

# A Parallel Direct Solver for the Simulation of Large-scale Power/Ground Networks

Venkataramanan (Ragu) Balakrishnan  
School of Electrical and Computer Engineering  
Purdue University

Joint work with **Stephen Cauley** and Cheng-Kok Koh

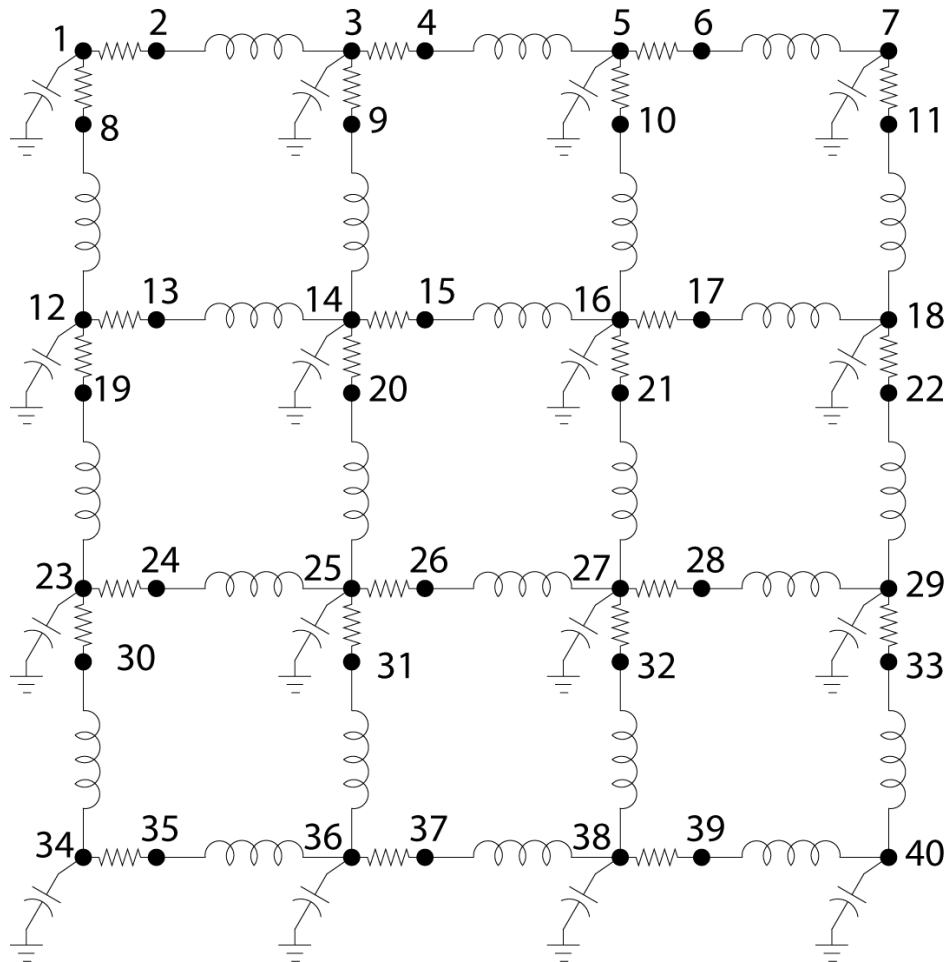
# Context and Motivation

- Application-oriented
- Addresses power mesh simulation in VLSI
  - Power provided at a grid of points
  - Very large grid size leads to a very large scale simulation problem: transient simulation to model switching
  - Accuracy important because of voltage scaling
- Challenges:
  - Prohibitive amounts of computation
  - Extraordinary demands on memory
  - Numerical instability

# Approach

- Regularity of power mesh structures leads to (approximation by) structured matrices
- Theory of structured matrices:
  - Advantages in computation
  - Parallelization

# 4x4 RLC Mesh Structure



# The model

- Standard state-space modeling:
  - State variables are the node voltages and branch currents
  - Resistance, capacitance, and inductance elements are coupled via incidence matrices
- Current sinks are attached to several nodes in order to model the switching activity

# State-space Model

$$\mathcal{G}x + C\dot{x} = B$$

$$\mathcal{G} = \begin{pmatrix} G & A_l^T \\ -A_l & 0 \end{pmatrix}, C = \begin{pmatrix} C & 0 \\ 0 & L \end{pmatrix},$$

$$x = \begin{pmatrix} v_n \\ i_l \end{pmatrix}, B = \begin{pmatrix} A_i^T I_s \\ 0 \end{pmatrix},$$

$$G = A_g^T R^{-1} A_g, \text{ and } C = A_c^T \hat{C} A_c.$$

$R, \hat{C}, L$  ~ Resistance, capacitance, and inductance.

# Nodal Analysis

- Trapezoidal method is used for numerical integration of underlying ordinary differential equations
- A uniform discretization of the time axis is assumed with resolution  $h$
- Defining  $v_n^k = v_n(kh)$  to be the voltage at the  $n$ th node, we may solve for  $v_n^{k+1}$  in terms of  $v_n^k$  through the Nodal Analysis (NA) equations

# NA Equations

- Linear system of equations that needs to be solved at each time step:

$$\underbrace{\left( G + \frac{2}{h}C + \frac{h}{2}S \right)}_K v_n^{k+1} = \underbrace{\left( -G + \frac{2}{h}C - \frac{h}{2}S \right)}_H v_n^k + A_i^T I_s^{k+1} + I_s^k - 2A_l^T i_l^k,$$

