Tu1A-2

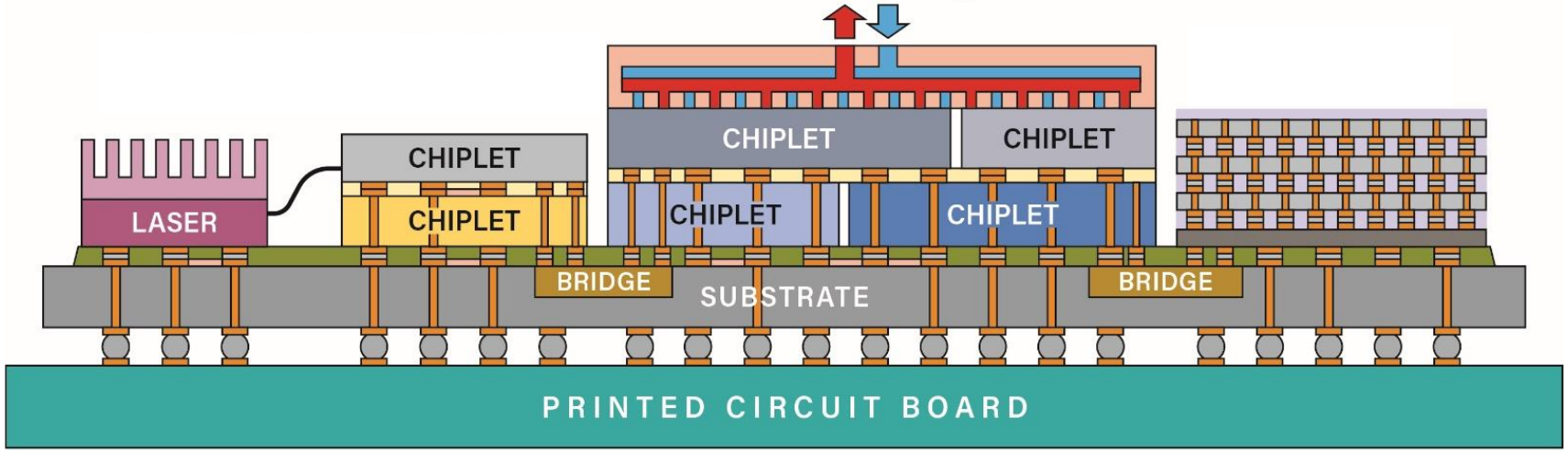
Modeling of Heterogeneously Integrated Systems: Challenges and Strategies for Rapid Design Exploration

Sai Sanjit Ganti[1], Michael Joseph Smith[1], Cheng-Kok Koh[2], Dan Jiao[2], and Ganesh Subbarayan[1]

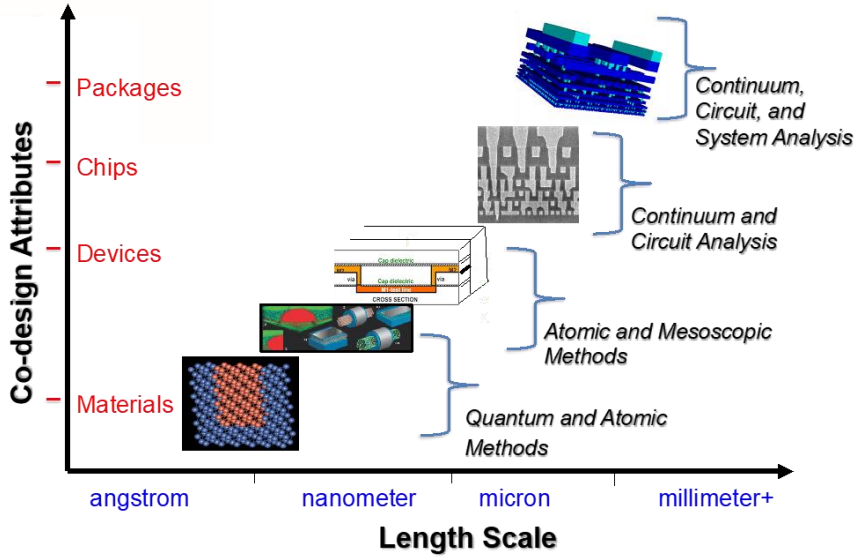
[1]School of Mechanical Engineering, Purdue University

[2]School of Electrical and Computer Engineering, Purdue University

Heterogeneously Integrated Systems



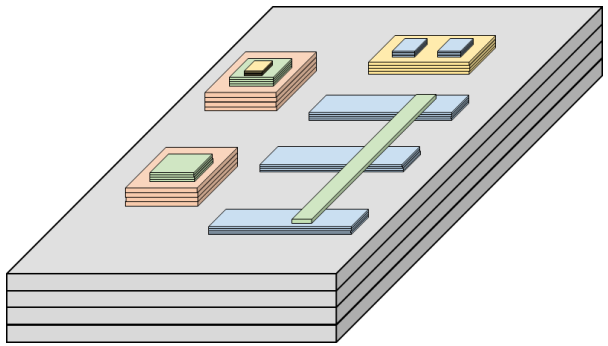
- System is a complex collection of chip packages
- System spans 7-8 orders in length!
- Multiphysics analysis critical to system design and reliability assessment

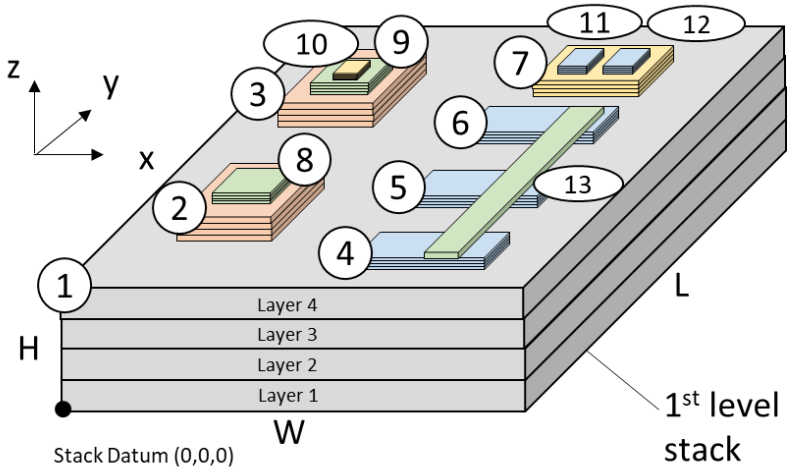


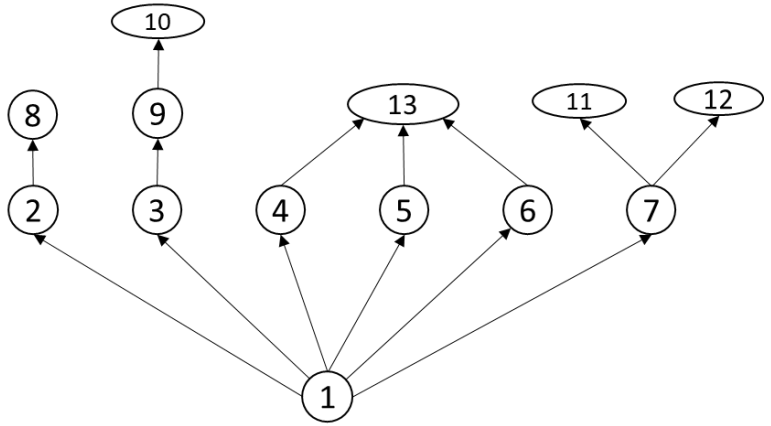
Outline

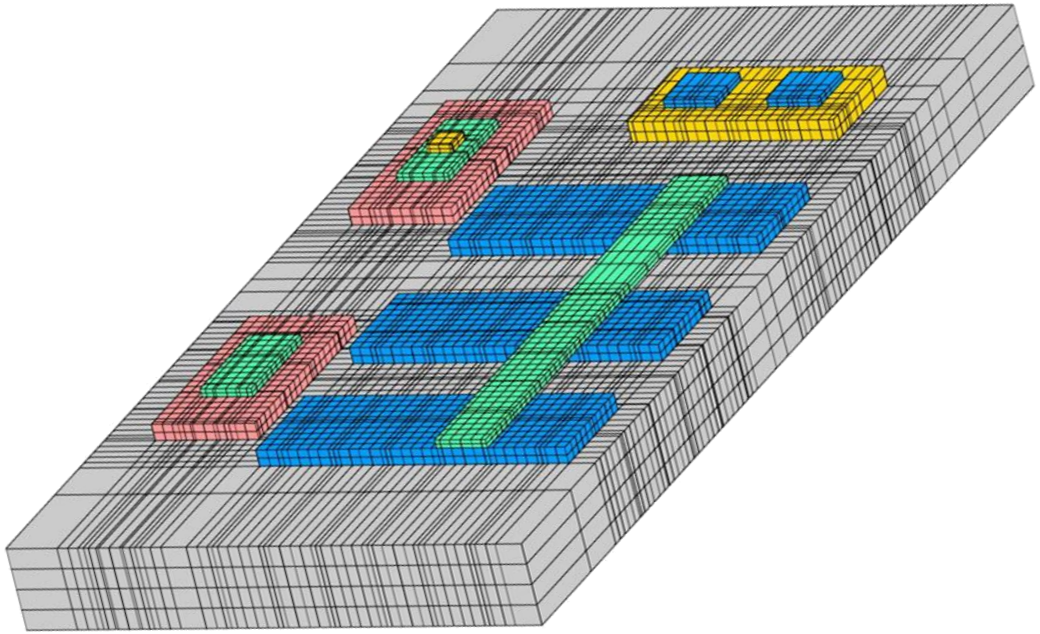
- Challenges to modeling heterogeneously integrated systems
  - Modeling geometry and automatic meshing for analysis
  - Accelerating solution – sparse solver and multigrid analysis
  - Domain decomposition and solution flexibility
- Physics informed neural networks for local modeling
  - Benchmarking against FE solution
  - Capturing complex powermaps
  - Variational formulation
  - Increasing the design space
- Conclusions

