [A/m^2]

ρ : volume charge density [C/m^3]



$\mathbf{E}$=E$_x$ $\hat{\boldsymbol{x}}$ + E$_y$ $\hat{\boldsymbol{y}}$+ E$_z$ $\hat{\boldsymbol{z}}$

$\mathbf{E}(x,y,z,t) = \mathrm{Re}\{e^{j\omega t}\tilde{E}\}$

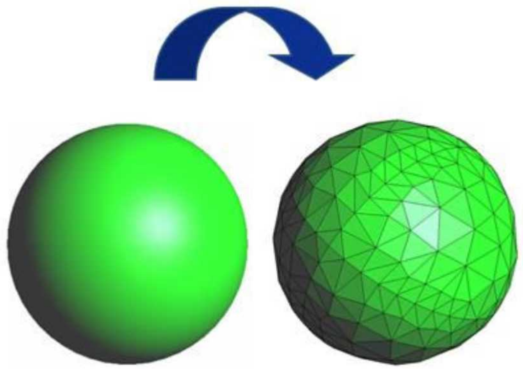
Constitutive relations:

$\mathbf{D} = \epsilon\mathbf{E} \qquad \epsilon$: permitivity [F/m]

$\mathbf{B} = \mu\mathbf{H} \qquad \mu$: permeability [H/m]

$\mathbf{J} = \sigma\mathbf{E} \qquad \sigma$: conductivity [S/m]

Integral Equation, Method of Moments



- Boundary Condition on Surface:

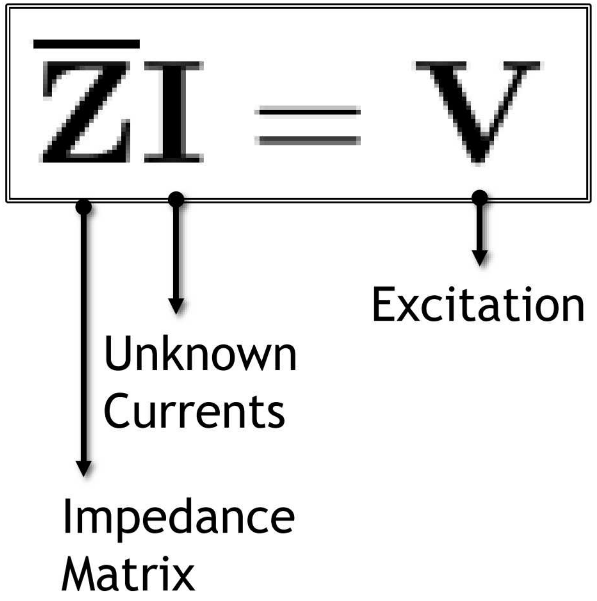
$$L(\bar{J}) = -\bar{E}^{scatt}_{tan} = \bar{E}^{inc}_{tan}$$

- The currents are expanded in terms of the Rao-Wilton-Glisson basis functions:

$$\bar{J}(r) = \sum_n I_n \bar{f}_n(r)$$

- Test the integral equation with the basis functions:

$$\langle \bar{f}_m, L(\bar{J})\rangle \Rightarrow \int_{surface} \bar{f}_m \cdot L(\bar{J})ds$$



$$\bar{Z}I = V$$

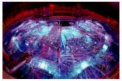
Impedance Matrix

Unknown Currents

Excitation

Sandia National Laboratories

Solution of Coupling Problems in Gemma



PRESENTED BY

Robert Pfeiffer

4/25/19

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

Overview

- Coupling problems
- Analysis techniques
  - Power Balance
  - Slot Model
  - Homogeneous Regions
    - Simulation using boundary element methods
    - PMCHWT and Müller formulations
- Status of coupling simulation in Gemma

Gemma: Next-Generation Electromagnetics Simulation Software

Gemma: Next-Generation Electromagnetics Simulation Software

- Successor to Sandia's EIGER (Electromagnetic Interactions GenERalized) code suite

Gemma: Next-Generation Electromagnetics Simulation Software

- Successor to Sandia's EIGER (Electromagnetic Interactions GenERalized) code suite
- Presently uses method of moments (MoM) with Rao-Wilton-Glisson (RWG) basis functions

Gemma: Next-Generation Electromagnetics Simulation Software

- Successor to Sandia's EIGER (Electromagnetic Interactions GenERalized) code suite
- Presently uses method of moments (MoM) with Rao-Wilton-Glisson (RWG) basis functions
- Use of Kokkos abstraction enables compilation for multiple different platforms

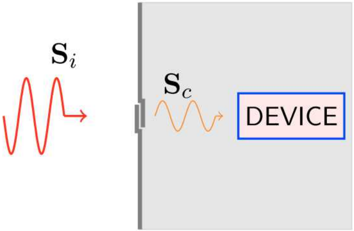
Gemma: Next-Generation Electromagnetics Simulation Software

- Successor to Sandia's EIGER (Electromagnetic Interactions GenERalized) code suite
- Presently uses method of moments (MoM) with Rao-Wilton-Glisson (RWG) basis functions
- Use of Kokkos abstraction enables compilation for multiple different platforms
- Currently being tested for OpenMP, MPI, and Cuda

Gemma: Next-Generation Electromagnetics Simulation Software

- Successor to Sandia's EIGER (Electromagnetic Interactions GenERalized) code suite
- Presently uses method of moments (MoM) with Rao-Wilton-Glisson (RWG) basis functions
- Use of Kokkos abstraction enables compilation for multiple different platforms
- Currently being tested for OpenMP, MPI, and Cuda
- Uses linear algebra algorithms from Sandia's Trilinos library

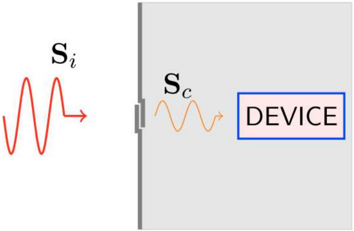
Coupling Simulations



Motivation:

- Devices in high-Q cavities vulnerable to EM interference coupled through small apertures
- Particularly at resonant frequencies
- Shielding effectiveness (SE) prediction becomes important design step

Coupling Simulations



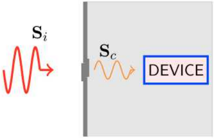
Motivation:

- Devices in high-Q cavities vulnerable to EM interference coupled through small apertures
- Particularly at resonant frequencies
- Shielding effectiveness (SE) prediction becomes important design step

Methods being investigated:

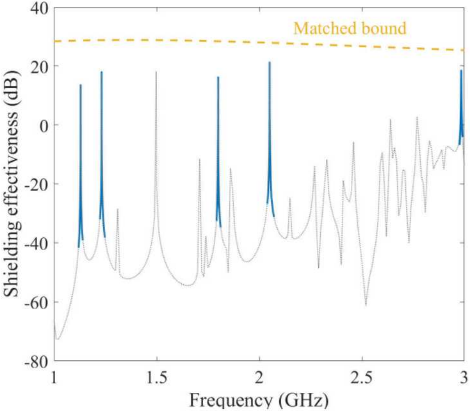
- Analytic
  - Power balance
- Full-Wave
  - PMCHWT formulation [J. Poggio and Miller, 1973]
  - Müller formulation [Müller, 1969]
  - Slot model

Coupling Problems: Bounding with Power Balance



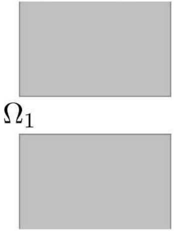
- Upper bound on shielding effectiveness obtained using statistical EM
- Transmitted power (P_t) equated to dissipated power (P_d)
- Shielding effectiveness calculated as ratio between power density of incident wave S_i and power density in cavity S_c [Hill, 2009]

$$\text{SE}(\mathbf{r}) = 10\log_{10}\left(\frac{S_c(\mathbf{r})}{S_i}\right)$$



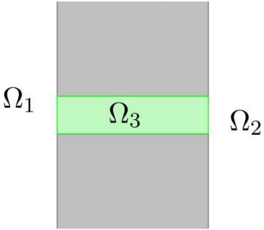
Coupling Problems: Full Wave Analysis

Single Region



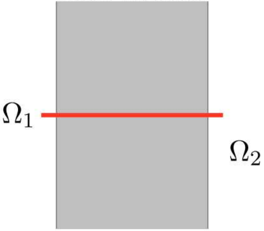
- Ω_1 includes inside, outside, gap
- Fails to capture coupling accurately

Slot Region



- Mesh slot volume as separate region
- Fewer interactions between inner/outer regions
- Still accurately capture slot geometry

Slot Model



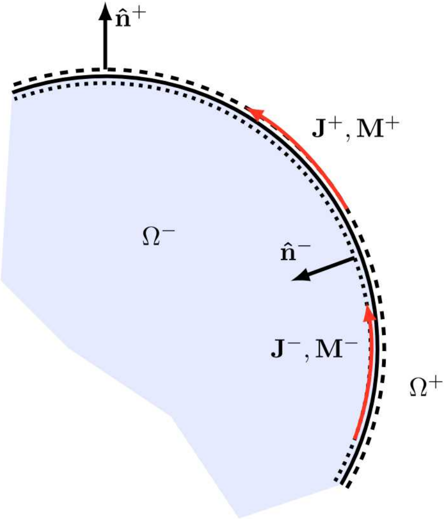
- Model slot as transmission line connecting Ω_1, Ω_2
- Same mesh can be used to test different slot parameters

BEM Solution of Multi-Region Problems

- Enforce tangential continuity of $\mathbf{E}$ and $\mathbf{H}$ on boundary
- Convenient to define $(\mathbf{J}^+, \mathbf{M}^+) = -(\mathbf{J}^-, \mathbf{M}^-)$
- Yields combined field formulation

$$\begin{aligned} \mathsf{EFIE}^+ + \alpha \mathsf{EFIE}^- \\ \mathsf{MFIE}^+ + \beta \mathsf{MFIE}^- \end{aligned} \tag{1}$$

- Choice of α, β affects accuracy, conditioning
  - $\alpha = \beta = 1$ gives PMCHWT formulation
  - $\alpha = \frac{\epsilon_r^-}{\epsilon_r^+}, \beta = \frac{\mu_r^-}{\mu_r^+}$ gives Müller formulation

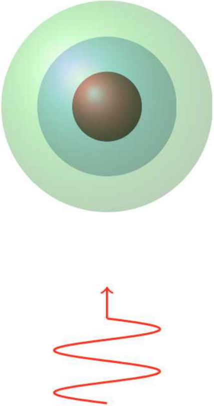


Reference Solutions

- Scattering from spherically symmetric objects described by Mie series[Harrington, 1961]

$$A_r = E_0 \frac{\cos\varphi}{\omega\mu} \sum_{n=1}^{\infty} \left(b_n \hat{H}_n^{(2)}(kr) + a_n \hat{J}_n \right) P_n^1(cos\theta)$$

$$F_r = E_0 \frac{\sin\varphi}{k} \sum_{n=1}^{\infty} \left(c_n \hat{H}_n^{(2)}(kr) + a_n \hat{J}_n \right) P_n^1(cos\theta) \tag{2}$$



- Applicable to
  - Conducting sphere
  - Homogeneous dielectric sphere
  - Piecewise homogeneous sphere with multiple dielectric/conducting layers

Comparison of PMCHWT and Müller Formulations

- PMCHWT
  - No $\frac{1}{2}\mathcal{I}$ term
  - Better accuracy with higher-contrast materials
  - Poorly conditioned

Comparison of PMCHWT and Müller Formulations

- PMCHWT
  - No $\frac{1}{2}\mathcal{I}$ term
  - Better accuracy with higher-contrast materials
  - Poorly conditioned
- Müller
  - Singularity reduction in $\mathcal{L}$
  - Better accuracy with low contrast materials
  - Well-conditioned

Comparison of PMCHWT and Müller Formulations

- PMCHWT
  - No $\frac{1}{2}\mathcal{I}$ term
  - Better accuracy with higher-contrast materials
  - Poorly conditioned
- Müller
  - Singularity reduction in $\mathcal{L}$
  - Better accuracy with low contrast materials
  - Well-conditioned

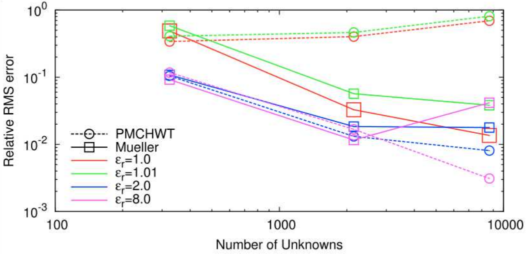


Figure: Near-field scattering of 3×10^7 Hz plane wave from homogeneous dielectric sphere characterized by ϵ_r.

Comparison of PMCHWT and Müller Formulations

- PMCHWT
  - No $\frac{1}{2}\mathcal{I}$ term
  - Better accuracy with higher-contrast materials
  - Poorly conditioned
- Müller
  - Singularity reduction in $\mathcal{L}$
  - Better accuracy with low contrast materials
  - Well-conditioned

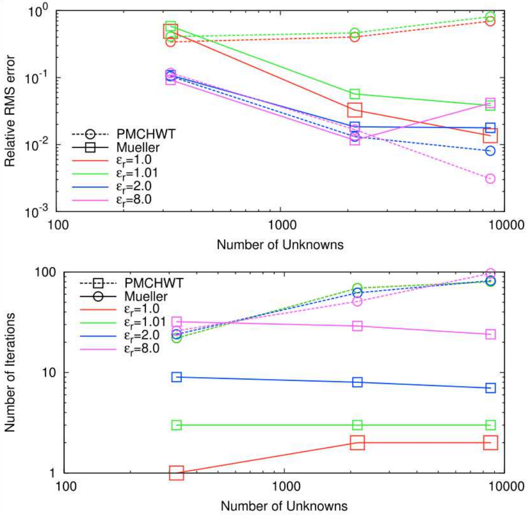
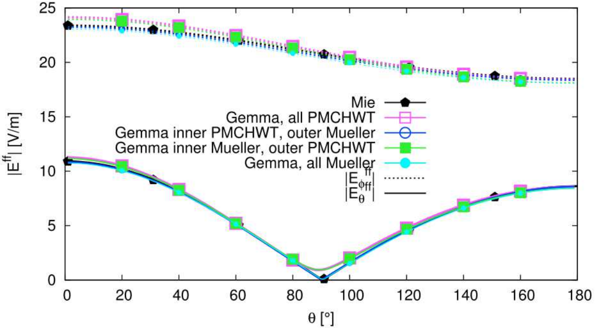
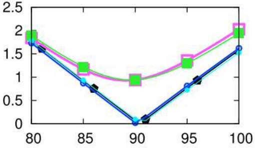
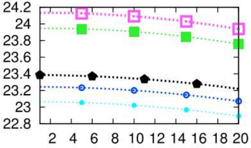


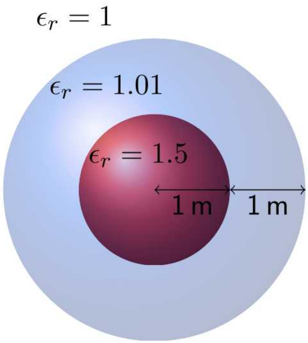
Figure: Near-field scattering of 3×10^7 Hz plane wave from homogeneous dielectric sphere characterized by ϵ_r.

Comparison of PMCHWT and Müller Formulations



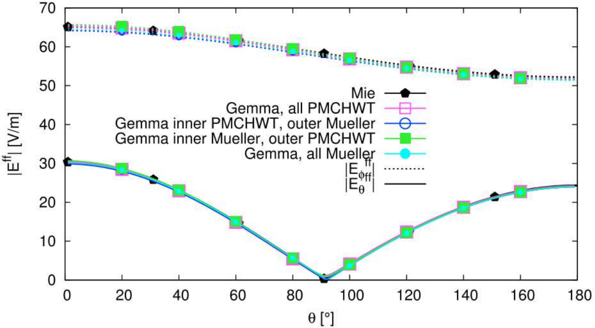


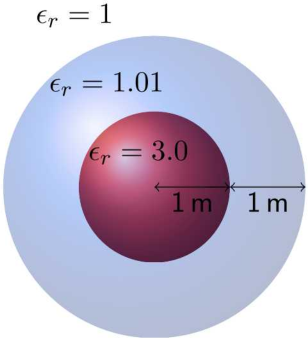


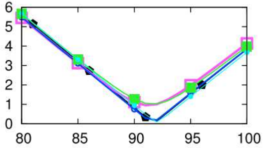


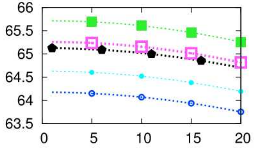
- $+z$-traveling wave, $\mathbf{E} = 377\,\mathrm{V\,m^{-1}}\hat{\mathbf{x}}$
- Frequency 30 MHz
- Far-field scattered $\mathbf{E}$ computed at $\varphi = 65^\circ$
- Sphere meshed with 3598 flat triangles

Comparison of PMCHWT and Müller Formulations



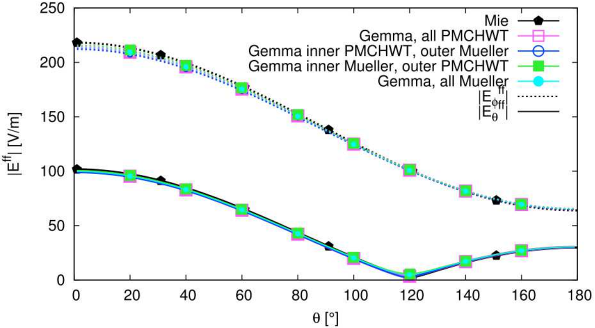


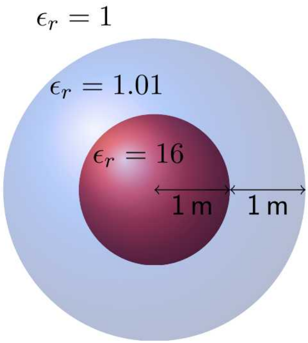


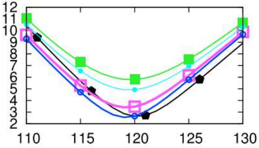


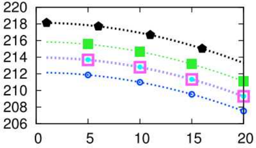
- $+z$-traveling wave, $\mathbf{E} = 377\,\mathrm{V\,m^{-1}}\hat{\mathbf{x}}$
- Frequency 30 MHz
- Far-field scattered $\mathbf{E}$ computed at $\varphi = 65^\circ$
- Sphere meshed with 3598 flat triangles

Comparison of PMCHWT and Müller Formulations



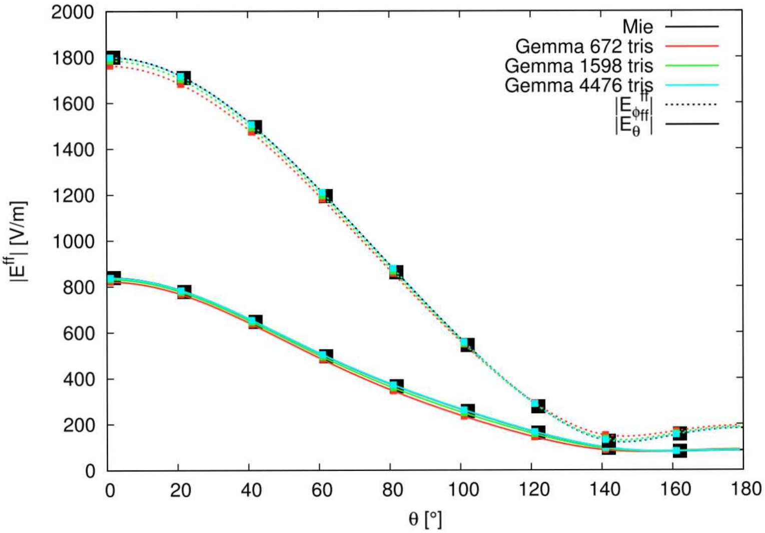


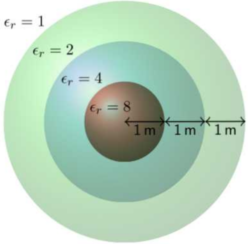




- $+z$-traveling wave, $\mathbf{E} = 377\,\mathrm{V\,m^{-1}}\,\hat{\mathbf{x}}$
- Frequency 30 MHz
- Far-field scattered $\mathbf{E}$ computed at $\varphi = 65^\circ$
- Sphere meshed with 3598 flat triangles

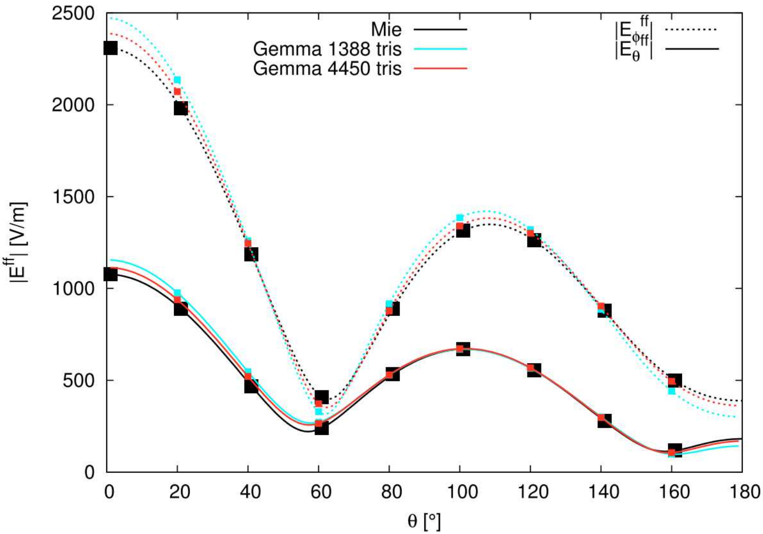
Gemma Multi-Region Results: 3-layer Sphere

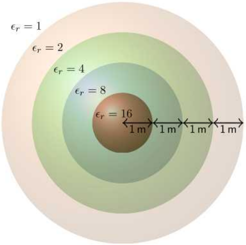




- $+z$-traveling wave, $\mathbf{E} = 377\,\mathrm{V\,m^{-1}}\hat{\mathbf{x}}$
- Frequency 30 MHz
- Far-field scattered $\mathbf{E}$ computed at $\varphi = 65^\circ$

Gemma Multi-Region Results: 4-layer Sphere





- $+z$-traveling wave, $\mathbf{E} = 377\,\mathrm{V\,m^{-1}}\hat{\mathbf{x}}$
- Frequency 30 MHz
- Far-field scattered $\mathbf{E}$ computed at $\varphi = 65^\circ$

Coupling Problems in Gemma

- At present Gemma solves dielectric problems with non-intersecting boundaries

Coupling Problems in `Gemma`

- At present `Gemma` solves dielectric problems with non-intersecting boundaries
- Correct solution obtained for 1-5 region problems

Coupling Problems in Gemma

- At present Gemma solves dielectric problems with non-intersecting boundaries
- Correct solution obtained for 1-5 region problems
- Coupling problems contain junctions

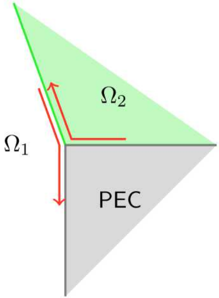


Figure : Multi-Region Junction with closed PEC. Currents represented by arrows correspond to same DOF.

Coupling Problems in Gemma

- At present Gemma solves dielectric problems with non-intersecting boundaries
- Correct solution obtained for 1-5 region problems
- Coupling problems contain junctions
- PMCHWT and Müeller require specific handling of different junction types [Ylä-Oijala et al., 2005]

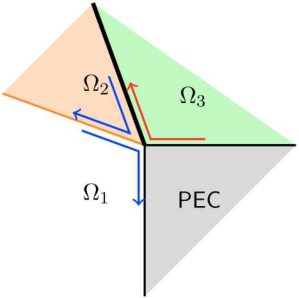


Figure : Multi-Region Junction with thin PEC (black). Different color arrows correspond to different DOF. (cf. [Ylä-Oijala et al., 2005, Fig. 6]

Current Status

- Done
  - PMCHWT and Müller formulations implemented on GPU
  - Multiple-region logic tested for 2-3 regions
- Remaining
  - Address MFIE integration accuracy
  - Generalize for arbitrary number and arrangement of regions (incl. junctions)
  - Incorporate slot model into `Gemma` and compare to meshed slot

References

Harrington, R. F. (1961).
Time-Harmonic Electromagnetic Fields.
McGraw-Hill.

Hill, D. A. (2009).
Electromagnetic Fields in Cavities, Deterministic and Statistical Theories.
Wiley & Sons, Inc., New Jersey.

J. Poggio, A. and Miller, E. (1973).
Integral equation solutions of three-dimensional scattering problems.

Müller, C. (1969).
Foundations of the Mathematical Theory of Electromagnetic Waves.
Springer-Verlag, Berlin.

Warne, L. K. and Chen, K. C. (1990).
Slot apertures having depth and losses described by local transmission line theory.
IEEE Transactions on Electromagnetic Compatibility, 32(3):185–196.

Ylä-Oijala, P. and Taskinen, M. (2005).
Well-conditioned müller formulation for electromagnetic scattering by dielectric objects.
IEEE Transactions on Antennas and Propagation, 53(10):3316–3323.

Ylä-Oijala, P., Taskinen, M., and Sarvas, J. (2005).
Surface integral equation method for general composite metallic and dielectric structures with junctions.
Progress in Electromagnetics Research, 52:81–108.

Slot Model

Slot geometry can be replaced by transmission line [Warne and Chen, 1990]

$$H_z^{>}(\rho_0^-, z) + \frac{1}{4Z_{\rho_0}} \frac{d^2}{dz^2} Im - Y_{\rho_0} I_m/4 = -H_z^{inc} \qquad (3)$$

$$H_z^{>}(\rho_0^-, z) = \frac{i}{\omega\mu_0} \left(\frac{d^2}{dz^2} + k^2 \right) \int_{-h}^{h} \frac{e^{ikR_0}}{4\pi R_0} I_m(z')dz' \qquad (4)$$

EM Boundary Element Equations for Homogeneous Regions

The equivalence principle [Harrington, 1961] gives us the integral equations on the surface S bounding region Ω_n [Ylä-Oijala and Taskinen, 2005]:

- EFIE$_n$:

$$\mathbf{T}\cdot\left[c_n\mathcal{L}_n(\mathbf{J}_n) + \mathcal{K}_n(\mathbf{M}_n)\right] - \frac{1}{2}(\hat{\mathbf{n}} \times \mathbf{T})\cdot\mathbf{M}_n = \mathbf{T}\cdot\mathbf{E}_n^{\mathsf{inc}} \quad (5)$$

- MFIE$_n$:

$$\mathbf{T}\cdot\left[d_n\mathcal{L}_n(\mathbf{M}_n) - \mathcal{K}_n(\mathbf{J}_n)\right] + \frac{1}{2}(\hat{\mathbf{n}} \times \mathbf{T})\cdot\mathbf{J}_n = \mathbf{T}\cdot\mathbf{H}_n^{\mathsf{inc}} \quad (6)$$

EM Boundary Element Equations for Homogeneous Regions

The equivalence principle [Harrington, 1961] gives us the integral equations on the surface S bounding region Ω_n [Ylä-Oijala and Taskinen, 2005]:

- EFIE$_n$:

$$\mathbf{T}\cdot\left[c_n\mathcal{L}_n(\mathbf{J}_n)+\mathcal{K}_n(\mathbf{M}_n)\right]-\frac{1}{2}(\hat{\mathbf{n}}\times\mathbf{T})\cdot\mathbf{M}_n=\mathbf{T}\cdot\mathbf{E}_n^{\text{inc}} \tag{5}$$

- MFIE$_n$:

$$\mathbf{T}\cdot\left[d_n\mathcal{L}_n(\mathbf{M}_n)-\mathcal{K}_n(\mathbf{J}_n)\right]+\frac{1}{2}(\hat{\mathbf{n}}\times\mathbf{T})\cdot\mathbf{J}_n=\mathbf{T}\cdot\mathbf{H}_n^{\text{inc}} \tag{6}$$

- Let $\mathbf{J}_2=-\mathbf{J}_1$ and $\mathbf{M}_2=-\mathbf{M}_1$. Solve combined system

$$\begin{gathered}\alpha_1\text{EFIE}_1-\alpha_2\text{EFIE}_2\\ \beta_1\text{MFIE}_1-\beta_2\text{MFIE}_2\end{gathered} \tag{7}$$

EM Boundary Element Equations for Homogeneous Regions

The equivalence principle [Harrington, 1961] gives us the integral equations on the surface S bounding region Ω_n [Ylä-Oijala and Taskinen, 2005]:

- EFIE$_n$:

$$\mathbf{T}\cdot\left[c_n\mathcal{L}_n(\mathbf{J}_n)+\mathcal{K}_n(\mathbf{M}_n)\right]-\frac{1}{2}(\hat{\mathbf{n}}\times\mathbf{T})\cdot\mathbf{M}_n=\mathbf{T}\cdot\mathbf{E}_n^{\text{inc}} \tag{5}$$

- MFIE$_n$:

$$\mathbf{T}\cdot\left[d_n\mathcal{L}_n(\mathbf{M}_n)-\mathcal{K}_n(\mathbf{J}_n)\right]+\frac{1}{2}(\hat{\mathbf{n}}\times\mathbf{T})\cdot\mathbf{J}_n=\mathbf{T}\cdot\mathbf{H}_n^{\text{inc}} \tag{6}$$

- Let $\mathbf{J}_2=-\mathbf{J}_1$ and $\mathbf{M}_2=-\mathbf{M}_1$. Solve combined system

$$\begin{aligned}\alpha_1\text{EFIE}_1-\alpha_2\text{EFIE}_2\\ \beta_1\text{MFIE}_1-\beta_2\text{MFIE}_2\end{aligned} \tag{7}$$

- Galerkin PMCHWT formulation:
  $\alpha_1=\alpha_2=\beta_1=\beta_2=1$, $\mathbf{T}_n=\mathbf{B}_n$

EM Boundary Element Equations for Homogeneous Regions

The equivalence principle [Harrington, 1961] gives us the integral equations on the surface S bounding region Ω_n [Ylä-Oijala and Taskinen, 2005]:

- EFIE$_n$:

$$\mathbf{T}\cdot\left[c_n\mathcal{L}_n(\mathbf{J}_n)+\mathcal{K}_n(\mathbf{M}_n)\right]-\frac{1}{2}(\hat{\mathbf{n}}\times\mathbf{T})\cdot\mathbf{M}_n=\mathbf{T}\cdot\mathbf{E}_n^{\text{inc}} \tag{5}$$

- MFIE$_n$:

$$\mathbf{T}\cdot\left[d_n\mathcal{L}_n(\mathbf{M}_n)-\mathcal{K}_n(\mathbf{J}_n)\right]+\frac{1}{2}(\hat{\mathbf{n}}\times\mathbf{T})\cdot\mathbf{J}_n=\mathbf{T}\cdot\mathbf{H}_n^{\text{inc}} \tag{6}$$

- Let $\mathbf{J}_2=-\mathbf{J}_1$ and $\mathbf{M}_2=-\mathbf{M}_1$. Solve combined system

$$\begin{aligned}\alpha_1\text{EFIE}_1-\alpha_2\text{EFIE}_2\\ \beta_1\text{MFIE}_1-\beta_2\text{MFIE}_2\end{aligned} \tag{7}$$

- Galerkin PMCHWT formulation:
  $\alpha_1=\alpha_2=\beta_1=\beta_2=1$, $\mathbf{T}_n=\mathbf{B}_n$
- N-Müller formulation:
  $\alpha_1=\epsilon_{r1},\alpha_2=\epsilon_{r2},\beta_1=\mu_{r1},\beta_2=\mu_{r2}$, $\mathbf{T}_n=\mathbf{B}_n\times\hat{\mathbf{n}}_n$

Rational Interpolation

- Frequency sweeps for complex/electrically large cavities expensive.
- Rational interpolation algorithm provided by OSU enables accurate and efficient resolution of SE peaks

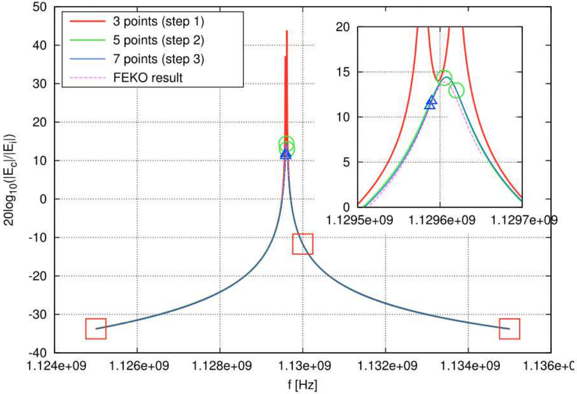
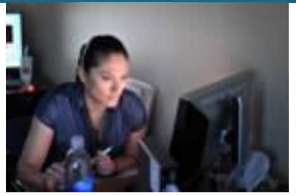
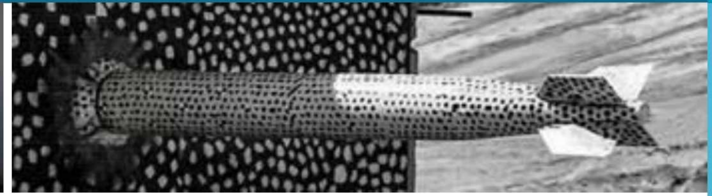


Figure : SE plot for cylinder using `Eiger` with rational interpolation algorithm. Comparison with Altair `FEKO` sweep across 46 uniformly spaced points.

Efforts toward Next Generation Platforms: Kokkos and Kokkos Kernels





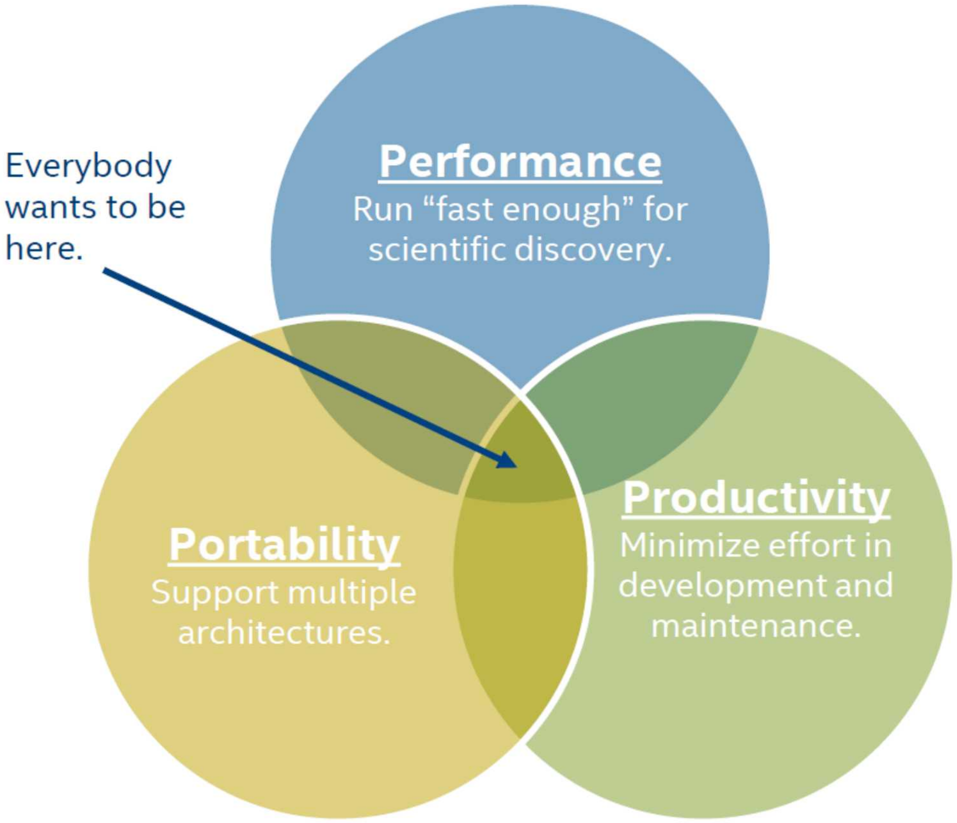
PRESENTED BY

Vinh Dang

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

Performance Portability

Minimizing Effort for Running Fast on Desired Architectures


Everybody wants to be here.
Performance
Run "fast enough" for scientific discovery.
Portability
Support multiple architectures.
Productivity
Minimize effort in development and maintenance.

