

GRID EMT

SIMULATION INITIATIVE

Structures in EMT Linear System and Adaptive Schur Solver

Presented by David Gleich (Purdue University)
Joint work with Zhiyao Xu (Purdue),
Joseph Francis (ORNL), Phani Marthi (ORNL)

August 20, 2026



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

Motivation

Solving linear systems fast in EMT simulation involves utilizing the structural property of the matrix, including:

- Block diagonal (**BD**) – fully decoupled structure
- Bordered Block Diagonal (**BBD**) – minimal coupling

This demo is about how graph and network techniques can be used to inform us the structure inside a linear system.

And how we can use that to accelerate a GPU solve



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

Linear system viewed as graph

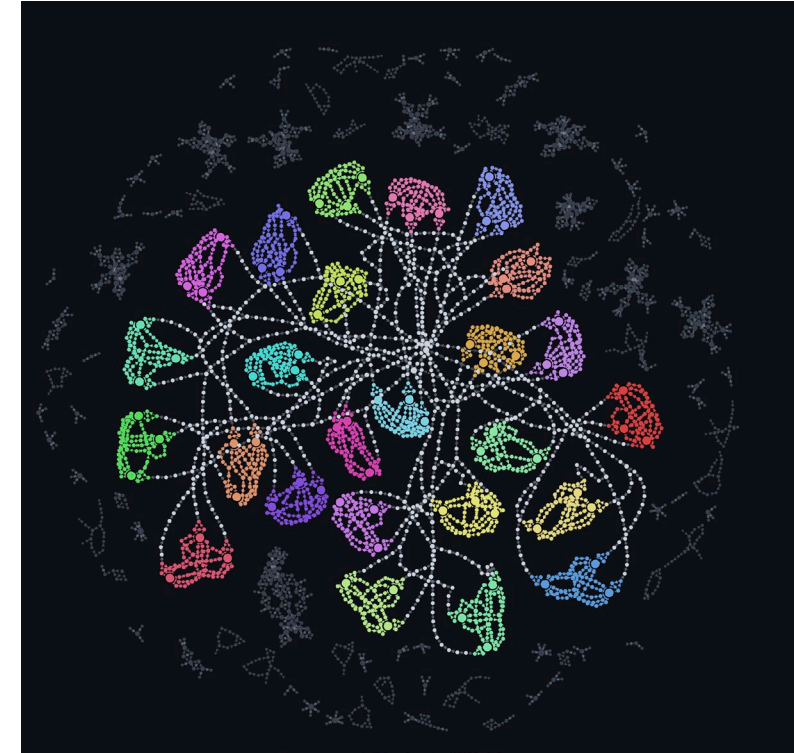
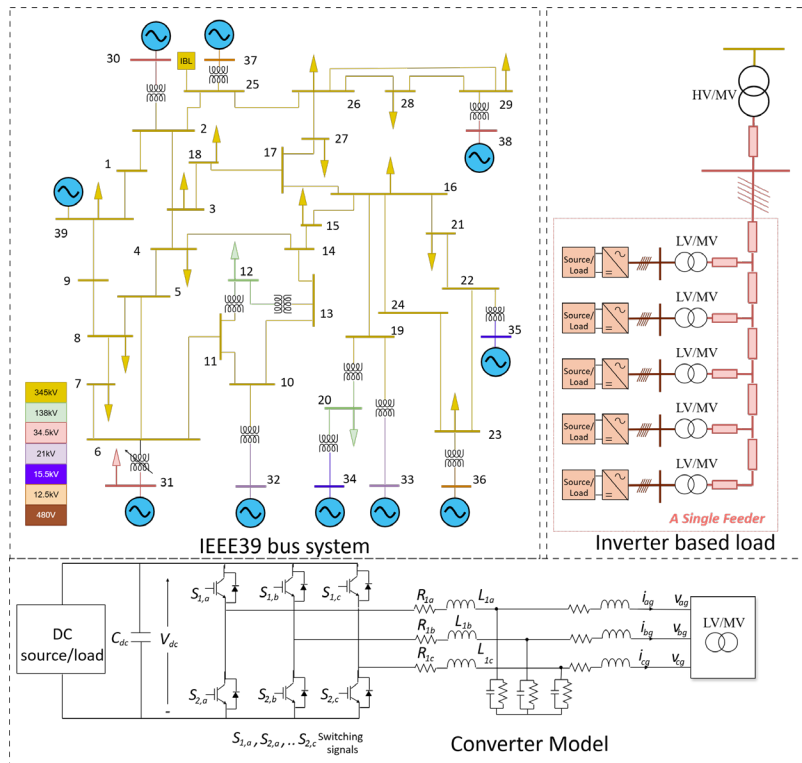
(1)
(2)
(3)
(4)
(5)