Geometry and Automated FE Meshing

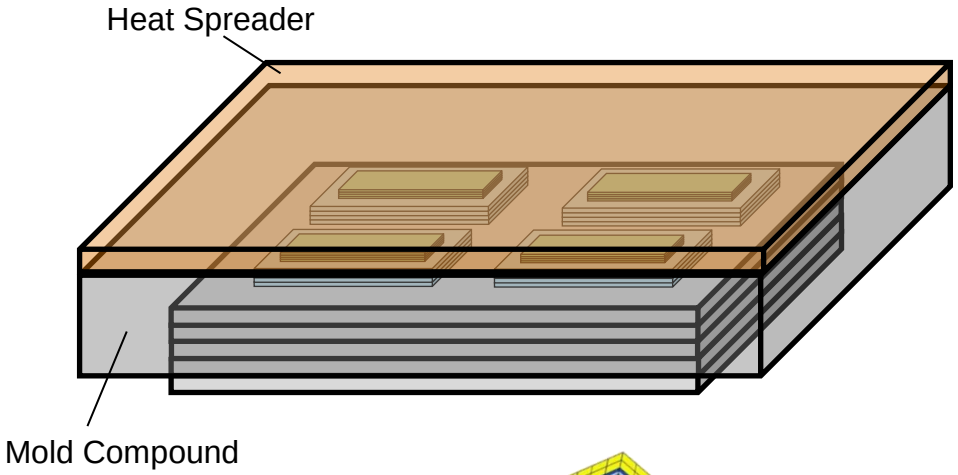


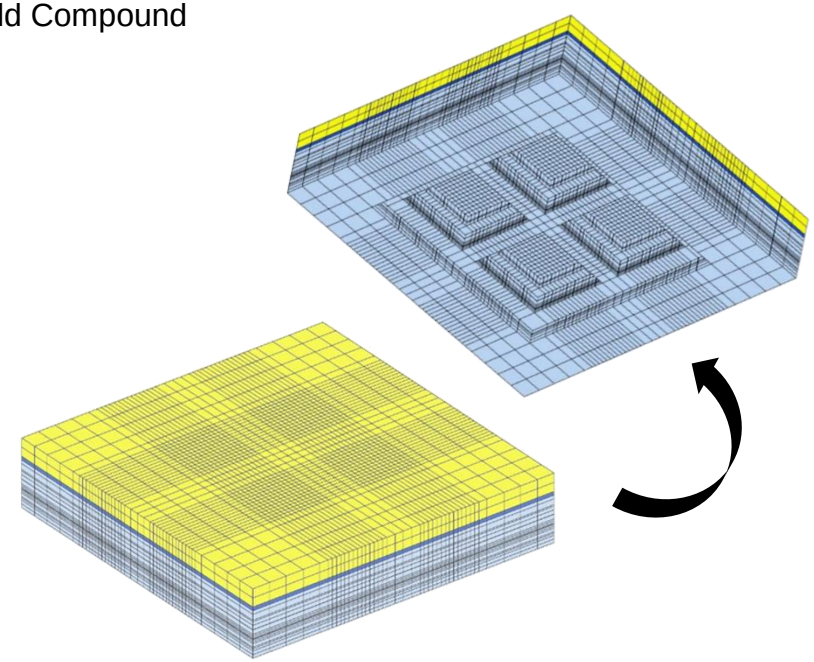






Example: Representational Details

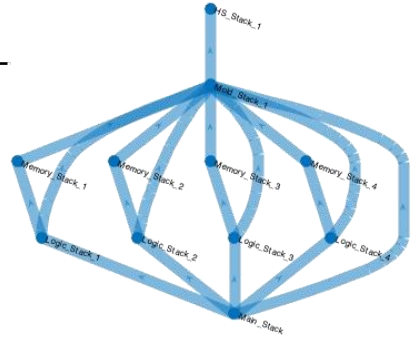




Object definition

Stack	Datum
Substrate	(0,0)
Logic	(1,1), (6.5,1), (1,6.5), (6.5,6.5)
Memory	(1.5,1.5), (7, 1.5), (1.5, 7), (7,7)
Mold Compound	(-3,-3, 0.2)
HS	(-3,-3)

Topology Graph

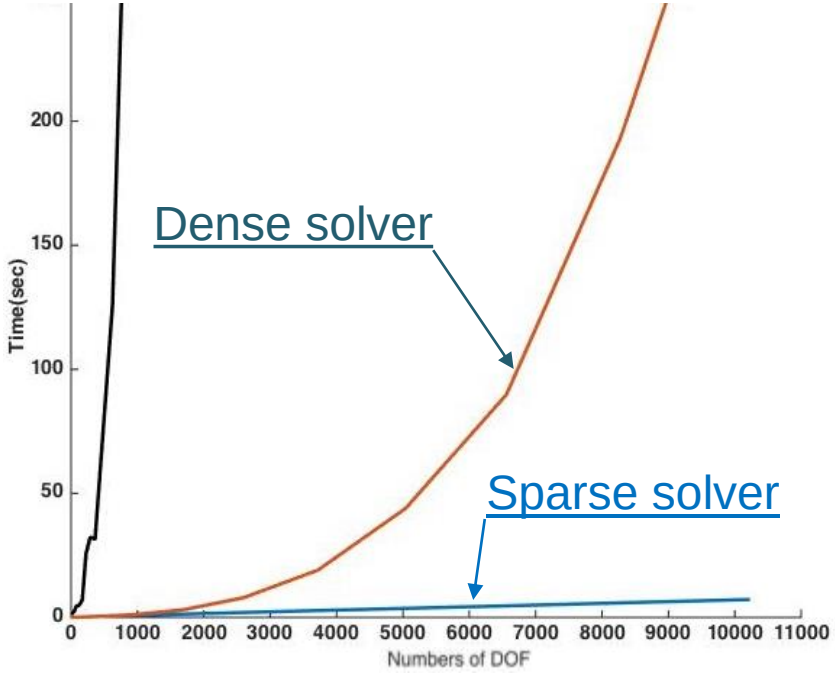


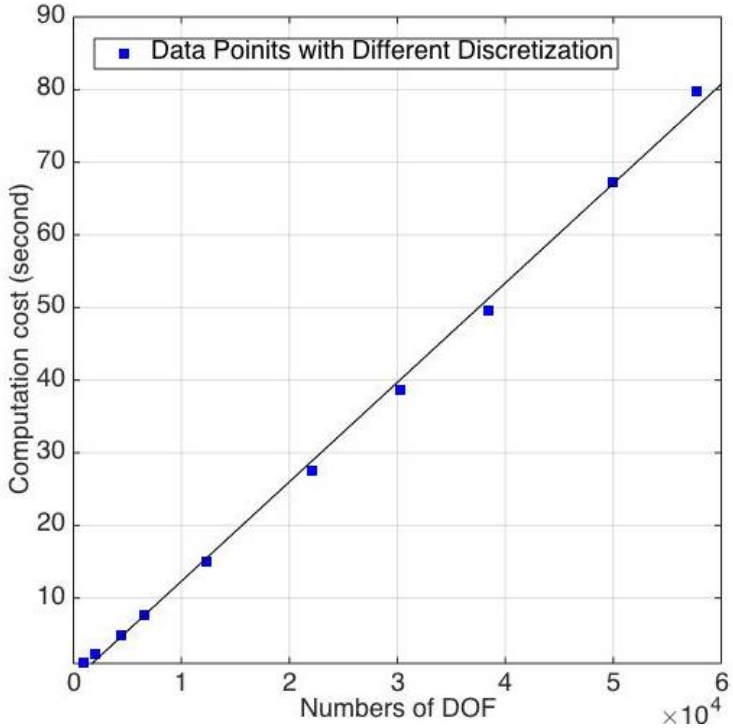
Memory Layer	Dimension (W/L/H)	Feature Size	Material
TIM 1	3.5/3.5/0.2	0.25/0.25/0.2	SAC 387
Device	3.5/3.5/0.4	0.25/0.25/0.4	Silicon

M/C Layer	Dimension (W/L/H)	Feature Size	Material
Mold	18/18/2.6	1.0/1.0/0.5	Copper

HS Layer	Dimension (W/L/H)	Feature Size	Material
TIM	18/18/0.2	1.0/1.0/0.2	SAC 387
Copper	18/18/1.0	1.0/1.0/0.2	Copper