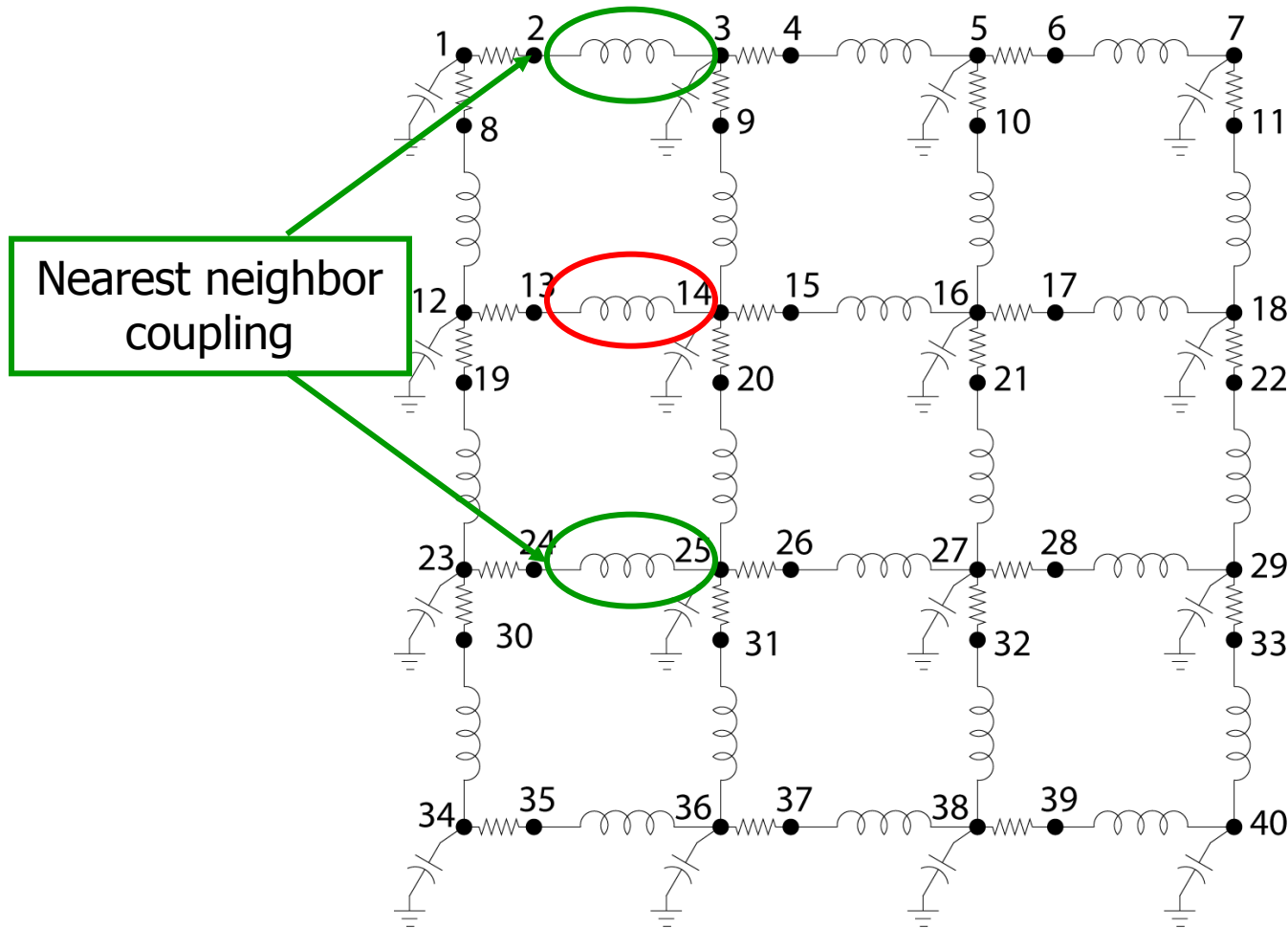
$$2A_l^T i_l^{k+1} = 2A_l^T i_l^k + hS v_n^{k+1} + v_n^k$$