We can view linear system as a network where:

- nodes are equations
- links are shared variables



A small grid example: IEEE39 with IBL

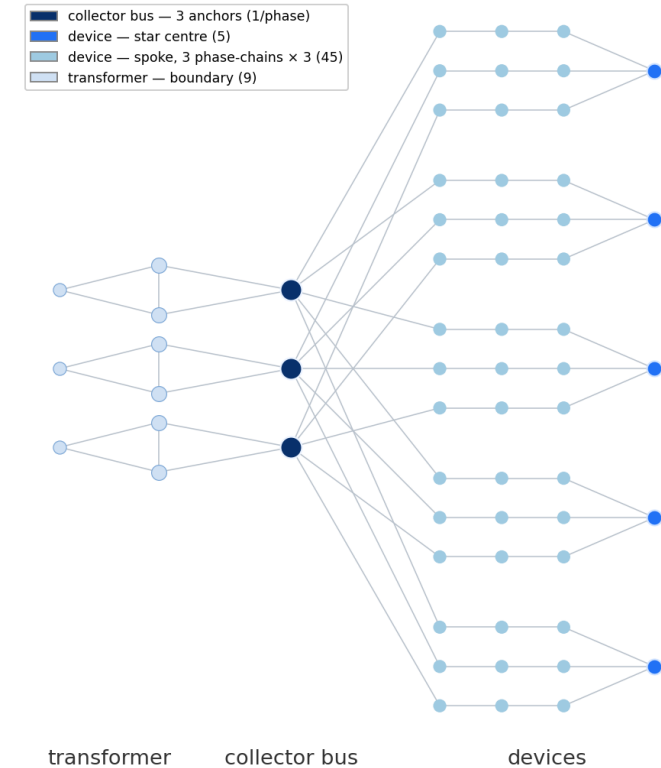


Example: repeating 62x62 pieces



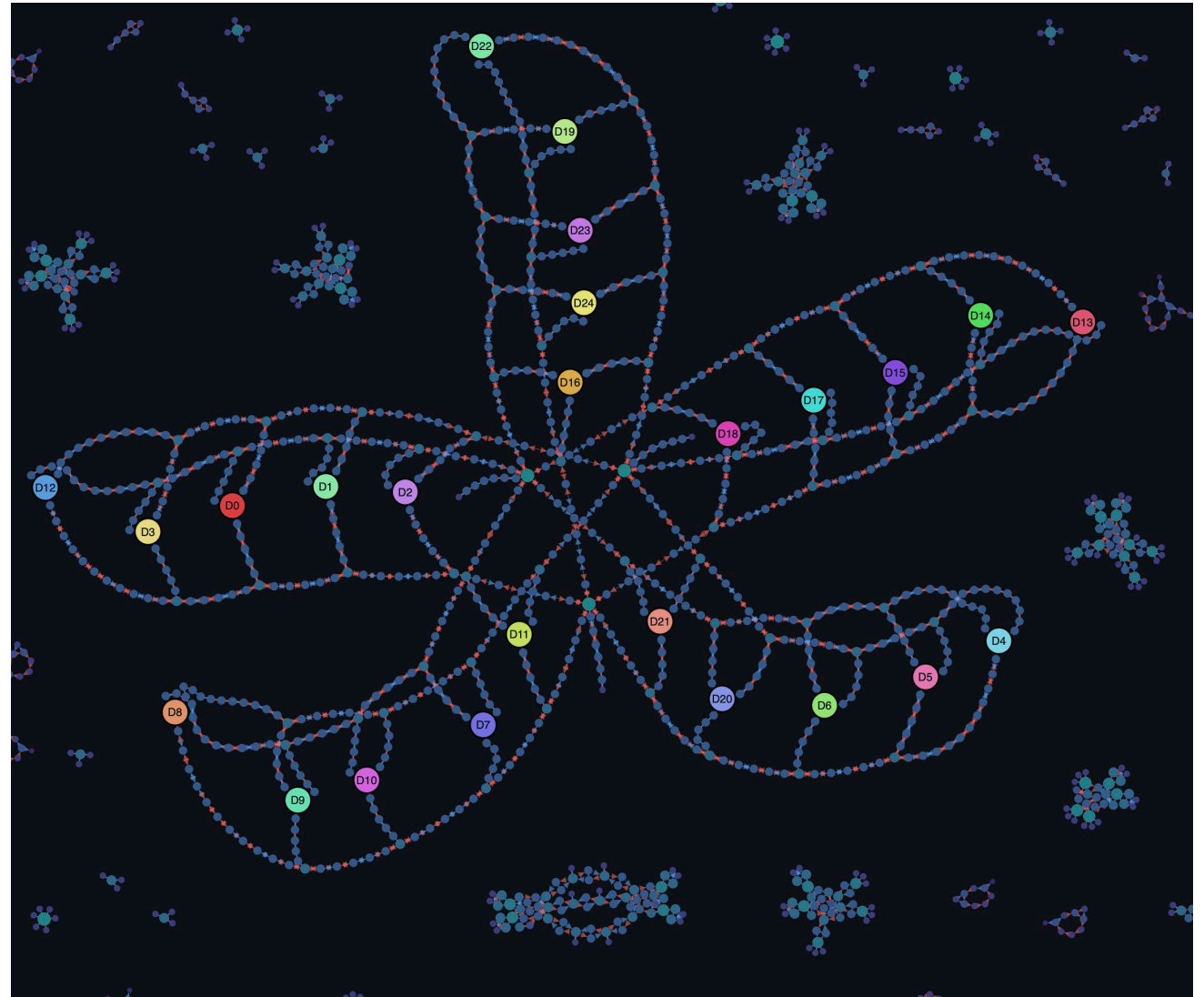
These are the inverters at the end of the distribution chain.

One IBL device block as physical topology — transformer · collector bus · 5 devices (62 nodes)



The grid topology and structure is visible in these visualizations

Here is the 5 leg feeder hosting the 125 inverters



Knowing the structure

Linear solver's knowledge of system structure is:

Notified, by the producer of the matrix

or

Analyzed, paying a cost per matrix change

(can be partially updated)