- **Performance / Productivity**
  Enable domain scientists to write high-performance codes with minimal tuning
- **Performance / Portability**
  Enable applications to run at different facilities, on different machine types
- **Portability / Productivity**
  Enable developers to program in one shared language/programming model

John Pennycook, Charlene Yang, and Jack Deslippe, "Quantitatively Assessing Performance Portability with Roofline", NeRSC. https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar025

Performance Portability Defined

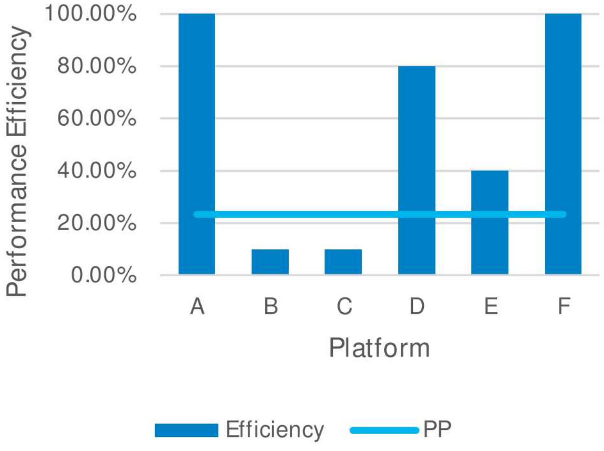
$$\Psi(a,p,H) = \begin{cases} \dfrac{|H|}{\sum_{i \in H} \dfrac{1}{e_i(a,p)}} & \text{if } i \text{ is supported } \forall i \in H \\ 0 & \text{otherwise} \end{cases}$$

$$e_i(a,p) = \frac{P_i(a,p)}{\min(F_i,\ B_i \times I_i(a,p))}$$

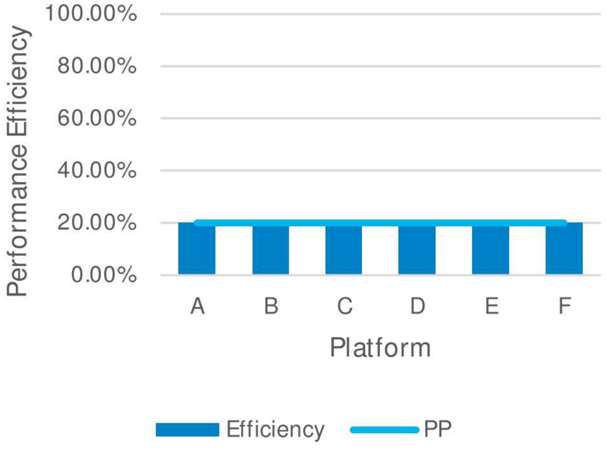
$e_i(a,p)$ = efficiency of application a for input problem p.

"The **harmonic mean** of an application's performance efficiency on a set of platforms for a given problem."

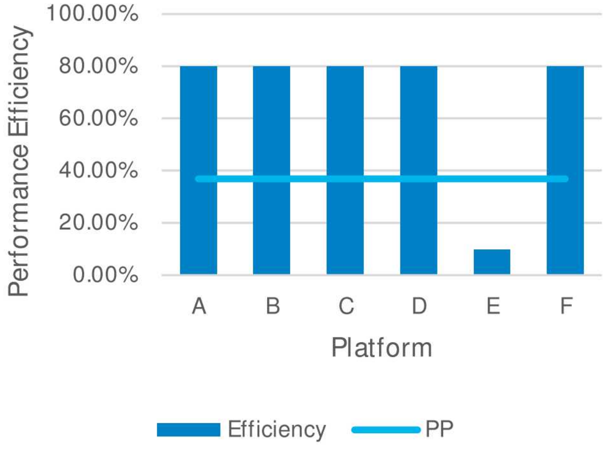
Application 1 – PP(*a,p,H*) = **23.30%**



Application 2 – PP(*a,p,H*) = **20.00%**



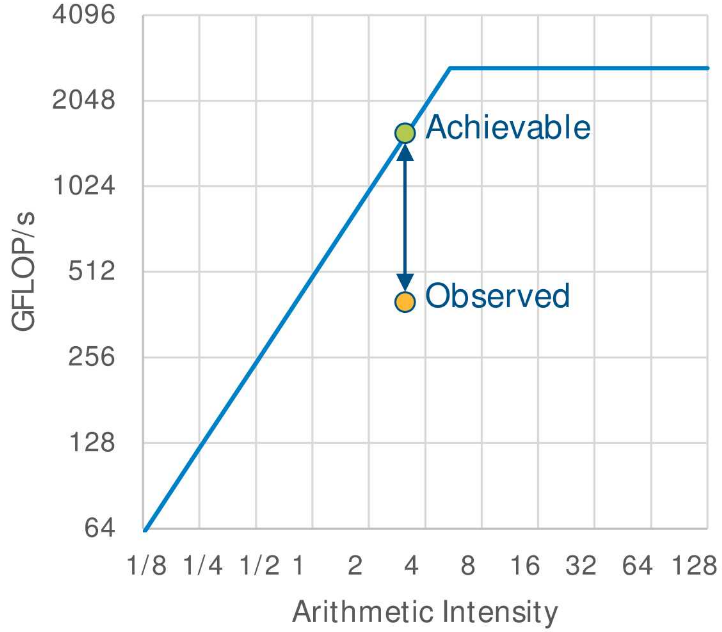
Application 3 – PP(*a,p,H*) = **36.92%**



John Pennycook, Charlene Yang, and Jack Deslippe, "Quantitatively Assessing Performance Portability with Roofline", NeRSC. https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar025

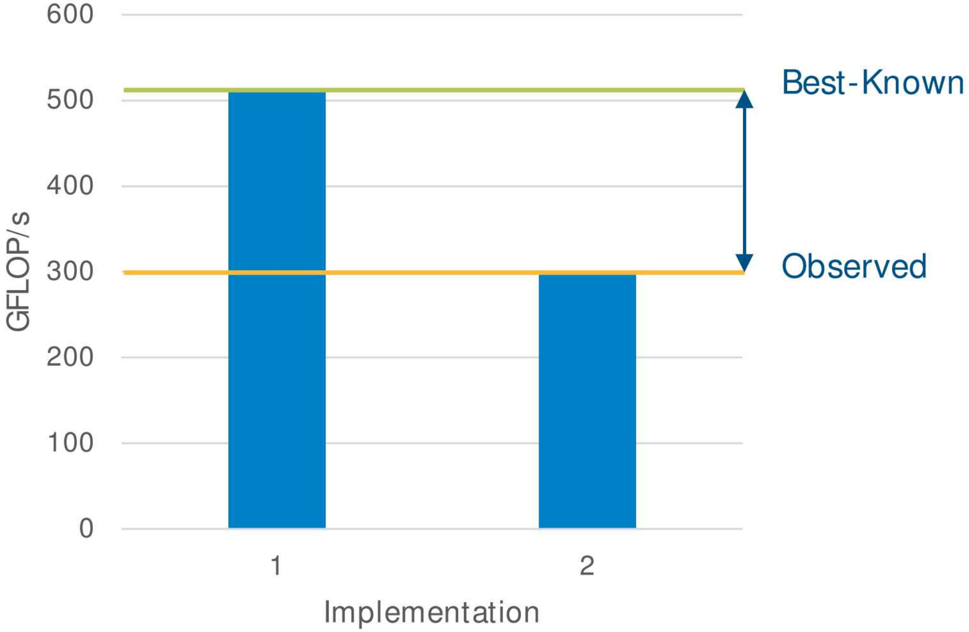
Architectural vs Application Efficiency

Architectural Efficiency



Represents how well an application utilizes each platform's resources

Application Efficiency

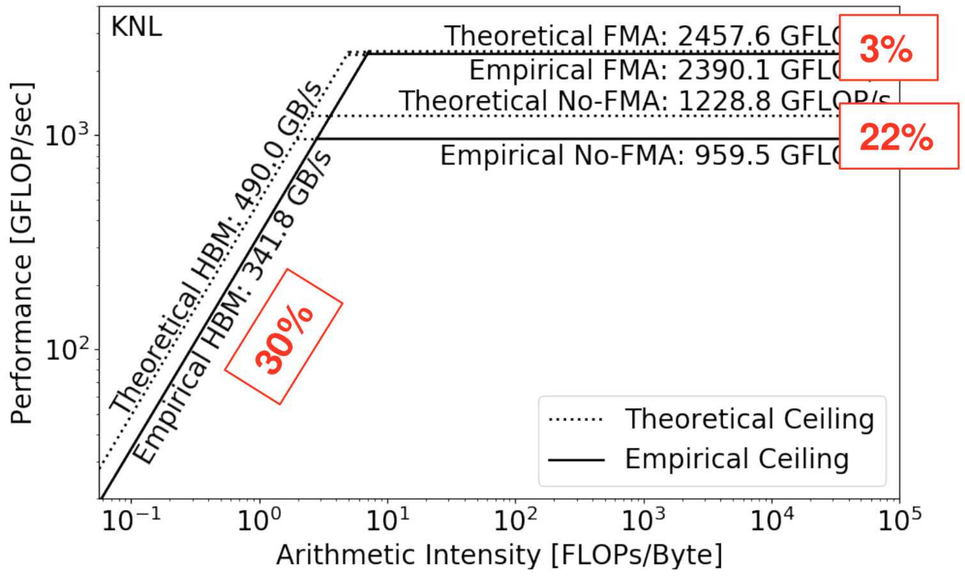


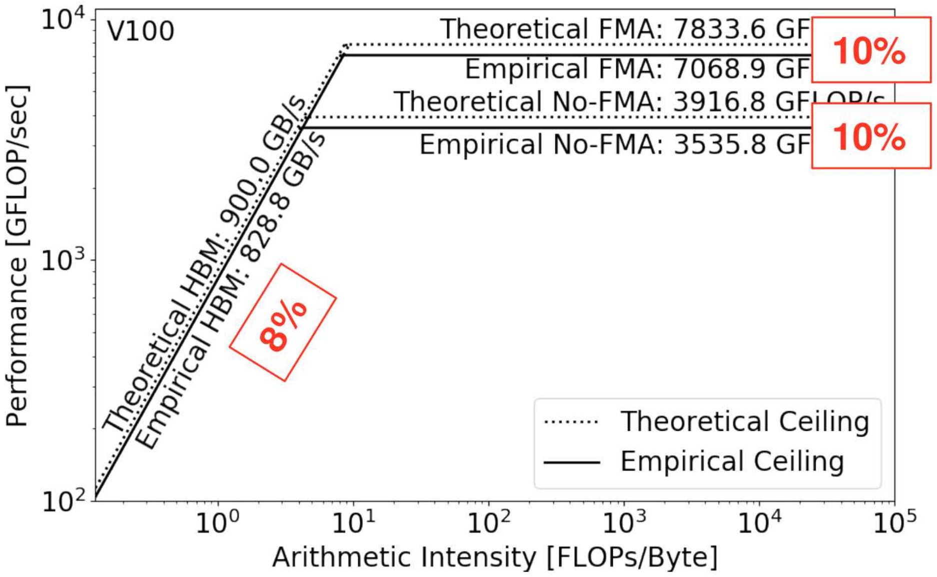
Represents whether an application uses appropriate algorithms on each platform

John Pennycook, Charlene Yang, and Jack Deslippe, "Quantitatively Assessing Performance Portability with Roofline", NeRSC. https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar025

Roofline Model

Theoretical and Empirical Rooflines for KNLs and V100s





$$\text{KNL FMA: } \frac{2390.1}{341.8} \text{ FLOPs/byte} \approx 7 \text{ FLOPs/byte}$$

$$\text{V100 FMA: } \frac{7068.9}{828.8} \text{ FLOPs/byte} \approx 8.5 \text{ FLOPs/byte}$$

$$\text{KNL no FMA: } \frac{959.5}{341.8} \text{ FLOPs/byte} \approx 2.8 \text{ FLOPs/byte}$$

$$\text{V100 no FMA: } \frac{3535.8}{828.8} \text{ FLOPs/byte} \approx 4.3 \text{ FLOPs/byte}$$

John Pennycook, Charlene Yang, and Jack Deslippe, “Quantitatively Assessing Performance Portability with Roofline”, NeRSC. https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar025

Required Arithmetic Intensities for Using Main Memory

- Haswell CPU
  - Theoretical bandwidth from system memory: 68 GB/s
  - Theoretical max FMA FLOPs: 185.6 GFLOPs/s
  - Required arithmetic intensity to load from system memory: 2.7 FLOPs/Byte
- V100
  - Empirical bandwidth from global memory: 828.8 GB/s
  - Empirical max FMA FLOPs: 7068.9 GFLOPs/s
  - Required arithmetic intensity to load from global memory: 8.5 FLOPs/Byte
- KNL
  - Empirical bandwidth from high bandwidth memory: 341.8 GB/s
  - Empirical max FMA FLOPs: 2390.1 GFLOPs/s
  - Required arithmetic intensity to load from high bandwidth memory: 7 FLOPs/Byte

Intel, *Intel Xeon Processor E5-2698 v3. https://ark.intel.com/content/www/us/en/ark/products/81060/intel-xeon-processor-e5-2698-v3-40m-cache-2-30-ghz.html*

Intel, *Export Compliance Metrics for Intel Microprocessors Intel Xeown Processors.* https://www.intel.com/content/dam/support/us/en/documents/processors/APP-for-Intel-Xeon-Processors.pdf

John Pennycook, Charlene Yang, and Jack Deslippe, "Quantitatively Assessing Performance Portability with Roofline", NeRSC. https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar025

Kokkos and Kokkos Kernels

Kokkos Overview

Kokkos is a **productive**, **portable**, **performant**, shared-memory programming model.

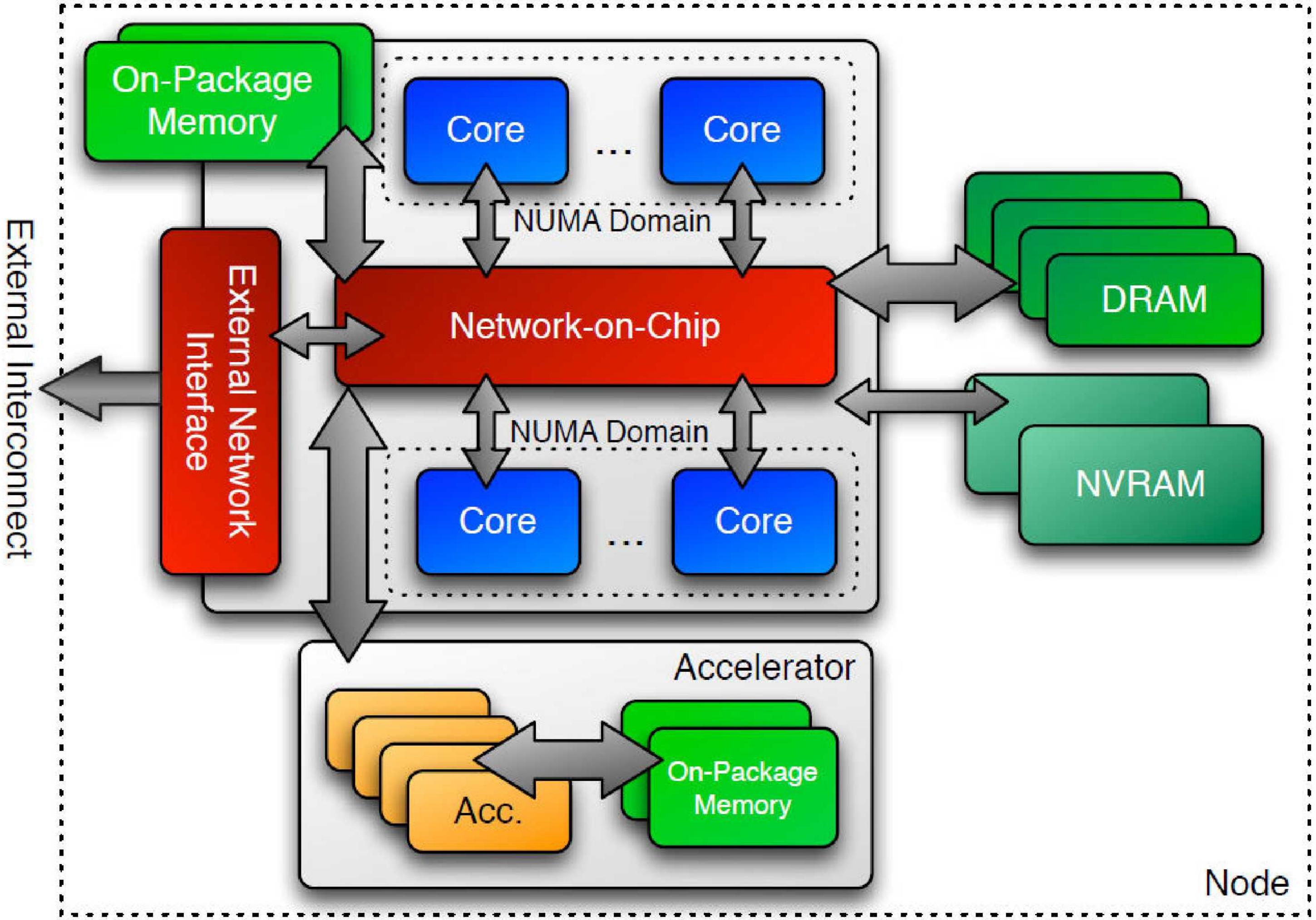
- ❑ is a C++ **library**, not a new language or language extension.
- ❑ supports **clear**, **concise**, **thread-scalable** parallel patterns.
- ❑ lets you write algorithms once and run on **many architectures**

  e.g. multi-core CPU, NVidia GPU, Xeon Phi, ...
- ❑ **minimizes** the amount of **architecture-specific implementation details** users must know.
- ❑ **solves the data layout problem** by using multi-dimensional arrays with architecture-dependent **layouts**

https://github.com/kokkos/kokkos-tutorials/blob/master/Intro-Short/Slides/KokkosTutorial_ATPESC18.pdf

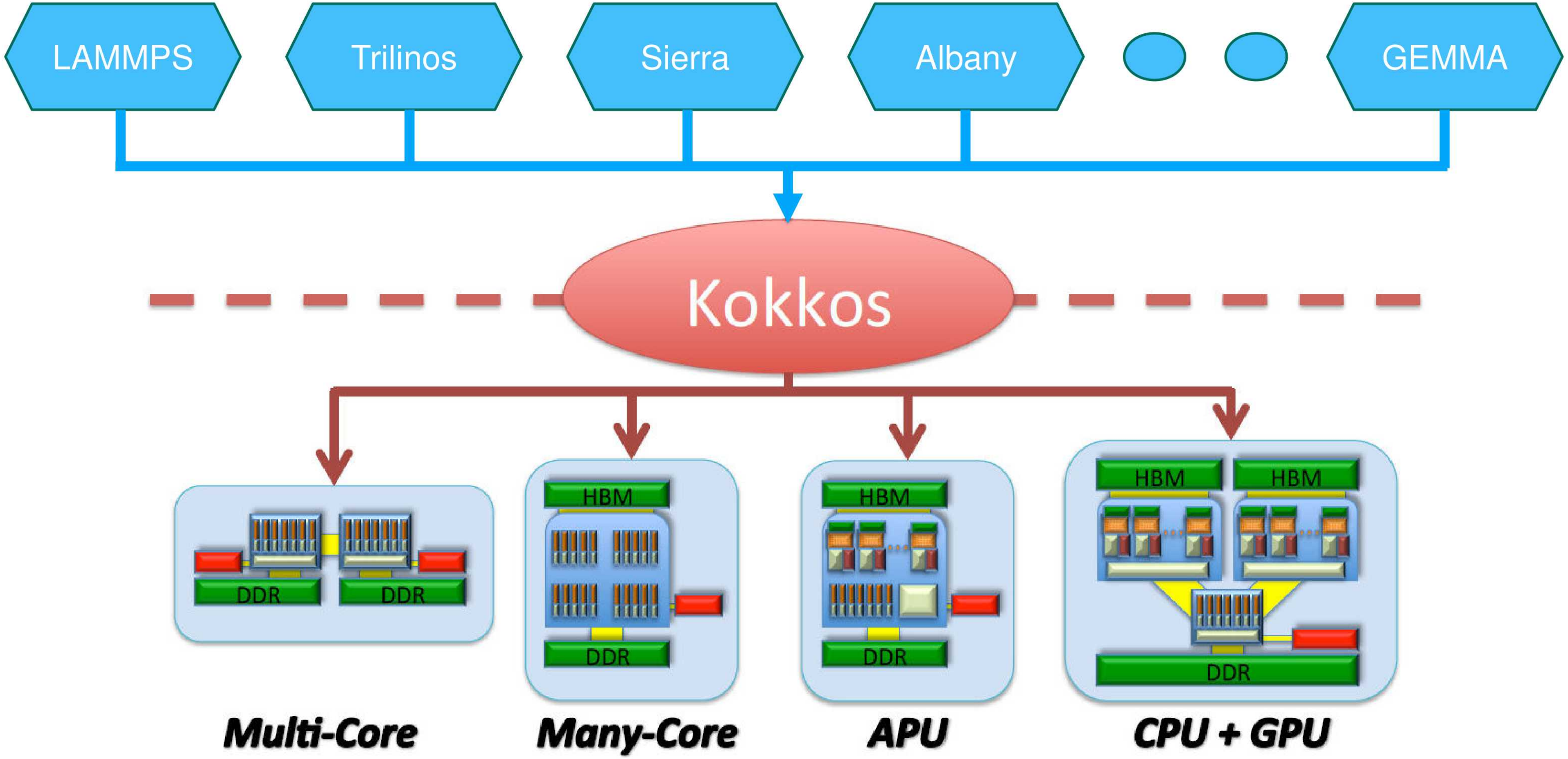
Kokkos – Node Design

Target machine:



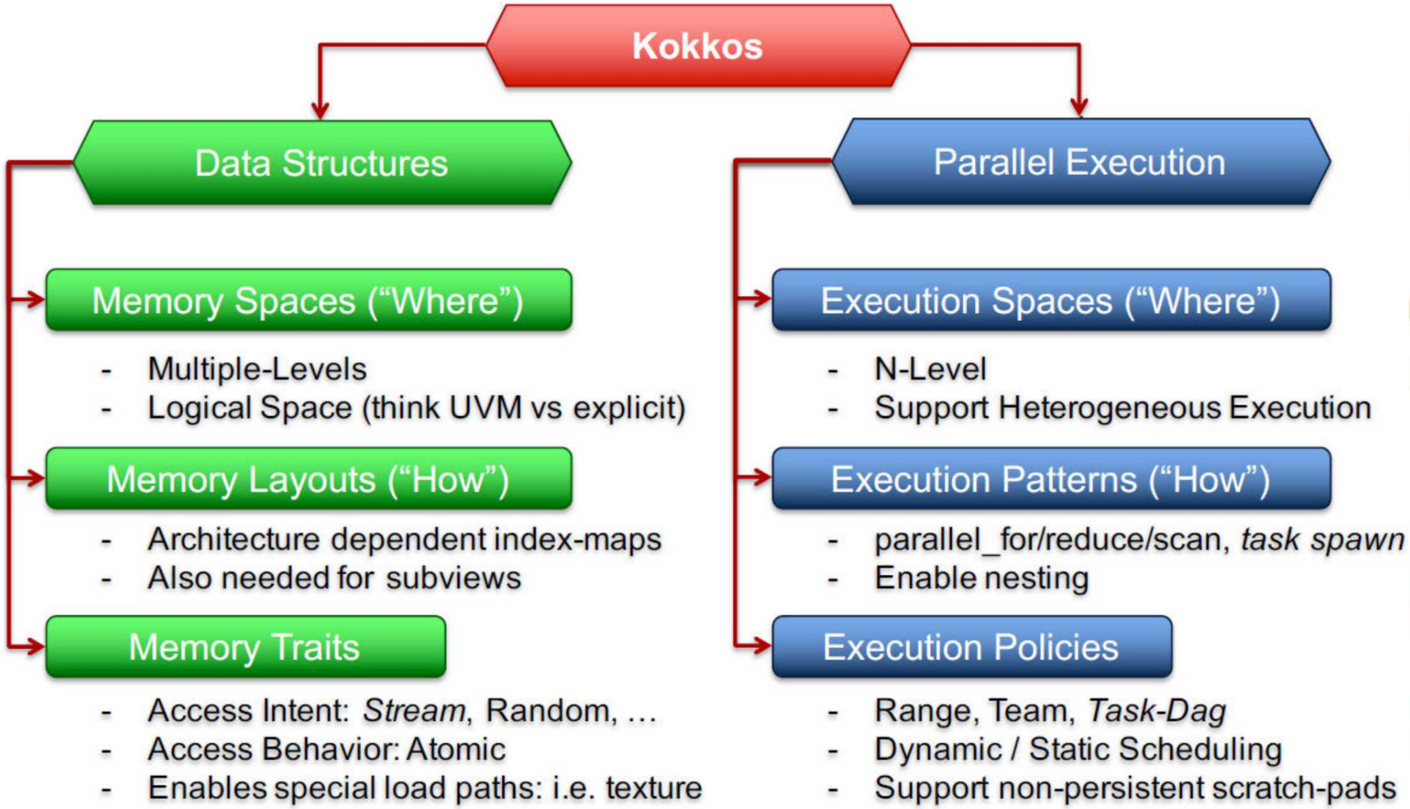
Christian Trott, "Kokkos: Capabilities Overview". https://github.com/kokkos/kokkos-tutorials/blob/master/KokkosCapabilities.pdf

An Abstraction Layer to Prevent Rewriting an Entire Code

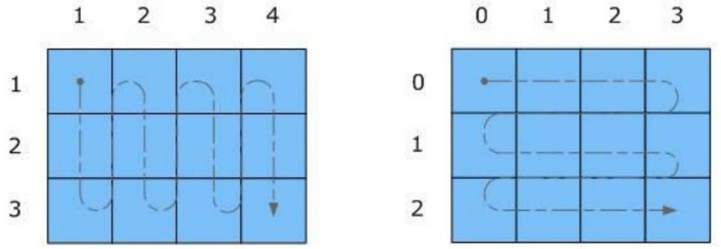


Christian Trott, “Kokkos: Capabilities Overview”. https://github.com/kokkos/kokkos-tutorials/blob/master/KokkosCapabilities.pdf

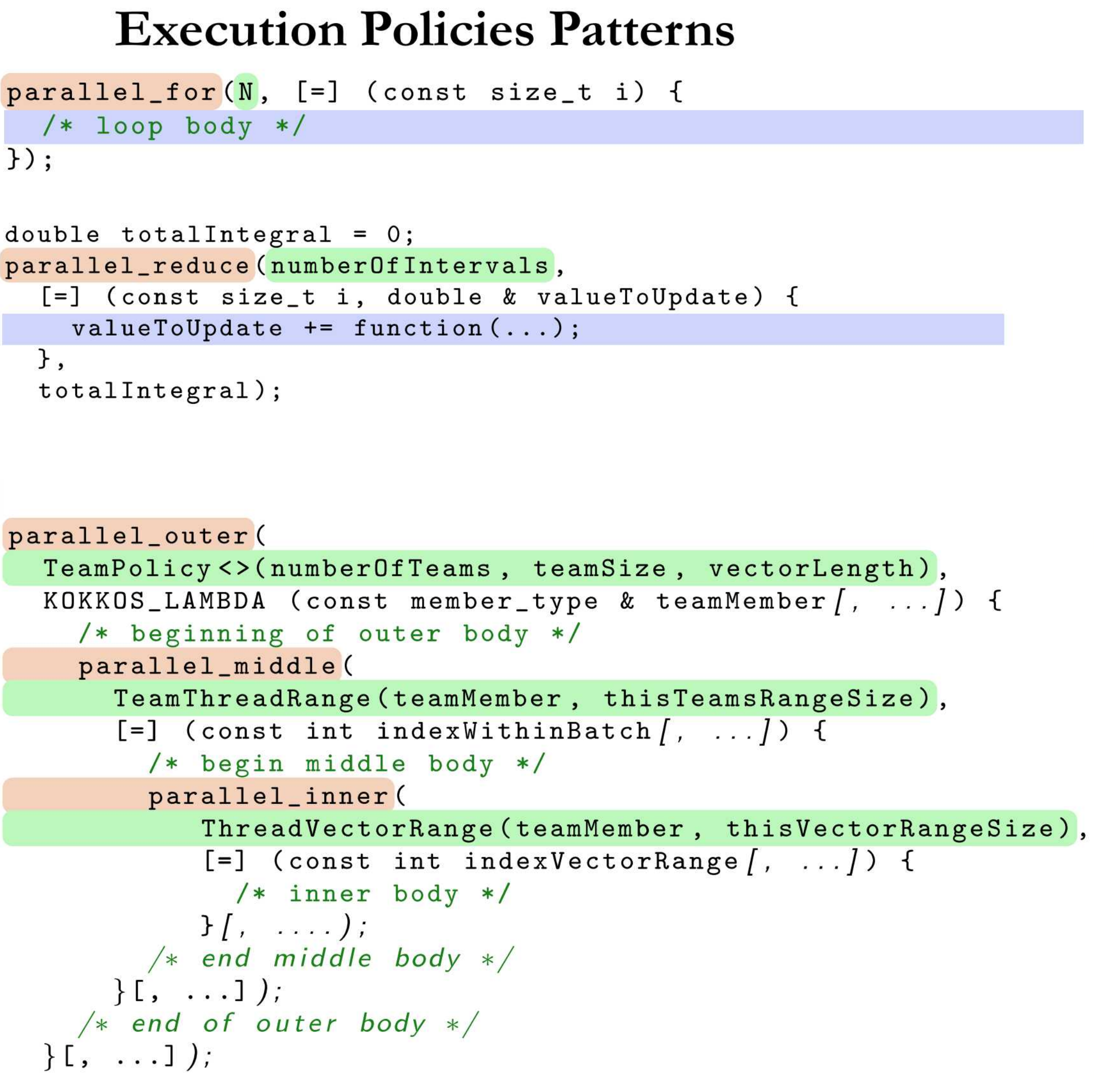
Kokkos Data Management and Execution



- Kokkos
  - Data Structures
    - Memory Spaces ("Where")
      - Multiple-Levels
      - Logical Space (think UVM vs explicit)
    - Memory Layouts ("How")
      - Architecture dependent index-maps
      - Also needed for subviews
    - Memory Traits
      - Access Intent: *Stream*, Random, …
      - Access Behavior: Atomic
      - Enables special load paths: i.e. texture
  - Parallel Execution
    - Execution Spaces ("Where")
      - N-Level
      - Support Heterogeneous Execution
    - Execution Patterns ("How")
      - parallel_for/reduce/scan, *task spawn*
      - Enable nesting
    - Execution Policies
      - Range, Team, *Task-Dag*
      - Dynamic / Static Scheduling
      - Support non-persistent scratch-pads



A: Column-major order (Fortran-style)   B: Row-major order (C-style)



Execution Policies Patterns

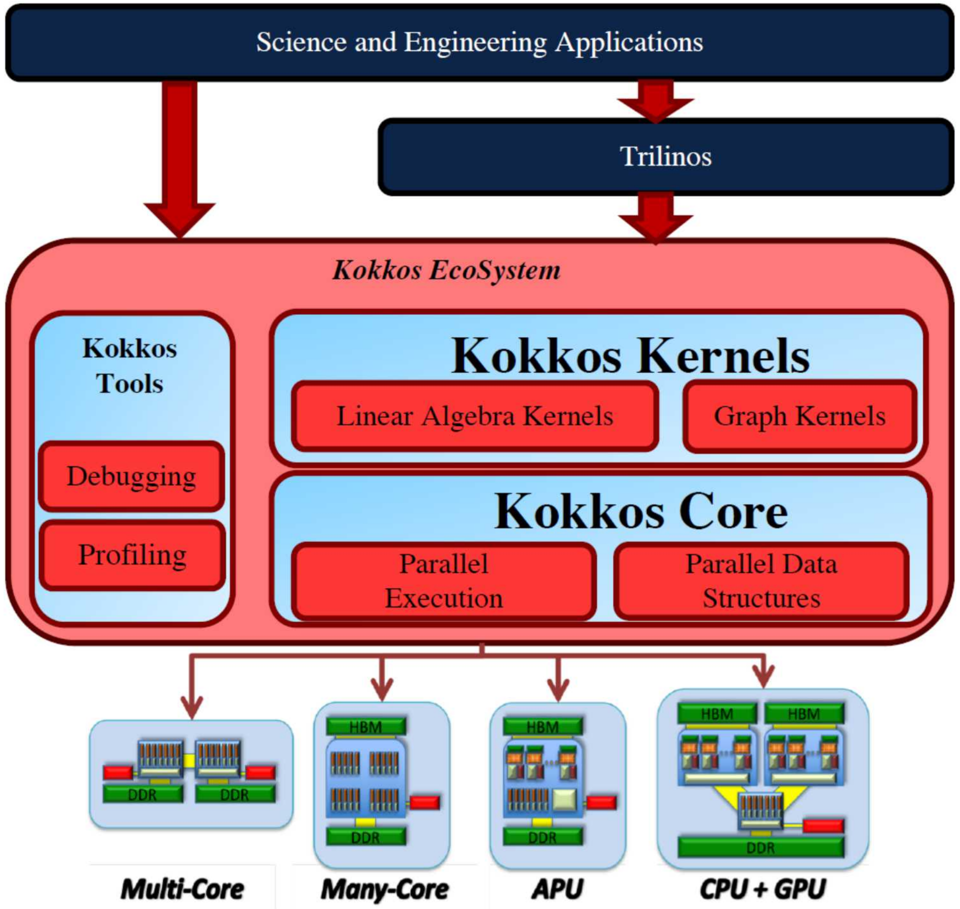
```
parallel_for(N, [=] (const size_t i) {
  /* loop body */
});

double totalIntegral = 0;
parallel_reduce(numberOfIntervals,
  [=] (const size_t i, double & valueToUpdate) {
    valueToUpdate += function(...);
  },
  totalIntegral);

parallel_outer(
  TeamPolicy<>(numberOfTeams, teamSize, vectorLength),
  KOKKOS_LAMBDA (const member_type & teamMember[, ...]) {
    /* beginning of outer body */
    parallel_middle(
      TeamThreadRange(teamMember, thisTeamsRangeSize),
      [=] (const int indexWithinBatch[, ...]) {
        /* begin middle body */
        parallel_inner(
          ThreadVectorRange(teamMember, thisVectorRangeSize),
          [=] (const int indexVectorRange[, ...]) {
            /* inner body */
          }[, ....);
        /* end middle body */
      }[, ...]);
    /* end of outer body */
  }[, ...]);
```

Christian Trott, "Kokkos: Capabilities Overview". https://github.com/kokkos/kokkos-tutorials/blob/master/KokkosCapabilities.pdf
Intel. *Developer Guide for Intel Math Kernel Library for Linux*. https://software.intel.com/en-us/node/528573

Kokkos Kernels

KokkosKernels is a library for node-level, performance-portable, computational kernels for sparse/dense linear algebra and graph operations, using the Kokkos shared-memory parallel programming model.