$$S = A_l^T L^{-1} A_l \quad (\text{Susceptance matrix})$$

# Numerical Challenges

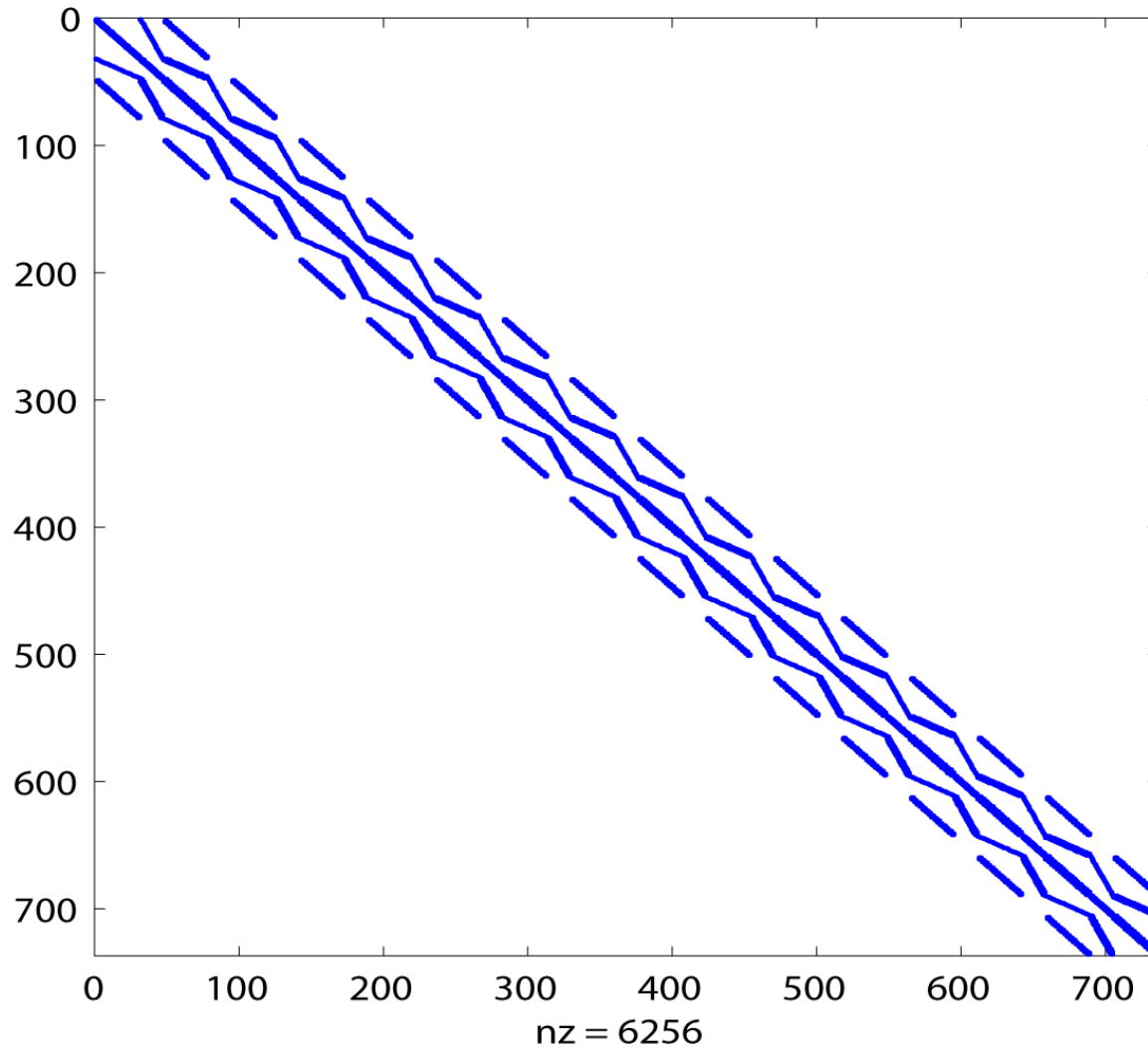
- Focusing on  $K = G + \frac{2}{h}C + \frac{h}{2}S$  :  $G$  and  $C$  are sparse, but  $S$  makes  $K$  dense
- Two questions:
  - Noting that  $S = A_l^T L^{-1} A_l$ , alternate sparse modeling strategies for reluctance matrix  $L^{-1}$  ?
  - If  $K$  can be sparsified, can we use *direct* methods to solve  $Kv = i$  ?

# 4x4 RLC Mesh Structure

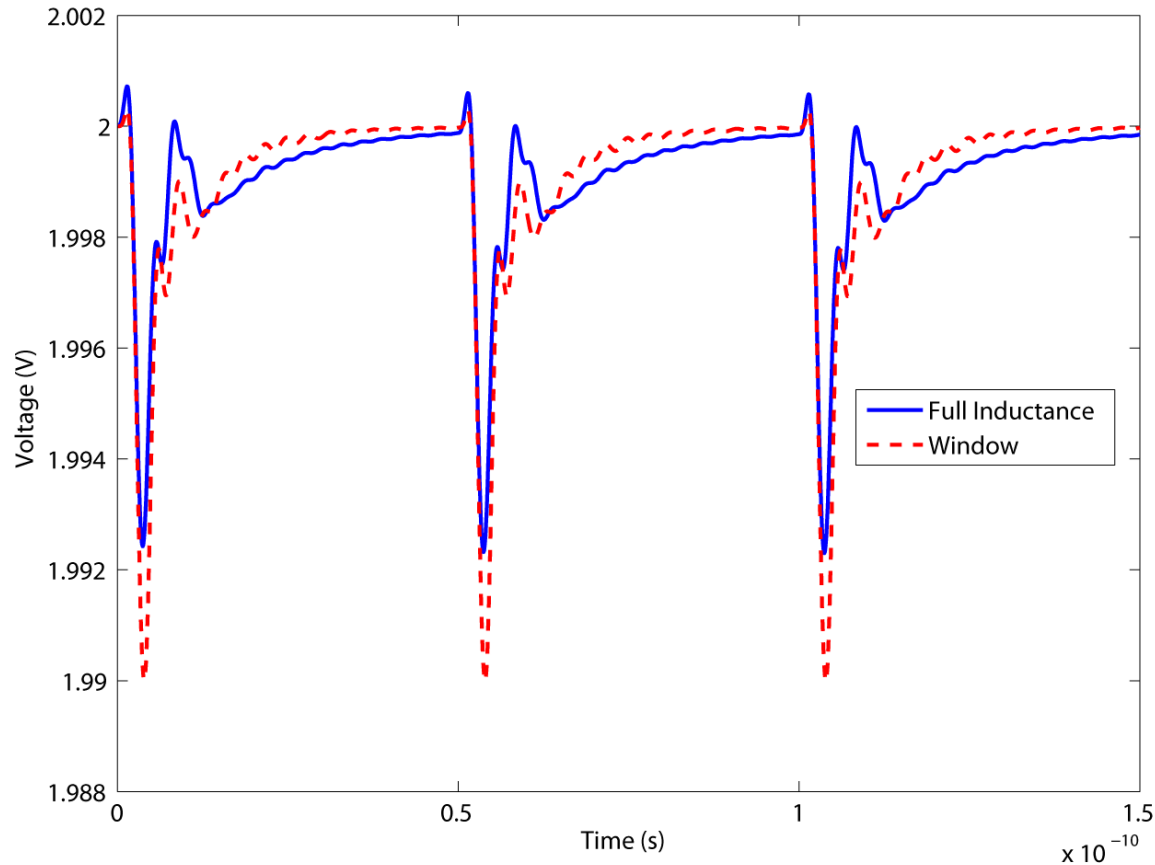


How accurate is the model?