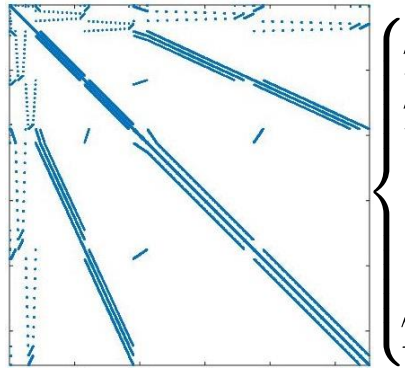
Accelerating Solution: Sparse Solver





$$[K]\{u\} = \{f\}$$

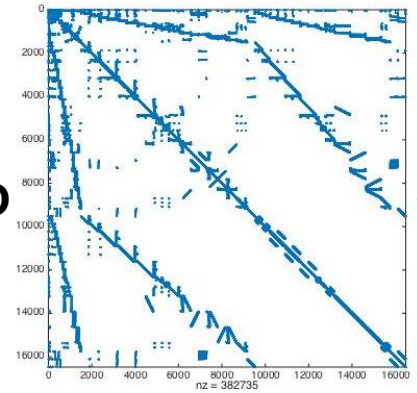
2D



$$\begin{Bmatrix} T_1 \\ T_2 \\ \cdot \\ \cdot \\ \cdot \\ T_n \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_n \end{Bmatrix}$$

Percentage of non-zero terms: ~1.08 %

3D



$$\begin{Bmatrix} T_1 \\ T_2 \\ \cdot \\ \cdot \\ \cdot \\ T_n \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_n \end{Bmatrix}$$

Percentage of non-zero terms: ~ 0.14 %

Accelerating Solution: Multigrid Analysis

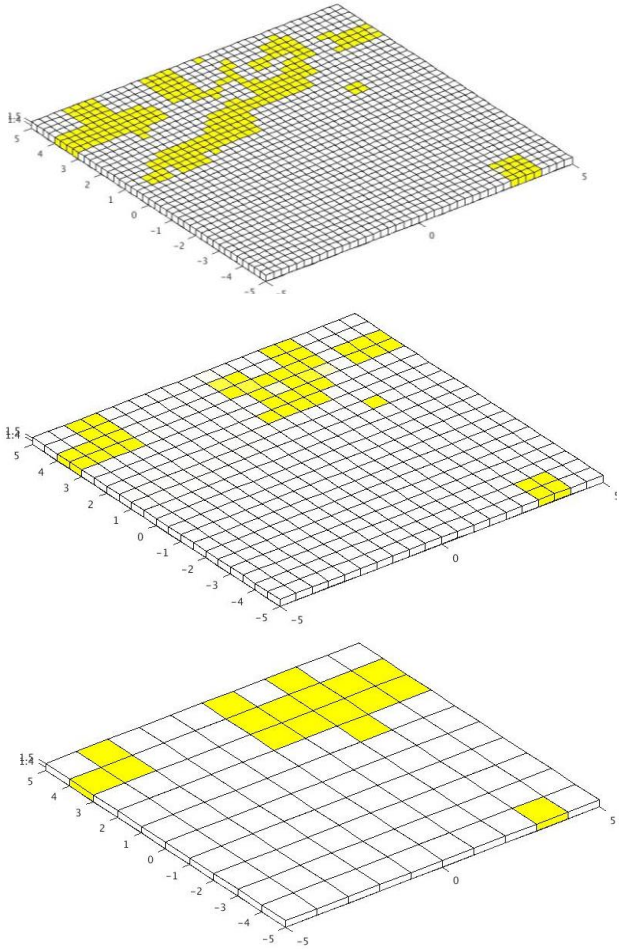
Level n

.
.
.

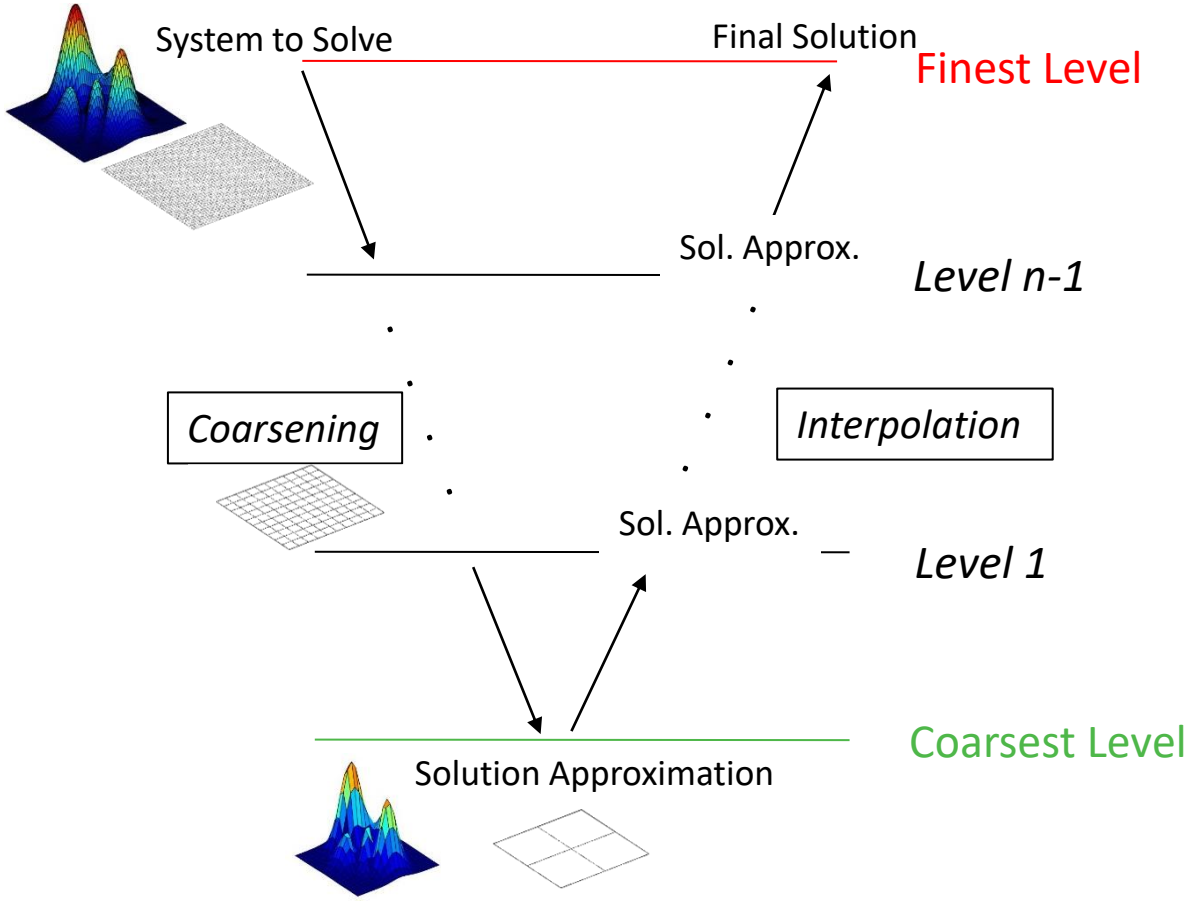
Level k

.
.
.