- Kokkos Kernels is available publicly both as part of Trilinos and as part of the **Kokkos ecosystem**
- Can be building block of a solver, linear algebra library that uses MPI and threads for parallelism, or it can be used stand-alone in an application.
- Generic implementations for various scalar types and data layouts
- Interfaces to Intel, NVIDIA and other vendor provided kernels available in order to leverage their high-performance libraries
- Several new kernels are being added as needed by the applications
  - E.g.: Distance-2 Coloring, Deterministic coloring, dense linear system solver, …
- Expand the scope of BLAS to hierarchical implementations.
- Download at https://github.com/kokkos/kokkos-kernels



Kokkos Kernels – Capabilities (1/2)

BLAS

- BLAS-1: abs(), axpy(), axpby(), dot(), fill(), mult(), nrm2(), scal(), …
- BLAS-2: gemv() (matrix-vector multiplication)
- BLAS-3: gemm() (matrix-matrix multiplication)
- BLAS-1 functions are available as multi-vector variants
- Extended BLAS: dense linear system solver (MAGMA Third-Party Library (TPL))

Sparse

- CSR-Sparse Matrix Class providing fundamental capabilities
- SPMV: Sparse Matrix Vector Multiply
- SpGEMM: Sparse Matrix Matrix Multiply; separate symbolic and numeric phase
- GS: Gauss-Seidel Method using graph coloring: symbolic, numeric, solve phases

Batched BLAS

- DGEMM: matrix-matrix multiplication
- DTRSM: triangular system solve
- DGETRF: LU factorization

Kokkos Kernels – Capabilities (2/2)

Graph

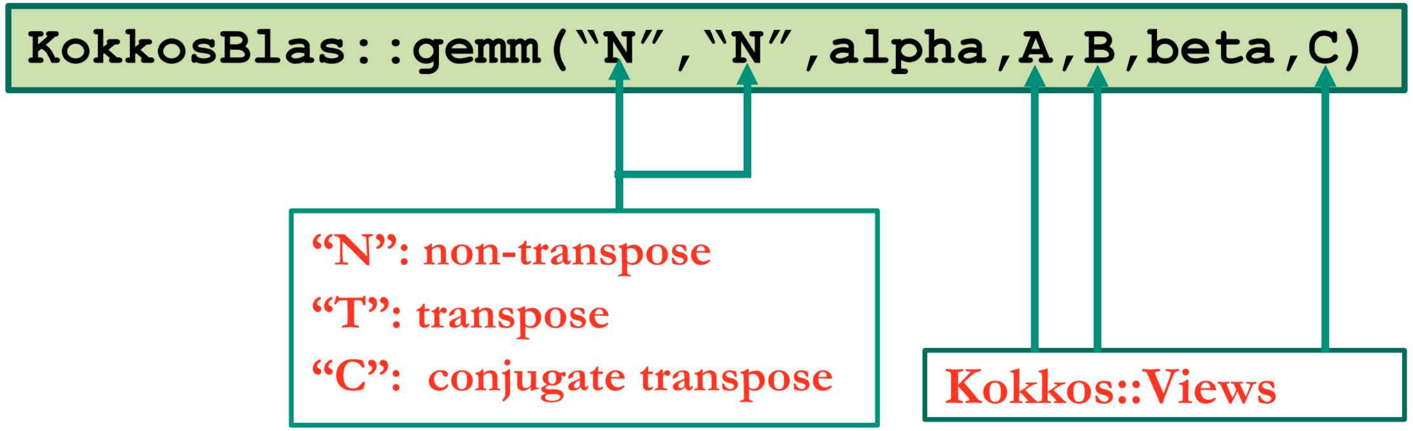
- Distance-1 and Distance-2 graph coloring
- Triangle enumeration for graph analytics
  - Using SpGEMM + Visitor Pattern: can be used to represent large problems

Micro BLAS

- Hierarchical Hardware requires hierarchy of function support
- Provide BLAS / SparseBLAS interface with hardware handles
- Example use-case: each CUDA block or KNL tile runs its own independent CG-Solve
- Team-level BLAS: abs(), axpy(), axpby(), dot(), mult(), nrm2(), scal(), update(), gemv(), …
- Kernel uses a CUDA block or all threads sharing a common L2 cache
- Utilization of local scratch

cuBLAS and MKL TPLs Support in Kokkos Kernels

- ❑ BLAS functions (Level-1, -2, and -3): KokkosKernels allows calling cuBLAS or MKL functions through KokkosKernels interfaces
  - cuBLAS: axpy, axpby, dot, nrm1 (asum), nrm2, nrm_inf (Ixamax), scal, gemv, gemm.
    More will be added when necessary
  - MKL: not yet, will be added soon (needed for Pliris)
- ❑ Example:



```
KokkosBlas::gemm(“N”,“N”,alpha,A,B,beta,C)
```

“N”: non-transpose
“T”: transpose
“C”: conjugate transpose

Kokkos::Views

performs matrix-matrix multiplication

```
C[i,j] = beta*C[i,j] + alpha*SUM_k(A[i,k]*B[k,j])
```

Equivalent cuBLAS functions: cublasSgemm, cublasDgemm, cublasCgemm, cublasZgemm

MAGMA TPL Support (GESV) in Kokkos Kernels (1/2)

- MAGMA is a collection of next generation linear algebra libraries for heterogeneous GPU-based architectures of multi/manycore CPUs and multi-GPUs.

- Key Features:
  - Multiple precision arithmetic support (S/D/C/**Z**)
  - **Hybrid** algorithms using both multicore CPUs and GPUs
  - Hybrid LAPACK-style functions:
    - Matrix factorizations: LU, Cholesky, QR, eigenvalue, SVD, ...
    - **Solve linear systems** and linear least squares, ...
    - Nearly all are synchronous: return on CPU when computation is finished
  - GPU BLAS and auxiliary functions:
    - Matrix-vector multiply, matrix norms, transpose (in-place and out-of-place), ...
    - Most are asynchronous: return immediately on CPU; computation proceeds on GPU
  - Wrappers around CUDA and cuBLAS:
    - BLAS routines (gemm, symm, symv, ...)
    - Copy host ⇔ device, queue (stream) support, GPU malloc & free, ...

- Nvidia Tensor Cores version of linear mixed-precision solver that is able to provide an FP64 solution with up to 4x speedup using the fast FP16 Tensor Cores arithmetic

MAGMA TPL Support (GESV) in Kokkos Kernels (2/2)

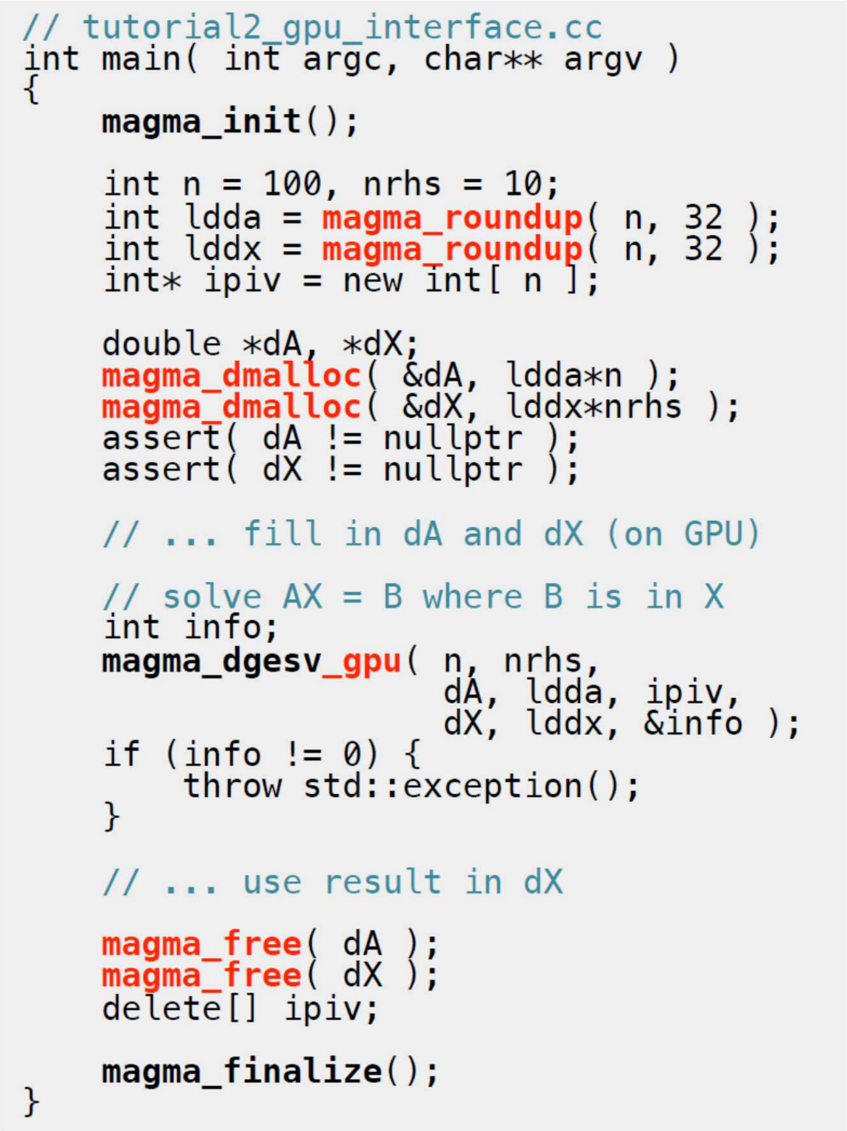
MAGMA GPU interface:

- Column-major layout
- Solve AX = B
- Example:

  Double precision, **GE**neral matrix **S**ol**V**e (**DGESV**)
- Input & output matrices in GPU device memory

Note:

- Set GPU stride (ldda) to multiple of 32 for better performance
- ipiv still in CPU memory

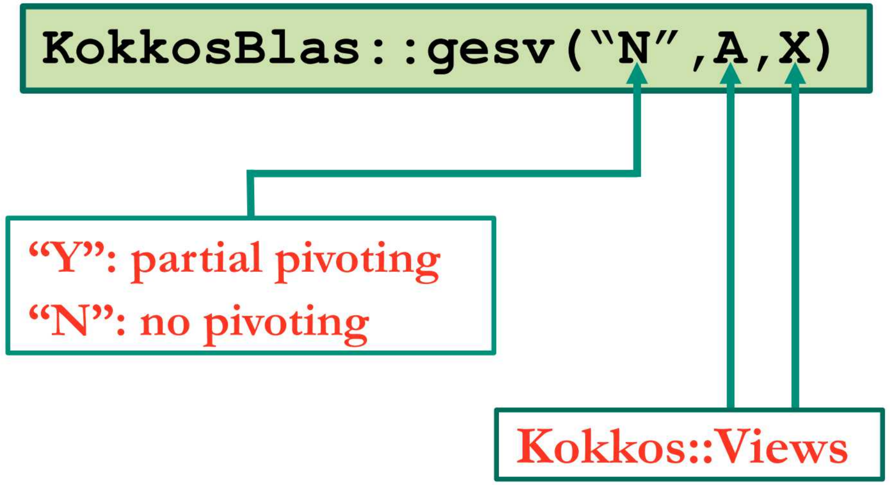


```
// tutorial2_gpu_interface.cc
int main( int argc, char** argv )
{
    magma_init();

    int n = 100, nrhs = 10;
    int ldda = magma_roundup( n, 32 );
    int lddx = magma_roundup( n, 32 );
    int* ipiv = new int[ n ];

    double *dA, *dX;
    magma_dmalloc( &dA, ldda*n );
    magma_dmalloc( &dX, lddx*nrhs );
    assert( dA != nullptr );
    assert( dX != nullptr );

    // ... fill in dA and dX (on GPU)

    // solve AX = B where B is in X
    int info;
    magma_dgesv_gpu( n, nrhs,
                     dA, ldda, ipiv,
                     dX, lddx, &info );
    if (info != 0) {
        throw std::exception();
    }

    // ... use result in dX

    magma_free( dA );
    magma_free( dX );
    delete[] ipiv;

    magma_finalize();
}
```

Source: ECP 2018 Annual Meeting Tutorials
https://icl.utk.edu/projectsfiles/magma/tutorial/ecp2018-magma-tutorial.pdf

MAGMA TPL support in KK:

- calling MAGMA functions through KK interfaces
- taking Kokkos::View instead of raw pointers
- initializing and finalizing MAGMA, allocating ipiv under the hood



```
KokkosBlas::gesv("N",A,X)
```

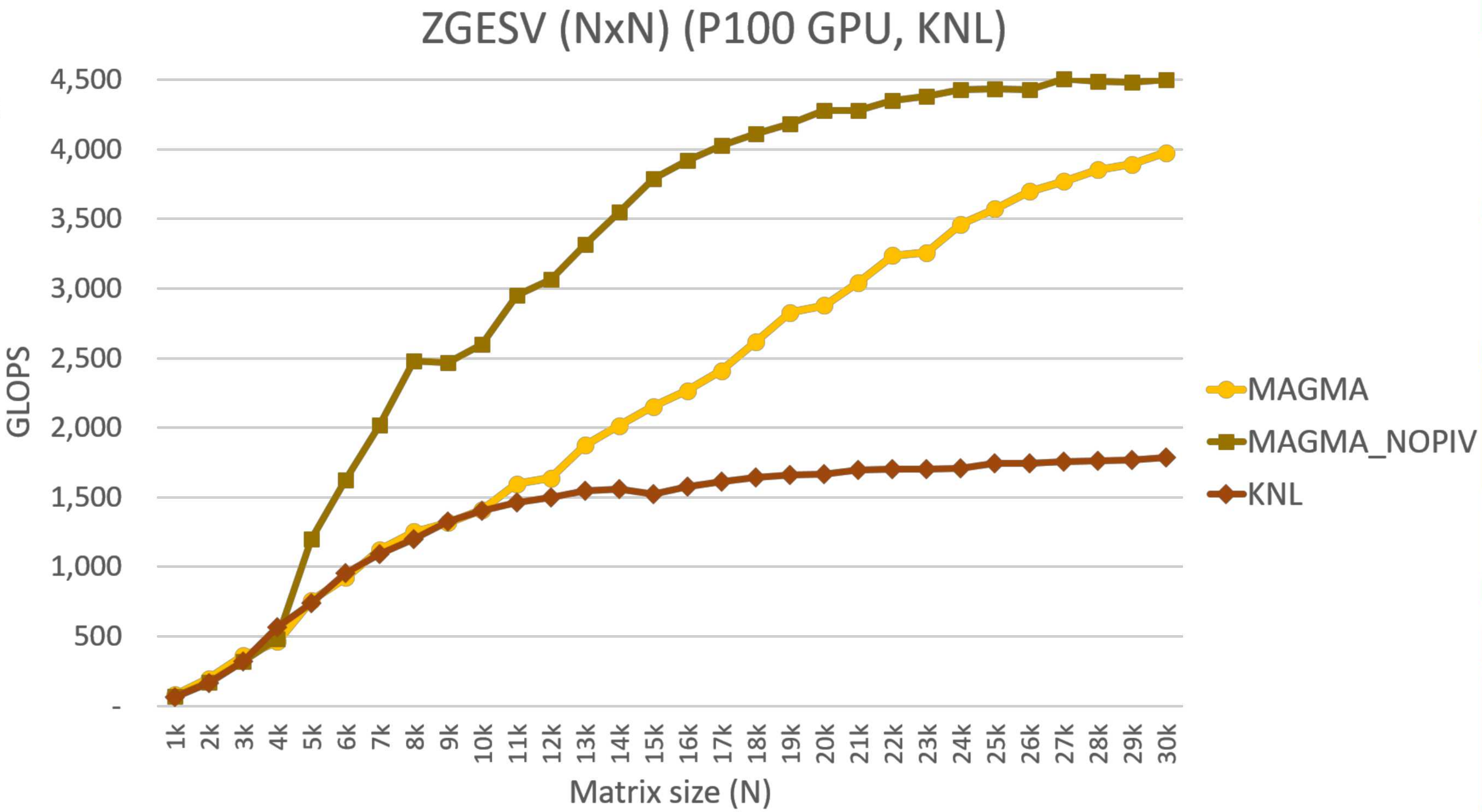
“Y”: partial pivoting
“N”: no pivoting

Kokkos::Views

Note: Used for solving problem fit in a single node in GEMMA

MAGMA ZGESV Performance Evaluation

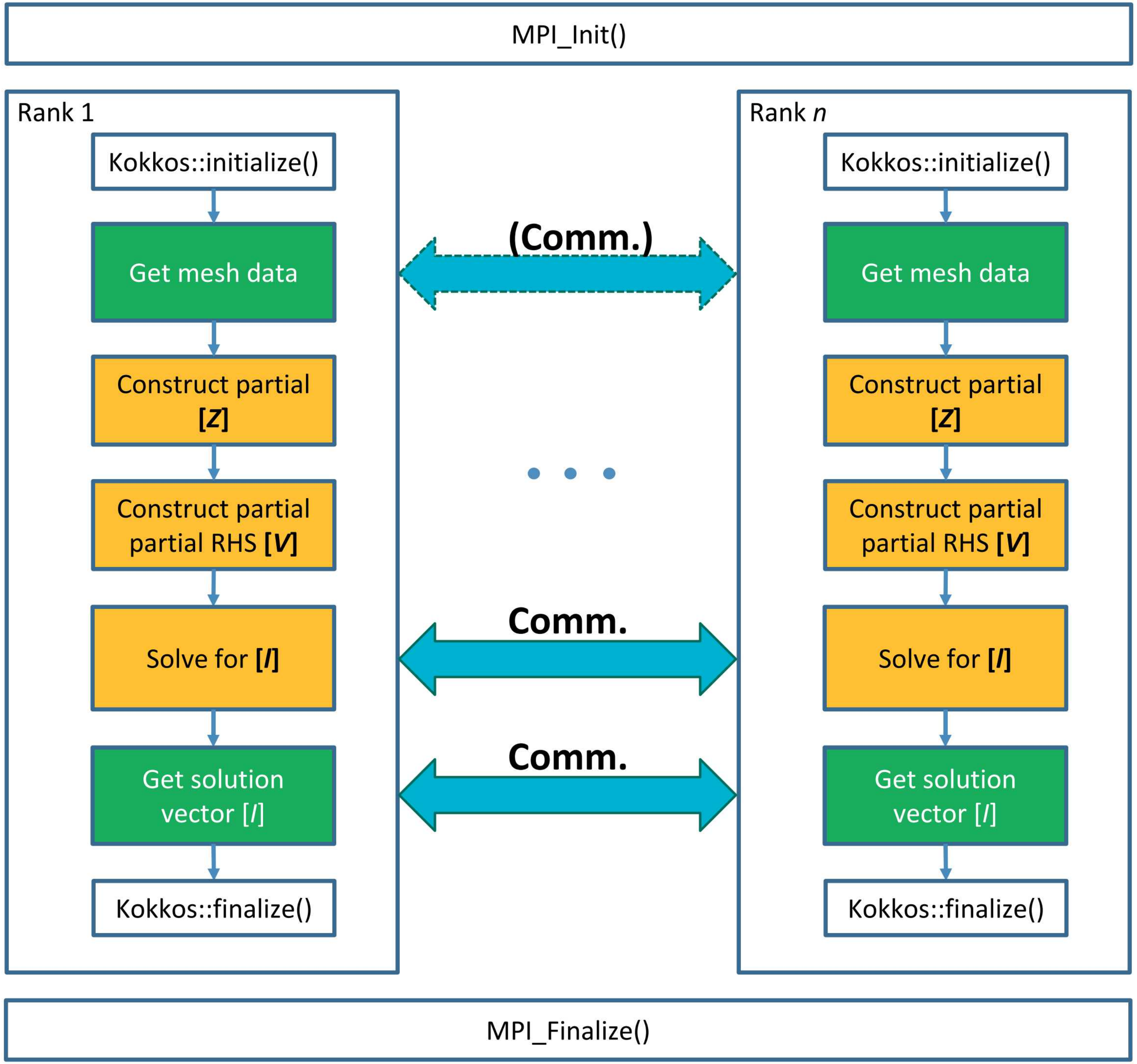
- ❑ Evaluation was performed through a Kokkos framework only
- ❑ Will redo with Kokkos Kernels TPL support when it gets merged
- ❑ Matrix A and vector B are randomly generated as Kokkos Views
- ❑ MAGMA: **magma_zgesv_gpu**, **magma_zgesv_nopiv_gpu** (on a single P100 Nvidia GPU, OMP_NUM_THREADS=64)
- ❑ MKL: **LAPACKE_zgesv** (on KNL, OMP_NUM_THREADS=68)



GFLOPS: **G**iga **F**loating **P**oint **O**perations **P**er **S**econd – The higher the better

Kokkos and/or Kokkos Kernels with MPI

Generic Framework



MPI_Init()

Rank 1

Kokkos::initialize()

Get mesh data

Construct partial **[*Z*]**

Construct partial partial RHS **[*V*]**

Solve for **[*I*]**

Get solution vector [*I*]

Kokkos::finalize()

(Comm.)

• • •

Comm.

Comm.

Rank *n*

Kokkos::initialize()

Get mesh data

Construct partial **[*Z*]**

Construct partial partial RHS **[*V*]**

Solve for **[*I*]**

Get solution vector [*I*]

Kokkos::finalize()

MPI_Finalize()

Next-Gen Pliris

Pliris: Parallel Dense Solver Package

- ❑ Performs LU factorization and solves a dense matrix equation on parallel computers using MPI
- ❑ The matrix is torus-wrap mapped onto the processors(transparent to the user) and uses partial pivoting during the factorization of the matrix
  - since the input matrix is not torus-wrapped, permutation of the results is performed to “unwrap the results”
- ❑ Each processor contains a portion of the matrix and the right hand sides determined by a distribution function to optimally load balance the computation and communication during the factorization of the matrix
  - no processor can have no more(or less) than one row or column of the matrix than any other processor
- ❑ Use old C-programming style with pointers and can run only on CPUs

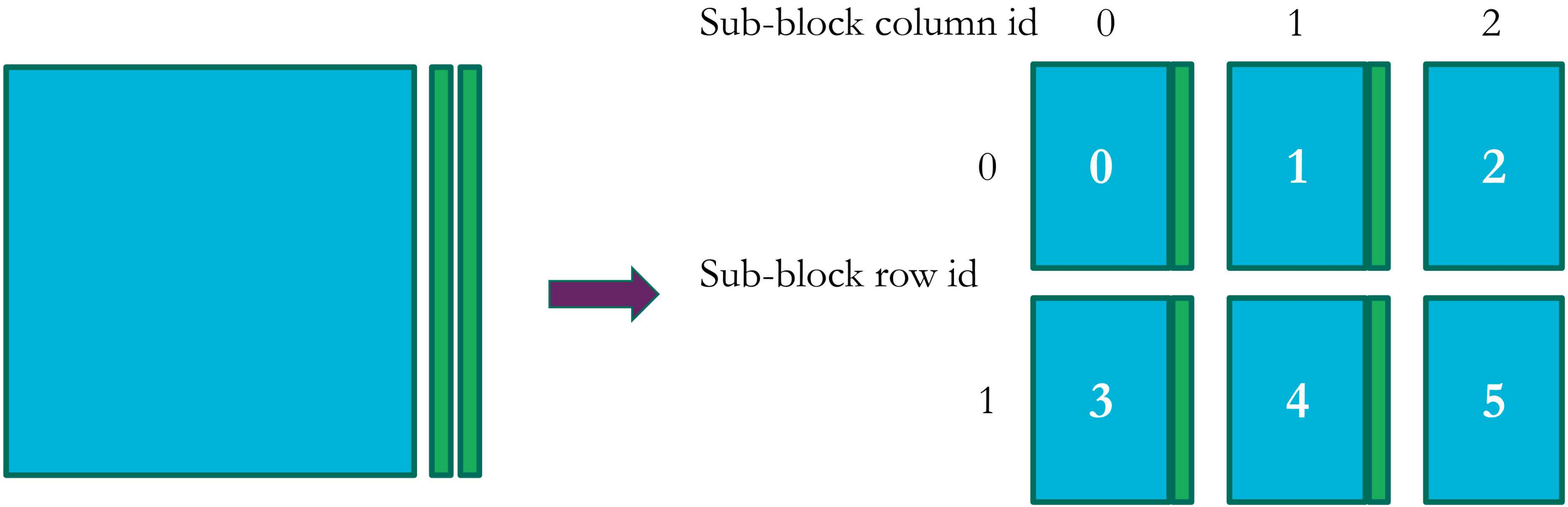
https://github.com/trilinos/Trilinos/tree/master/packages/pliris

Pliris – Example of Workload Distribution

Total number of processors = 6

Number of processors for a row = 3

Number of right-hand sides = 2



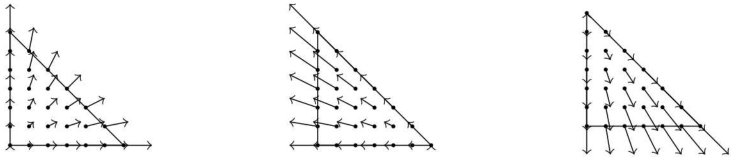
Next-Gen Pliris (in progress)

- ❑ C++ 11 (driving toward newer C++ standards)
- ❑ Templated implementation for code readability and maintenance
- ❑ Supporting multiple target compilers and architectures
  - Re-factor Pliris for use on MIC and GPU processors
  - Use Kokkos and Kokkos Kernels
- ❑ Non-even distribution may be necessary for load balancing in the future applications

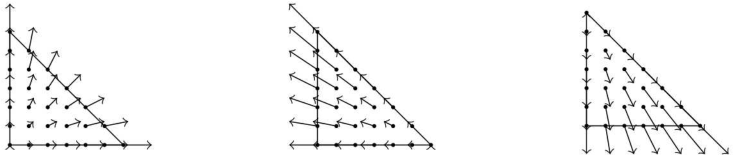
Arithmetic Intensity for EFIE MoM

MoM EFIE Arithmetic Intensity when Considering Triangle Pairs

Test triangle:



Source triangle:

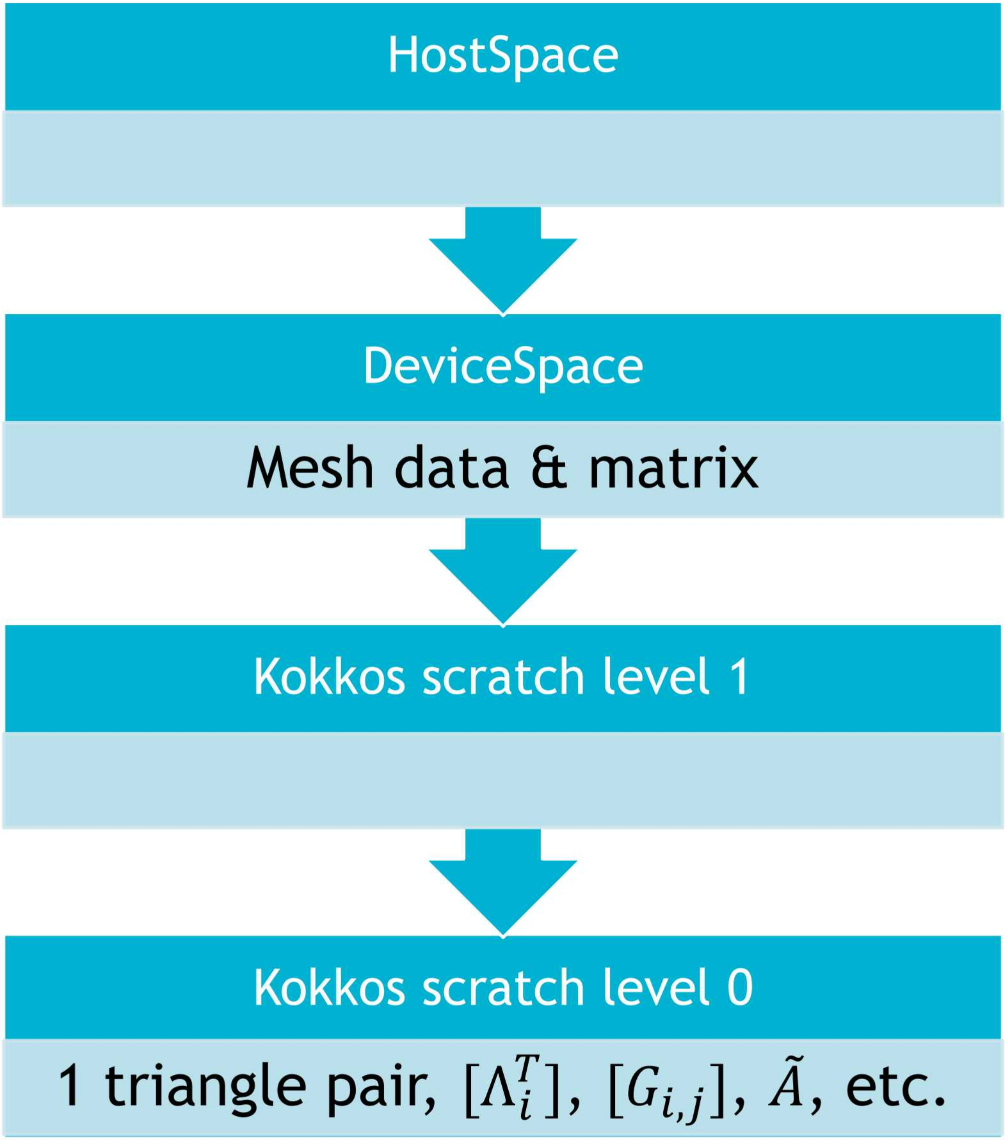


- **Fills 1/4th of 9 EFIE matrix entries:** Compute the 4D integral $\iint G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$ over an triangle pair where T and S loop through the 3 half-basis functions on each triangle.
- Assume the triangle pair requires loading 520 bytes = 6 vertices (18 doubles) + eps & mu (2 complexes) + connectivity information (14 ints) + 9 matrix contributes (18 complexes since atomic)
- With some reuse of data, the arithmetic intensity is $\frac{2335}{520}$ FLOPs/byte ≈ 4.5 FLOPs/byte

	add, sub, mul (1 FLOPs)	div (4 FLOPs)	exp, sincos, sqrt (8 FLOPs)	estimated FLOPs	Total for 3x7 points and 3 half-basis
$\Lambda(r)$	16	2		24	720 (= FLOPs x 10 x 3)
$\nabla \cdot \Lambda$		1		4	12 (= FLOPs x 3)
$G(r, r')$	7	2	3	39	819 (= FLOPs x 21)
$G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$	8			8	504 (= FLOPs x 21 x 3)
Elemental mapping	28			28	280 (= FLOPs x 10)
Total	59	5	3	108	2335

Considering Triangle Pairs via Quadrature Inner Product

- Inner product over quadrature to form a 3x3 block of matrix contributions: $\tilde{A} = [\Lambda_1^T \quad \cdots \quad \Lambda_3^T] \begin{bmatrix} G_{1,1} & \cdots & G_{1,7} \\ \vdots & & \vdots \\ G_{3,1} & \cdots & G_{3,7} \end{bmatrix} \begin{bmatrix} \Lambda_1^S \\ \vdots \\ \Lambda_7^S \end{bmatrix}.$
- 3x3 block of complexes $\tilde{A}$ is scattered to nonconsecutive matrix entries. It contains 18 doubles, i.e., 144 bytes.
- $[\Lambda_i^T]$, $[\Lambda_j^S]$, $[\nabla \cdot \Lambda^T]$, $[\nabla \cdot \Lambda^S]$, $[G_{i,j}]$ vectors and matrix contain 162 doubles, i.e., 1296 bytes.
- Requires 1528 bytes of Kokkos scratch level 0.
- GPU details: Only 1 triangle pair considered by a warp.



HostSpace

DeviceSpace

Mesh data & matrix

Kokkos scratch level 1

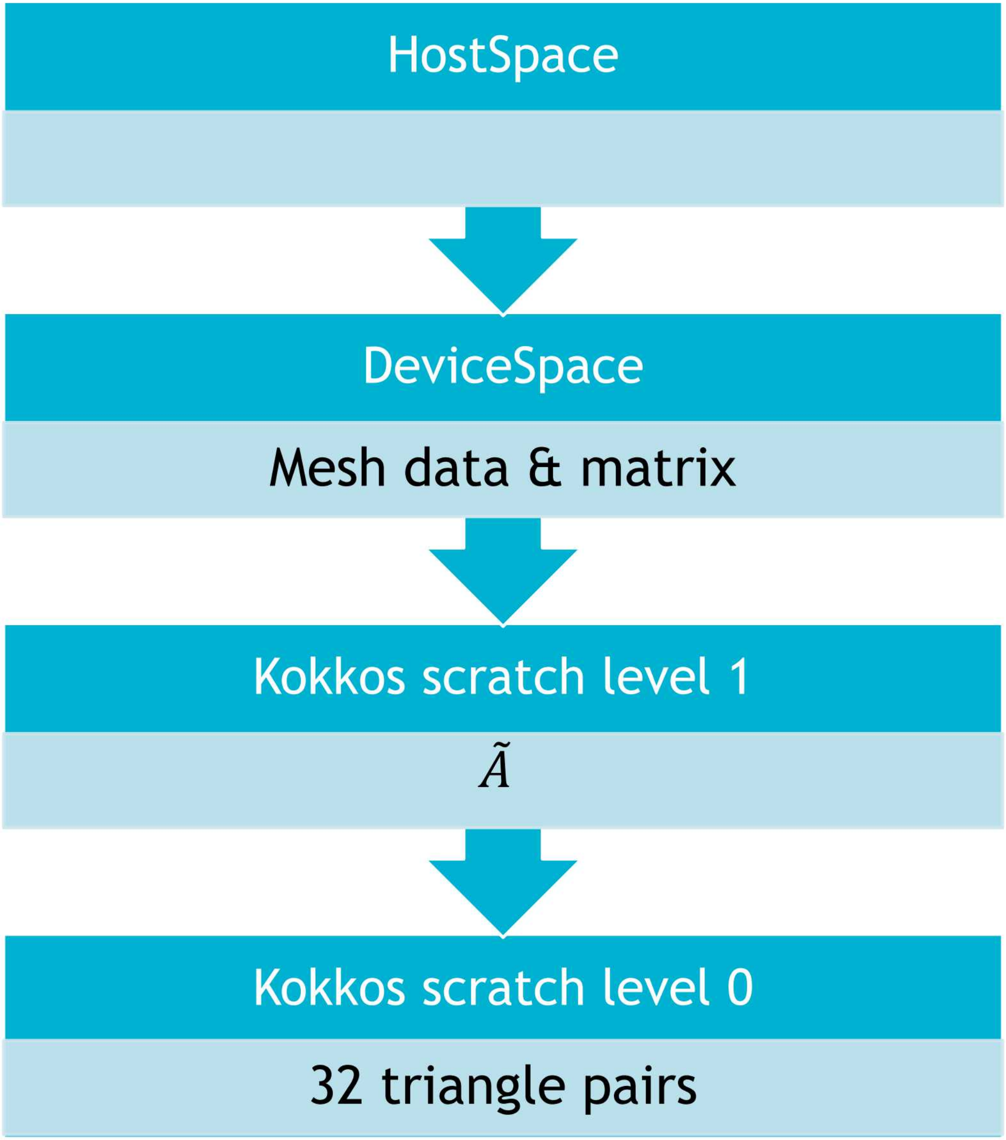
Kokkos scratch level 0

1 triangle pair, $[\Lambda_i^T]$, $[G_{i,j}]$, $\tilde{A}$, etc.