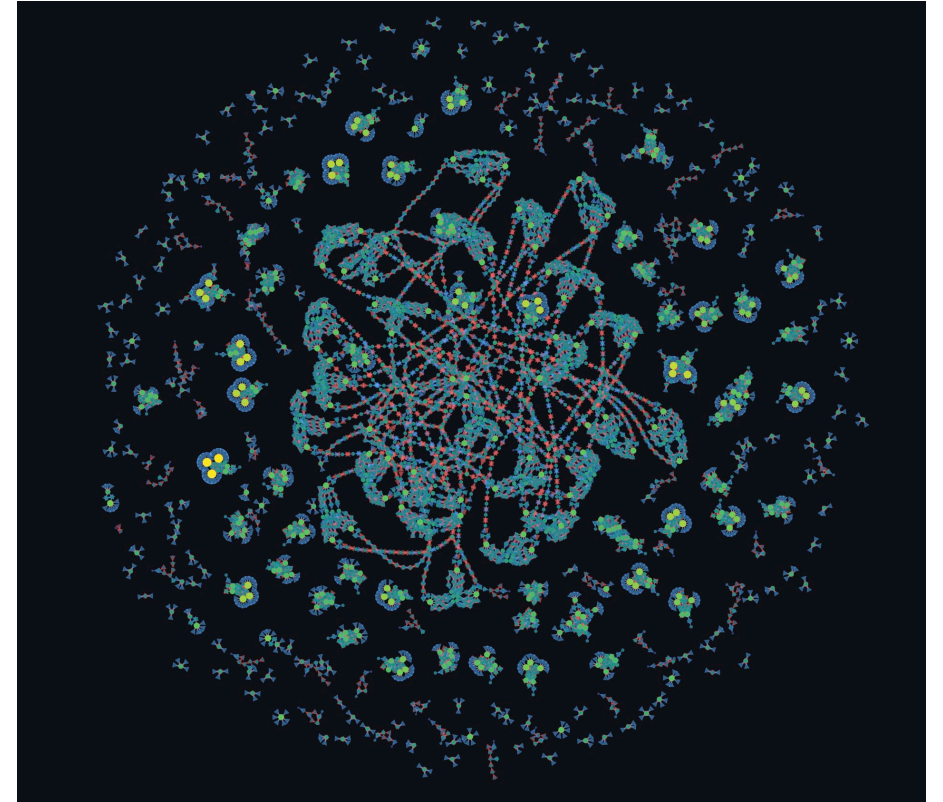


U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

Analyze Structure

BFS to get connected components. (reachability)
Reordering to get BD matrix structure.



Analyze structure

Inside large blocks, set high-degree nodes as hub / connection-line candidates (most coupling), set the rest as interior blocks (least coupling).

Results in a BBD / arrowhead matrix:

Bordered Block Diagonal / Block Arrowhead Matrix



Analyze structure

The overall system could be a BD of BBD matrices, that looks like

Upon entry value ch

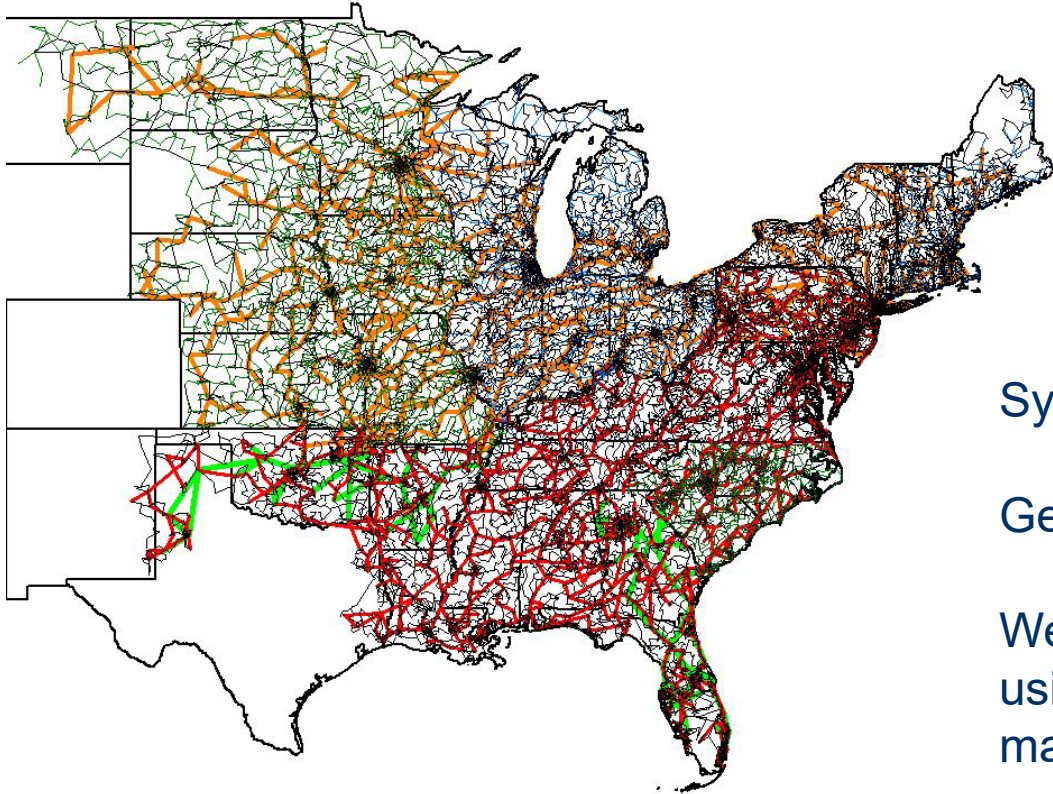


ed.