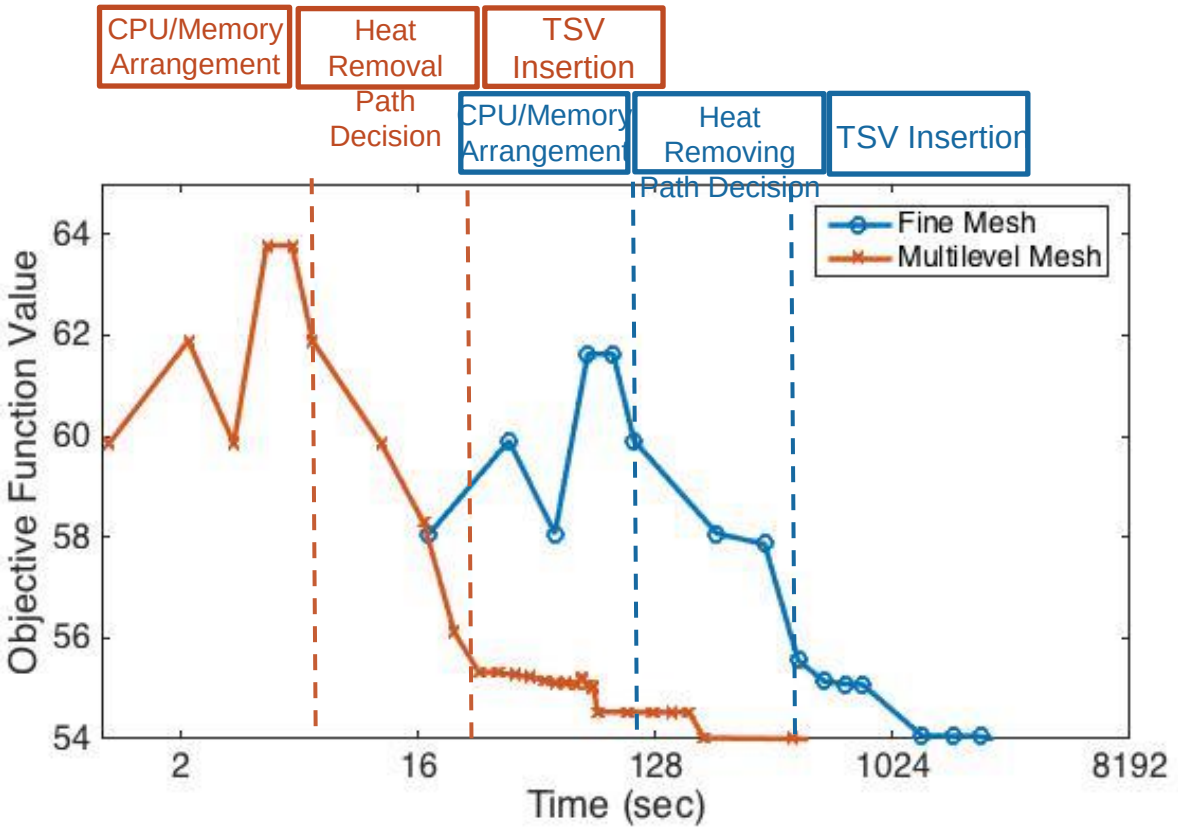
Level 0



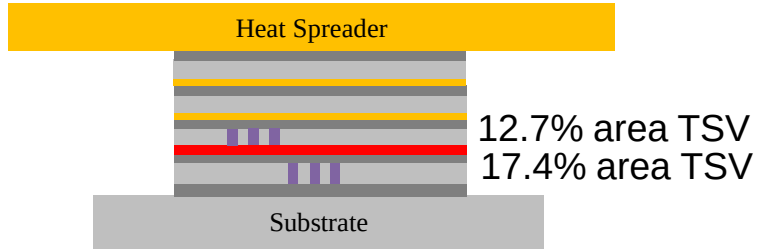
Common V-Cycle for MG:



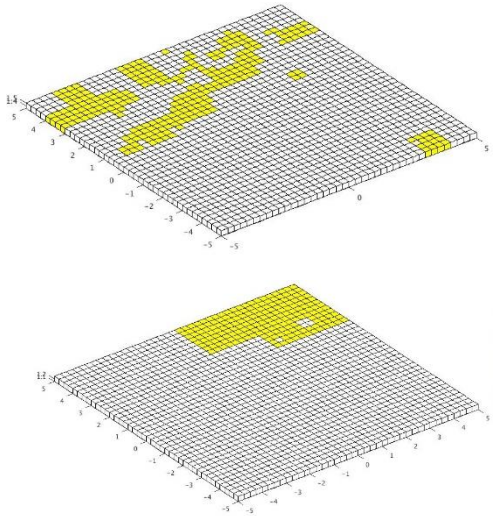
Thermal Design using Multigrid Analysis



7.5x more efficient than regular optimization

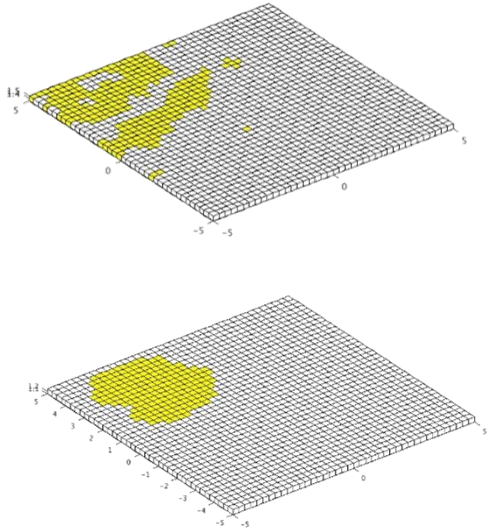


Adaptive Grid Solution



Final $T_{max} = 54.0.^{\circ} C$

Fine-mesh Solution

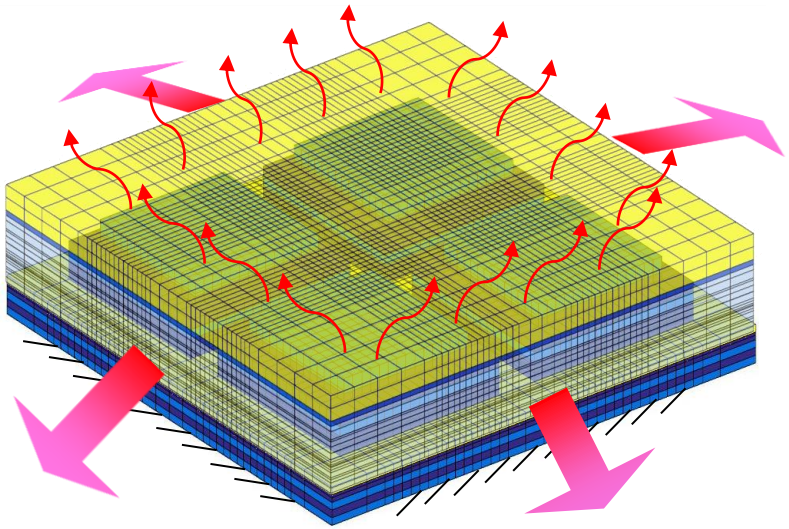


Final $T_{max} = 53.9.^{\circ} C$

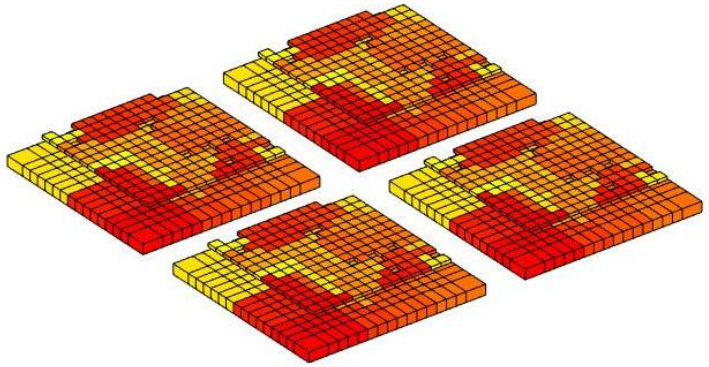
C. -P. Chen, Yifan Weng and G. Subbarayan, "Topology optimization for efficient heat removal in 3D packages," *2016 15th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, Las Vegas, NV, USA, 2016, pp. 238-244, doi: 10.1109/ITHERM.2016.7517556.