Considering Triangle Pairs via a Loop over Triangle Pairs

- Instead of storing the $[\Lambda_i^T]$, $[\Lambda_j^S]$, $[G_{i,j}]$ vectors and matrix required for $\tilde{A} = [\Lambda_1^T \quad \cdots \quad \Lambda_3^T] \begin{bmatrix} G_{1,1} & \cdots & G_{1,7} \\ \vdots & & \vdots \\ G_{3,1} & \cdots & G_{3,7} \end{bmatrix} \begin{bmatrix} \Lambda_1^S \\ \vdots \\ \Lambda_7^S \end{bmatrix}$, compute them on the fly.
- Requires 7424 bytes of Kokkos scratch level 0 and 4608 bytes of Kokkos scratch level 1.

	Total FLOPs per triangle pair
$\Lambda(r)$	1512 (= FLOPs x 21 x 3)
$\nabla \cdot \Lambda$	252 (= FLOPs x 21 x 3)
$G(r, r')$	2457 (= FLOPs x 21 x 3)
$G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$	504 (= FLOPs x 21 x 3)
Elemental mapping	1764 (= FLOPs x 21 x 3)
Total	6489



HostSpace

DeviceSpace

Mesh data & matrix

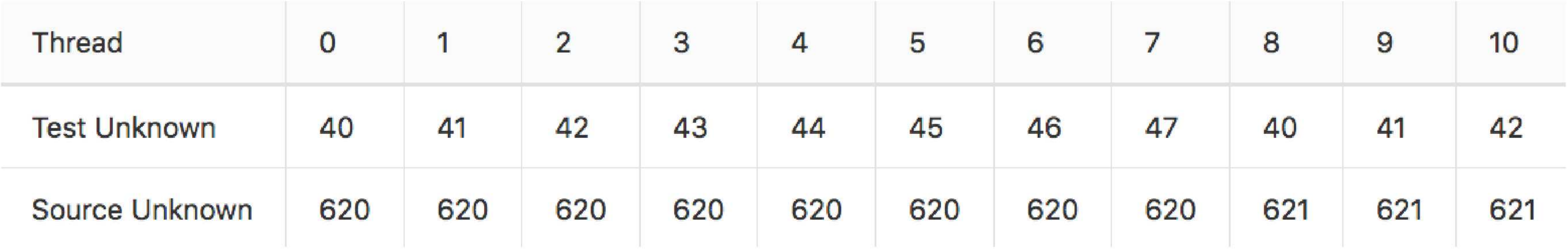
Kokkos scratch level 1

$\tilde{A}$

Kokkos scratch level 0

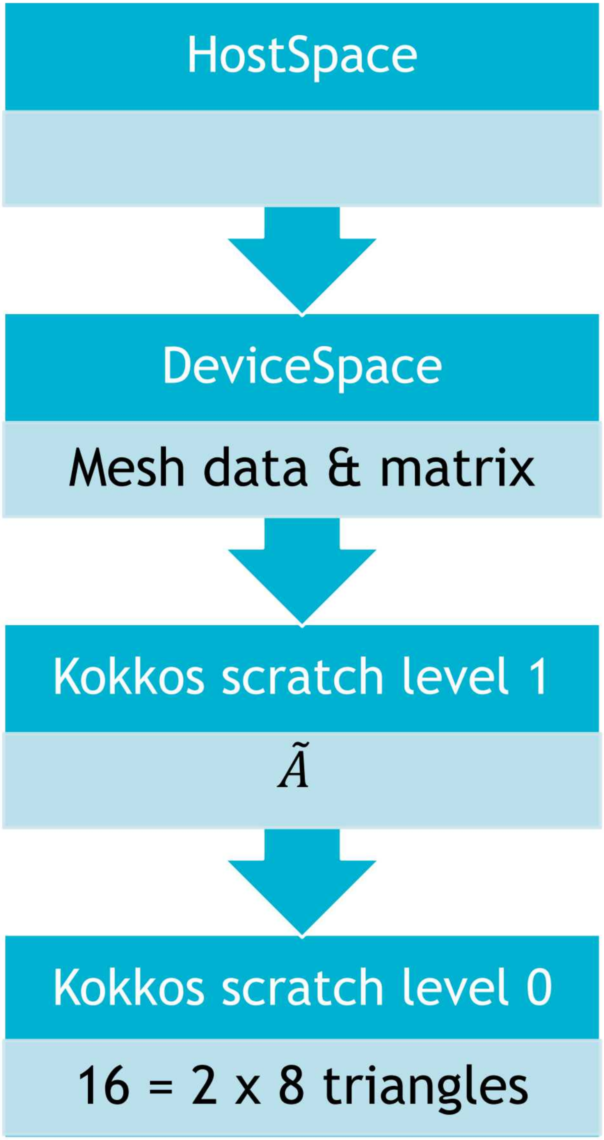
32 triangle pairs

Considering Triangle Pairs via a Loop over Triangles



Thread	0	1	2	3	4	5	6	7	8	9	10
Test Unknown	40	41	42	43	44	45	46	47	40	41	42
Source Unknown	620	620	620	620	620	620	620	620	621	621	621

- Instead of loading 32 triangle pairs, load 8 test and 8 source triangles to make 64 triangle pairs on the fly.
- Reduces memory use in Kokkos scratch level 0 by reducing the required number of vertices, but maybe not the other information.
  - Previous slide's 32 triangle pairs require 192 vertices, which is 576 doubles or 4608 bytes.
  - This slide's 16 triangles require 48 vertices, which is 144 doubles or 1152 byes.
- Required FLOPs largely unaffected.



HostSpace

DeviceSpace

Mesh data & matrix

Kokkos scratch level 1

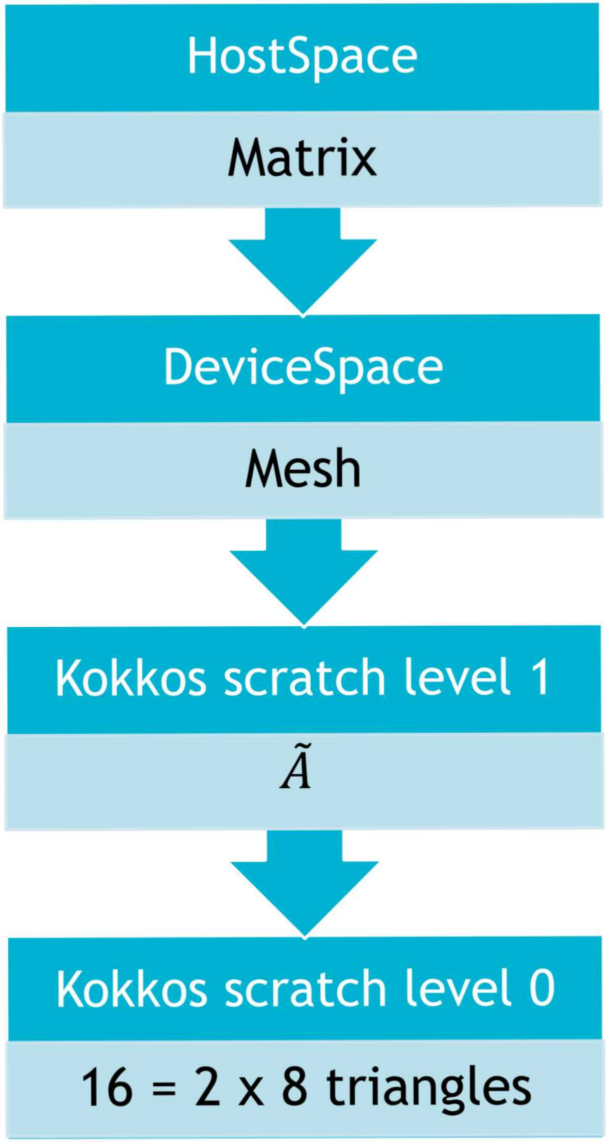
$\tilde{A}$

Kokkos scratch level 0

16 = 2 x 8 triangles

Considering Triangle Pairs while Storing the Matrix in HostSpace

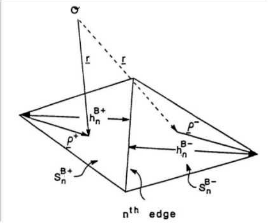
- Send $\tilde{A}$ to HostSpace with instructions on where to put it in the matrix.
- $\tilde{A}$ is $3x3$ and each entry goes to a different location in the matrix.
- Information required by the host to scatter $\tilde{A}$ is 216 bytes = 9 entries of $\tilde{A}$ (9 complexes) + 9 matrix coordinates (18 ints).
- GPU details:
  - Data reuse: the algorithm requires 2335 FLOPs. Its arithmetic intensity is $\frac{2335}{216}$ FLOPs/byte ≈ 10.8 FLOPs/byte.
  - No data reuse: the algorithm requires 6489 FLOPs. Its arithmetic intensity is $\frac{6489}{216}$ FLOPs/byte ≈ 30 FLOPs/byte.
  - Feasible if (the GPU max FLOPs divided by the GPU-HostSpace bandwidth) is less than the arithmetic intensity.



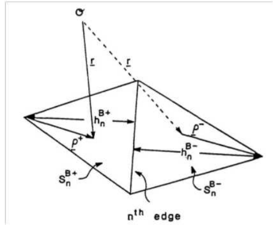
John Pennycook, Charlene Yang, and Jack Deslippe, "Quantitatively Assessing Performance Portability with Roofline", NeRSC. https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar025

MoM EFIE Arithmetic Intensity when Considering Basis Pairs

Test basis:



Source basis:



- **Fills EFIE matrix entry ($\boldsymbol{T}, \boldsymbol{S}$):** Compute the 4D integral $\iint G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$ over 4 triangle pairs, 2 of which support basis T and 2 of which support basis S.

- Assume for 4 triangle pairs, the triangle pair requires loading 652 bytes = 12 vertices (36 doubles) + eps & mu (2 complexes) + connectivity information (16 ints) + 1 matrix contributes (1 complex)

- With some reuse of data, the arithmetic intensity is $\frac{5004}{652}$ FLOPs/byte ≈ 7.7 FLOPs/byte

	add, sub, mul (1 FLOPs)	div (4 FLOPs)	exp, sincos, sqrt (8 FLOPs)	estimated FLOPs	Total for 3x7 points and 2x2 triangles
$\Lambda(r)$	16	2		24	480 (= FLOPs x 10 x 2)
$\nabla \cdot \Lambda$		1		4	16 (= FLOPs x 4)
$G(r, r')$	7	2	3	39	3276 (= FLOPs x 21 x 4)
$G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$	8			8	672 (= FLOPs x 21 x 4)
Elemental mapping	28			28	560 (= FLOPs x 10 x 2)
Total	59	5	3	108	5004

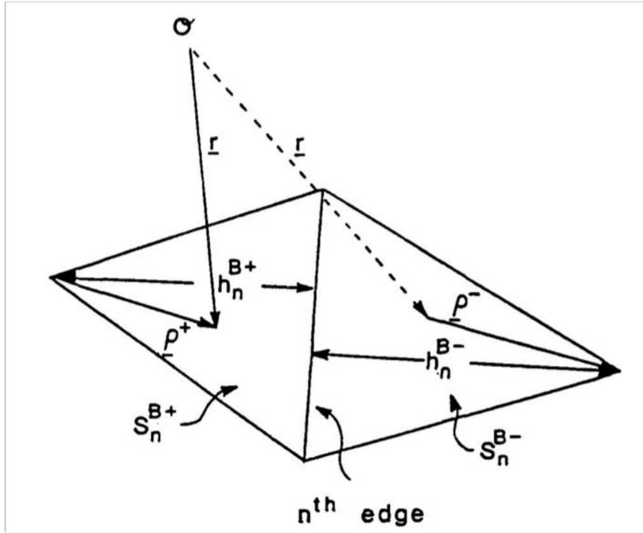
Considering Basis Pairs without Reuse

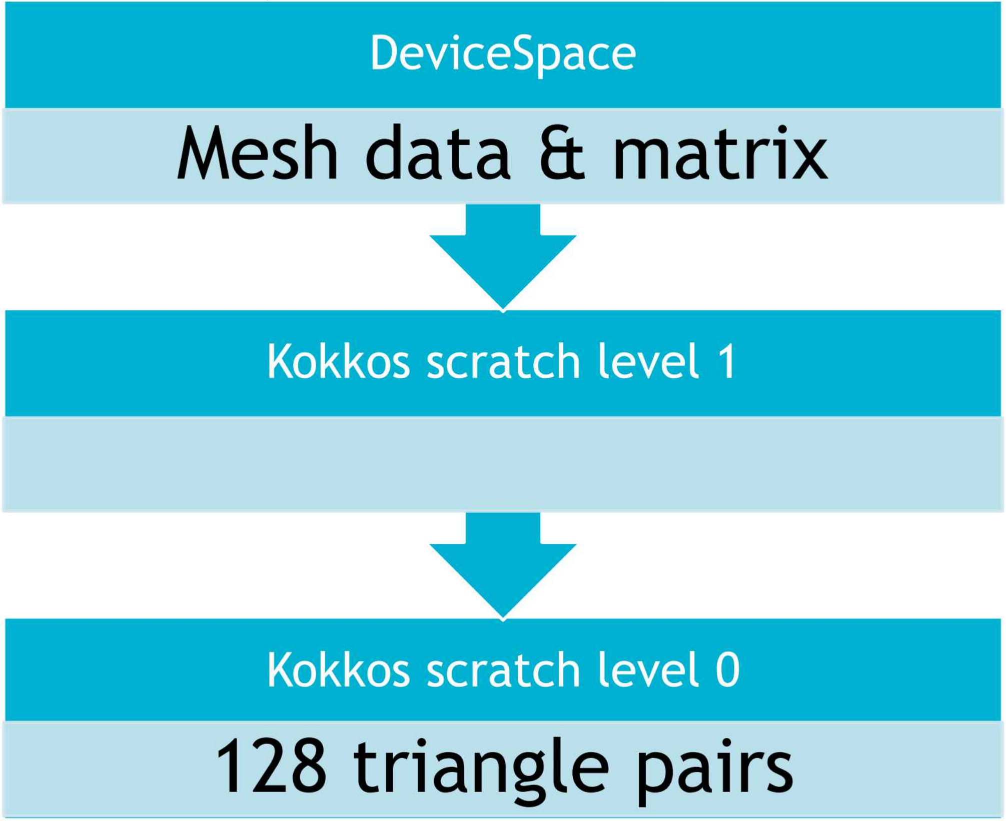
- Since each basis pair is supported by 4 triangle pairs, 32 basis pairs requires 128 triangle pairs, i.e., 20864 bytes in Kokkos scratch level 0.

- With T and S fixed for a single edge of each element, the contribution is still given by

$$[\Lambda_1^T \quad \cdots \quad \Lambda_3^T] \begin{bmatrix} G_{1,1} & \cdots & G_{1,7} \\ \vdots & & \vdots \\ G_{3,1} & \cdots & G_{3,7} \end{bmatrix} \begin{bmatrix} \Lambda_1^S \\ \vdots \\ \Lambda_7^S \end{bmatrix}$$

	Total FLOPs per basis pair (no reuse)
$\Lambda(r)$	2016 (= FLOPs x 21 x 4)
$\nabla \cdot \Lambda$	336 (= FLOPs x 21 x 4)
$G(r, r')$	3276 (= FLOPs x 21 x 4)
$G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$	672 (= FLOPs x 21 x 4)
Elemental mapping	2352 (= FLOPs x 21 x 4)
Total	8652




DeviceSpace

Mesh data & matrix

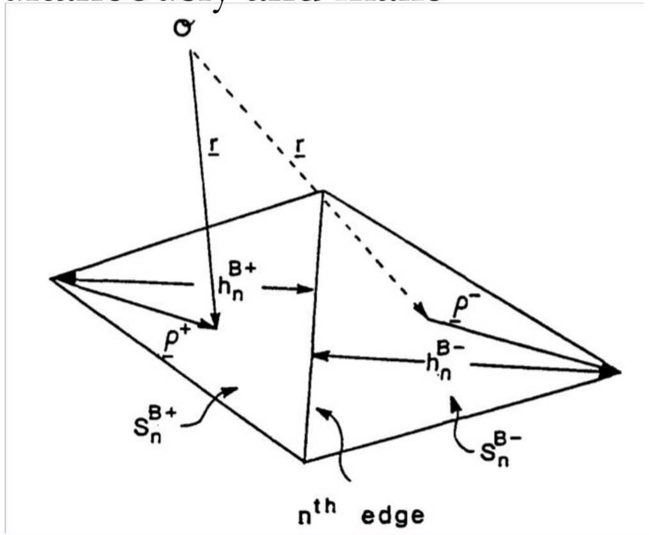
Kokkos scratch level 1

Kokkos scratch level 0

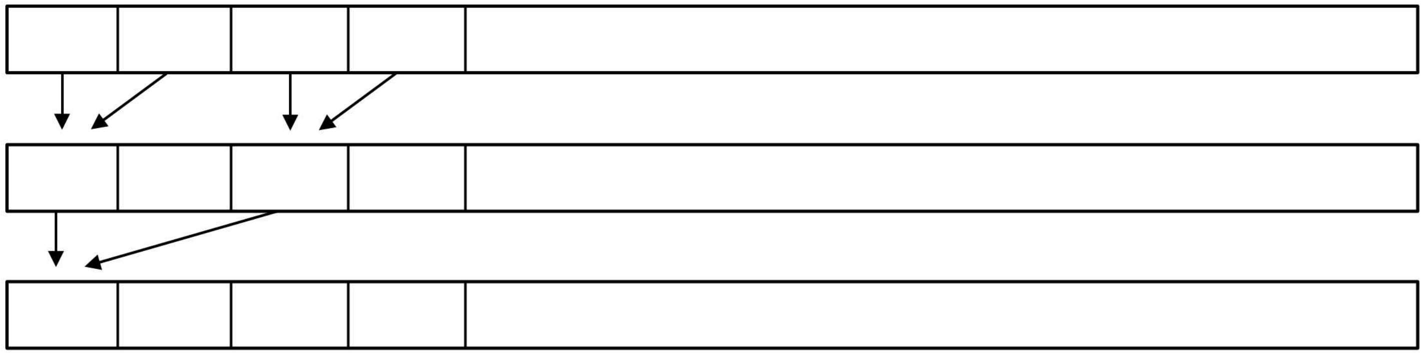
128 triangle pairs

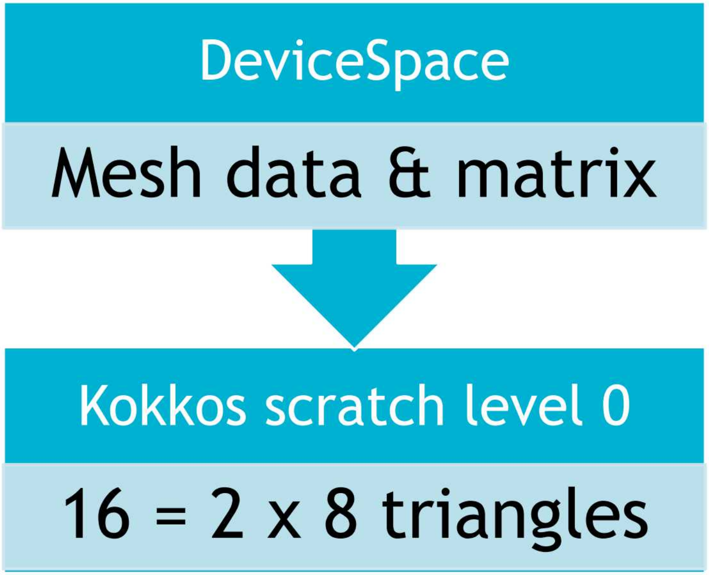
Considering Basis Pairs Making Triangle Pairs on the Fly

- If not reusing data, consider the 4 triangle pairs supporting the basis pair simultaneously and make triangle pairs on the fly.
  - 16 triangles require 48 vertices = 144 doubles = 1152 byes.
  - 128 triangle pairs require 768 vertices = 2304 doubles = 18432 bytes.



Thread	0	1	2	3	4	5	6	7	8	9	10
Test Unknown	6	6	6	6	7	7	7	7	8	8	8
Test Element	+	-	+	-	+	-	+	-	+	-	+
Source Unknown	600	600	600	600	600	600	600	600	600	600	600
Source Element	+	+	-	-	+	+	-	-	+	+	-





DeviceSpace

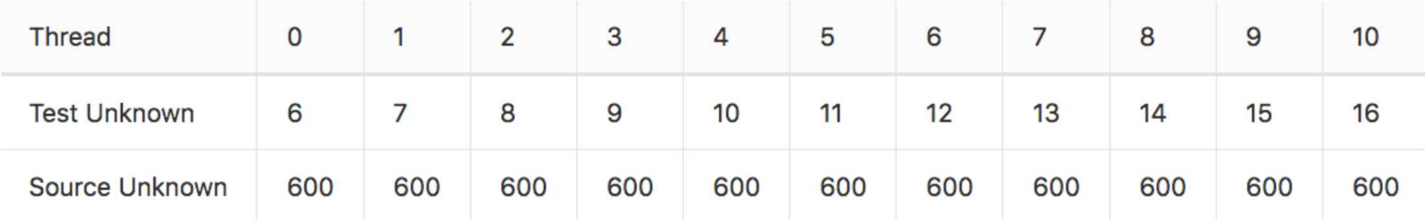
Mesh data & matrix

Kokkos scratch level 0

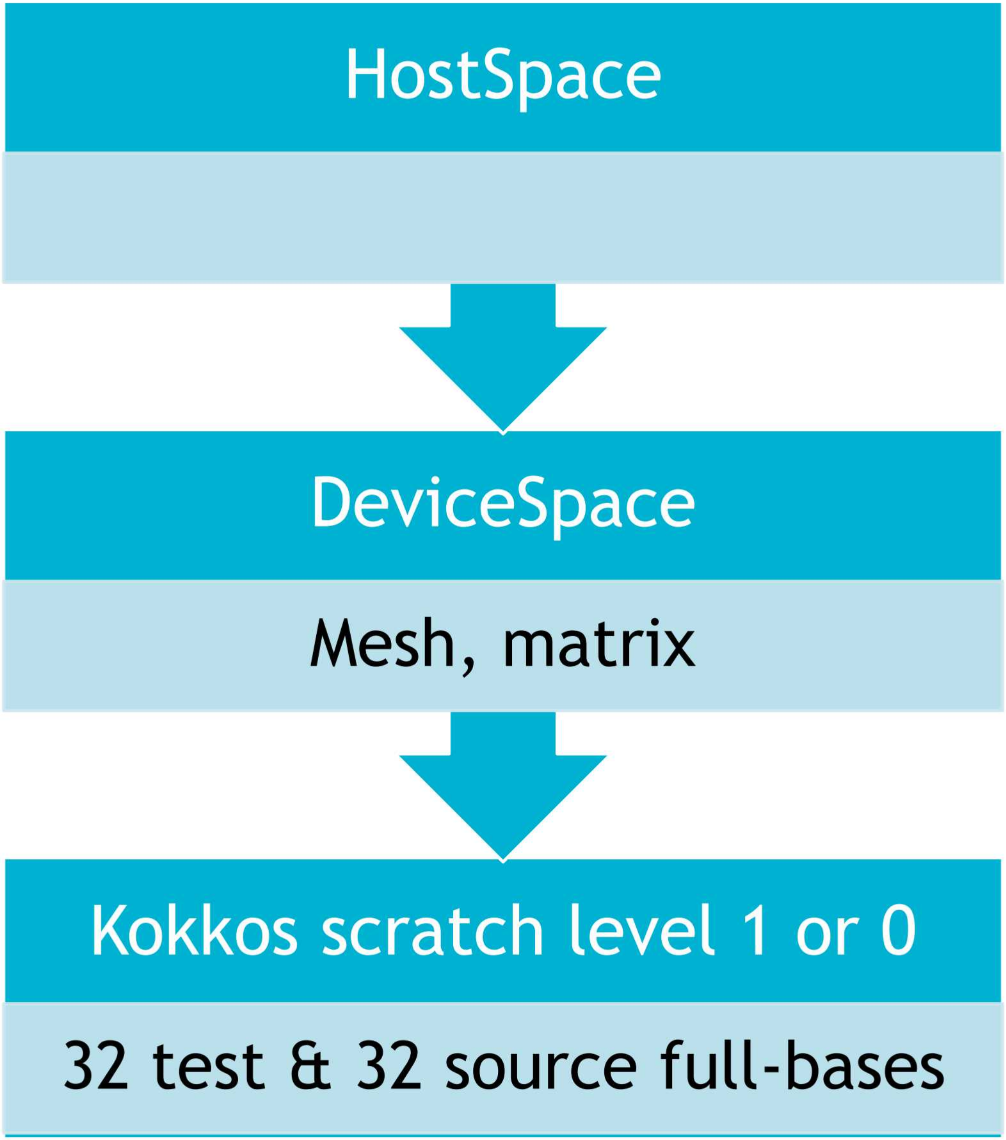
16 = 2 x 8 triangles

Considering Basis Pairs via an Outer Product of Bases

- Instead of loading Basis pairs, load 32 test and 32 source bases to make 1024 basis pairs on the fly.
  - Loading 1024 basis pairs requires 24576 vertices = 73728 doubles = 589824 bytes.
  - Loading 64 bases requires 768 vertices = 2304 doubles = 18432 byes.
- Fills a 32x32 block of the system matrix.
- GPU details: Load the information for 1 test basis and all 32 source bases for a given warp. This is broadcasting and coalesced memory access, respectively.



Thread	0	1	2	3	4	5	6	7	8	9	10
Test Unknown	6	7	8	9	10	11	12	13	14	15	16
Source Unknown	600	600	600	600	600	600	600	600	600	600	600



HostSpace

DeviceSpace

Mesh, matrix

Kokkos scratch level 1 or 0

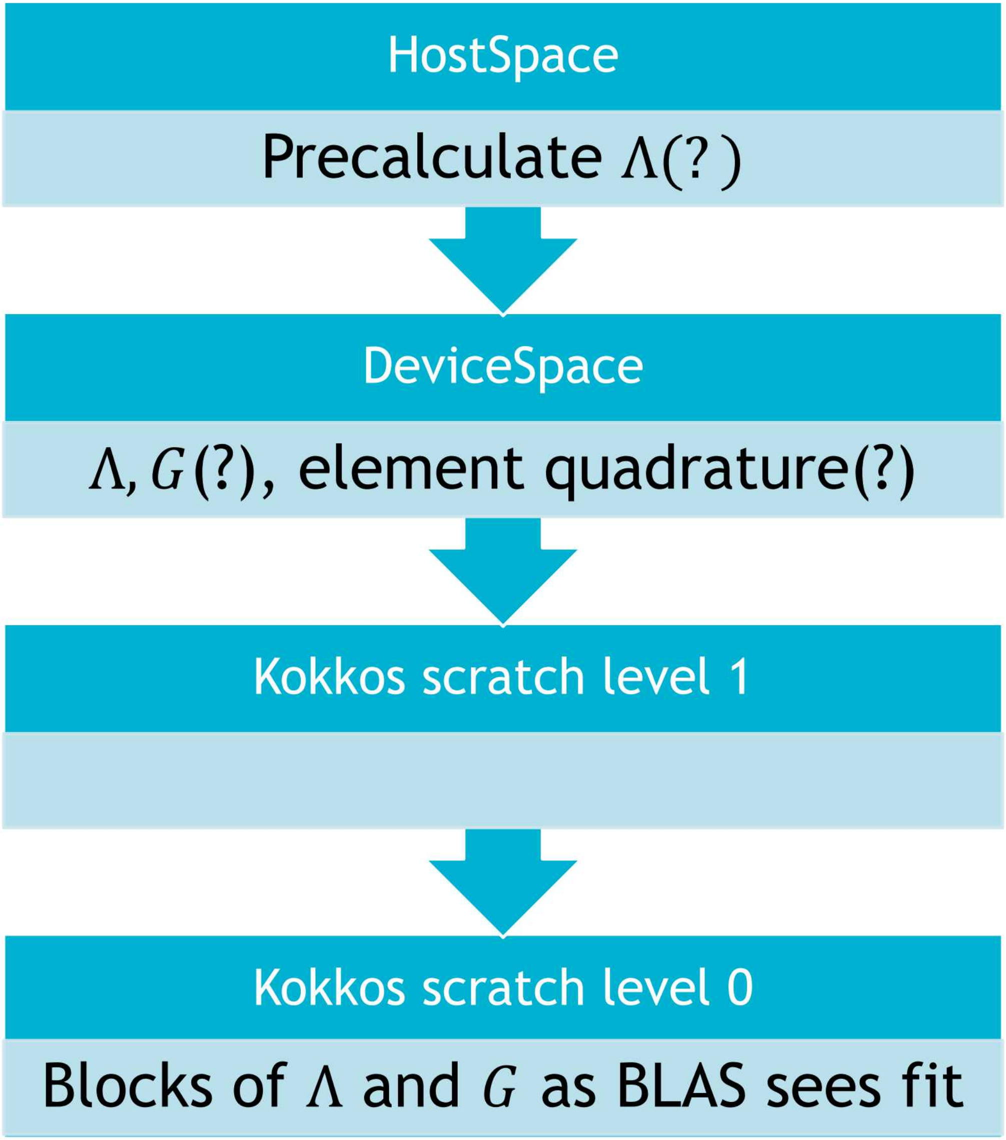
32 test & 32 source full-bases

Considering Basis Pairs via Precomputing and Calling BLAS

- Precompute Λ & G.
- Use BLAS to perform a contraction over some of the dimensions of the following tensors:

$$[\Lambda_1^{Ti} \quad \cdots \quad \Lambda_3^{Ti}] \begin{bmatrix} G_{1,1}^{Ti,Sj} & \cdots & G_{1,7}^{Ti,Sj} \\ \vdots & & \vdots \\ G_{3,1} & \cdots & G_{1,1}^{Ti,Sj} \end{bmatrix} \begin{bmatrix} \Lambda_1^{Sj} \\ \vdots \\ \Lambda_7^{Sj} \end{bmatrix}$$

- Λ_1^{Ti} is a matrix. 1st dimension: (x,y,z) coordinates. 2nd dimension: Full-basis $[Ti]$. Same for all other Λ. Full-bases $[Ti]$ and [Sj] .
- $G_{1,1}^{Ti,Sj}$ is a 3-dimensional tensor. 1st dimension: real and imaginary parts. 2nd and 3rd dimensions:
- Contract over the shown quadrature point dimension and the (x,y,z) coordinates of the Λs.
- For *N* test full-basis and *N* source full-basis, fills a continuous NxN block of the system matrix.
- The *G* tensor contains $41N^2$ doubles and the Λs contain 30 $30N$, where BLAS likes large *N*.
- Opting for instead precomputing quadrature points and basis requires $60N$ instead of $41N^2 + 30N$.



HostSpace

Precalculate $\Lambda(?)$

DeviceSpace

$\Lambda, G(?)$, element quadrature(?)

Kokkos scratch level 1

Kokkos scratch level 0

Blocks of Λ and G as BLAS sees fit

Question: Arithmetic Intensity in an MPI Environment?

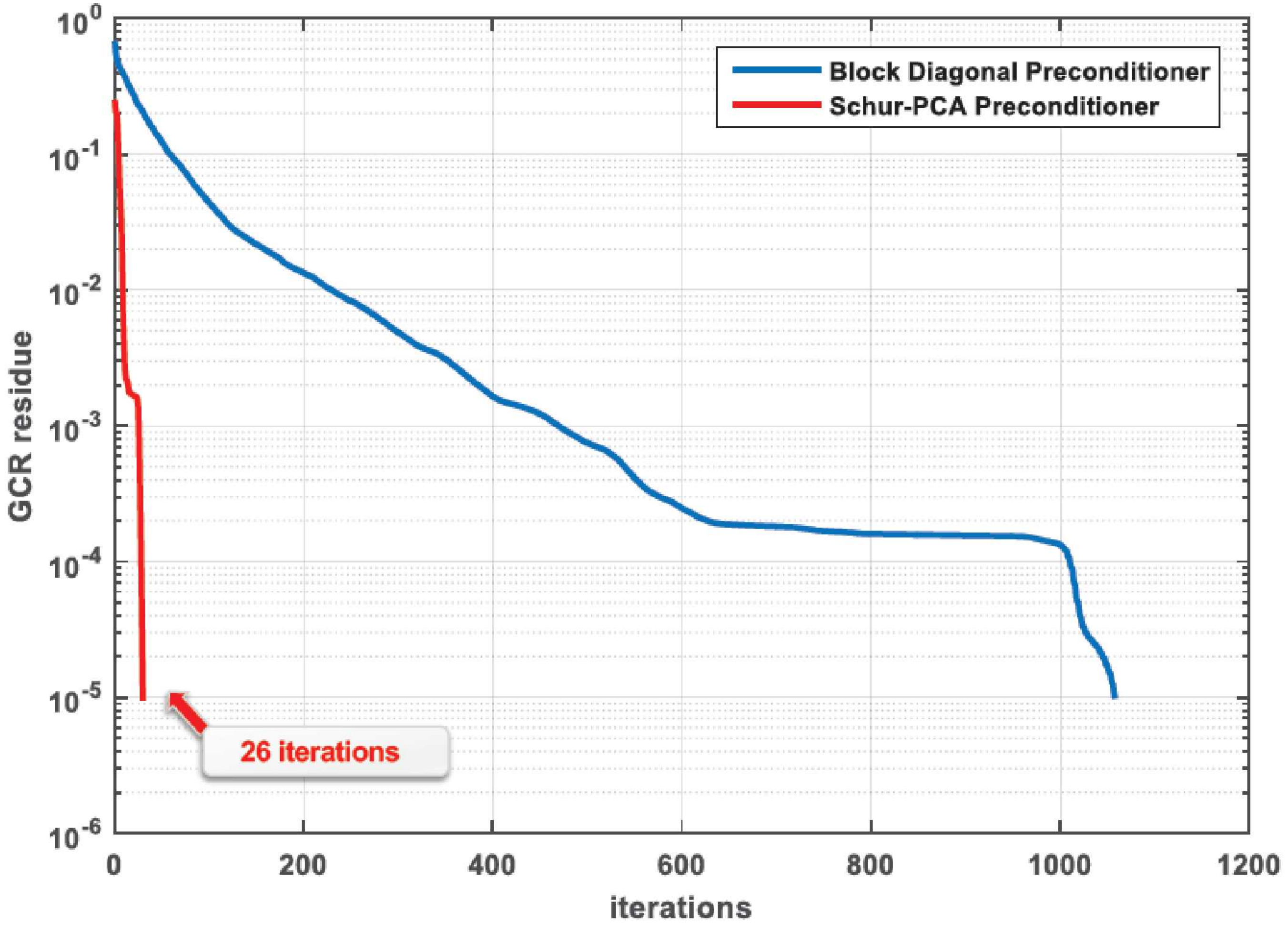
What if Bound by MPI Communication Instead of Memory Bandwidth?