A_i may have similar BBD structure or even $A_i = A_j$



ACTIVSg70k: a nation scale synthetic grid



Synthetic 70k bus grid of eastern U.S.

Generation and load profiles are similar to real public data.

We assemble a 1.8 million by 1.8 million linear system from it using an Claude to generate EMT-like structures similar to matrices from REINTEGRATE.

ACTIVSg70k synthetic grid, Texas A&M Electric Grid Test Case Repository (electricgrids.engr.tamu.edu)



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

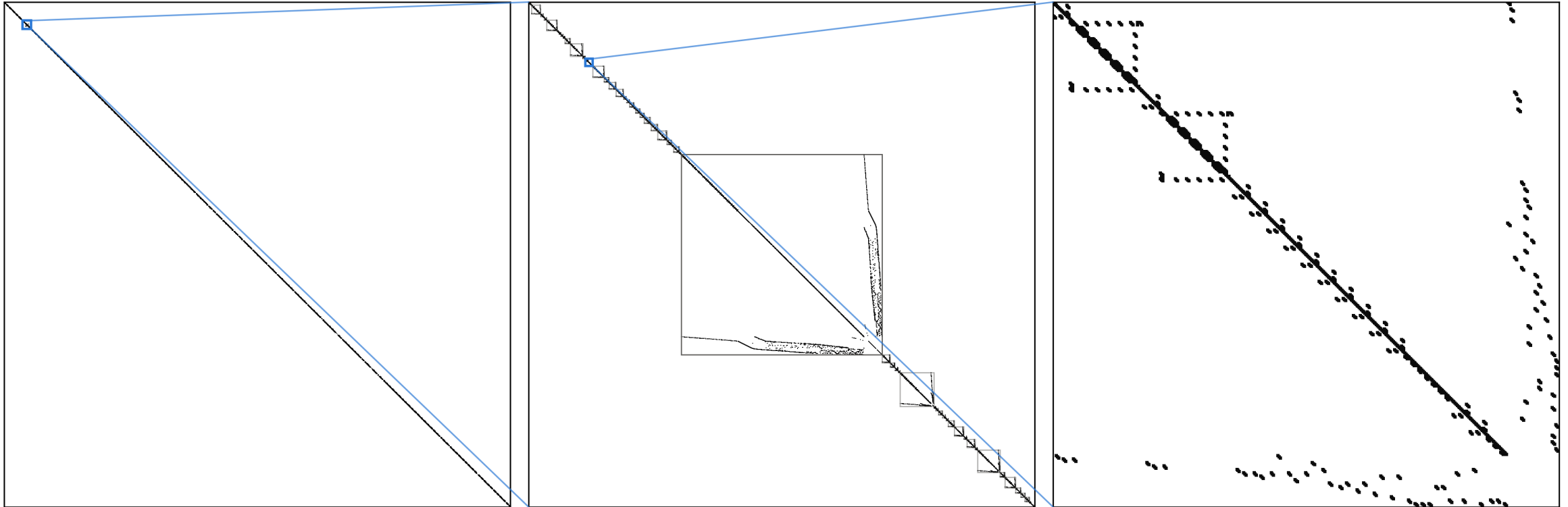
Linear system from ACTIVSg70k

Very sparse, averages less than 3 entries per row