# 16x16 RLC Mesh Structure



# 16x16 RLC Mesh Structure



Can we include more coupling terms to improve accuracy?

# Sparse Approximate Inverse

- Technique due to M. Grote, et al. to form a preconditioner for iterative linear solvers
- Given an inductance matrix  $L$  the SPAI method can be used to form another matrix  $M$  in order to match the inverse under the Frobenius norm:

$$\|LM - I\|_F^2 = \sum_{i=1}^n \|LM - I e_i\|_2^2,$$

$$\min \|Lm_i - e_i\|_2^2, \quad i = 1, 2, \dots, n$$

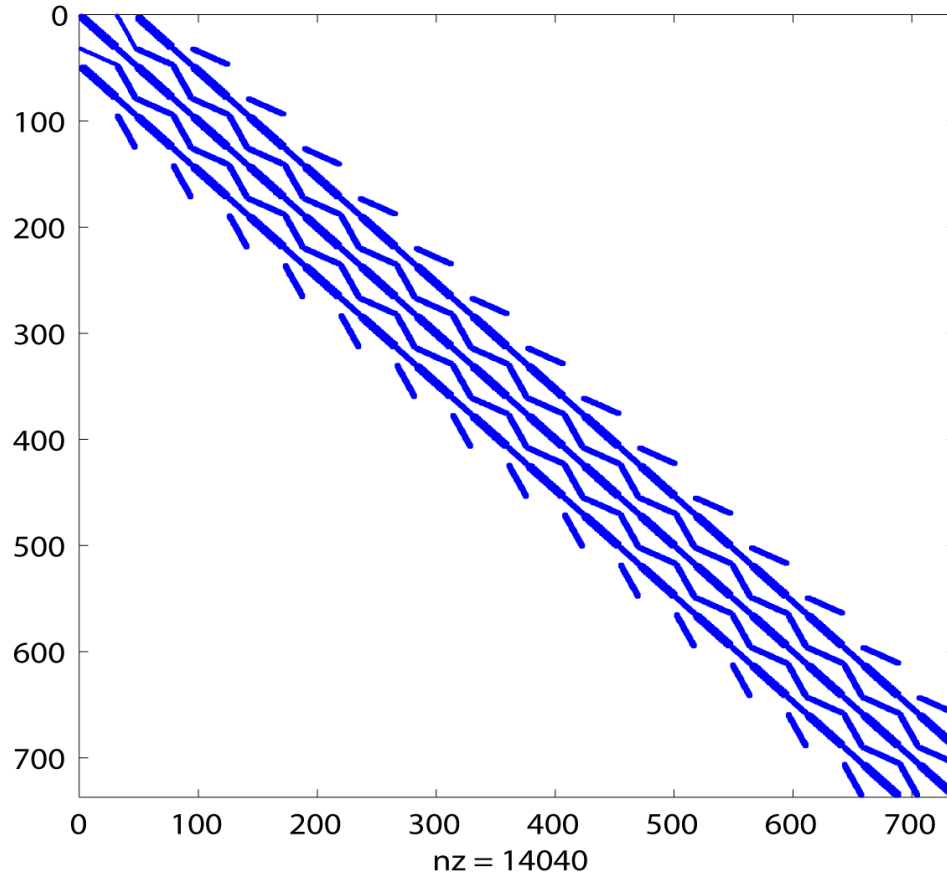
# Sparse Approximate Inverse

- The diagonal entries  $M_{i,j}$  are always included
- Drop tolerance  $\tau$  allows for control of sparsity:  
Drop  $M_{i,j}$  if

$$|M_{i,j}| \leq (1 - \tau) \max_j |M_{i,j}|$$

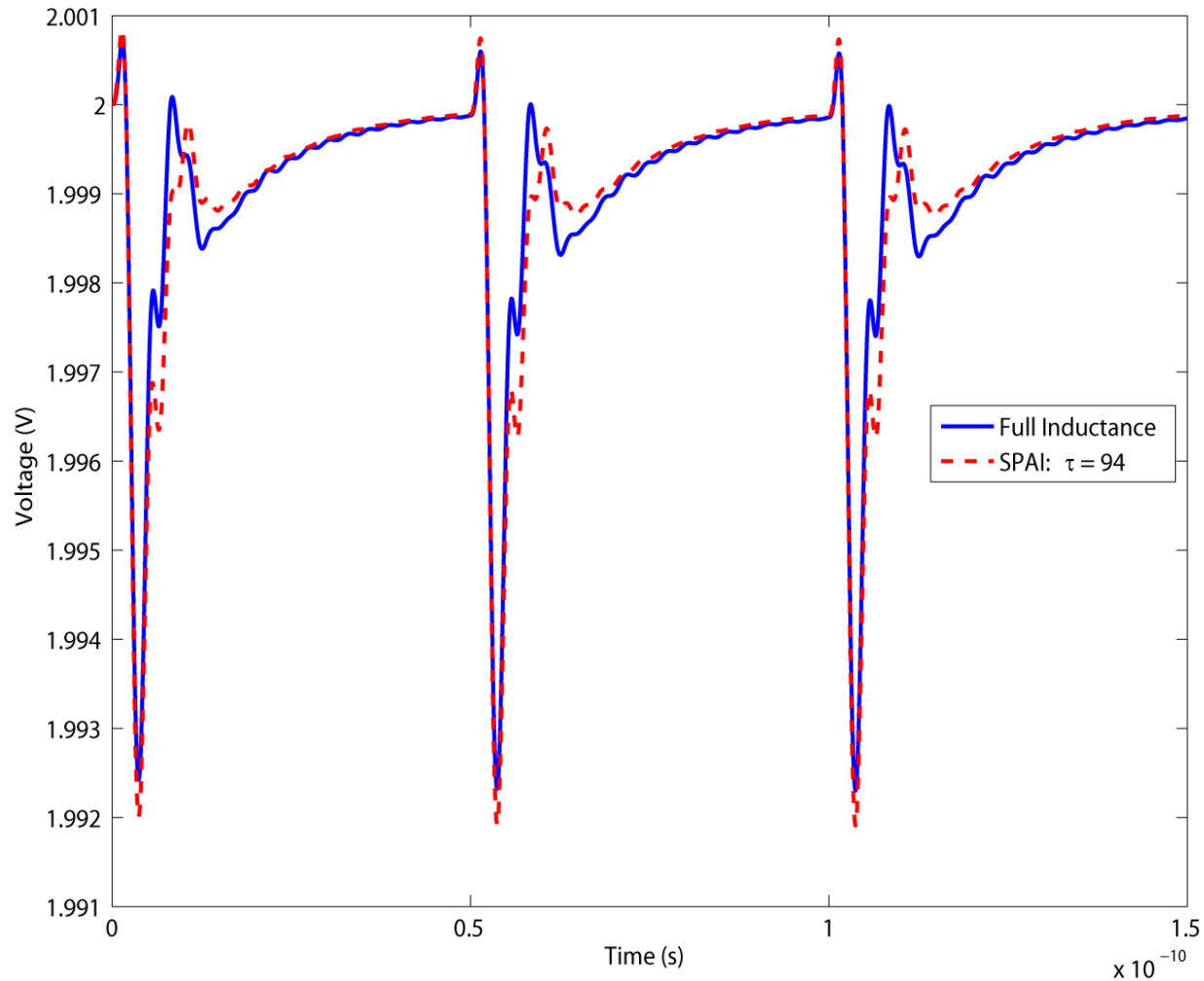
- If  $\tau$  is close to zero, this criterion would prevent fill-in and result in a matrix that is very sparse
- If  $\tau = 1$  the entire pattern of  $L^{-1}$  will be considered