- MPI scatter can be expensive; when is it appropriate?
- MLFMA example
  - Has $O(N \log N)$ complexity and $O(N)$ storage requirement, but that means the complexity is $C * N \log N$ for some constant C. If C is a function of MPI bandwidth when the algorithm is implemented, C can be quite high.

Ideas for increasing Arithmetic Intensity

Idea 1: Decrease Working Precision

- Helps because it lowers the number of bytes but keeps the number of FLOPs constant
- Example:
  - CUBIT default output has error of 1*10^-6
  - Consider the slotted cylinder
    - Matrix has a condition number of 1.1 x 10^{-7}. An LU solver needs double precision to get 5 digits
    - Schur-PCA solver might only need single precision to obtain error of 10^{-4}.
  - Fill with full RWG can lose 1 significant digit due to subtraction after calculating integrals over each element
  - Should we use single precision until we scatter to the system matrix? Can we perform the solve in single precision?



Jin-Fa Lee and Chung Lee, "Schur PC with Octree ", 4/9/2019.

Idea 2: Decrease Memory Requested / Increase the FLOPs

- Reduce accuracy to reduce bytes loaded
  - Low accuracy solution and far coupling calculation require only 1 quadrature point (centroid) on each triangle
  - Store element centroid instead of 3 vertices
- Reuse data stored in the cache to reduce bytes loaded
  - Classical approach: Loop by elements instead of by unknowns
  - More recent: sort unknowns in an FMM like way to increase chance of reuse, but have to avoid bank conflicts
- Use high order basis to increase the FLOPs performed
  - More computationally demanding for loading same element data
  - Don't need to obtain exponential convergence of p-refinement in order to increase arithmetic intensity; just need to do as well as h-refinement

Fin

Backup Slides

Required Bytes

- Previously stored in a cache: 3 test quadrature points and 7 source quadrature points
- Input: 188 bytes = 23 doubles = 6 vertices (18 doubles) + eps (1 complex) + mu (1 complex) + wavenumber (1 complex)
- Output: 16 bytes = 2 doubles (1 complex)

$$= \iint G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$$

- Total bytes read and written for one element pair: 204

Λ's FLOPS

- $\Lambda_0(r) = \frac{r-v_0}{|h_0|} = \frac{(r_x, r_y, r_z)-(v_{0x}, v_{0y}, v_{0z})}{|h_0|}$
  - $|h_0| = \frac{2*area}{|e_0|} = \frac{2*|e_0 \times e_1|}{|e_0|}$ where $e_0 = (v_{2x}, v_{2y}, v_{2z}) - (v_{1x}, v_{1y}, v_{1z})$
  - A cross product requires 3 subtractions and 6 multiplications
  - $\Lambda_0(r)$ requires 9 subtractions, 7 multiplications, and 2 divisions
  - Counting division as 4 flops, $\Lambda_0(r)$ requires 24 flops
- $\nabla \cdot \Lambda_0 = \frac{2}{|h_0|}$
  - $\nabla \cdot \Lambda_0$ requires 1 division given $|h_0|$ from $\Lambda_0(r)$
  - $\nabla \cdot \Lambda_0$ requires 4 flops

G's FLOPS

- $G(r,r') = \frac{e^{-jk|r-r'|}}{4\pi|r-r'|}$
  - 2 divisions: $\frac{e^{-jk|r-r'|}}{4\pi|r-r'|} = \frac{re(e^{-jk|r-r'|})+j\ im(e^{-jk|r-r'|})}{4\pi|r-r'|}$
  - 1 exp and 1 sincos: $e^{-jk|r-r'|} = e^{re(-jk|r-r'|)}\left(\cos\left(im(-jk|r-r'|)\right) + j\sin(im(-jk|r-r'|)\right)$
  - 2 multiplications: $-jk|r-r'| = im(k)|r-r'| - j\ re(k)|r-r'|$
  - 2 multiplications: $4\pi|r-r'|$
  - 3 subtractions and 1 square root: $|r-r'| = \sqrt{(r_x, r_y, r_z) - (r'_x, r'_y, r'_z)}$
  - $\mathrm{G}(r,r')$ requires 3 subtractions, 4 multiplications, 2 divisions, 1 exponential, and 1 square root
  - Counting exp and sincos as 8 flops, $\mathrm{G}(r,r')$ requires 31 flops

Kernel FLOPS

- $G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$
  - 2 multiplications: $G * x = (re(G) + j\, im(G)) * x$
  - 2 additions and 3 multiplications: $\Lambda^T \cdot \Lambda^S = \Lambda_x^T \Lambda_x^S + \Lambda_y^T \Lambda_y^S + \Lambda_z^T \Lambda_z^S$
  - 1 multiplication: $\nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S$
  - The kernel requires 2 additions and 6 multiplications, i.e., 8 flops
- Single quadrature point / weight
  - 6 additions and 9 multiplications: Elemental mapping $\xi_0 v_0 + \xi_1 v_1 + \xi_2 v_2$
  - 3 subtractions and 6 multiplications: Jacobian $|e_0 \times e_1|$
  - 4 multiplications: kernel * quadrature weights * Jacobian
  - Calculating point / weight information requires 6 additions, 3 subtractions, and 19 multiplications, i.e., 28 FLOPS.

Integral Arithmetic Intensity

- Evaluating $G(\Lambda^T \cdot \Lambda^S + \nabla \cdot \Lambda^T * \nabla \cdot \Lambda^S)$ at a quadrature point on the element and multiplying by the corresponding weight requires 8 additions, 15 subtractions, 36 multiplications, and 5 divisions, 1 exponential, and 1 square root, i.e., 95 FLOPS
- Repeating this for 3 test and 7 source points, $\iint G(\Lambda^T \cdot \Lambda^S +$

Example: Arithmetic Intensity Matrix Multiplication

Matrix Multiplication Arithmetic Intensity

- For two 1x1 matrices, one must load 2 doubles and store 1 double to perform the matrix-matrix multiplication. 1 FLOP is performed. The arithmetic intensity is $\frac{1}{24} \approx 0.04$ FLOPs/Byte.
- For two NxN matrices, $3N^2$ doubles (8 bytes each) are moved. Each entry of the resulting matrix requires N multiplications and $N - 1$ additions, i.e., $2N - 1$ FLOPs. Doing this for all N^2 entries of the result, the arithmetic intensity is $\frac{N^2(N-1)}{24N^2} = (N - 1)/24$ FLOPs/Byte.
- For sufficiently large N, the algorithm is compute bound. Is there enough cache to make a large enough N? No.
- For example, consider a V100 GPU with 96 KB of shared memory. Assuming all of it can be used for the three matrices, $96000 = 24N^2$, which gives $N \leq 63$. Limiting ourselves to this size, the best arithmetic intensity for this matrix-matrix multiplication algorithm on a V100 is $\frac{N-1}{24} = \frac{62}{24} \approx 2.6$ FLOPs/Byte.

Gemma Verification and Validation Efforts

May 1-2, 2019

Adam Jones, Aaron Pung, Salvatore Campione, and William Langston, Electromagnetic Theory

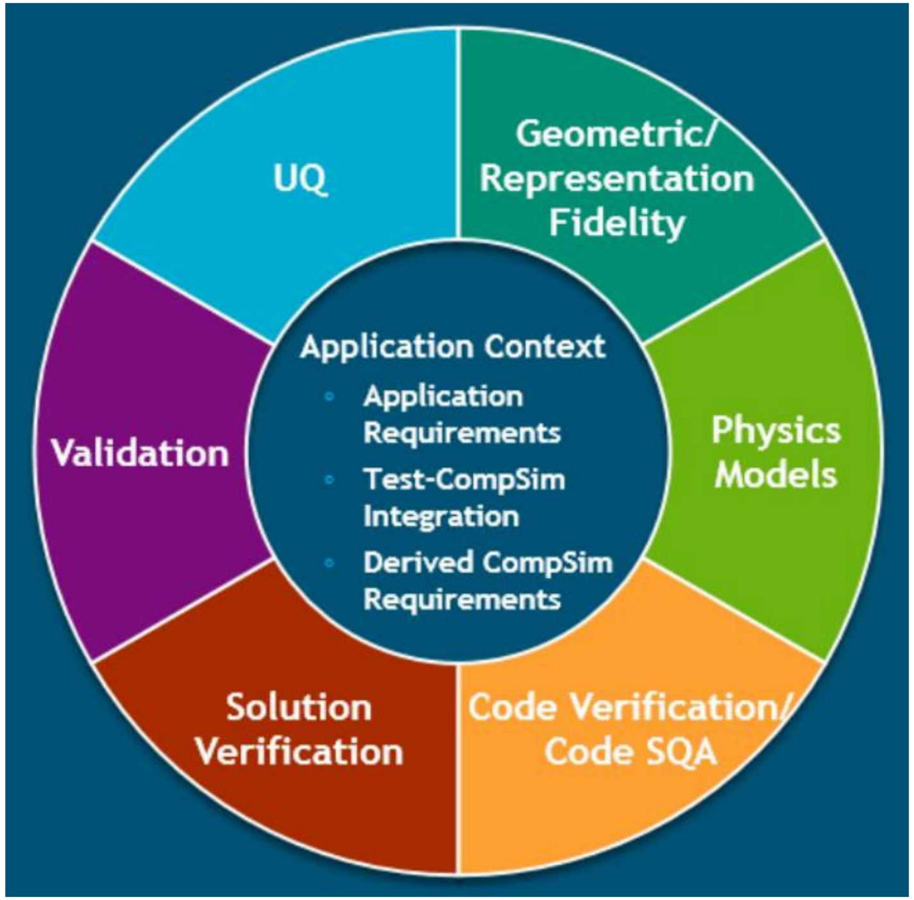
Sandia National Laboratories

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

Export Controlled Information WARNING: This document contains technical data whose export is restricted by the Atomic Energy Act of 1954, as amended, 42 U.S.C. § 2011, et seq. Violations of these export laws are subject to severe criminal penalties

The V&V Process

- An iterative process with feedback between V&V elements
- Uncertainty Quantification
- Geometric/Representation Fidelity
  - Errors arise in using a model/mesh to approximate continuous structures
  - Defeaturing, for example, will introduce geometric fidelity errors
- Physics Models
  - How close are the equations to the physically realized solution?
- Code Verification
  - How well does the code represent the physics model?
- Solution Verification
  - Does the solution converge to the right answer?
  - If so, how rapidly does it converge?
- Validation
  - Comparison of simulation results to experimental data



Overview of RCS Calculations

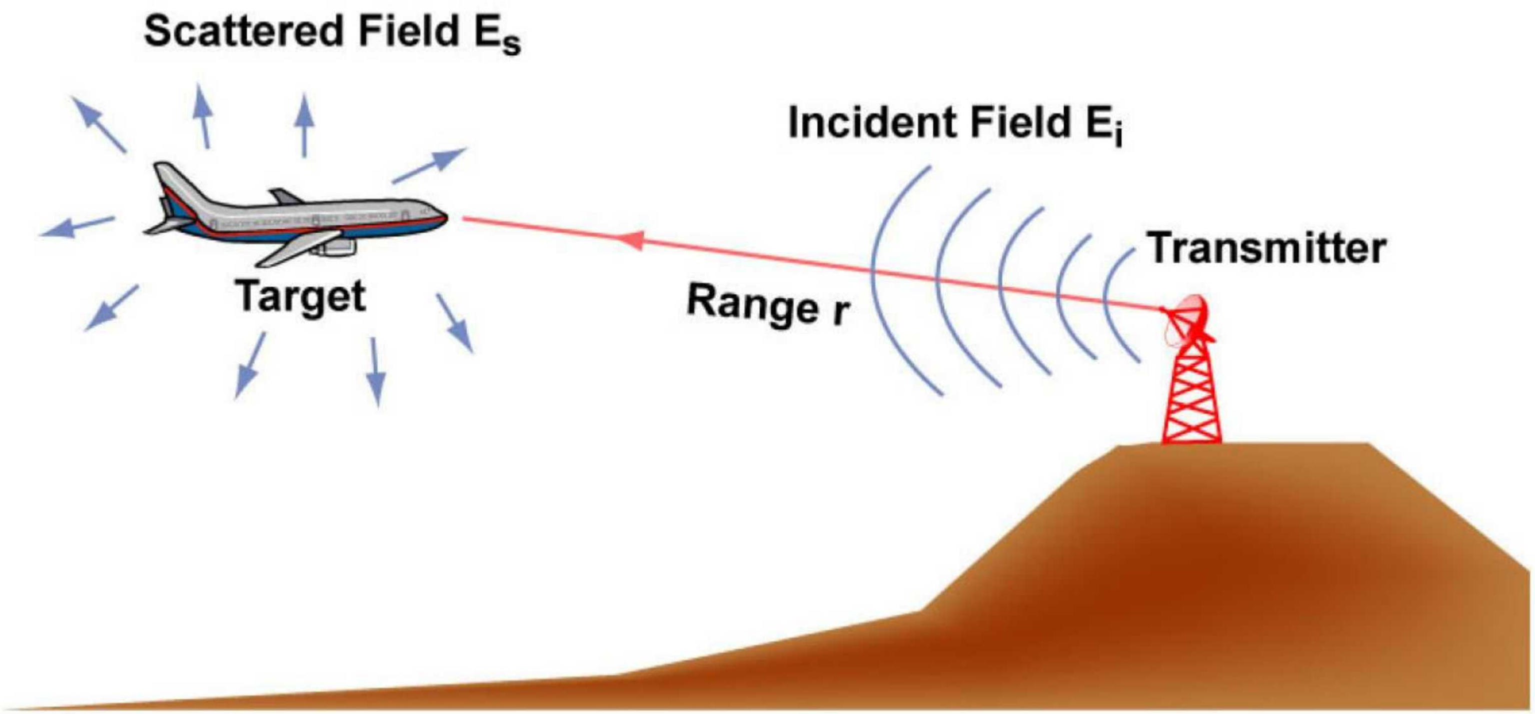


Figure by MIT OCW.

$$\text{RCS} = \lim_{r \to \infty} 4\pi r^2 \frac{|E_s|^2}{|E_i|^2} \qquad \text{(Unit: Area)}$$

Radar Cross Section (RCS) is the hypothetical area, that would intercept the incident power at the target, which if scattered isotropically, would produce the same echo power at the radar, as the actual target.

Overview of RCS Calculations (2)

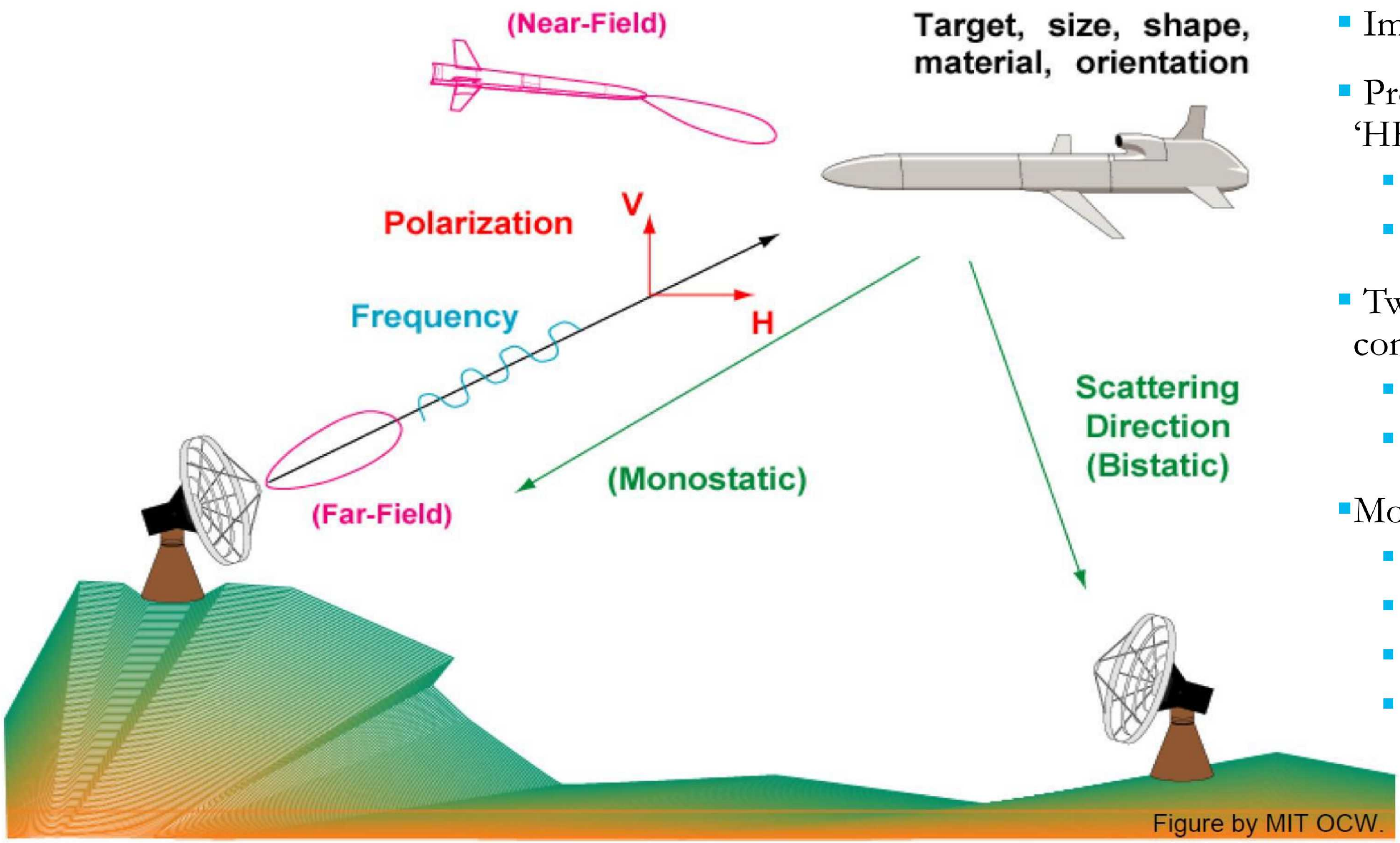
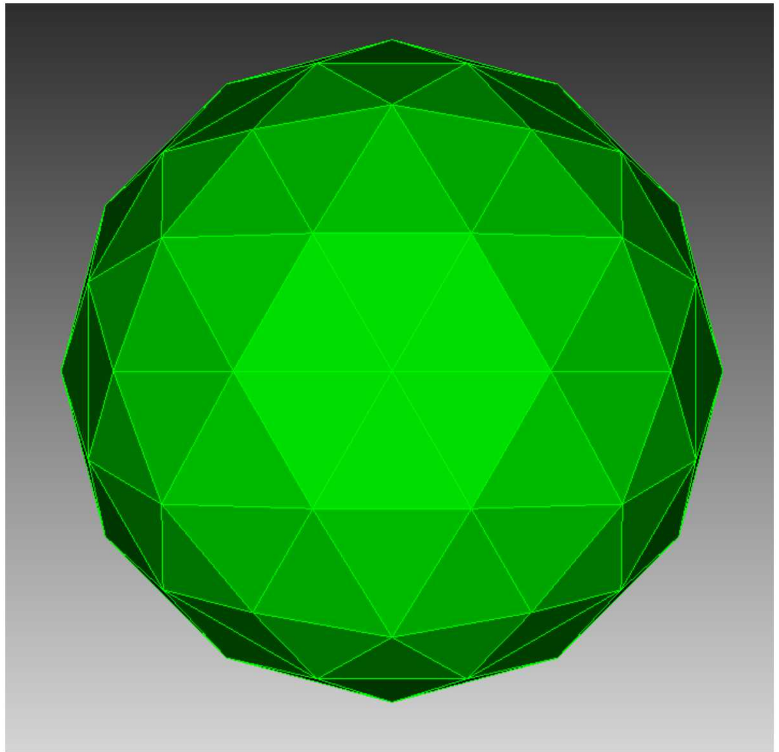

(Near-Field)
Target, size, shape, material, orientation
Polarization
V
H
Frequency
(Far-Field)
(Monostatic)
Scattering Direction (Bistatic)

Figure by MIT OCW.

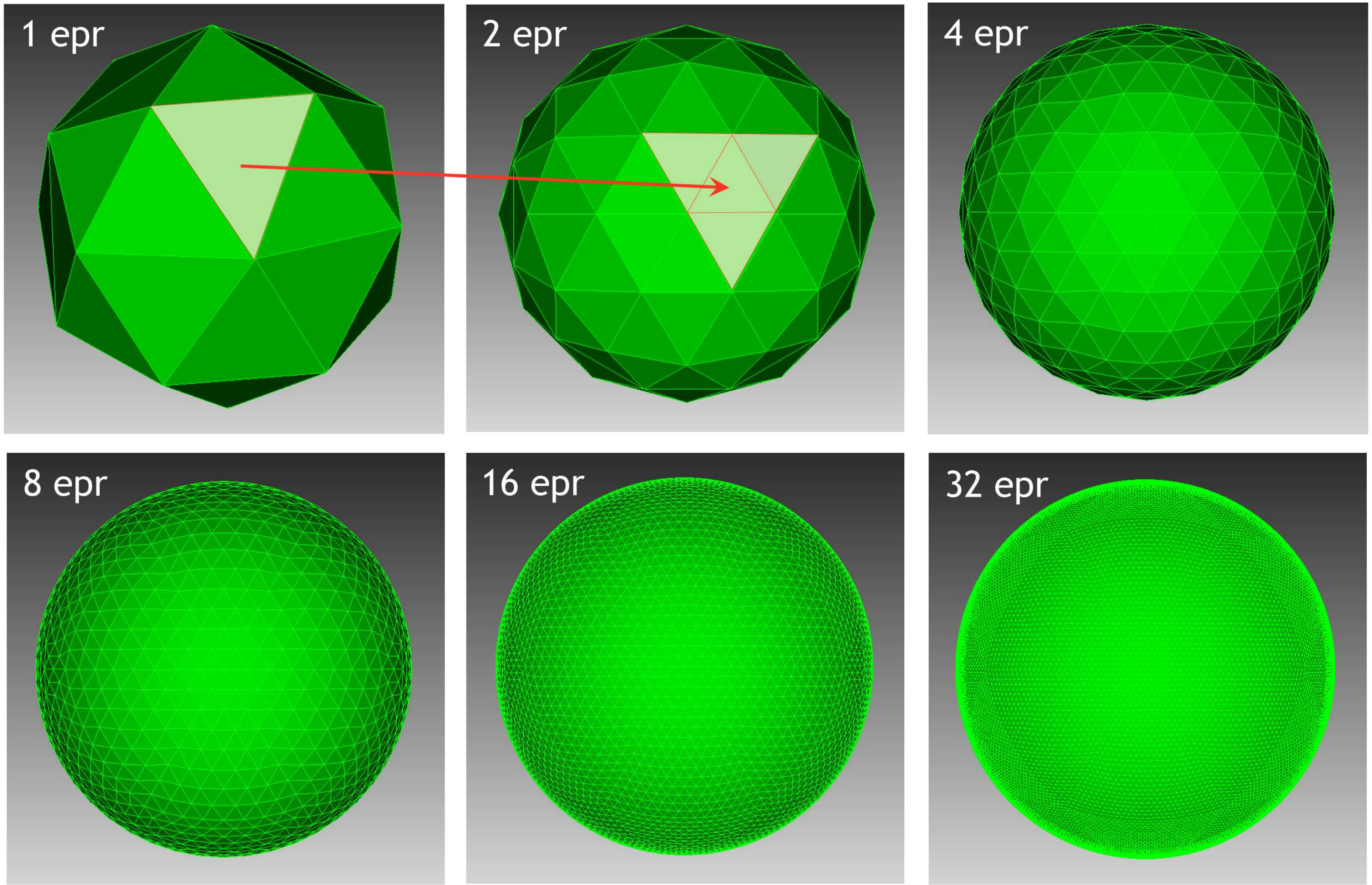
- Image from Ref. [1]
- Presented results are for 'HH'
  - H polarized input
  - H polarized return
- Two problems are considered
  - Cone-sphere
  - Mie Sphere
- More left to the future
  - NASA almond
  - Double ogive
  - Ogive
  - Business card

RCS Problem 2: Mie Sphere

- A perfect PEC sphere with a radius of 1 meter is meshed at different densities to test convergence of the numerical estimate of the RCS value.
- An analytical solution for the Mie Sphere scattering problem exists and will be used, in this case, as our benchmark.
  - Most cases do not have an analytical solution and rely on a) a densely meshed structure and/or b) experimental data
- Useful for identifying numerical instability thanks to the stability of the expected result.
  - HH and VV results should be identical
  - Results should not vary with angle of incidence (phi or theta)
- Squarish mesh used for historical reasons
- Simulations start at low frequency (30 MHz)
- Experimental data not as necessary as other cases
  - Can vary frequency, sphere size, etc. with impunity
- Useful for exploring the parameter space and making comparisons



Mie Sphere: Details of the Meshing scheme

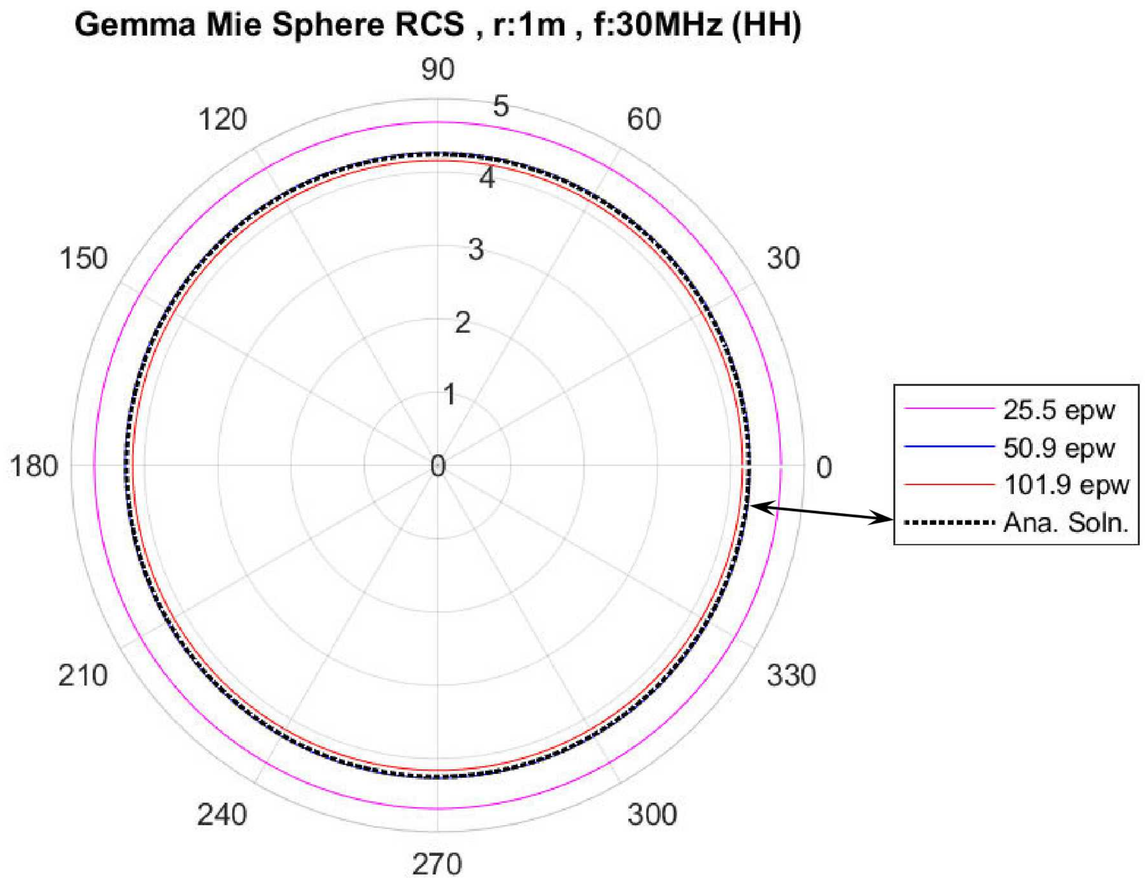


- Radius (R) = 1 meter
- Frequency (f) = 30 MHz
- Initial element edge length ≈ sphere radius (1 epr)
- Mesh densified by mesh refinement; four elements per initial element
- EPR scales as 2^N
- Unknown (edge currents) scales as $13.5 \cdot 4^N$
- Number of elements scales as $9 \cdot 4^N$
- At 30 MHz, $\lambda = 10m$, $EPW = 10 \cdot EPR$

Convergence Study Background

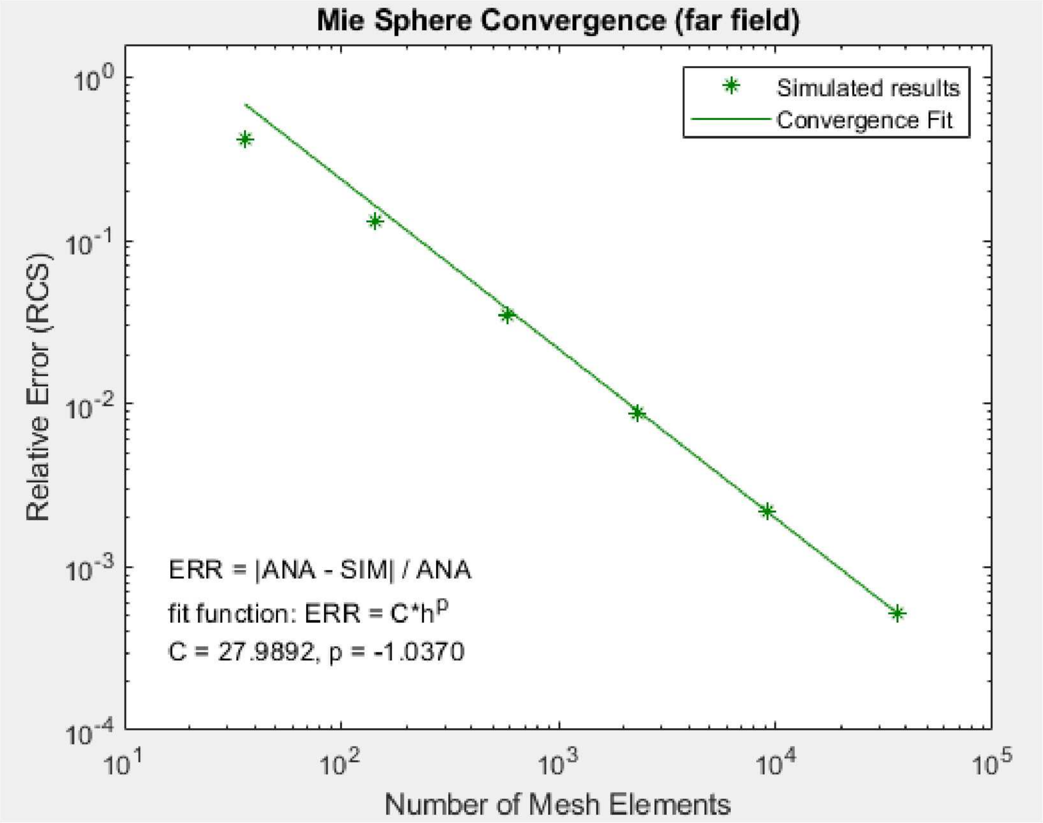
- Order of Accuracy (OA): quantifies the rate of convergence of our numerical solution to the true solution
  - As a function of element size h, we expect the error to follow $E(h) = \left|u - u_{ref}\right| \leq Ch^n$ where n is our convergence rate
- For the Mie Sphere, an analytical solution is used to provide a gold standard to estimate the error
- For other cases (e.g. EMCC structures), the mesh with the greatest number of elements is often used as a reference
- L1, L2, and L infinity norms are used to quantify error in the simulation
  - $L_1 = \frac{1}{N}\sum_{n=1}^{N}|u_n - u_{ref}|$, corresponds to the average absolute error over the evaluation domain
  - $L_2 = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|u_n - u_{ref}|^2}$, Euclidean or RMS error
  - $L_\infty = max|u_n - u_{ref}|$, maximum absolute error in the evaluation domain
- These norms are calculated on the RCS and surface current values and used to evaluate the OA of our solution

Mie Sphere: Polar Plots of the Simulated RCS values



- Incidence angle explored as a variable initially
- As expected, variance as a function of angle quickly goes to zero as the mesh begins to approximate the sphere
- Angle of incidence eliminated as a variable in further simulations

Mie Sphere: Convergence Behavior – RCS



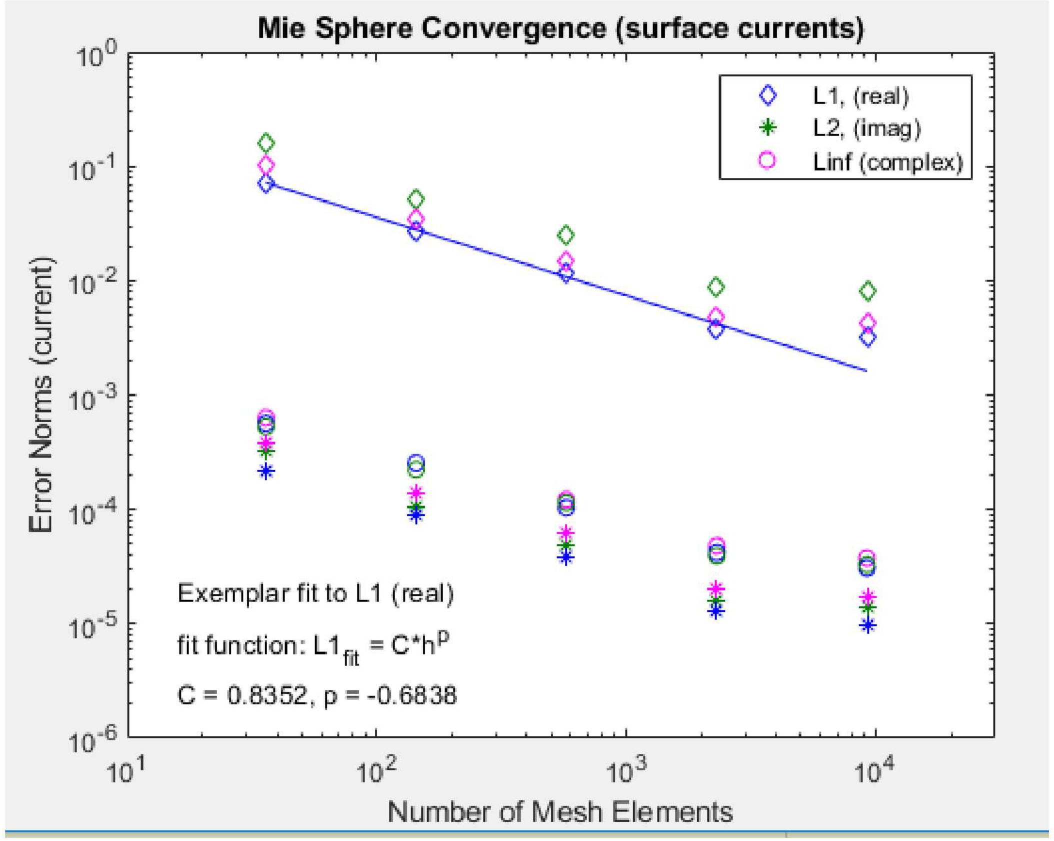
Summary of Results

- Far field RCS values used to compare against the analytical model
- Results are shown to be 1st order accurate (as expected)
- Relative errors down to 5e-4 interrogated thusfar

Future Work

- Extend study to increasingly dense meshes in an attempt to interrogate the floor of the error values
- Explore additional polarization components to verify their behavior
- Evaluate behavior across the frequency band of interest

Mie Sphere: Convergence Behavior – Surface Currents



Summary of Results

- Norms (L1, L2, Linf) used to evaluate error in the surface current values of the simulation
- Real, imaginary, and complex current values are separately evaluated as they could show different behavior based on the algorithms employed; here, they show the similar trends

Future Work

- Extend study to increasingly dense meshes in an attempt to interrogate the floor of the error values
- Explore additional polarization components to verify their behavior
- Evaluate behavior across the frequency band of interest

Mie Sphere: Behavior as a Function of Frequency (from Ref. 1)

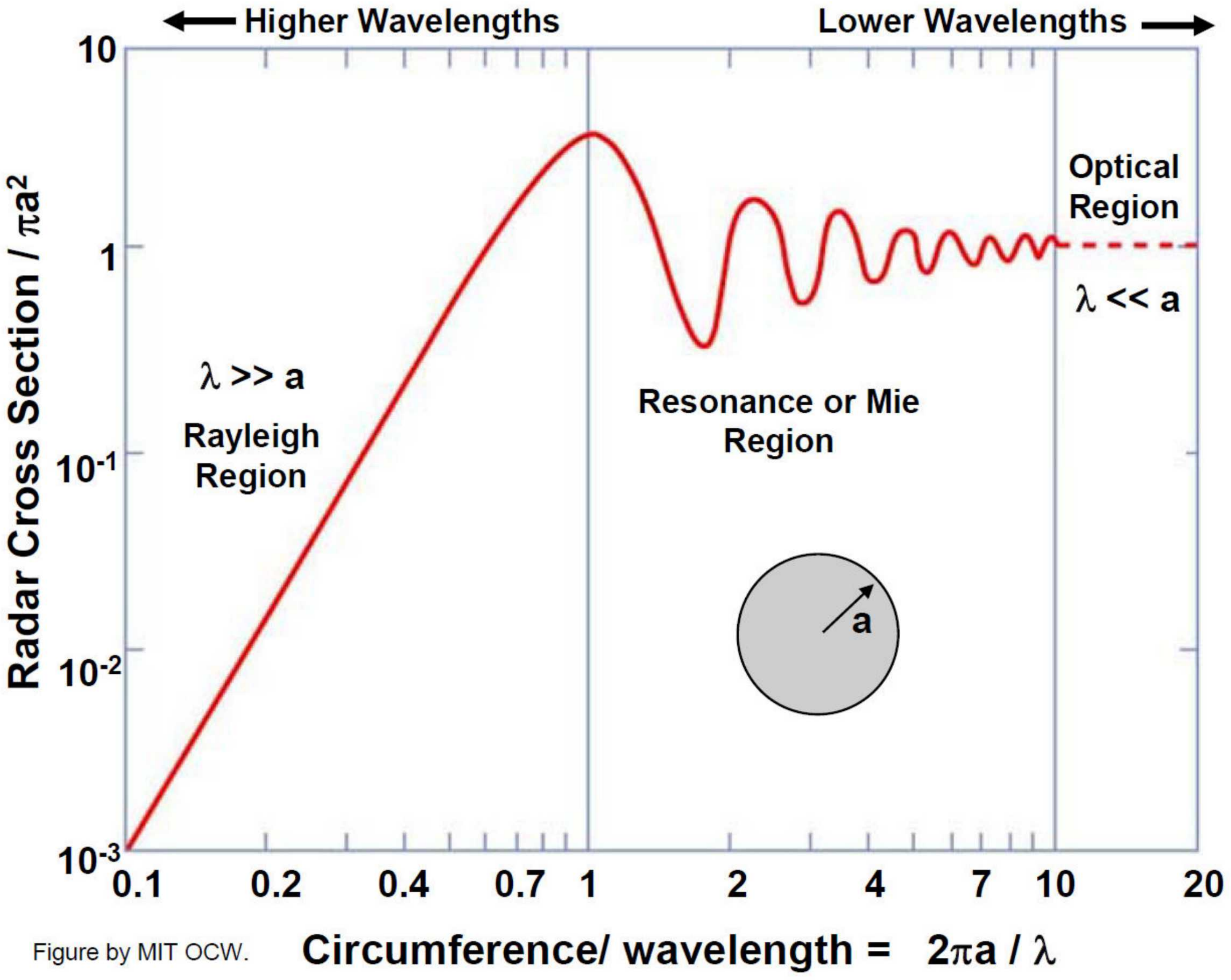


Figure by MIT OCW.