Accelerated FE Solution

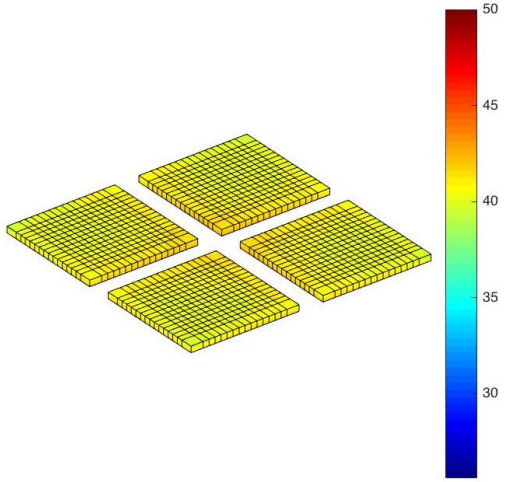
Geometry, Mesh and Boundary Conditions



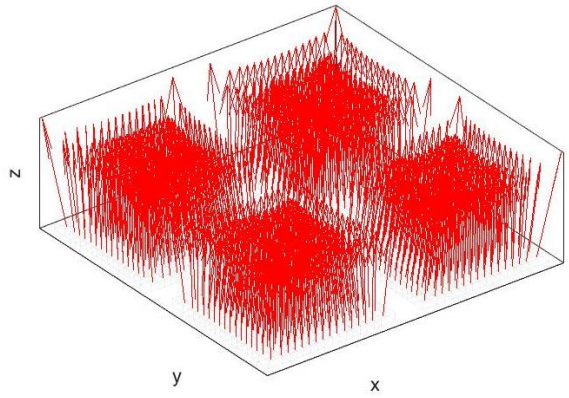
Specify Power Map



Temperature Solution

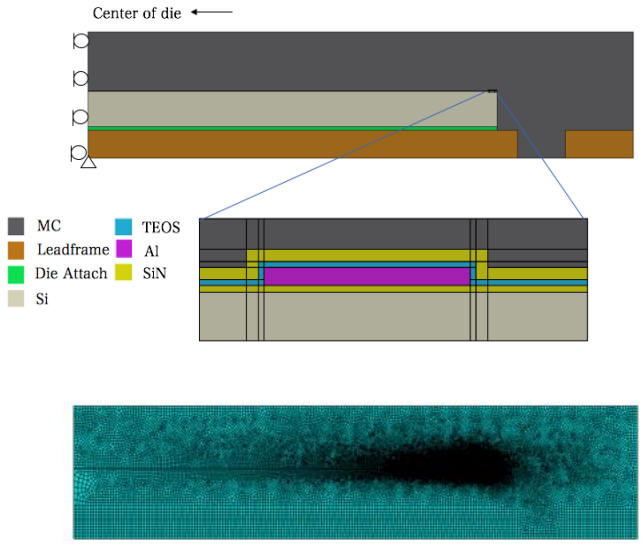


Heat Flux Solution

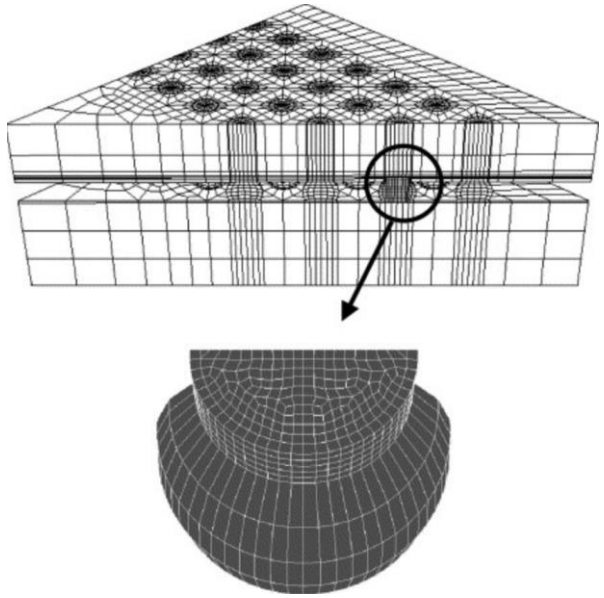


Flexibility in Solution: Decomposition

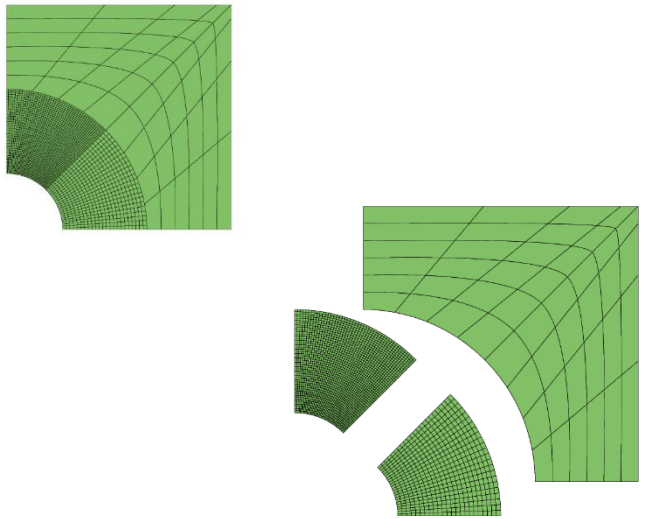
Fully Refined Models



Global-Local Modeling



Domain Decomposition



	Full-Model	Global-Local	Domain Decomposition
Computationally efficient	✗	✓	✓
Reusable models for reduced setup time	✗	✗	✓
Allow subdomains with different discretizations and commercial solvers	✗	✗	✗

Two-Way Coupled DDM Solution

Variational Principle:

$$\delta I \overset{\text{def}}{=} \delta I^G + \delta I^L + \delta I^\Gamma = 0$$

$$\delta I^G(u^G) = \int_{\Omega^G} \sigma^G : \delta\varepsilon^G d\Omega - \int_{\Omega^G} \bar{f}^G : \delta u^G d\Omega - \int_{\partial\Omega^G} \bar{t}^G : \delta u^G dS$$

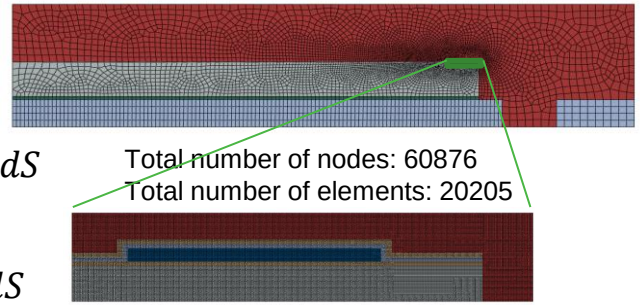
$$\delta I^L(u^L) = \int_{\Omega^L} \sigma^L : \delta\varepsilon^L d\Omega - \int_{\Omega^L} \bar{f}^L : \delta u^L d\Omega - \int_{\partial\Omega^L} \bar{t}^L : \delta u^L dS$$

$$\delta I^\Gamma(\lambda) = \int_\Gamma \lambda(\delta u^G - \delta u^L) d\Gamma + \int_\Gamma \delta\lambda(u^G - u^L) d\Gamma$$

Necessary Conditions:

$$\nabla \cdot \sigma + \bar{f} = 0 \text{ on } \Omega_G \cup \Omega_L$$
$$\sigma \cdot \boldsymbol{n} = \bar{t} \text{ on } \Gamma_n$$
$$u^L = u^G \text{ on } \Gamma$$
$$\lambda = -\sigma^G \cdot \boldsymbol{n} \text{ or } \delta u^G = 0 \text{ on } \Gamma$$
$$\lambda = \sigma^L \cdot \boldsymbol{n} \text{ or } \delta u^L = 0 \text{ on } \Gamma$$

Interface Lagrange multiplier field used to connect force and displacement fields



$$\mathbf{u^L} = \mathbf{P^{LG}}\mathbf{u^G}$$

$$\mathbf{F^G} = -\mathbf{P^{LG}}^{\mathrm{T}}\mathbf{F^L}$$

Discretization:

$$u^L = N^L \boldsymbol{u}^L$$
$$u^G = N^G \boldsymbol{u}^G$$
$$\delta\lambda = N^\lambda \boldsymbol{u}^\lambda$$

Interface Compatibility Condition:

Choose: $N^\lambda = N^L$

$\mathbf{C}^{\boldsymbol{L}}_{ij} = \int_\Gamma N_i^L(\xi)\, N_j^L(\xi)\, \mathrm{d}\xi$

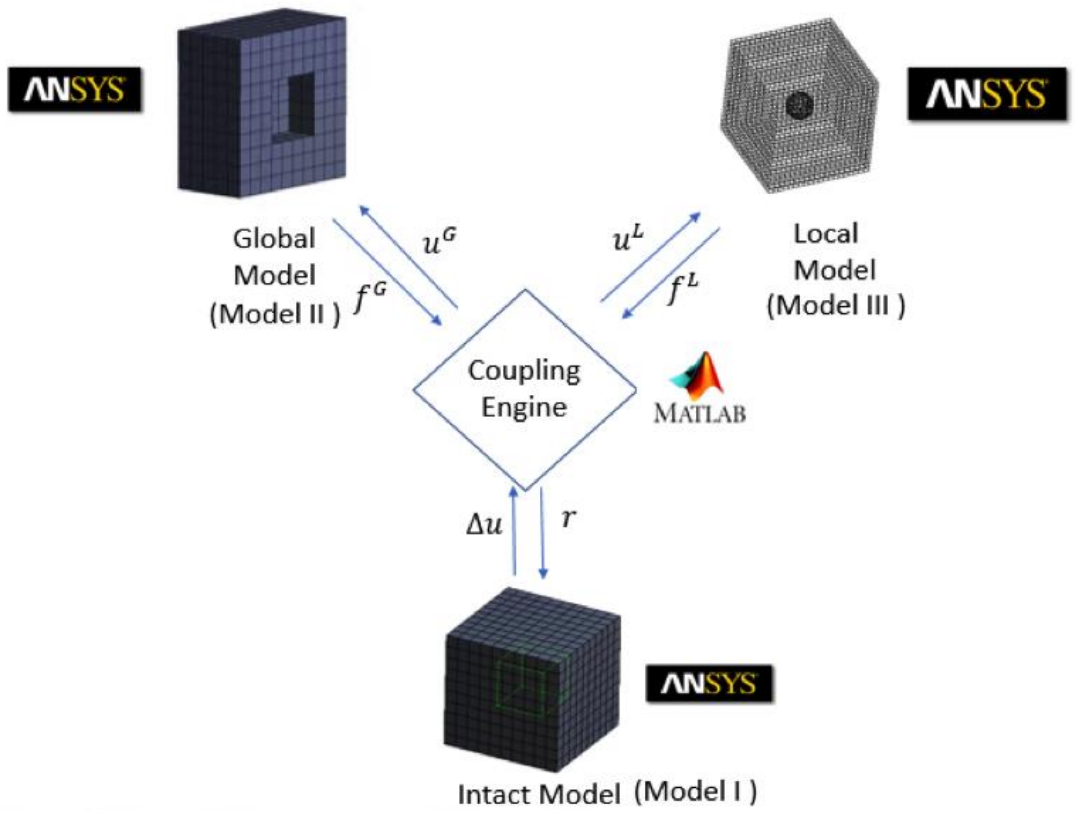
$\mathbf{C}^{\boldsymbol{G}}_{ij} = \int_\Gamma N_i^L(\xi)\, N_j^G(\xi)\, \mathrm{d}\xi$

$$\mathbf{P^{LG}} = (\mathbf{C}^{\mathrm{L}})^{+}\mathbf{C}^{\mathrm{G}}$$

$\mathbf{P^{LG}}$: Mesh projection matrix

Y. Chen, S. S. Ganti and G. Subbarayan, "A Computational Strategy for Code- and Mesh-Agnostic Nonlinear Global–Local Analysis," in *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 5, pp. 740-759, May 2022, doi: 10.1109/TCPMT.2022.3167651.