full system: $n = 1,812,163$

zoom $\times 60$: a 29k window

zoom $\times 4,090$: one block, $s = 443$

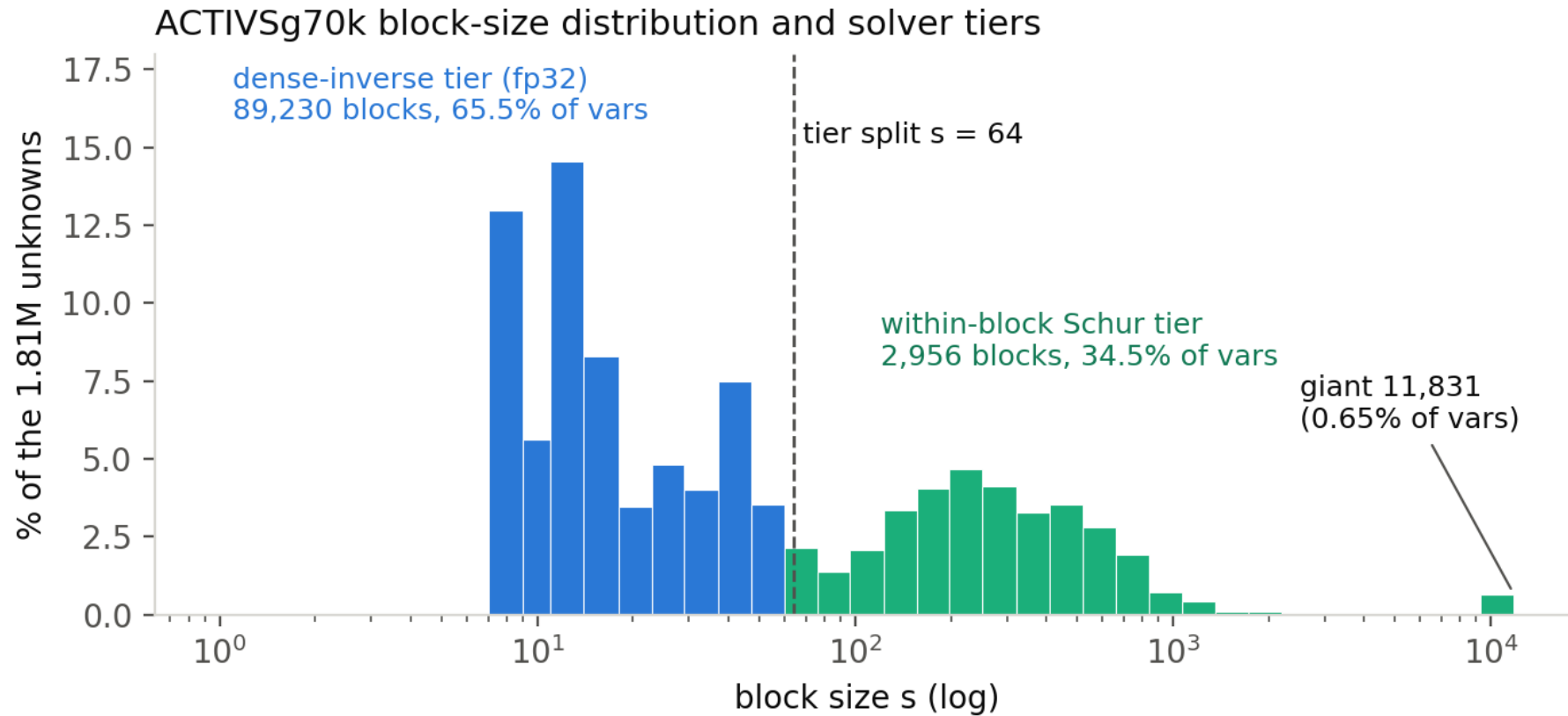


This matrix is systematically generated with AI assistance from Texas A&M ACTIVSg70k synthetic powergrid. It is expected to reflect the structure, sparsity, and other properties of smaller EMT linear systems generated by various DAE methods.



U.S. DEPARTMENT
of **ENERGY** | Office of
Electricity

Linear system from ACTIVSg70k

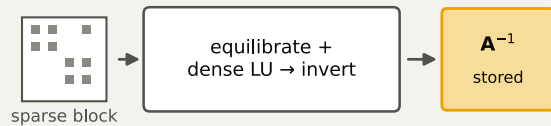


Solver pipeline

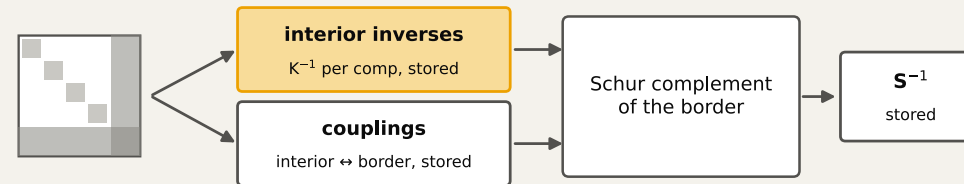
SETUP — once per topology, on CPU (OpenMP), all arithmetic in fp64