Rayleigh Region

$\lambda >> a$

$\sigma = k / \lambda^4$

Mie or Resonance Region

Oscillations

Backscattered wave interferes with creeping wave

Optical Region

$\lambda << a$

$\sigma = \pi a^2$

Surface and edge scattering occur

Simulation geometry

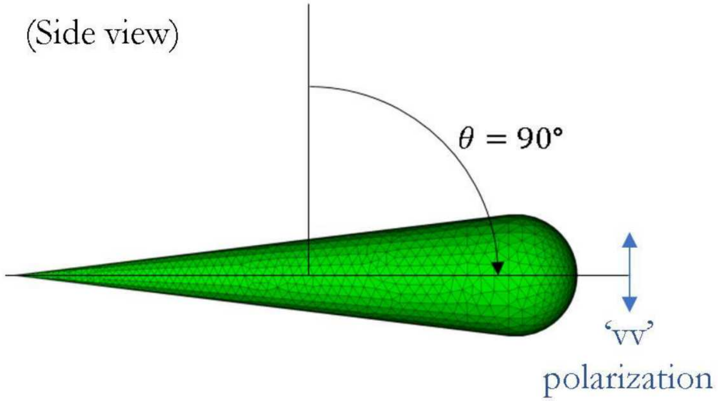
The source strikes the cone-sphere geometry normal to the spherical region ($\theta = 90°$).

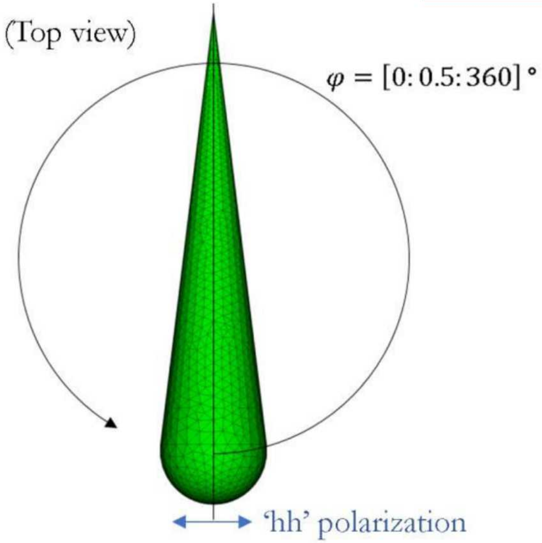
As the simulation progresses, the source is swept in φ from 0° to 360° in 0.5° increments.

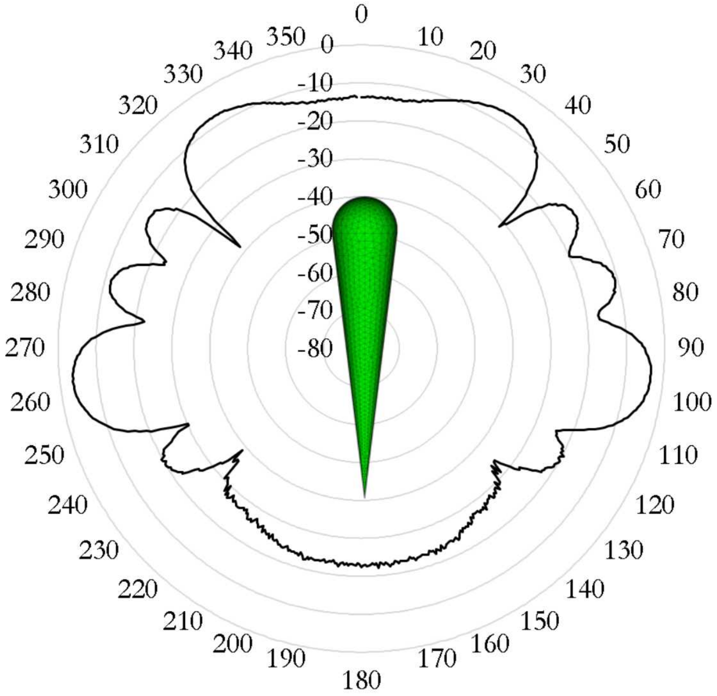
"hh polarization" indicates both the incident and analyzed RCS signal are horizontally polarized.

This produces a far-field plot of radar cross-section (RCS) in decibels per square meter [dBsm] vs φ.

In comparison to the Mie sphere, this provides solution verification, since an analytic solution is unavailable.





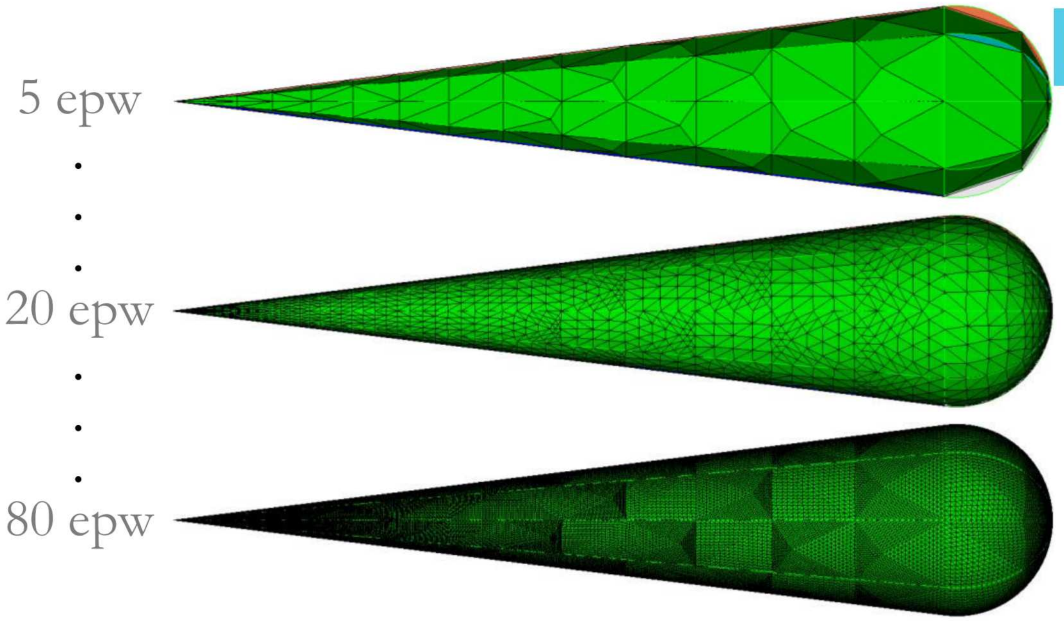


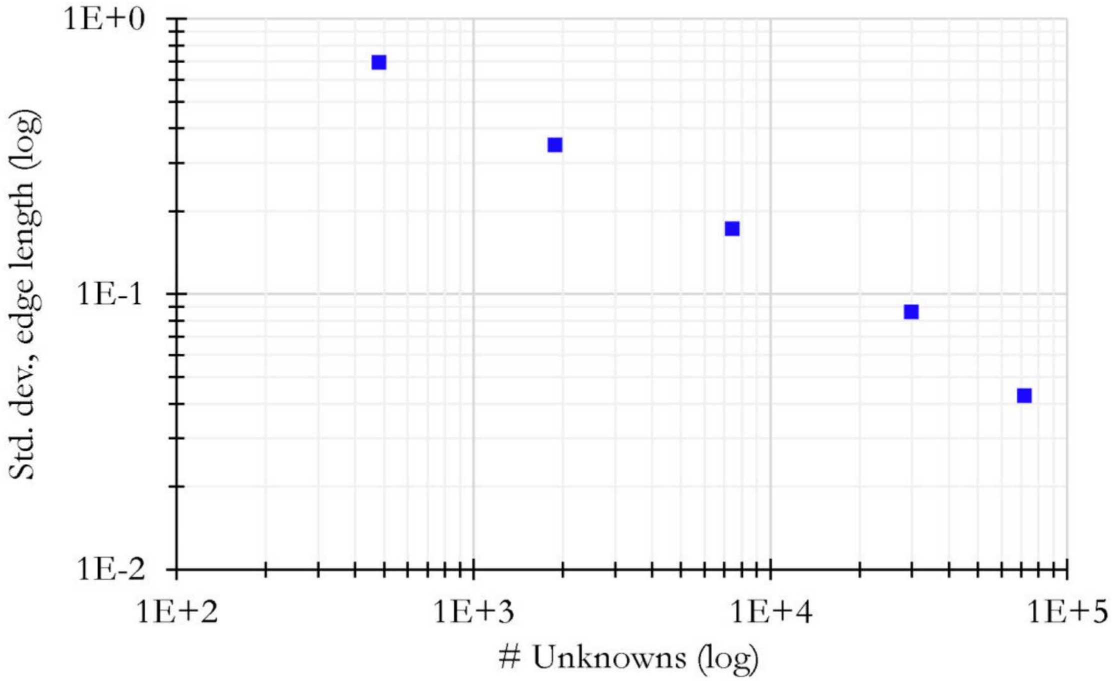
Mesh density

For a single source frequency, mesh density is increased from 5 elements per wavelength (epw) to 80.

In each subsequent mesh, mesh density increases by a factor of 2 using CUBIT's "refine" function, ensuring the original mesh nodes are present in the highest mesh density.

Coarse meshes produce large variation in element sizes; as mesh density increases, greater uniformity is seen in the edge lengths.



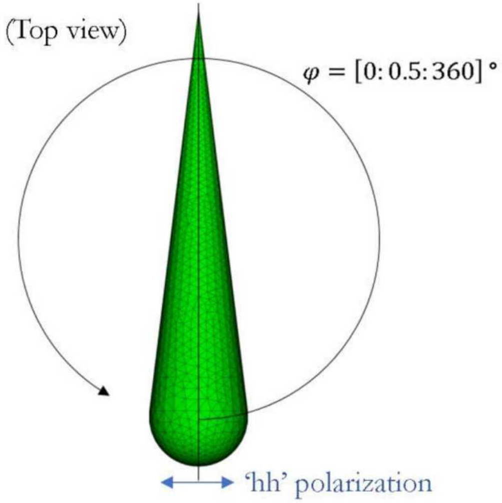


RCS Error

Using the 40epw mesh as a “golden standard”, RCS error was computed for the 5, 10, and 20epw mesh densities using:

$$ERR = \frac{|40epw - Current\ mesh|}{|40epw|}$$

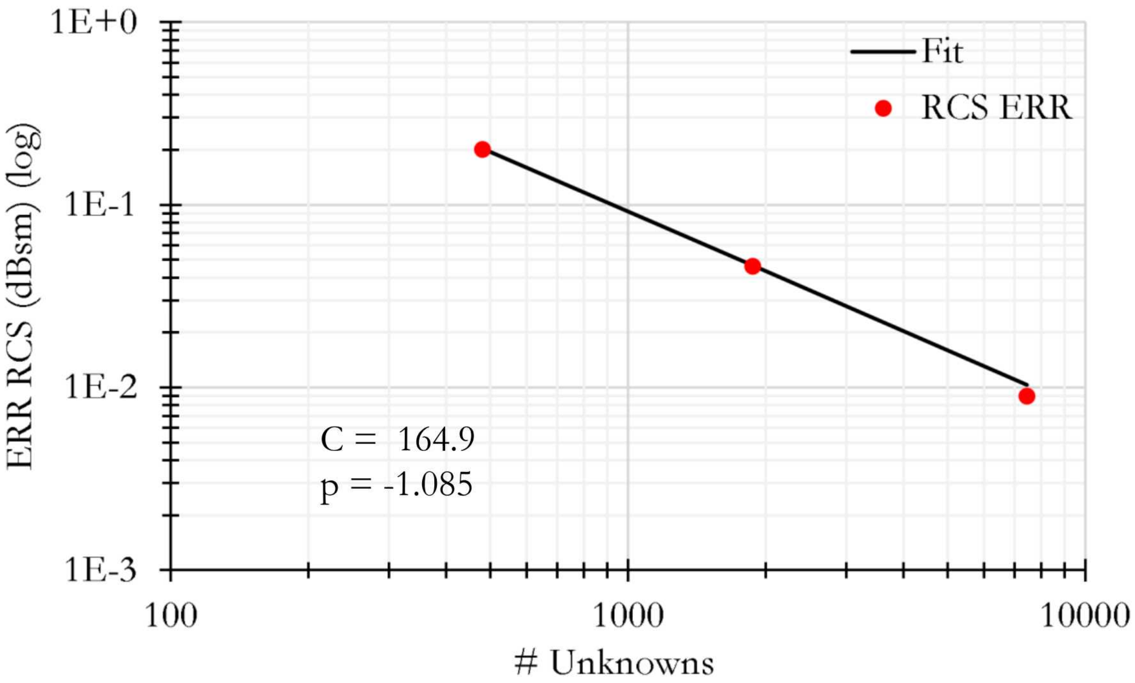
and fit using $ERR = C \cdot epw^p$, as shown in the plot below.



Future studies will examine the convergence of the surface currents using the L_1 and L_∞ norms, similar to the Mie sphere:

$$L_1\ Error = \frac{\iint_S |J_{40}(\boldsymbol{r}) - J_{sim}(\boldsymbol{r})|\, dS}{\iint_S |J_{40}(\boldsymbol{r})|\, dS}$$

$$L_\infty\ Error = \frac{|\max(|J_{40}(\boldsymbol{r})|) - \max(|J_{sim}(\boldsymbol{r})|)|}{\max(|J_{sim}(\boldsymbol{r})|)}$$



Backup Slides

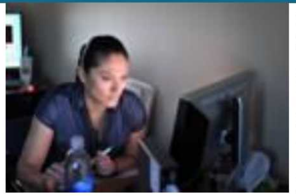
References

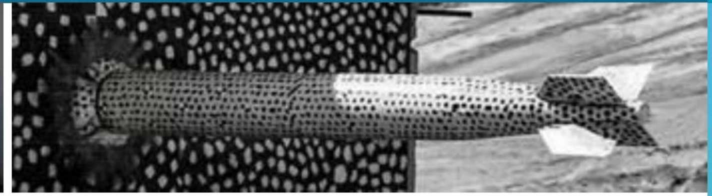
1) R.M. O’Donnell, ‘Radar Systems Engineering Lecture 7 – Part 1, Radar Cross Section’ IEEE New Hampshire Section 2010
2) FEKO application note: Benchmark Radar Targets for the Validation of Computational Electromagnetics Programs (2010)
3) R.E. Jorgenson and J.D. Kotulski, “A Set of Verification Test Cases for EIGER: Plane Wave Scattering from a Sphere,” SAND2004-4816

Mie Sphere: Mesh Characteristics

N	EPR	# Unknowns	Mesh Elements	EPW
1	1	54	36	10
2	2	216	144	20
3	4	864	576	40
4	8	3456	2304	80
5	16	13824	9216	160
6	32	55296	13824	320

Sensitivity Analysis Effort in High-Quality Factor Cavities





PRESENTED BY

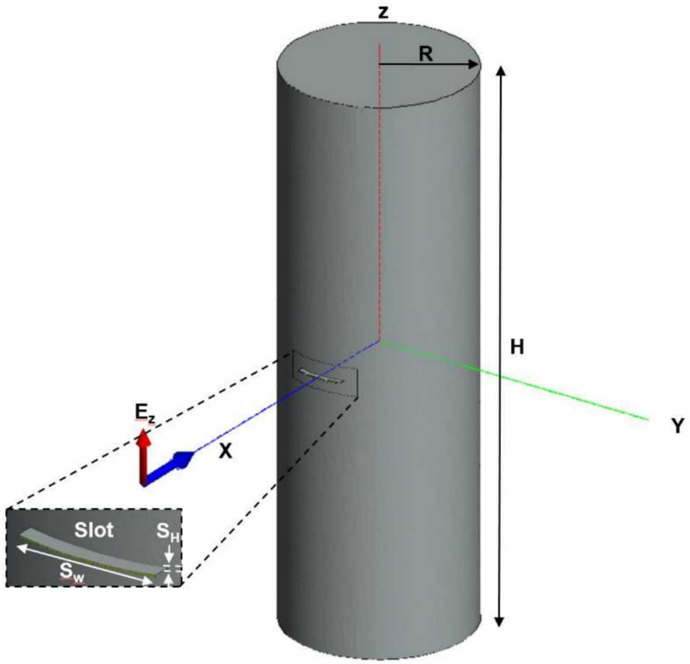
Salvatore Campione

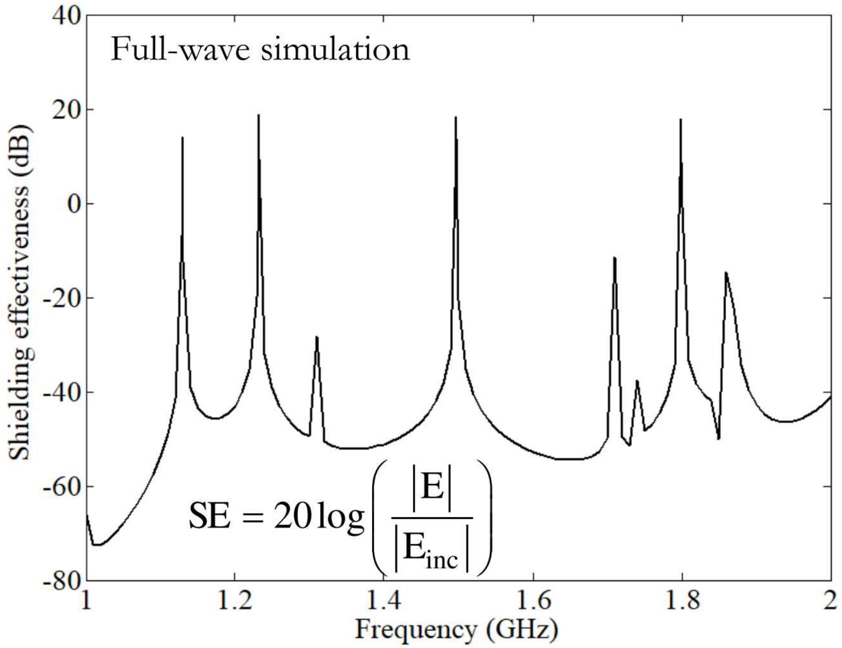
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

- Motivation
- EIGER Simulations / Rational interpolation
- Sensitivity Analysis
  - Methods
  - Canonical Cavity Code and Results
  - Power Balance Matched Bound Code and Results
- Unmatched Power Balance Case
- Comparison with Experiments Toward Validation
- Conclusions/Future Work

Motivation: Canonical Cylindrical Cavity Example





$$SE = 20\log\left(\frac{|E|}{|E_{inc}|}\right)$$

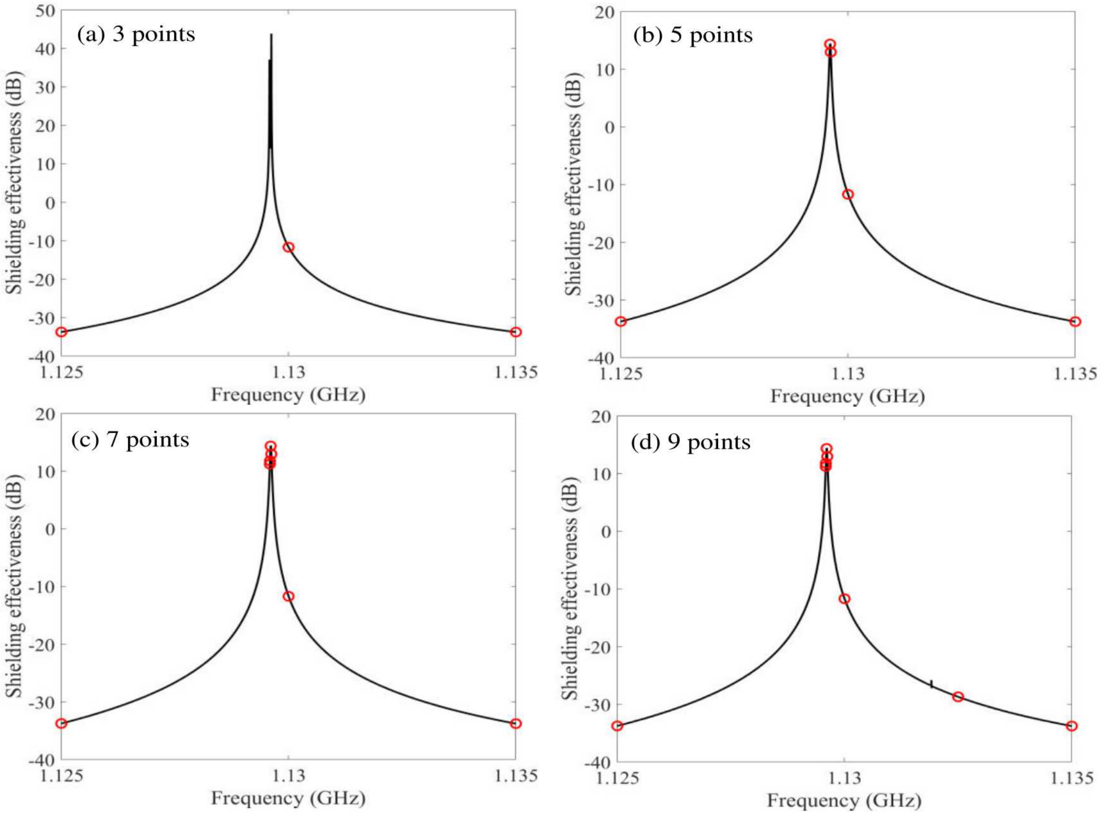
- External fields couple through the slot → Large fields (and thus large SE) >> 0 dB are observed near resonant modes.

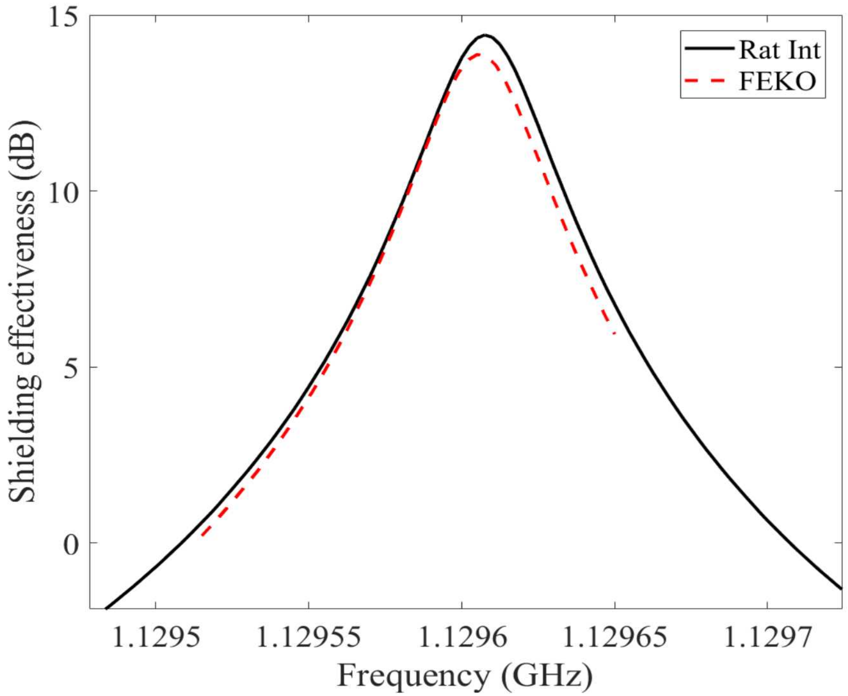
- These deterministic full-wave simulations are very sensitive to the parameters used (e.g. geometry, materials, etc.), and require many frequency points to resolve the high-Q resonant peaks.

We aim to determine, through a sensitivity analysis, the parameters that largely affect SE, with the goal of achieving a range determination and distribution

EIGER and Rational Interpolation

- High-Q resonances may require >100 frequency simulations to fully determine peak value and quality factor
- However, the need of large frequency discretization may hinder a sensitivity analysis where enough samples are required
- We applied a rational interpolation algorithm from OSU where only a handful of points (<20) are needed to resolve a high-Q resonance. Using HPC machines, a resonant peak is resolved in 10 minutes run time per frequency point





Sensitivity Analysis: Basics

- Sensitivity analysis is a way to identify which uncertain inputs are responsible for the variation in the output. There are many ways to assess input sensitivity. Below are two methods that were used:

1) **One-at-a-time variation:** inputs are varied one-at-a-time with all other inputs held fixed.

   - This provides a measure of the variation of the output, but it does not allow for interactions between inputs

2) **Variance Decomposition:** inputs are varied at the same time and the amount of variation in the output is decomposed and attributed to each input. Two measures are calculated:

   - **First-order sensitivity indices** - the proportion of the uncertainty in the output that is explained by the uncertainty in a single input.

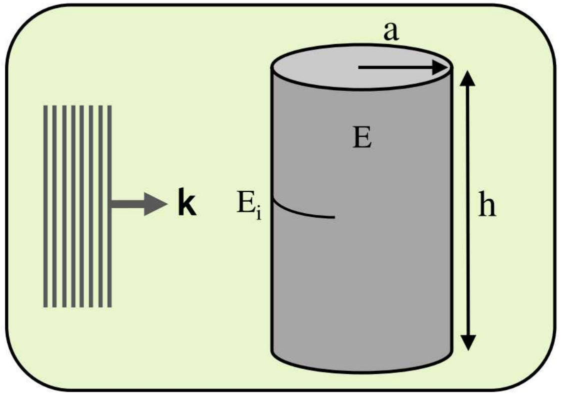
   - **Total-order sensitivity indices** - the proportion of the uncertainty in the output that is explained by the uncertainty in an input and its interactions with other inputs.

Sensitivity Analysis: Two Codes Considered

- Canonical cavity code
  - Uses an analytical code for fully-enclosed, highly-resonant cavities
  - Initially implemented to build the sensitivity analysis framework to the cavity problem and gauge Q factor, E field and frequency sensitivity vs cavity radius, height and conductivity
- Power balance code
  - Uses matched bound formulation
  - Allows for the coupling problem to be analyzed, bringing in the slot dimensions in the sensitivity analysis, in addition to cavity radius, height and conductivity

Canonical Cavity

- Assuming that the slot acts as an excitation source for the modes, but does not perturb the resonant modes



- Cylindrical cavity resonant TM mode frequencies (Closed cavity)

$$\omega_{m,p,n} = \frac{1}{a\sqrt{\mu\varepsilon}}\sqrt{\left(j_{m,p}^2 + \left(\frac{n\pi a}{h}\right)^2\right)}$$

- Cylindrical cavity quality factor (Closed Cavity)

$$Q_{m,p,n} = \omega_{m,p,n}\frac{W_{m,p,n}}{P_{m,p,n}} = \frac{\eta}{2R_S}\frac{\sqrt{\left(j_{m,p}\right)^2 + \left(n\pi a/h\right)^2}}{1+\delta_n a/h}$$

- Field expressions (Closed cavity)

$$E_\rho = -\frac{1}{k^2-\alpha^2}A\frac{j_{m,p}}{a}J'_m\left(j_{m,p}\rho/a\right)\cos\left[m\left(\phi-\phi_0\right)\right]\frac{n\pi}{h}\sin\left(n\pi z/h\right) \quad 0<\rho<a,\ 0<z<h$$

$$E_\varphi = \frac{1}{k^2-\alpha^2}\frac{1}{\rho}AJ_m\left(j_{m,p}\rho/a\right)m\sin\left[m\left(\phi-\phi_0\right)\right]\frac{n\pi}{h}\sin\left(n\pi z/h\right) \quad 0<\rho<a,\ 0<z<h$$

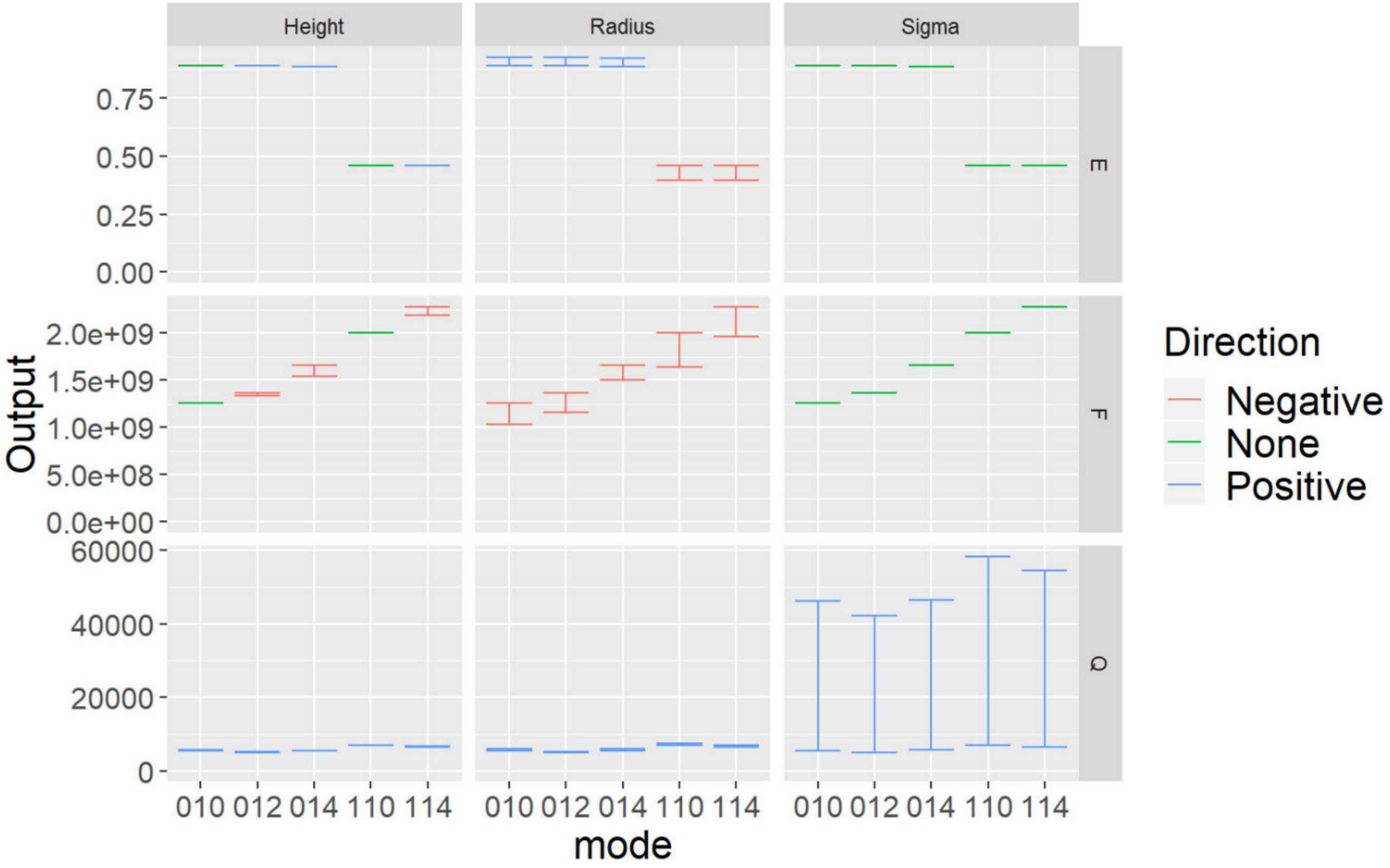
$$E_z = AJ_m\left(j_{m,p}\rho/a\right)\cos\left[m\left(\varphi-\varphi_0\right)\right]\cos\left(n\pi z/h\right) \quad 0<\rho<a,\ 0<z<h$$

$$|E| = \sqrt{|E_\rho|^2 + |E_\phi|^2 + |E_z|^2}$$

Sensitivity Analysis: Results (Canonical Cavity Code)

Inputs: Radius, Height and Conductivity

Output 1: E field, frequency, Q factor



E field and frequency most sensitive to radius, Q factor most sensitive to conductivity.