Accelerating the Decomposed Solution



Residual force

$$r = P^T f^L + f^G$$

Line search

$$u_{k+1} = u_k + \alpha K_k^{-1} r$$

SR1 Update:

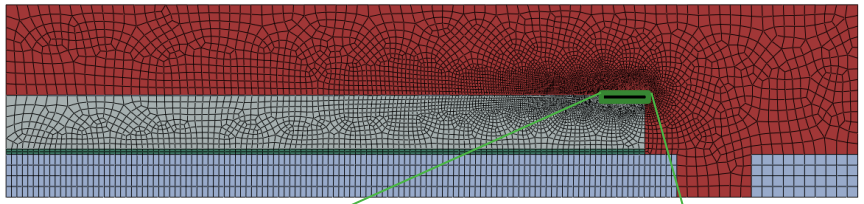
$$\mathbf{K}_k^{-1} r_n = K_{k-1}^{-1} r_n + K_{k-1}^{-1} r_k \frac{\mathrm{r}_k^T (K_{k-1}^{-1} r_n)}{r_k^T (\Delta \mathbf{u}_k - \mathbf{K}_{k-1}^{-1} r_k)}$$

DFP Update:

$$\mathbf{K}_k^{-1} r_n$$
$$= K_{k-1}^{-1} r_n + (-\Delta u_k^T \Delta r_k + \Delta r_k^T K_{k-1}^{-1} \Delta r_k) \frac{\Delta u_k \Delta u_k^T}{(\Delta u_k^T \Delta r_k)^2} r_n$$
$$- \frac{(K_{k-1}^{-1} \Delta r_k \Delta u_k^T + \Delta u_k \Delta r_k^T K_{k-1}^{-1}) r_n}{\Delta u_k^T \Delta r_k}$$

Y. Chen, S. S. Ganti and G. Subbarayan, "A Computational Strategy for Code- and Mesh-Agnostic Nonlinear Global–Local Analysis," in *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 5, pp. 740-759, May 2022, doi: 10.1109/TCPMT.2022.3167651.

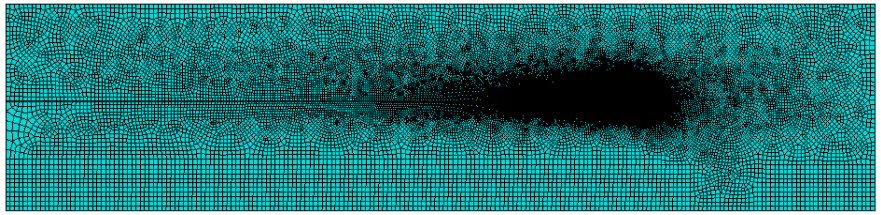
Example: Cyclic Stress Evolution in BEOL



Total number of nodes: 60876
Total number of elements: 20205

Total number of nodes: 88249
Total number of elements: 29120

Abaqus Global-Local Model

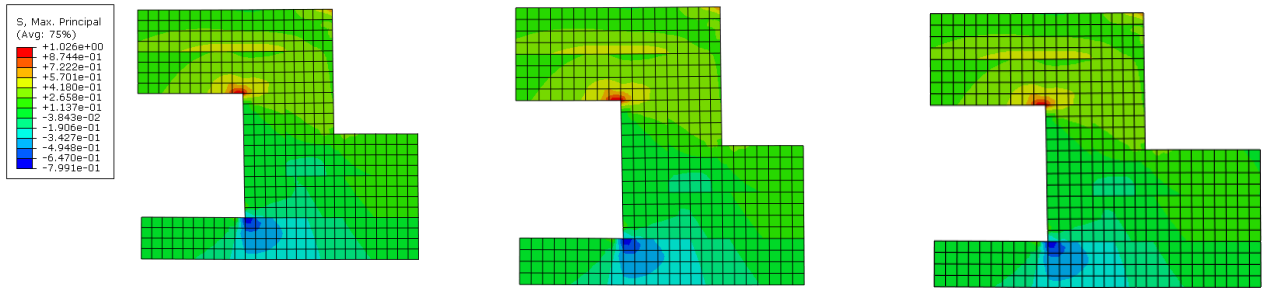


Abaqus Full Model

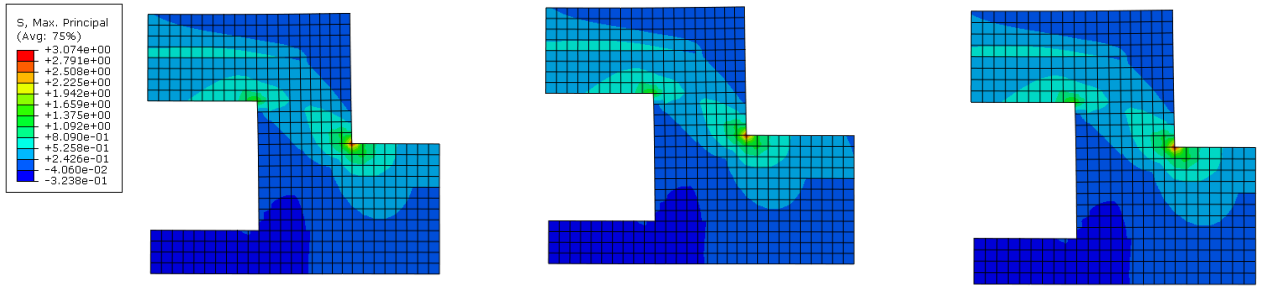
Total number of nodes: 532049
Total number of elements: 177184

Principal Stress in TEOS (20th Cycle)

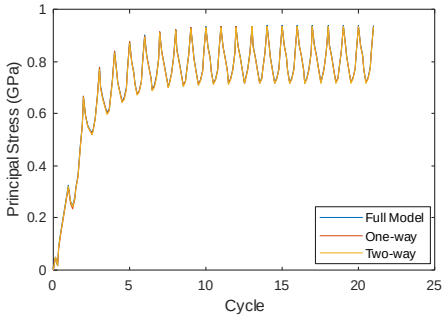
-65°



150°



One-way | Two-way | Full



Physics Informed Neural Networks

- The Neural Network formulations that solve PDEs have been coined Physics Informed Neural Networks (PINNs)

$$\begin{aligned} &\mathcal{L}u = f, \quad x \in \Omega \\ &\mathcal{B}u = g, \ x \in \partial\Omega \end{aligned} \longrightarrow \min_{\theta} \int_{\Omega} \|\mathcal{L}N(x;\theta) - f\|^2 dV + \alpha \int_{\partial\Omega} \|\mathcal{B}N(x;\theta) - g\|^2 dS$$

$$or: \quad \min_{\theta} \int_{\Omega} \left\|\mathcal{L}\widetilde{N}(x, N(x;\theta)) - f\right\|^2 dV$$

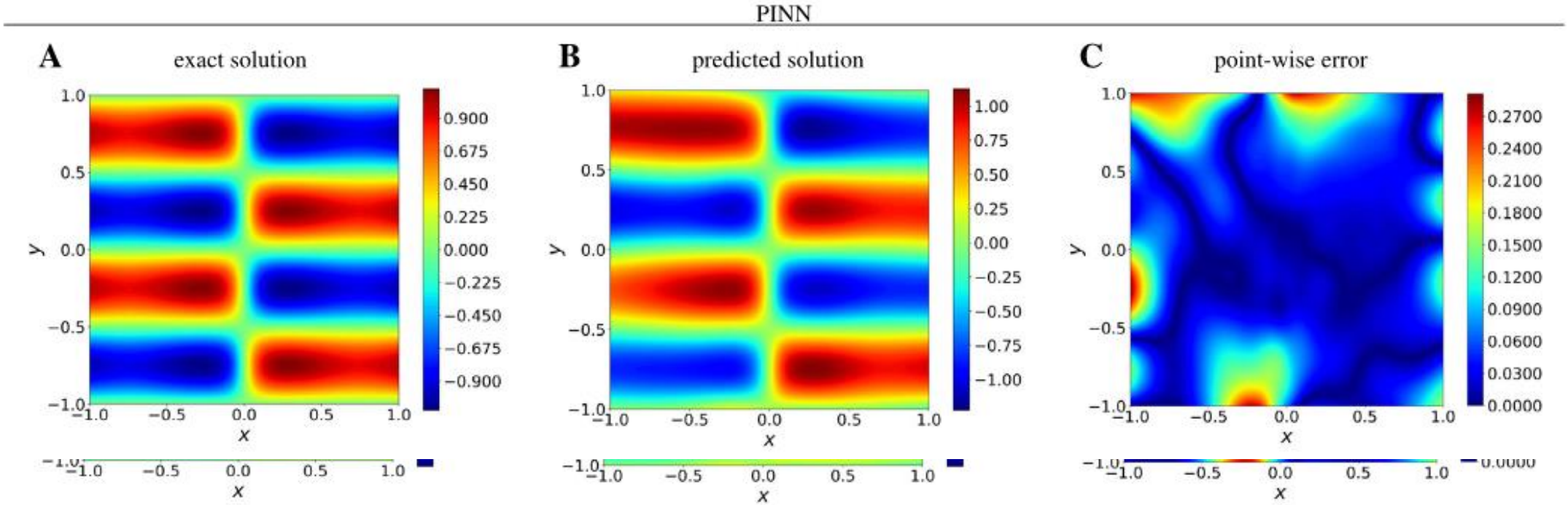
- Network N has trainable parameters θ
- $\widetilde{N}$ is a function of network N and domain variables that automatically satisfies the boundary conditions
- Minimize by evaluating the error at colocation points using automatic differentiation and tuning θ with back propagation

M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed Neural Networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. doi:10.1016/j.jcp.2018.10.045

PINNs for Decomposed Domains?