# Block Tridiagonal Approximation

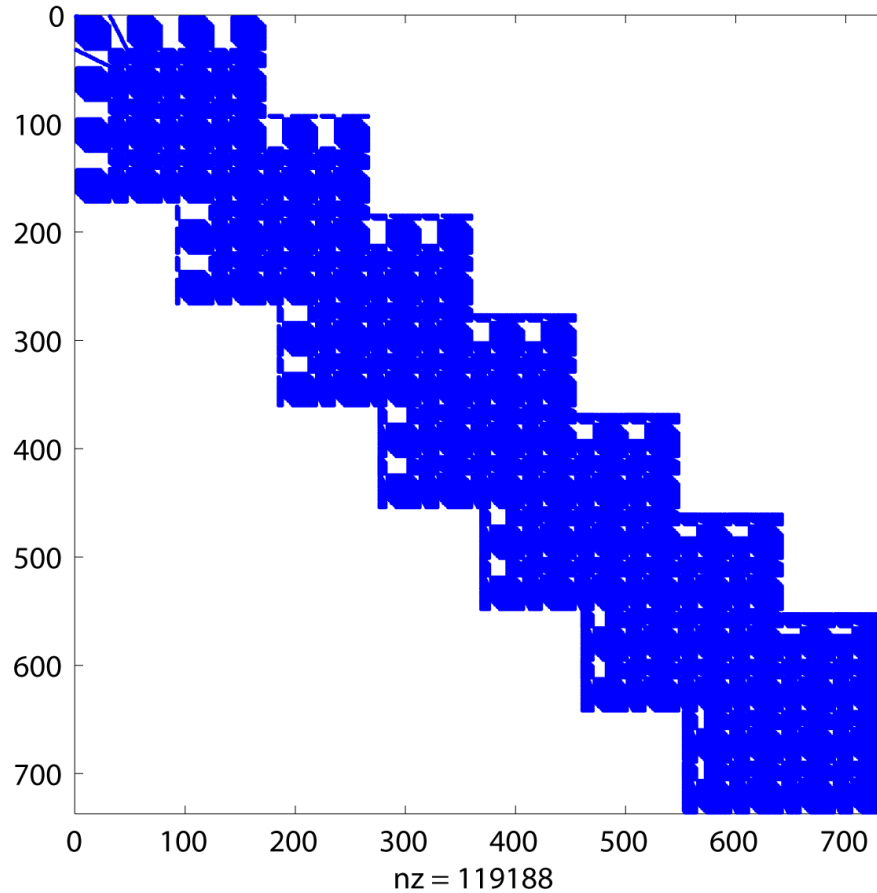


$$\tau = 0.94$$

# 16x16 RLC Mesh Structure

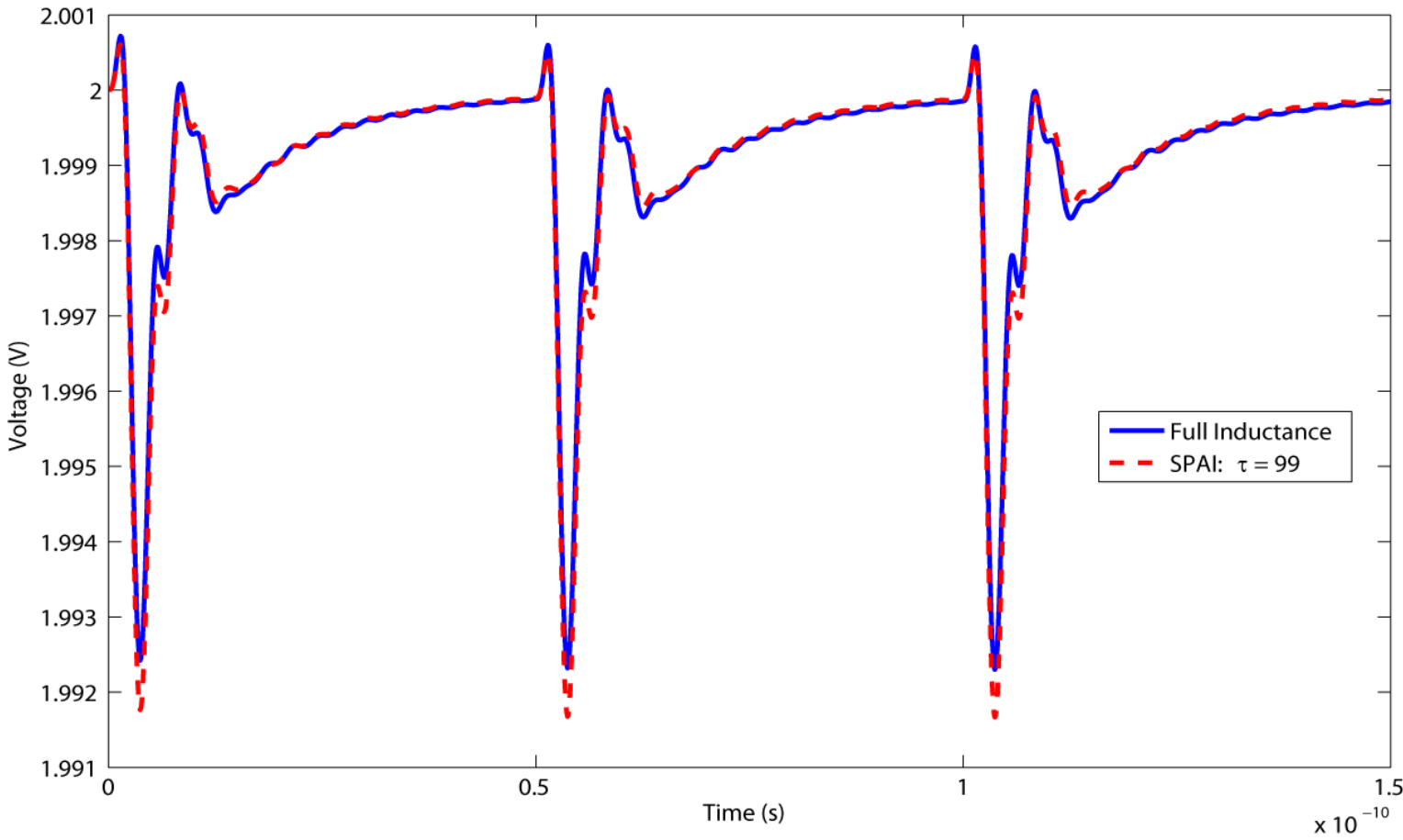


# Block Tridiagonal Approximation



$$\tau = 0.99$$

# 16x16 RLC Mesh Structure



# Direct Methods for $Kv = i$

- For block-tridiagonal  $K$ , there exists a *compact* representation for its inverse
- Suppose

$$K = \begin{bmatrix} A_1 & -B_1 & & & \\ -B_1^T & A_2 & -B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -B_{N_y-2}^T & A_{N_y-1} & -B_{N_y-1} \\ & & & -B_{N_y-1}^T & A_{N_y} \end{bmatrix}$$

# Compact Representation for $K^{-1}$

$$K^{-1} = \begin{pmatrix} D_1 & D_1 S_1 & \cdots & D_1 \prod_{k=1}^{N_y-1} S_k \\ R_1 D_1 & D_2 & \cdots & D_2 \prod_{k=2}^{N_y-1} S_k \\ \vdots & \vdots & \ddots & \vdots \\ \left( \prod_{k=N_y-1}^1 R_k \right) D_1 & \left( \prod_{k=N_y-1}^2 R_k \right) D_2 & \cdots & D_{N_y} \end{pmatrix}$$

$$R_1 = A_1^{-1} B_1,$$

$$R_i = A_i - B_{i-1}^T R_{i-1}^{-1} B_i, \quad i = 2, \dots, N_y - 1,$$