Sensitivity Analysis: Results (Canonical Cavity Code)

Inputs: Radius, Height and Conductivity

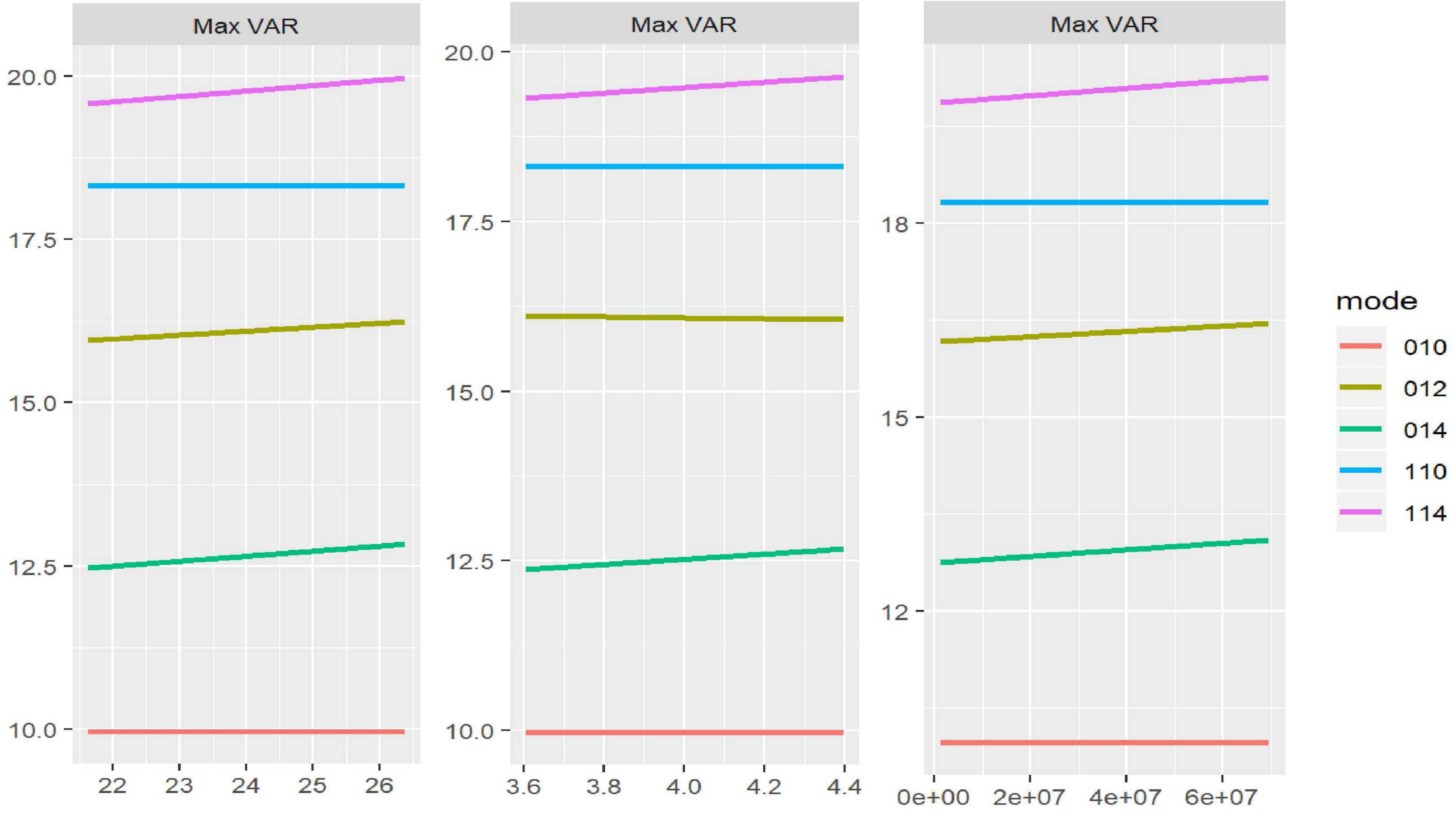
Output 2: VAR $VAR = {|E|^2}/{\langle |E|^2 \rangle}$

Cavity Height
[21.6in, 26.4in]

Cavity Radius
[3.6in, 4.4in]

Cavity Sigma
[1e6, 7e7]

VAR not very sensitive to radius, height or conductivity.



- These do not take into account the coupling problem

Power Balance Bounding Method (Matched Bound)

- The matched bound has been computed using conservation of power arguments
  - ➢ Assumes that the matched received power of the aperture is delivered to be absorbed by the interior cavity walls

$$\sigma_{\text{rec}} S_0 = \sigma_{\text{wall}} S$$

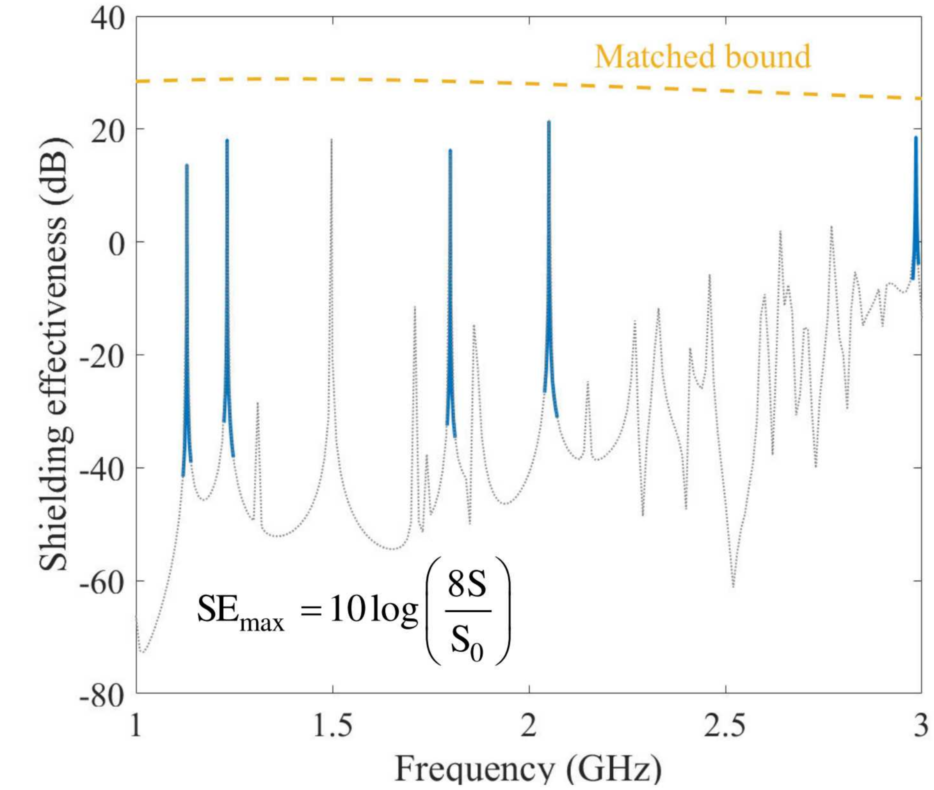
σ_{rec} = matched slot cross section

S_0 = incident power flux density

σ_{wall} = wall loss cross section

S = interior power density

  - ➢ Additional loss mechanisms can be accounted for to further improve the bound



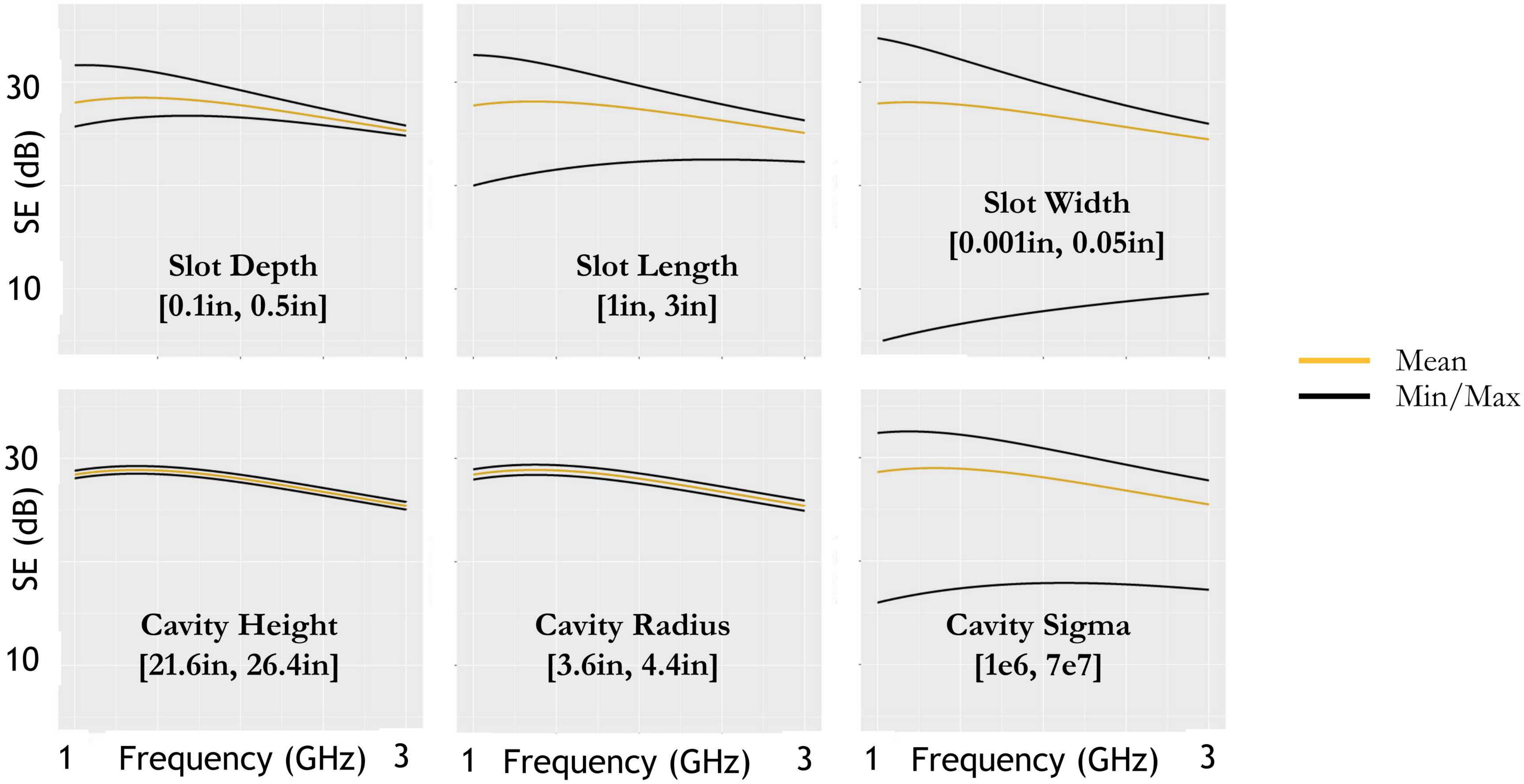
$$SE_{\max} = 10\log\left(\frac{8S}{S_0}\right)$$

- Cavity SE is always below this bound, providing guidelines for maximum achievable levels of interior fields.

We use this power balance bound to perform a sensitivity analysis to determine the parameters that lead to large SE variations

Sensitivity Analysis: Results (Power Balance Matched Bound)

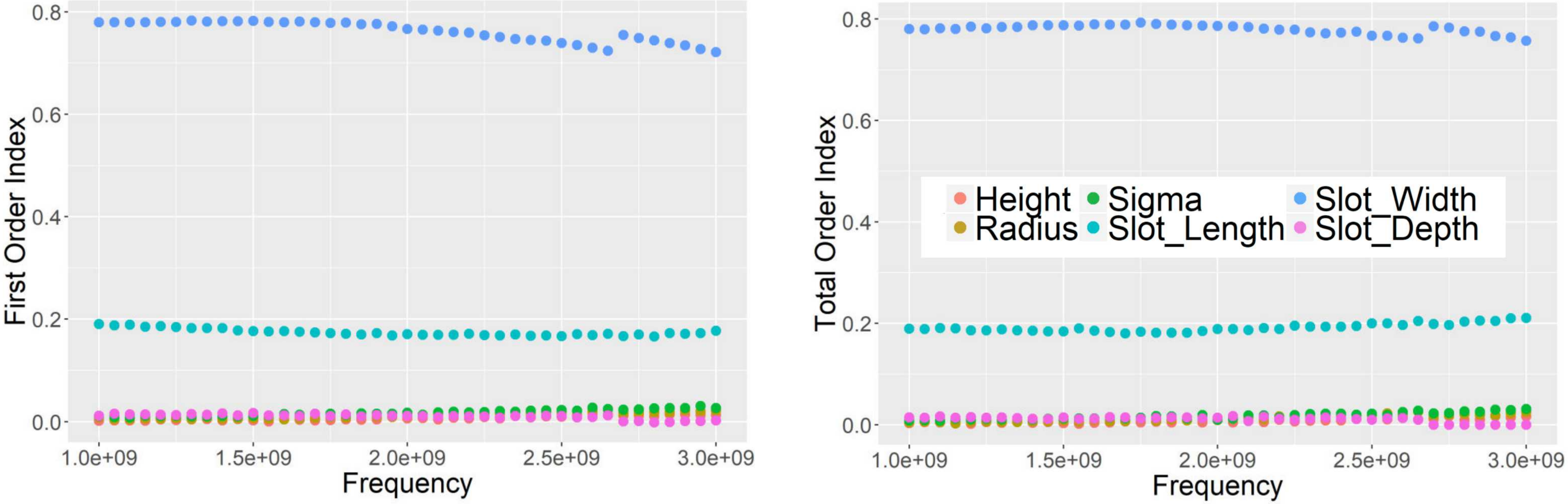
- SE vs frequency changing only one parameter at a time



Sensitivity Analysis: Results (Power Balance Matched Bound)

- First and total order variance decomposition (all parameters varied at once)

Height	Radius	Sigma	Slot Length	Slot Width	Slot Depth
[21.6in,26.4in]	[3.6in,4.4in]	[2.5e7,3.5e7]	[1.5in,2.5in]	[0.005in,0.025in]	[0.2in,0.3in]



- The fact that these two are nearly the same shows that the parameters do not interact; also, the major variations in SE are induced by the slot parameters (length, width, depth).

- We will focus only on the slot parameters from now on and try to define a computational experiment in full-wave simulations

Computational Experiment: Basics

There can be many goals of a UQ analysis. Two common goals are:

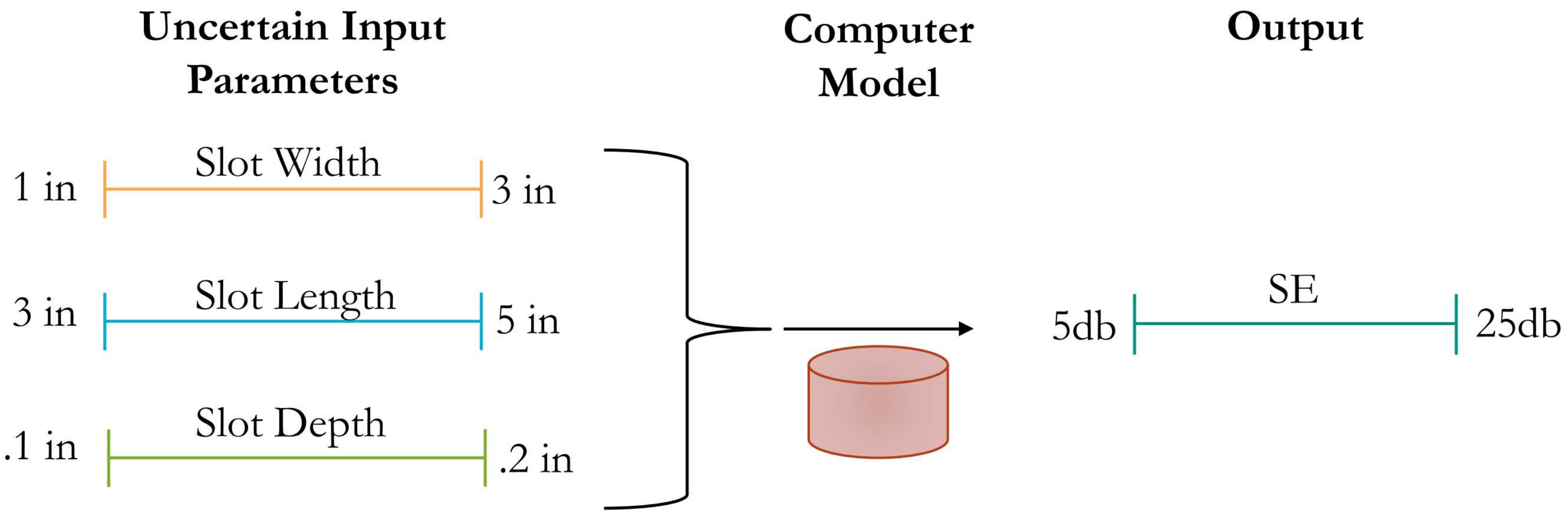
1. Estimate the **min/max (range) of the output** based on the range of input variables.

2. Estimate a **distribution of the output** based on distributions of the input.

The choice of the goal is generally tied to requirements. It is important to consider how the results of the UQ analysis will be used.

Example to Illustrate Goal #1

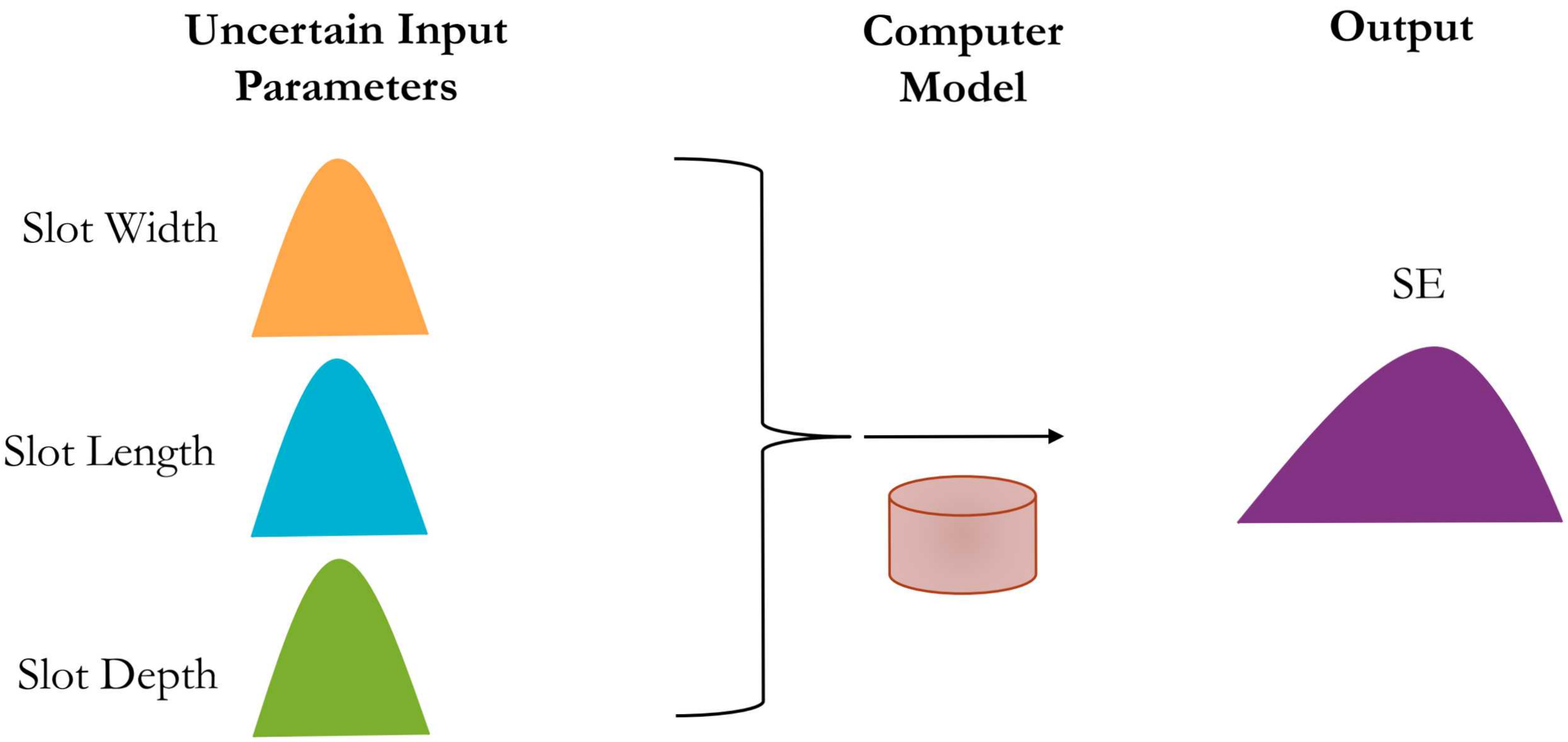
Simulating a Cavity with a Slot



We estimate the SE to be between 5 and 25db. However, we don’t know how likely SE values are between that range. Is it likely to be close to 25db? 5db? Somewhere in between?

Example to Illustrate Goal #2

Simulating a Cavity with a Slot



Here we can use the distribution of the output to answer questions like “what is the probability SE is greater than 20db?” and “what is the 99th percentile of SE?”

Computational Experiment: Sampling Plan

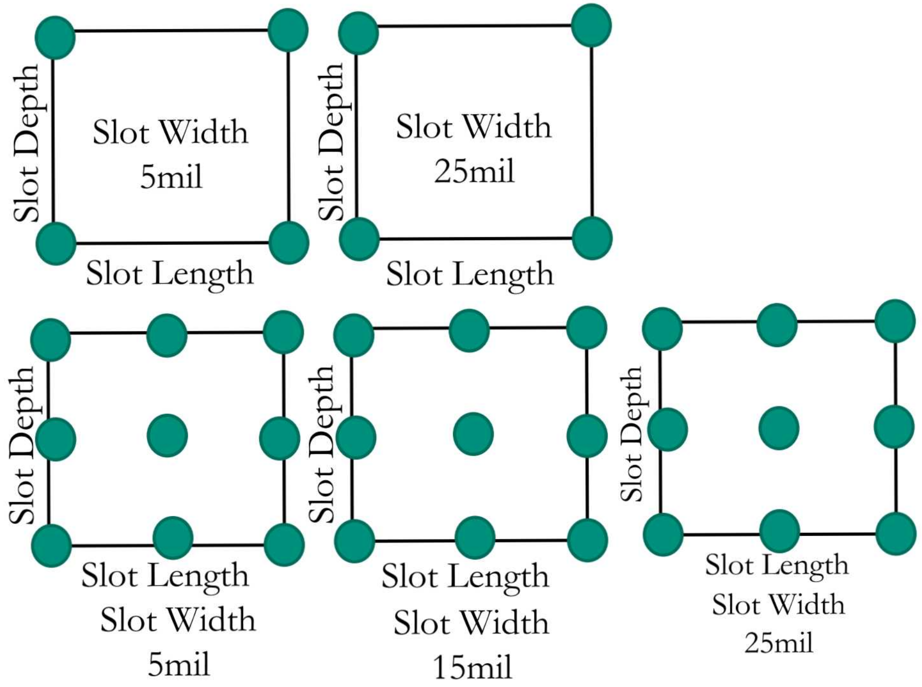
The choice of sampling will depend on the goal of the analysis.

1) **Design of Experiments Sampling** – Often used in physical experiments.
   - Advantages: Relatively small number of runs needed.
   - Disadvantages: Requires assumptions about input/output relationship.

2) **Monte Carlo Based Sampling** – Often used in computer experiments.
   - Advantages: Flexible method for dealing with more complex input/output relationships. Can be used to propagate input uncertainty to get a distribution on the output.
   - Disadvantages: Requires more model runs to adequately cover the input space.

Design of Experiments Methods

Full Factorial – All combinations of all factors (inputs) are run through the model.



2^3 Full Factorial

8 Runs
Linear

3^3 Full Factorial

27 Runs
Quadratic

4^3 Full Factorial

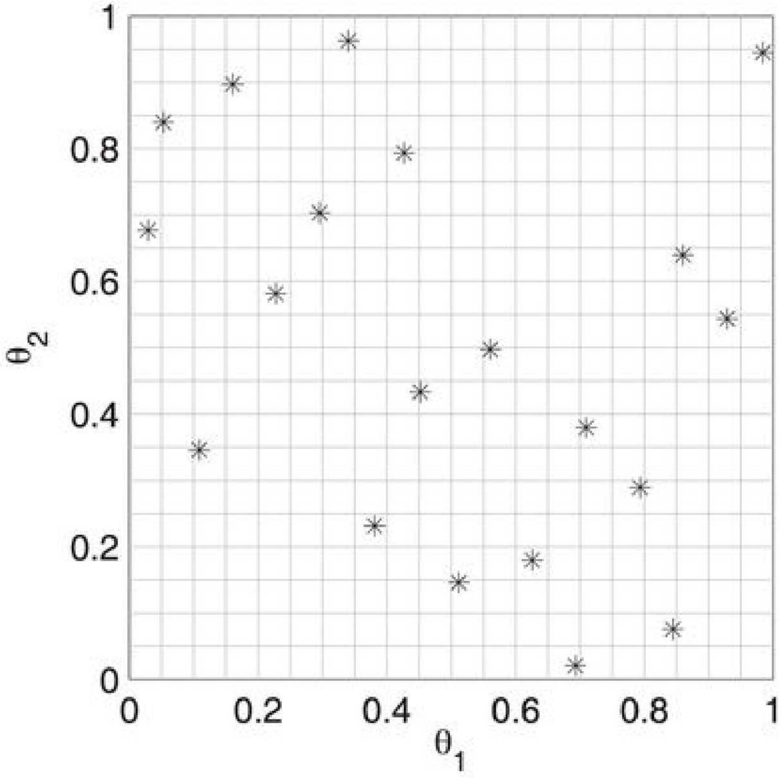
…

64 Runs
Cubic

Note: There are other design options that aim to be more efficient (i.e., use less runs) than these.
Each design has its advantages/disadvantages.

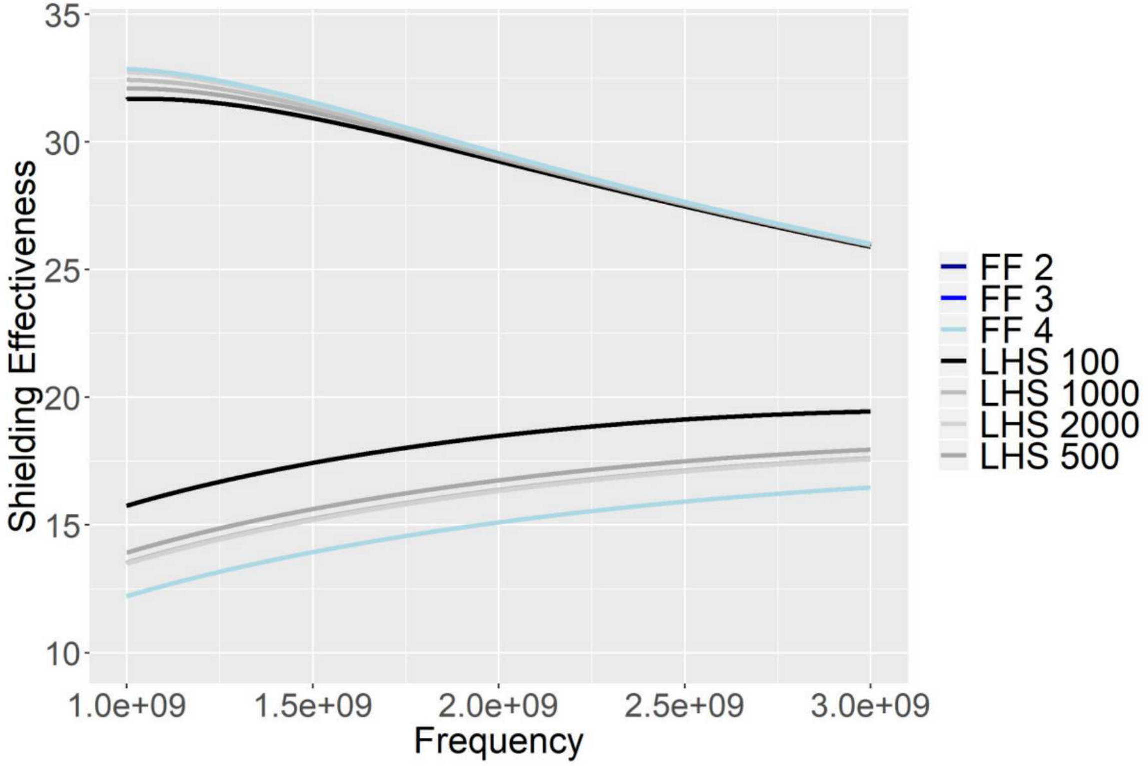
Monte Carlo Based Methods

Latin Hypercube Sampling (LHS) is a random sampling method that aims to sample the inputs evenly across the input space. It is more efficient than random sampling or doing a very fine grid (e.g., a 15^3 full factorial).



Computational Experiment: Results with Power Balance

The power balance code was run using $2^3, 3^3$, and 4^3 full factorials, as well as LHS using sample sizes of 100, 500, 1000 and 2000.



All full factorials gave the same result, indicating that the input/output relationship is likely approximately linear. Many LHS samples are needed to estimate the extremes of the output.