- Given PINNs' ability to learn complex and non-linear behavior, they are natural candidates for local models in domain decomposition

Non-Homogenous 2D Poisson Equation PINN:

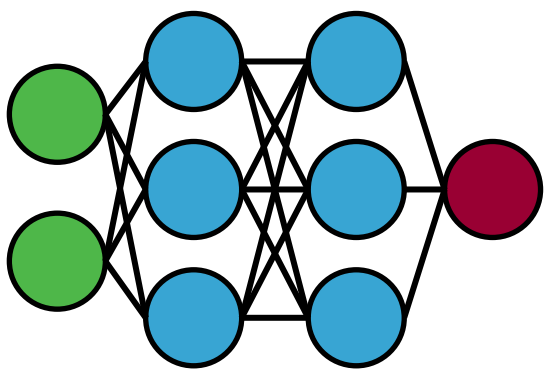


E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis, "HP-VPINNs: Variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, 2021. doi:10.1016/j.cma.2020.113547

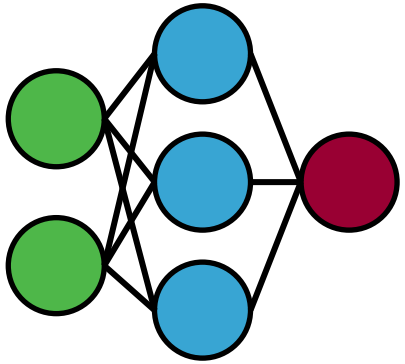
PINN Benchmark Solution

- Use of PINNs for Domain Decomposition and other methods is only practical if they are either faster or more accurate than conventional methods such as Finite Element solutions
- A PINN solver was developed and benchmarked against an FE Solution

Network Notation

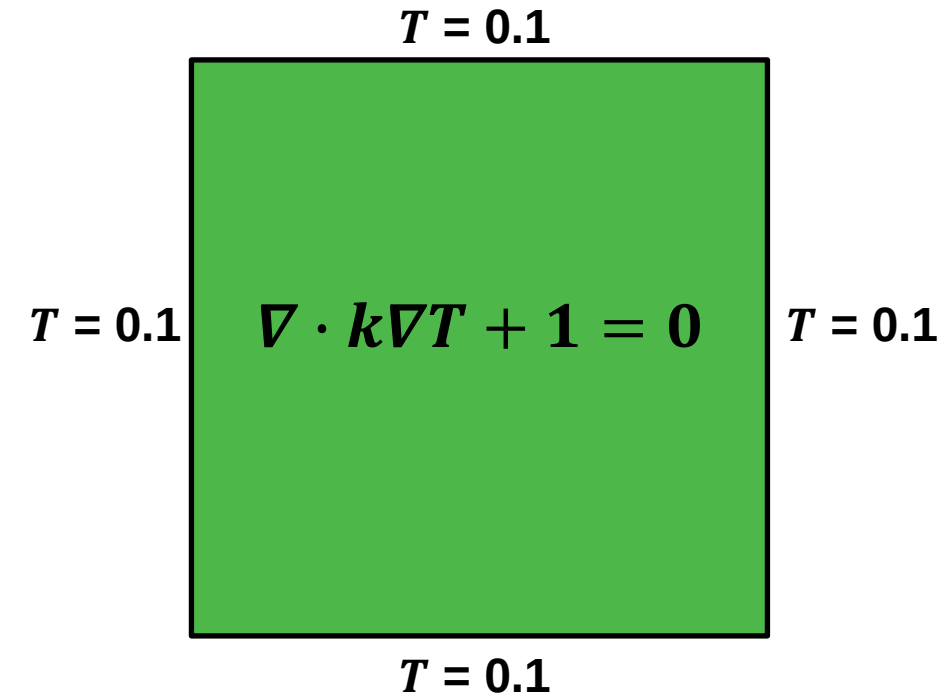


"3:3"



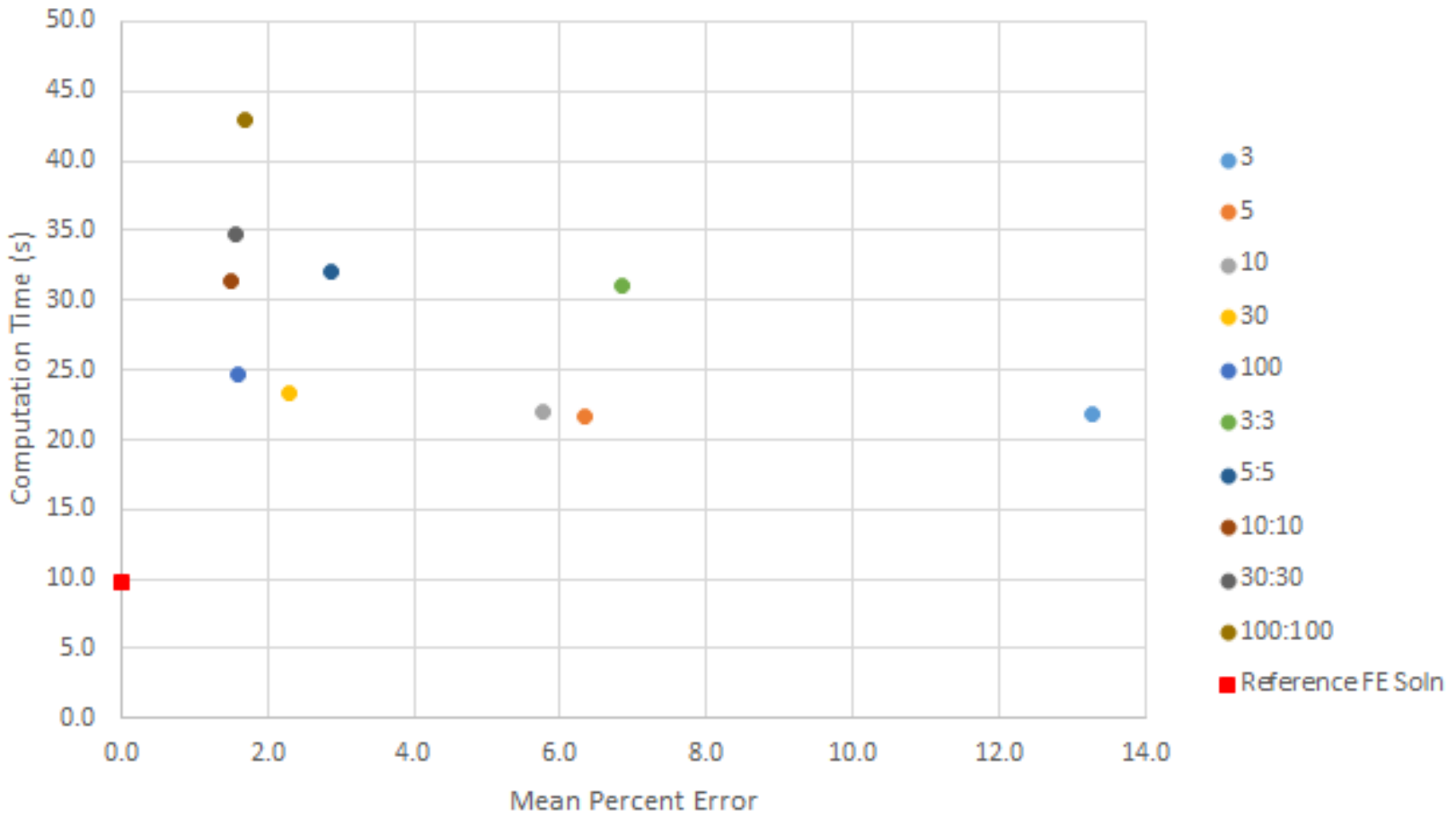
"3"

Benchmark Problem

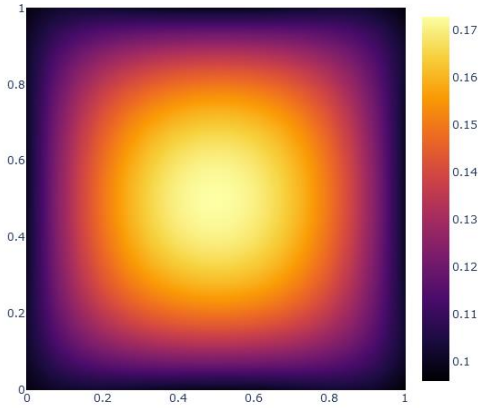


PINNs Vs. Finite Elements

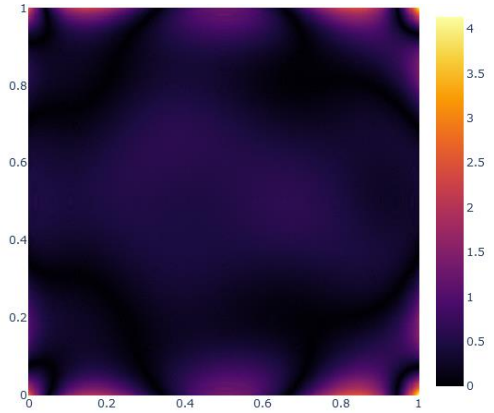
Training Time vs Domain Percent Error After 10,000 Training Iterations