$$D_1 = A_1 - S_1 B_1^T^{-1},$$

$$S_{N_y-1} = B_{N_y-1} A_{N_y}^{-1},$$

$$D_{i+1} = A_{i+1} - S_{i+1} B_{i+1}^T^{-1} I + B_i^T D_i S_i, \quad i = 1, \dots, N_y - 2,$$

$$S_i = B_i A_{i+1} - S_{i+1} B_{i+1}^T^{-1}, \quad i = N_y - 2, \dots, 1,$$

$$D_{N_y} = A_{N_y}^{-1} I + B_{N_y-1}^T D_{N_y-1} S_{N_y-1}.$$

# Compact Representation

$$\text{tri}(K^{-1}) = \begin{pmatrix} D_1 & Z_1 & & & & \\ Z_1^T & D_2 & & & & \\ & \ddots & \ddots & & & \\ & & & D_{N_y-1} & Z_{N_y-1} & \\ & & & Z_{N_y-1}^T & D_{N_y} & \\ & & & & & \end{pmatrix} = \begin{pmatrix} D_1 & D_1 S_1 & & & & \\ R_1 D_1 & D_2 & & & & \\ & \ddots & \ddots & & & \\ & & & D_{N_y-1} & D_{N_y-1} S_{N_y-1} & \\ & & & R_{N_y-1} D_{N_y-1} & D_{N_y} & \end{pmatrix},$$

$$D_i S_i = Z_i \Rightarrow S_i = D_i^{-1} Z_i, \quad i = 1, \dots, N_y - 1,$$

$$R_i D_i = Z_i^T \Rightarrow R_i = Z_i^T D_i^{-1}, \quad i = 1, \dots, N_y - 1,$$

The compact representation can be completely determined from the block tridiagonal entries of the inverse.

