

GRID EMT

SIMULATION INITIATIVE

SuperLU & Its Impact in EMT

Sherry Li, Tianyi Shi

Lawrence Berkeley National Laboratory



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

Linear solvers in EMT simulations

- Sparse direct solvers used most frequently.
- Some approaches partition the static and time-varying parts of the matrices and only update factorizations for the time-varying parts.
- Solvers perform reorderings to various forms, e.g. bordered block diagonal, to exploit the form for faster factorization and solves.
- Graph partitioning algorithms can be used to speed up factorizations and other operations on these matrices.



New opportunities for linear solvers

- Exploit matrix structures via batching on GPUs
- Exploit low precision hardware on GPU
- Explore preconditioned iterative solvers



Solution Workflow

- For stability and efficiency, need to solve transformed linear system:

$$Ax = b \rightarrow P_c(P_r(D_r A D_c)) P_c^T P_c D_c^{-1} x = P_c P_r D_r b$$
$$\tilde{A} \tilde{x} = \tilde{b}$$

- Preprocessing:
 - Reordering for nice block structure, e.g., BTF
 - Equilibration: diagonal scalings D_r and D_c
 - Numerical pivoting: row permutation matrix P_r
 - Sparsity preserving ordering: column permutation matrix P_c
- Numerical steps:
 - Sparse LU of
 - Sparse triangular solve
 - Iterative refinement

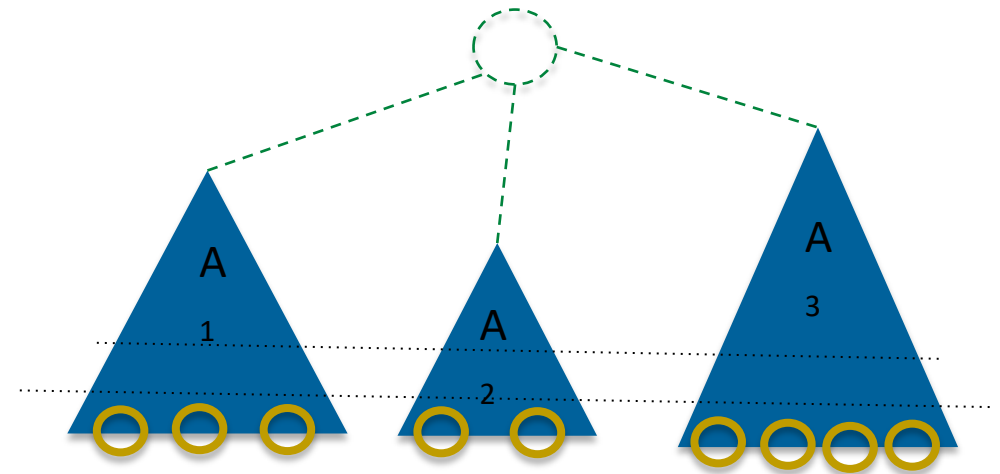


Batching to Improve GPU Performance

For a BD form $A = \begin{bmatrix} A_1 & & \\ & A_2 & \\ & & A_3 \end{bmatrix}$ Solve the diagonal blocks in parallel

- Restructure factorization in levels of the elimination trees
- Each level has many panels, calling
 - Batched LU
 - Batched triangular solves
 - Batched matrix-matrix multiply
 - Batched Schur complement updates

Elimination forest-of-trees – based on $\text{struct}(A+A')$



Level-by-level batch



Low Precision LU, High Precision Iterative Refinement (IR)

Iterative refinement

		Dense	Sparse (typical 3D problems)
Solve $Ax = b$ using LU factorization in	single precision (expensive)	$O(N^3)$	$O(N^2)$
IR iteration: $r := b - Ax$	double precision (cheap)	$O(N^2)$	$O(N)$
solve: $U^{-1}L^{-1}d = r$	single precision (cheap)	$O(N^2)$	$O(N^{4/3})$
$x := x + d$	double precision (cheap)	$O(N)$	$O(N)$

Stopping criteria:

1. A set number of iterations reached
2. A set threshold of accuracy reached (such as machine precision for double)
3. The error is not reduced by at least half

Demmel, Hida, Kahan, Li, Mukherjee, Riedy, "Error Bounds from Extra Precise Iterative Refinement," TOMS, 2005

Demmel, Hida, Li, Riedy, "Extra-precise Iterative Refinement for Overdetermined Least Squares Problems," TOMS, 2008

Carson, Higham, "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions," SISC, 2018



Example: IEEE39_IBL, on single CPU or GPU

- $m = 3336$, $nnz = 10200$ (average ~ 3 nonzeros per row)
- NVIDIA A100; AMD MI100

	Factor second	Solve second
NVIDIA double	0.0216	0.001
NVIDIA single LU + double IR	0.019	0.002
AMD double	0.0304	0.001
AMD single LU + double IR	0.0268	0.002
KLU CPU	0.0016	0.0001

SuperLU_DIST is not fast for small matrices because of heavy machineries for

- Distributed data structures
- MPI parallel decomposition
- Multi-GPU design



Example: IEEE39_IEEE118_IBL, on single CPU or GPU

- $m = 6378$, $nnz = 19009$ (average ~ 3 nonzeros per row); 3 diagonal blocks
- Condition number: original $5.26e+48$; after equilibration $4.41e+28$; Can obtain good residual error after IR

	Factor second	Solve second
NVIDIA double	0.0228	0.001
NVIDIA single LU + double IR	0.019	0.002
CPU double	0.048	0.001
CPU single LU + double IR	0.043	0.002
KLU CPU	factor failed (status 1) = KLU_SINGULAR	



Mixed precision GMRES-IR

LU factorization in **single precision (expensive)**

Initialize x_0 (e.g., $x_0 = U^{-1}L^{-1}b$) in single precision

Iteration:

$$r_i = b - Ax_i$$

solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ using GMRES

$$\text{update } x_i = x_i + d_i$$

double precision (cheap)

single precision (cheap)

double precision (cheap)



U.S. DEPARTMENT
of **ENERGY**

Office of
Electricity

Example: IEEE39_IBL and IEEE39_IEEE118_IBL on GPU

- Backward error (residual):

$$\text{Berr} = \max_i \frac{|b - Ax|_i}{(|A||x| + |b|)_i}$$

IEEE39_IBL

	IR second	IR steps	Berr batch 1	Berr batch 2
NVIDIA double IR	0.001	2	4.83e-8	1.66e-10
NVIDIA double GMRES-IR	0.0492	4	1.51e-15	2.67e-16

IEEE39_IEEE118_IBL

	IR second	IR steps	Berr batch 1	Berr batch 2	Berr batch 3
NVIDIA double IR	0.002	2	4.83e-8	1.80e-10	1.66e-10
NVIDIA double GMRES-IR	0.2034	4	3.80e-12	5.17e-14	7.45e-15

Preconditioned GMRES

- LU factorizations in **double precision**
- Use L and U factors as preconditioners for GMRES to iteratively solve the linear system

	Total second	GMRES iters	Berr batch 1	Berr batch 2	Berr batch 3
IEEE39_IBL	0.032	51	7.54e-15	6.52e-16	
IEEE39_IEEE118_IBL	0.2562	269	1.90e-12	3.32e-14	2.41e-15



Ongoing and next steps

- Integrate into RE-INTEGRATE code with plug-n-play capability (will showcase in the demo session)
- Benchmark larger systems
- Fast sparse LU updating





U.S. DEPARTMENT *of* ENERGY

Office of Electricity