100:100 Temperature

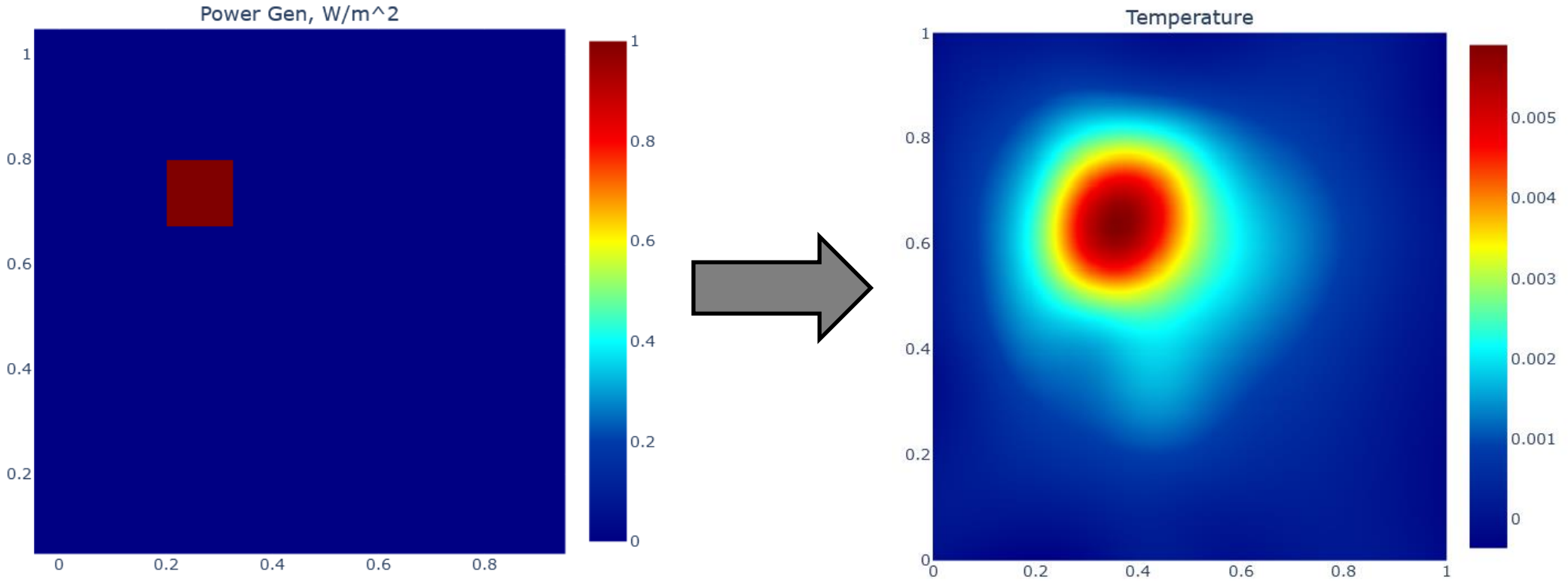


100:100 Percent Error



Capturing Complex Powermaps

- Sharp, non-linear, transitions can be learned by PINNs
- This is important for simulating problems with measured inputs such as a discrete power map



Variational Formulation

- The strong form loss function of traditional PINNs can be recast into a variational form
- This increases the computational efficiency of training by eliminating second order gradients
- There is no analog to automatic differentiation for integration so quadrature must be used during training

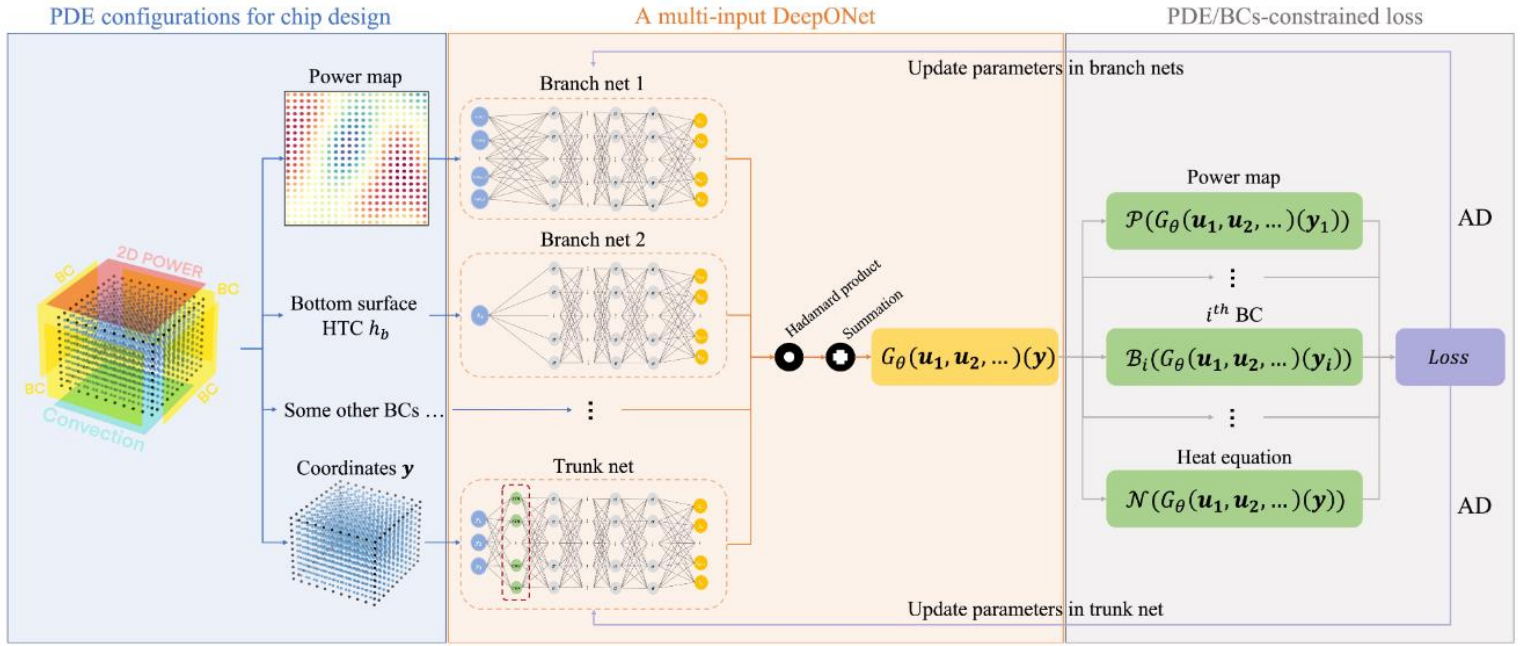
$$\begin{aligned} \nabla \cdot k\nabla T &= f \quad \epsilon\Omega \\ T &= \bar{T} \quad \epsilon\Gamma_T \\ q &= \bar{q} \quad \epsilon\Gamma_q \\ q &= -k\nabla T \end{aligned}$$

$$k\int_\Omega w\nabla^2 T d\Omega = \int_\Omega wf d\Omega \qquad k\int_\Omega \nabla(w\nabla T)d\Omega - k\int_\Omega \nabla w\nabla T d\Omega = \int_\Omega wf d\Omega$$

$$k\int_\Gamma \hat{n}\cdot(w\nabla T)d\Omega - k\int_\Omega \nabla w\nabla T d\Omega = \int_\Omega wf d\Omega \qquad Choose\ w = 0\ on\ \Gamma \qquad k\int_\Omega \nabla w\nabla T d\Omega + \int_\Omega wf d\Omega = 0$$

Increasing the Design Space

- While finding the solution to a single linear PDE with PINNs is slower and less accurate than finite elements, the network can be re-formulated to solve multiple problems at once
- Recently, a network was trained to solve the 3D poison equation with arbitrary boundary conditions and power generation



- **Training: 2-10 hours**
- **Evaluation: 0.001-0.01 seconds**

Z. Liu et al., "DeepOHeat: Operator Learning-based Ultra-fast Thermal Simulation in 3D-IC Design," https://doi.org/10.48550/arXiv.2302.12949

Conclusions

- Heterogeneous systems are a complex collection of chip packages and require multiple strategies to address geometry spanning 7-8 orders of magnitude in length scale
  - Developed general data structure for geometry modeling, accelerated solvers and domain decomposition strategies for flexible solution
- While PINNs present exciting new ways to solve PDEs, they do not show efficiency or accuracy advantages compared to Finite Element Methods when used to approximate Linear PDEs using their strong or variational forms
- Even if training time is large, PINNs' evaluation time is faster than FE solution time
- To efficiently explore a design space, more model parameters (i.e. conductivity or power generation fields) must be cast as inputs to the network or a deep neural network can be trained on Finite Element simulations