# Matrix Inversion Complexity

- Assuming (block size, # blocks) :  $(N_x, N_y)$
- Computational complexity:  $O(N_x^3 N_y)$
- Memory complexity:  $O(N_x^2 N_y)$
- Can the matrix inversion problem be distributed?

# Divide and Conquer Approach

$$A = \begin{pmatrix} S_1 & \\ & S_2 \end{pmatrix} + \underbrace{XY^T}$$

Low rank

$$A^{-1} = \begin{pmatrix} S_1^{-1} & \\ & S_2^{-1} \end{pmatrix} + \text{Correction terms}$$

- Compute inverse of block-tridiagonal matrices  $S_i$
  - Adjust for low rank correction term
- (Procedure continued recursively)

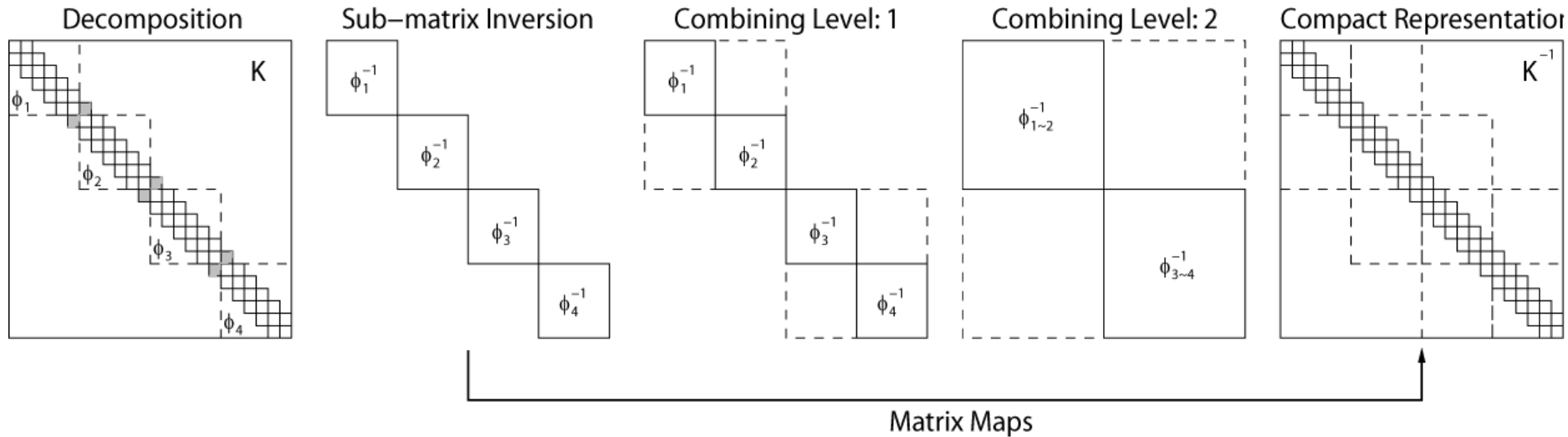
# Applying low-rank corrections

$$A = \begin{pmatrix} S_1 & \\ & S_2 \end{pmatrix} + \underbrace{XY^T} = \begin{array}{|c|c|} \hline \text{yellow } S_1^{-1} & \text{red } \\ \hline \hline & \text{blue } \\ \hline \hline \text{cyan } S_2^{-1} & \text{red } \\ \hline \hline \end{array} + \begin{array}{|c|c|c|} \hline \text{red } & \text{blue } & \text{red } \\ \hline \hline \hline \end{array}$$

## ■ Observations:

- Corrections to entries of inverse can be expressed as functions of first block-row and last block-column of sub-problem solutions
- Functions depend on “corner blocks” of sub-problem solutions; can be represented by a few  $N_x \times N_x$  matrices (“matrix maps”)

# Parallel Divide-and-conquer



- Need only block-tridiagonal part
- When combining, intermediate entries not explicitly calculated
- “Matrix maps”, which depend on  $N_x \times N_x$  matrices are updated at each combining step; these express final block tridiagonal entries in terms of the entries of the lowest-level inverses

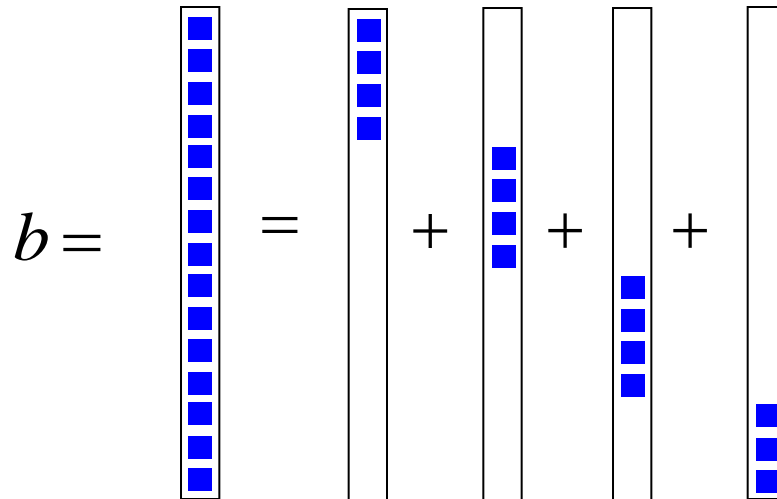
# Performance

- Twelve  $N_x \times N_x$  maps needed
- Computational complexity:  $O\left(\frac{N_x^3 N_y}{p_y} + N_x^3 \log_2 p_y\right)$
- Memory complexity:  $O\left(\frac{N_x^2 N_y}{p_y} + N_x^2\right)$
- Still need to evaluate:  $Kx = b \rightarrow x = K^{-1}b$
- Can parallelize matrix vector multiply

# Parallel Matrix-Vector Multiply

- Approach:

- Separate the problem:  $x = K^{-1}b = K^{-1} \hat{b}^{(1)} + \dots + \hat{b}^{(p)}$



- Create several recursions, passing information to neighboring processors when necessary.