STRUCTURE — today: BFS finds decoupled blocks; in large ones, degree $\geq 5 \rightarrow$ border, another BFS splits the interior
— future: the assembler reports structure and unit scales

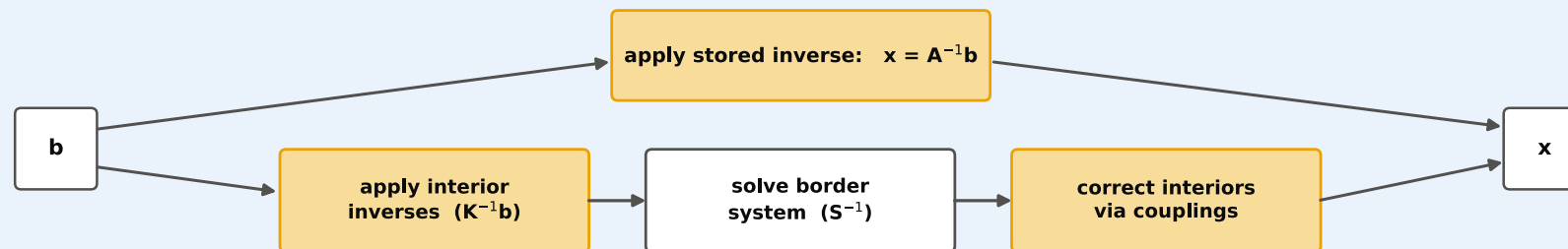
TIER A blocks ≤ 64 — dense inverse



TIER B blocks > 64 — within-block Schur



PER-RHS — every time step, on GPU as one CUDA graph, streaming the stored data once per solve



 data stored on the GPU in fp32



Solver pipeline

A CUDA graph encodes:

- Parallel block-matrix-inverse application
 - Border-coupling-inverse application
 - Interior correction based on coupling
- within a single launch.

Sparse blocks exploit sparsity, dense inverse application promotes parallelism.



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

Handling changing matrices (LHS)

We expected structured variation *within* each block.

- Each inverter block will only exist in a handful of states
- Generator blocks could be approximated by a nearby block from a handful of states.

This is currently planned for future work.



Key properties of solver

- 0.9s fresh setup from CSR, mostly CPU; 245 μ s per solve on GPU (RHS on GPU).
- KLU solve was $\sim$ 6000 μ s per RHS on CPU.
- CUDA GPU accelerates matrix routines (highly parallel blocks, explicit inverse).
- Despite dense storage and dense routine, sparsity is exploited by the BD and residing BBD structure.
- GPU Bandwidth, not throughput, is the bottleneck.
- FP32-stored inverse doubles bandwidth performance, against FP64.



Current programming interface

- We have designed a KLU-like interface to make it easy to put into REINTEGRATE.

```
GblsSolver* Gbls_Create (config);           // Create solver instance with configuration
int          Gbls_Setup  (slv, n, rowptr, colidx, vals); // Setup, or update existing setup
int          Gbls_Solve  (slv, b, x);         // Solve for given RHS
void         Gbls_Free   (slv);
```



Learning about EMT matrices: Hardware constrained performance and system limits

- This formulation of problem roughly scales linearly with the VRAM bandwidth.
- If producer and consumer of RHS is on CPU, the PCIe roundtrip **triples** the per RHS solve time.
- A **unified memory** computing unit with **HBM** can: **eliminate PCIe roundtrip**, and **linearly accelerate** per solve performance. E.g. AMD instinct, NVIDIA GB200



Next Steps

- Design a general structure communication interface.
- Investigate how the knowledge of structure and conditioning and push the precision down further.
- Formulate an optimization problem of hardware-constrained EMT matrix solve.
- Test the performance on various hardware settings.



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity



U.S. DEPARTMENT *of* ENERGY

Office of Electricity