Computational Experiment: EIGER Simulation with Rational Interpolation

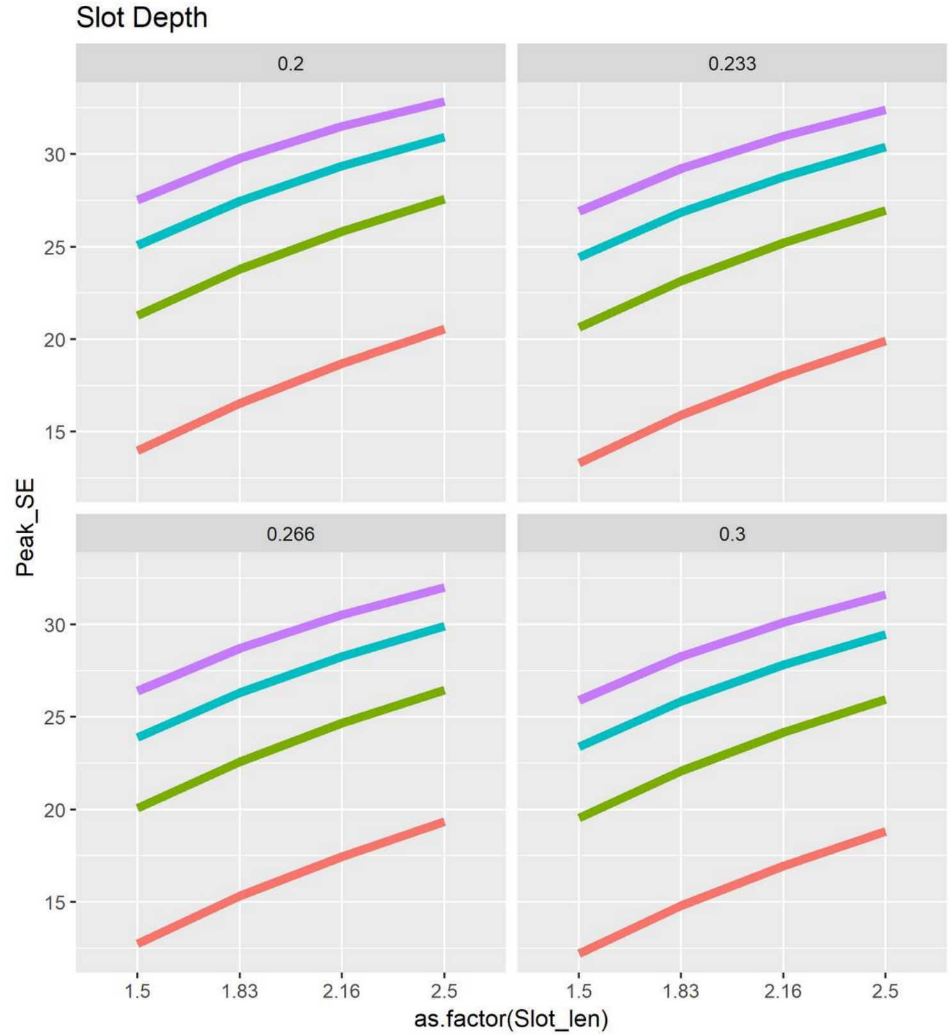
- Based on the results from the power balance code, an EIGER simulation with rational interpolation was run using a 4^3 full factorial (64 runs) with the slot parameters specified at the following ranges:

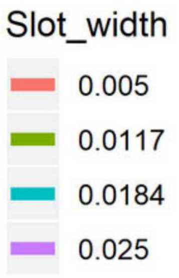
Slot Length [1.5in, 2.5in]

Slot Width [0.005in, 0.025in]

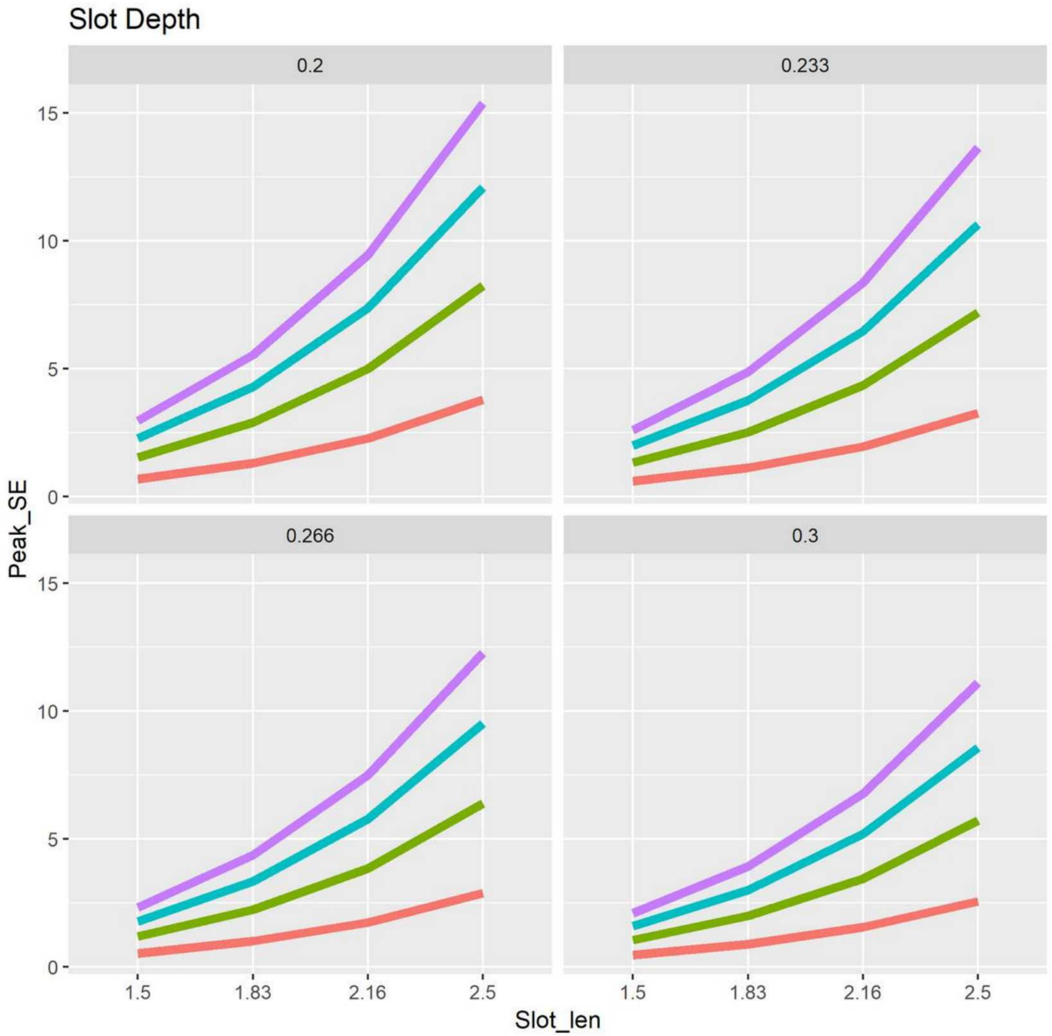
Slot Depth [0.2in, 0.3in]

Power balance matched bound





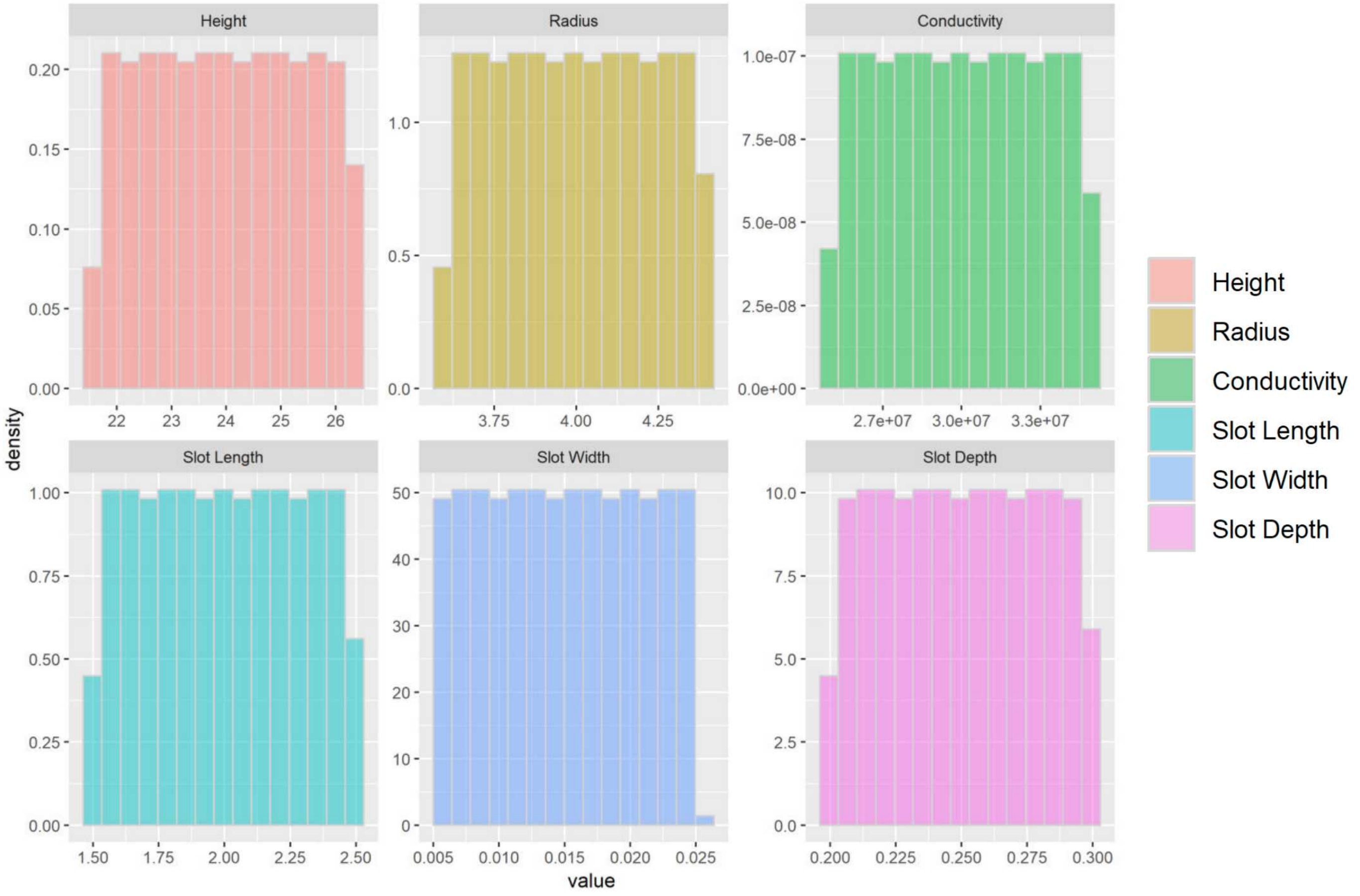
Full-wave



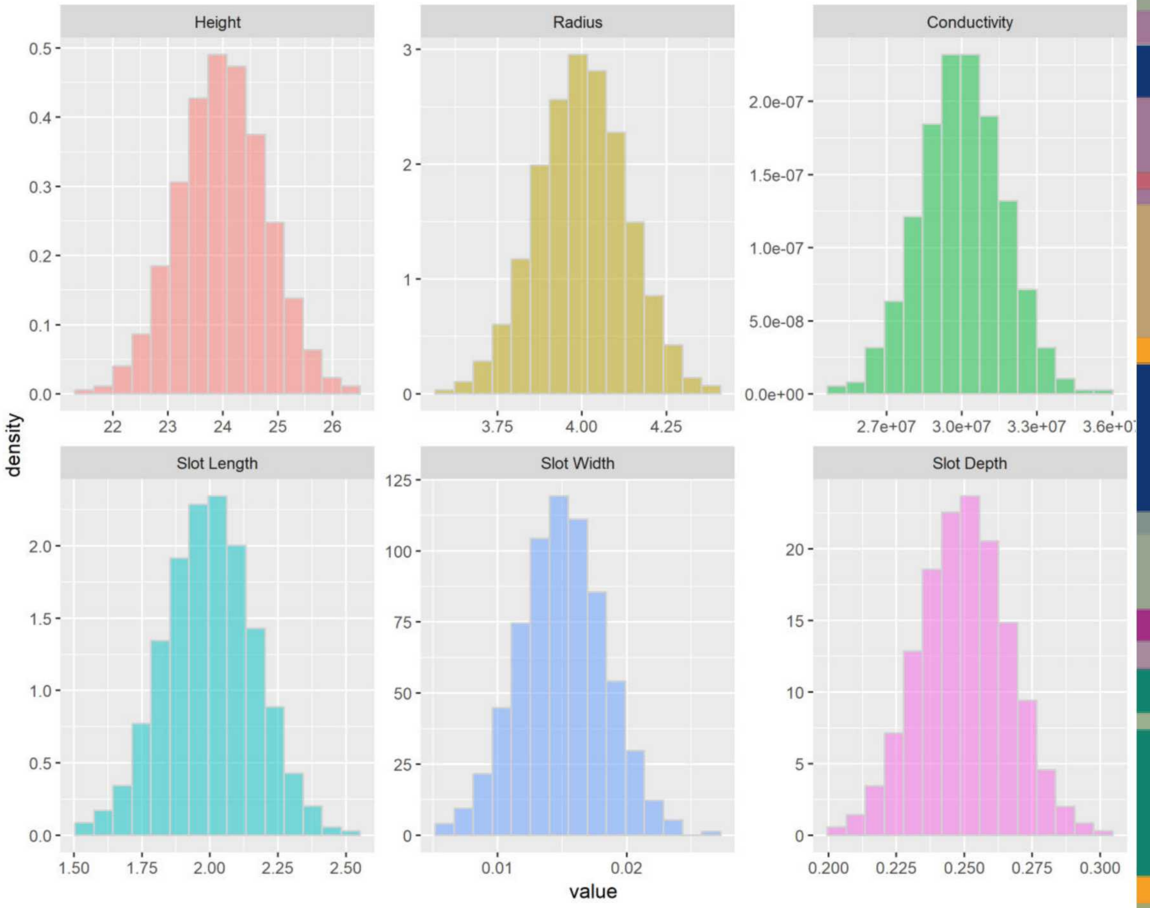
Distribution analysis

- We compute here distributions assuming uniform and normal distributions of the input parameters

Uniform

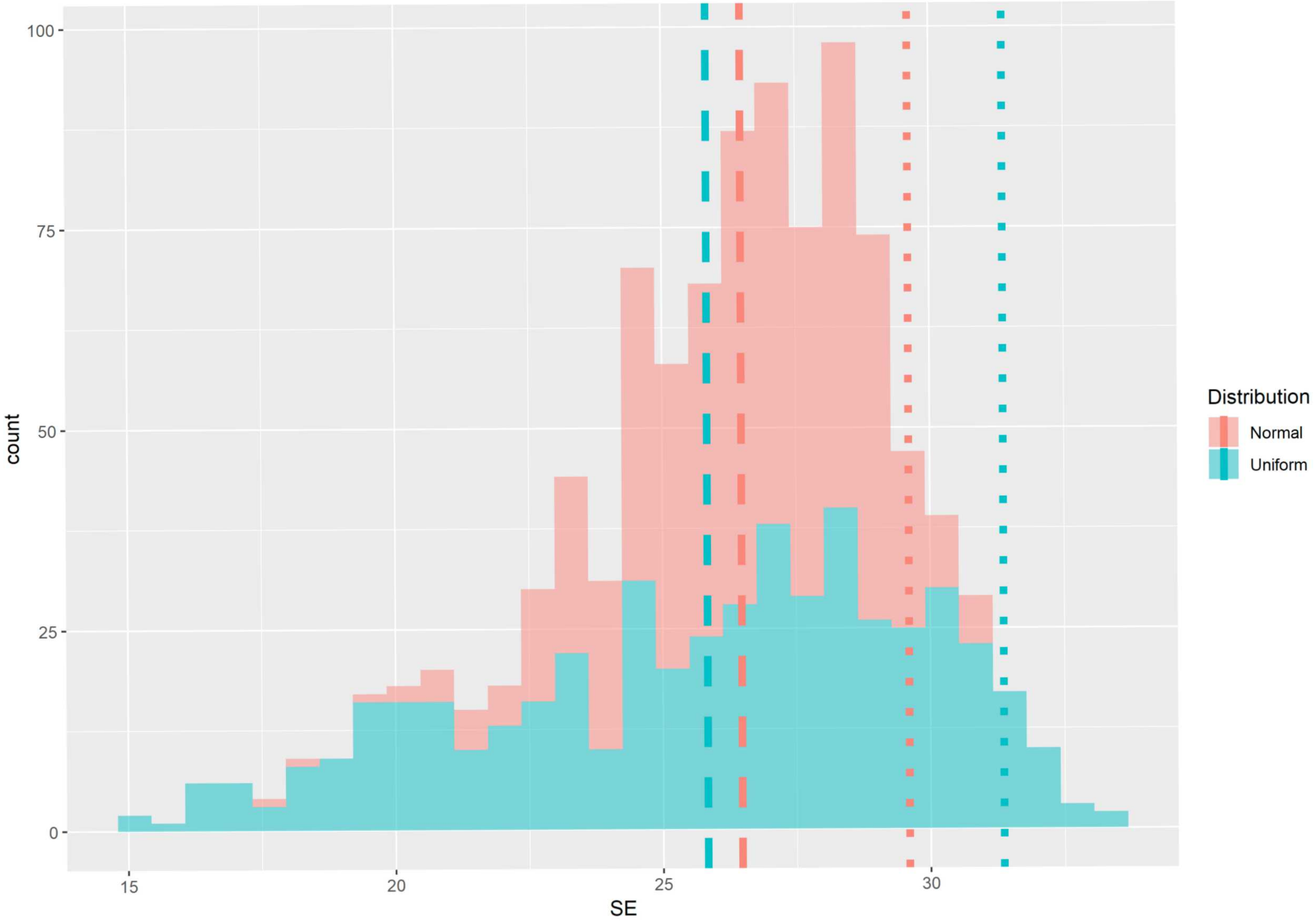


Normal



Distribution analysis

- We compute here distributions assuming uniform and normal distributions of the input parameters



Unmatched Bound Case

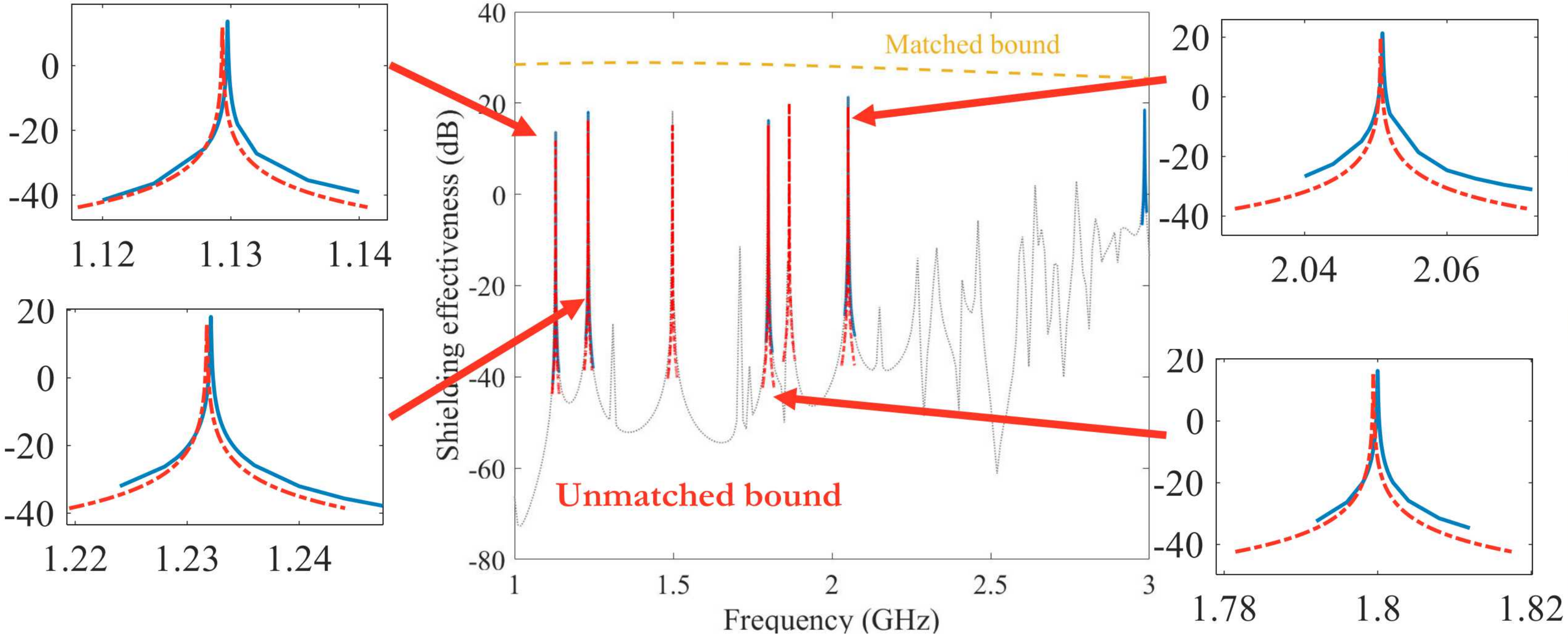
- In this case, the model:
  - Assumes we are below the slot resonance, so the slot behaves inductively
  - Ignores interior loading of the slot and wall losses
  - As an approximation, is driven by the exterior short circuit current density for an infinite cylinder

$$E_{\rho} = -\frac{1}{k^2-\alpha^2} A_{m,p,n} \frac{j_{m,p}}{a} J'_{m}\left(j_{m,p}\rho/a\right)\cos\left[m(\phi-\phi_0)\right]\frac{n\pi}{h}\sin\left(n\pi z/h\right)$$

$$E_{\varphi} = \frac{1}{k^2-\alpha^2}\frac{1}{\rho} A_{m,p,n} J_{m}\left(j_{m,p}\rho/a\right) m\sin\left[m(\phi-\phi_0)\right]\frac{n\pi}{h}\sin\left(n\pi z/h\right)$$

$$E_{z} = A_{m,p,n} J_{m}\left(j_{m,p}\rho/a\right)\cos\left[m(\phi-\phi_0)\right]\cos\left(n\pi z/h\right)$$

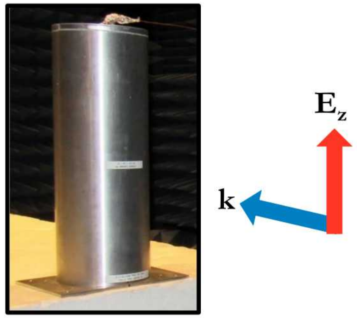
$$A_{m,p,n} \propto L_{slot} K_z^{sc} \frac{k_{m,p,n}^2-\alpha^2}{k^2+kk_{m,p,n}(1+i)/Q_{m,p,n}-k_{m,p,n}^2}$$

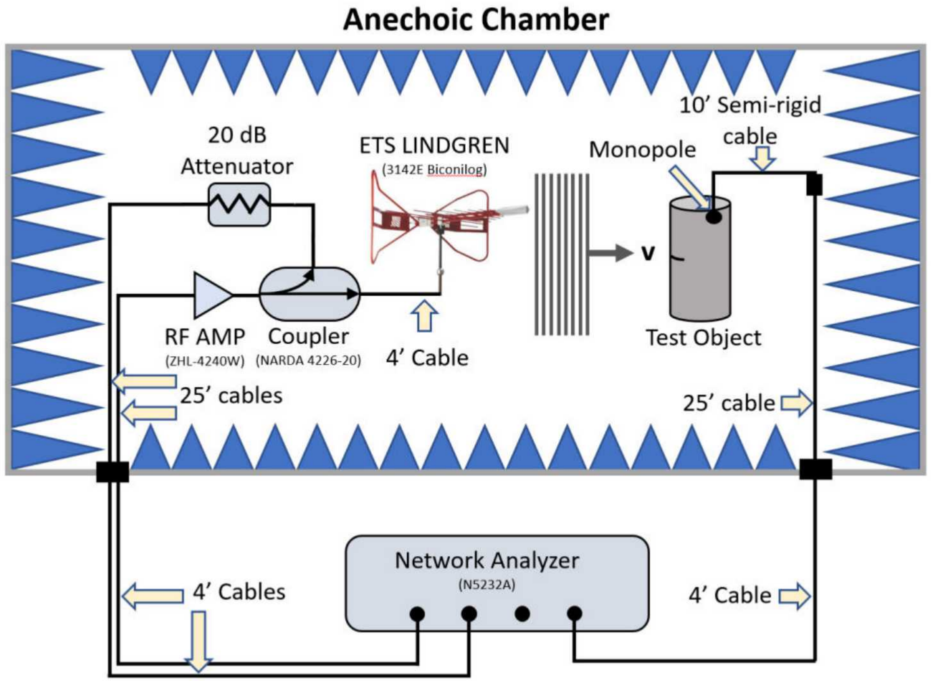


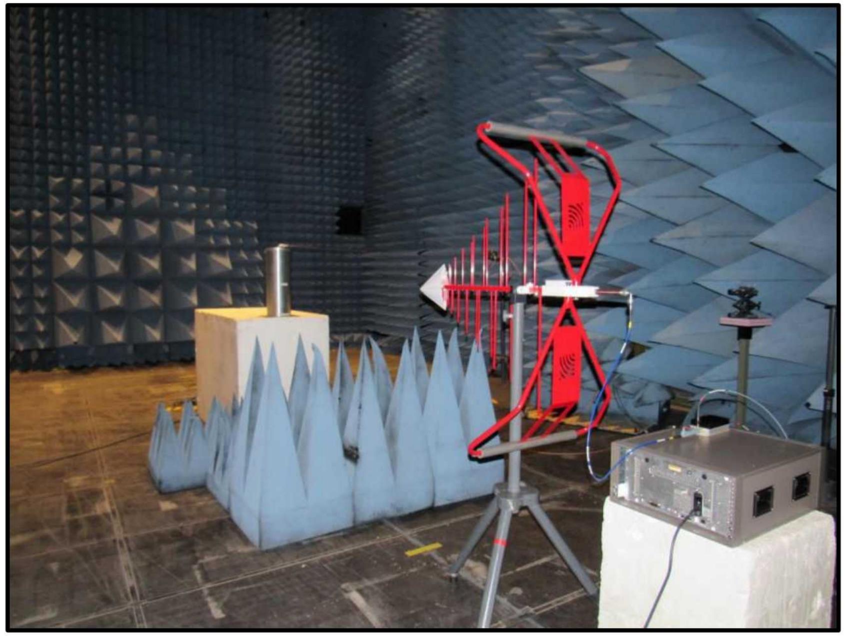
- This model (red curves) is in very good agreement with full-wave simulations and better estimates the peak SE value at each resonance frequency

Shielding Effectiveness Measurement Setup

- Measurement setup bandwidth from 1-4 GHz limited by the RF amplifier
- Incident field strength at the aperture slot was measured separately at the same frequency points in order to provide the SE normalization
- Noise floor limited by the long RF cables from the cylinder to the network analyzer
- Electric field oriented in the z-direction perpendicular to the slot







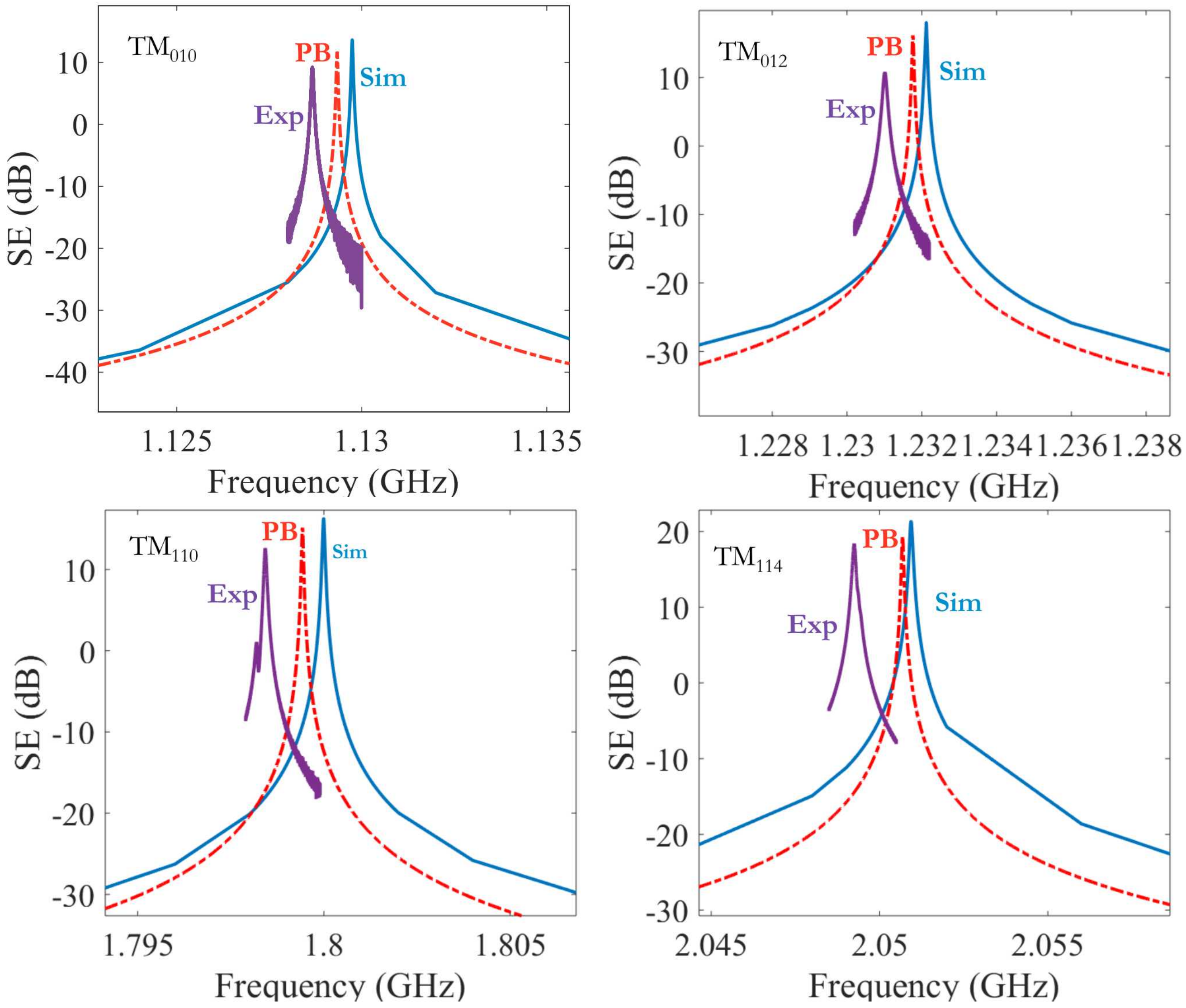
S. Campione et al., Sandia National Laboratories Report, SAND2018-10548, Albuquerque, NM (2018)

Measured & Simulated Empty Cavity SE Results

- Measured mode frequency shift from simulation can be explained by a change in cylinder radius of only 50 μm

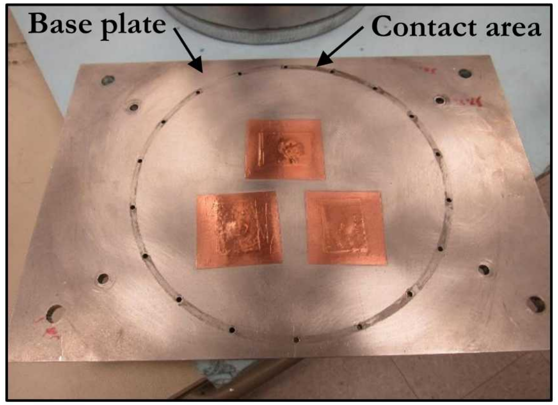
- Measured SE and Q are lower than simulation likely due to some joint resistance in the fabricated cylinder (next slide)

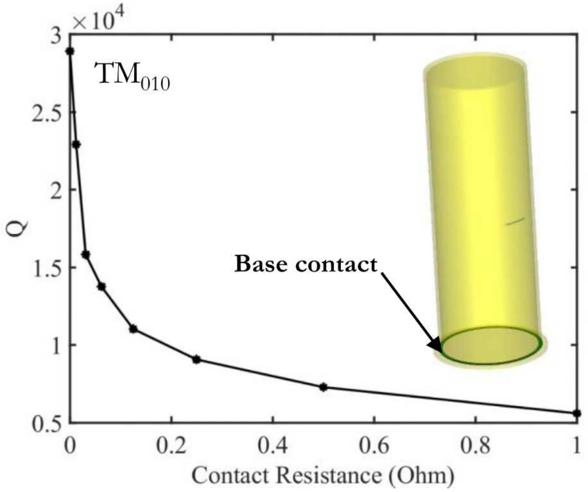
Mode	Peak $SE_{Sim.}$ (dB)	Peak $SE_{Meas.}$ (dB)	$Q_{Sim.}$	$Q_{Meas.}$
TM_{010}	13.6	9.25	29268	18327
TM_{012}	18.1	10.85	27027	10662
TM_{110}	16.3	12.65	35109	22885
TM_{114}	21.3	18.42	27994	19017

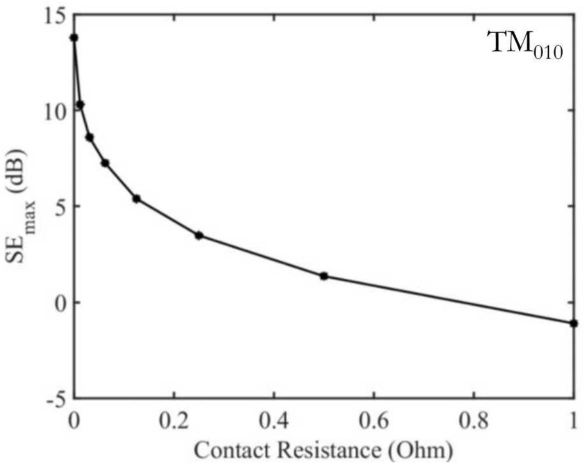


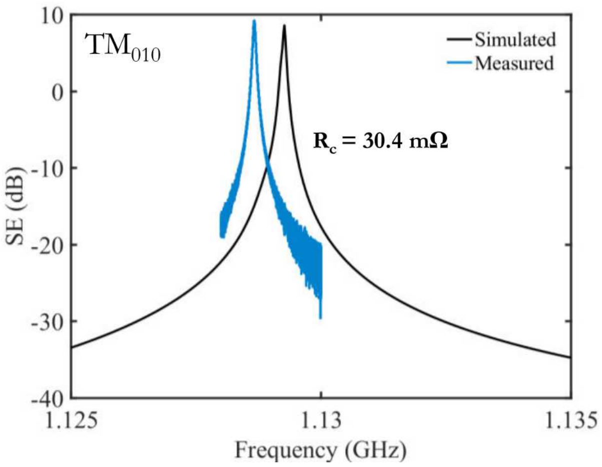
Cylinder Joint Resistance

- Fabricated cylinder has joint at the bottom where the base plate is screwed in
- This joint has some contact resistance that was not initially included in the model/simulations
- A reasonable fit with the TM_{010} measurements is achieved with only 30.4 mΩ of joint resistance
- High-Q cavities are very sensitive to additional resistance between joints





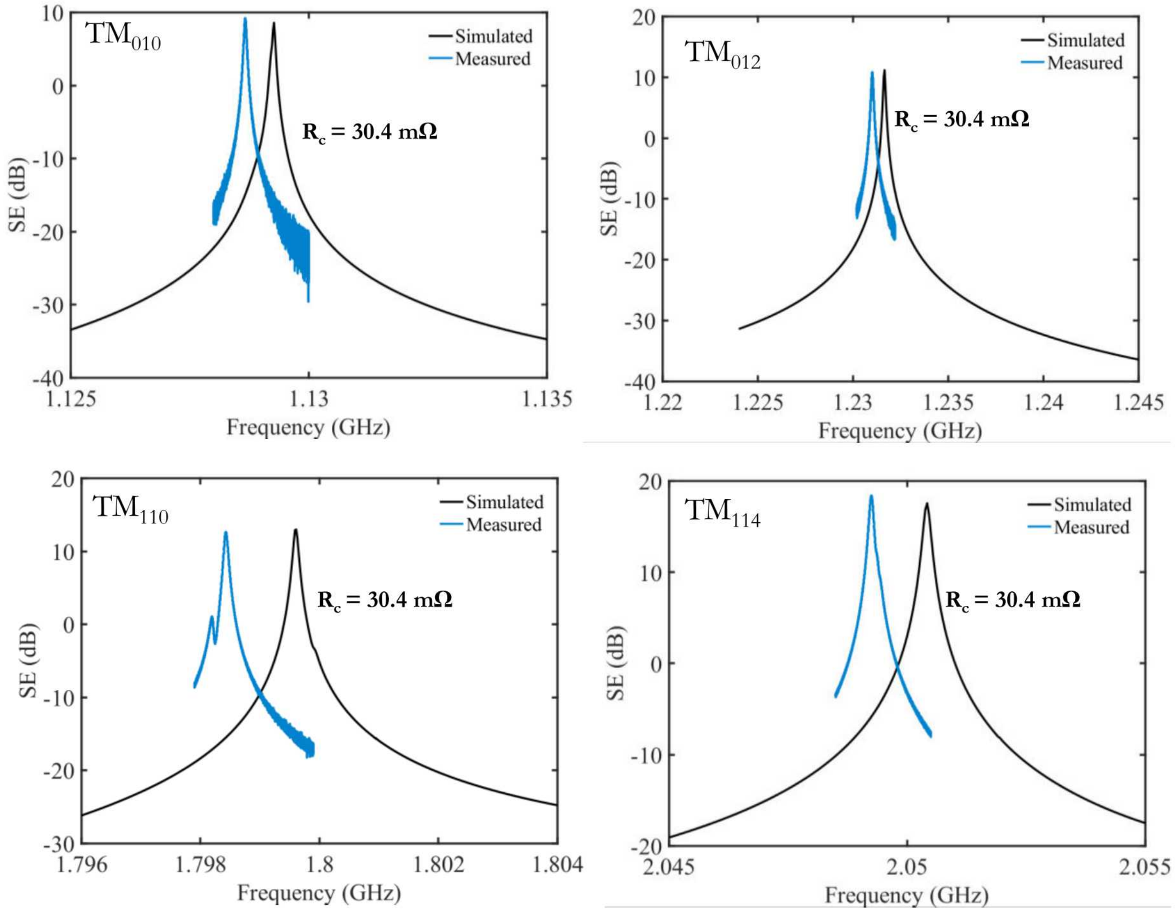




Cylinder Joint Resistance Continued

- The contact resistance fitted for the TM_{010} mode has been applied to higher order modes, leading to agreement with experiments
- This further validates the presence of such loss mechanism

S. Campione et al., Sandia National Laboratories Report, SAND2018-10548, Albuquerque, NM (2018)



Mode	$f_{Sim.}$ (GHz)	$f_{Meas.}$ (GHz)	Peak $SE_{Sim.}$ (dB)	Peak $SE_{Meas.}$ (dB)	$Q_{Sim.}$	$Q_{Meas.}$
TM_{010}	1.1293	1.1287	8.60	9.25	15825	18327
TM_{012}	1.2317	1.2310	11.19	10.85	10697	10662
TM_{110}	1.7996	1.7984	13.00	12.65	19334	22885
TM_{114}	2.0504	2.0493	17.56	18.42	12994	19017

Conclusions

- Rational interpolation greatly decreases the number of simulations required to resolve a resonance peak.
- HPC can be utilized to further decrease runtime.
- Sensitivity analysis on the power balance code informed that SE is most sensitive to the slot parameters.
- Performed comparisons with experiments toward validation.
- Use of more accurate power balance codes will increase fidelity.