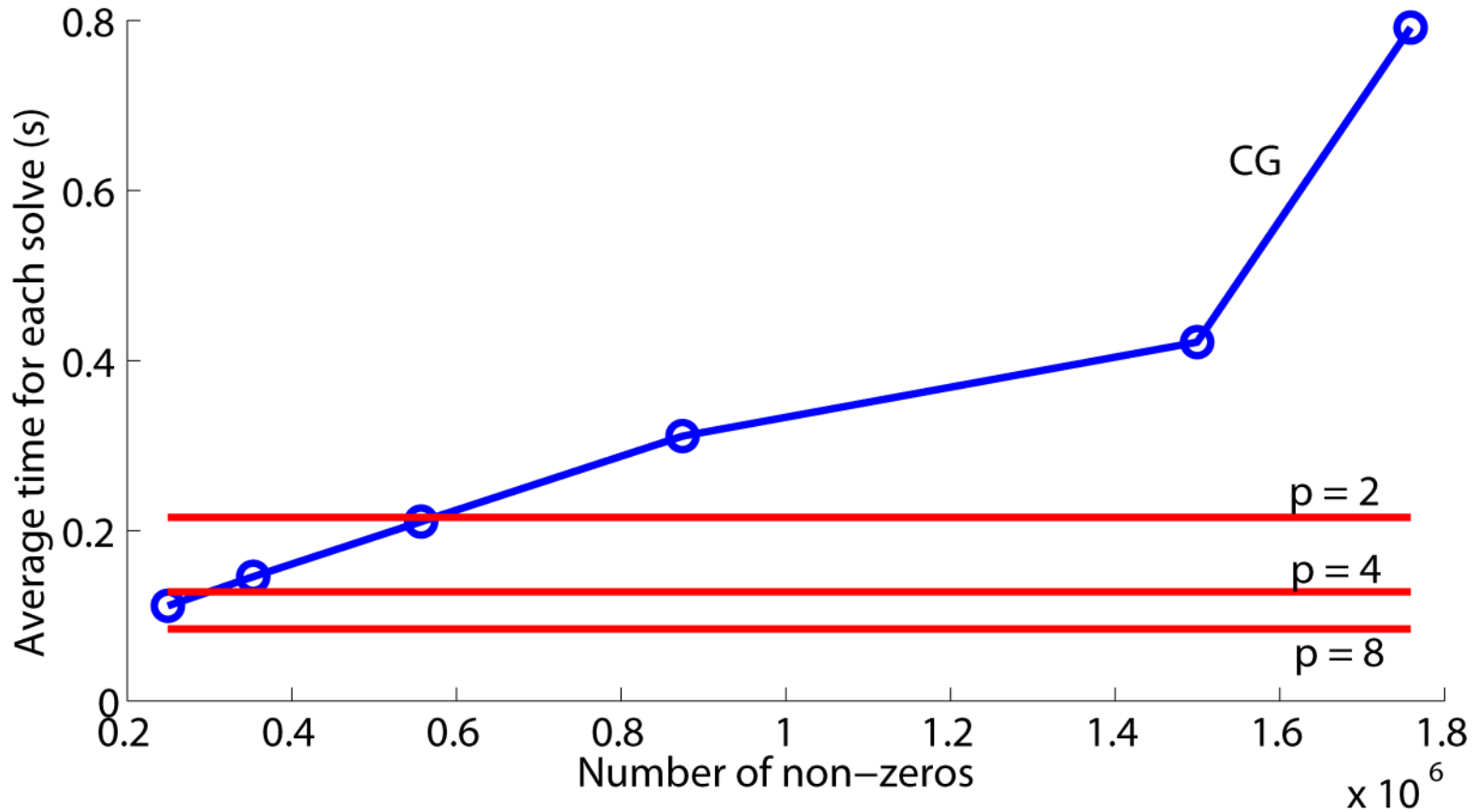
# Memory Limitations of Direct Methods for Simulation

Data		Our Algorithm	LU	UMF
16 × 16	$\tau = 0.99$	2.01E-03	2.78E-03	2.61E-03
32 × 32	$\tau = 0.99$	1.45E-02	2.25E-02	1.90E-02
48 × 48	$\tau = 0.99$	4.43E-02	6.21E-02	6.03E-02
64 × 64	$\tau = 0.99$	8.73E-02	1.41E-01	1.48E-01
128 × 128	$\tau = 0.97$	7.29E-01	1.25E+00	7.71E-01
192 × 192	$\tau = 0.90$	1.50E+00	1.67E+00	-
256 × 256	window	2.25E+00	-	-
384 × 384	window	6.57E+00	-	-
512 × 512	window	1.08E+01	-	-

TABLE III

PERFORMANCE OF DIRECT ALGORITHMS ACROSS MESH SIZE. TRANSIENT STEP SOLVE TIMES ARE SHOWN IN SECONDS; THE LACK OF MEMORY SCALABILITY FOR EXISTING DIRECT APPROACHES IS SHOWN THROUGH AN INABILITY TO PERFORM THE LARGER SIMULATIONS.

# Scalability of Iterative Methods



# Conclusions

- Currently employed techniques trade-off accuracy for simulation time via iterative schemes
- Our algorithm is a non-iterative tool that facilitates the simulation of large mesh structures through a divide-and-conquer approach
- The scalability of the divide-and-conquer method is demonstrated by the absence of increases to time for the primary computational task for transient simulation:
  - when considering the inclusion of these additional coupling terms, through the SPAI method
- The divide-and-conquer approach allows for the simulation of a 512x512 RLC mesh with a speed-up factor over 9x when compared to the CG method with ILU