Progress Report for Year 2

College of William and Mary
Algorithms and software

Andreas Stathopoulos (PI)
Kostas Orginos (Co-PI)
Lingfei Wu (CS Ph.D. student)
Jesse Laeuchli (CS Ph.D. student)
Arjun Gambhir (Physics Ph.D. student)

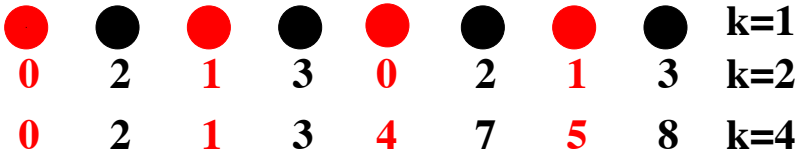
The SOW points

Summary table of tasks to be performed at the College of William and Mary
Konstantinos Orginos, Andreas Stathopoulos (PI)

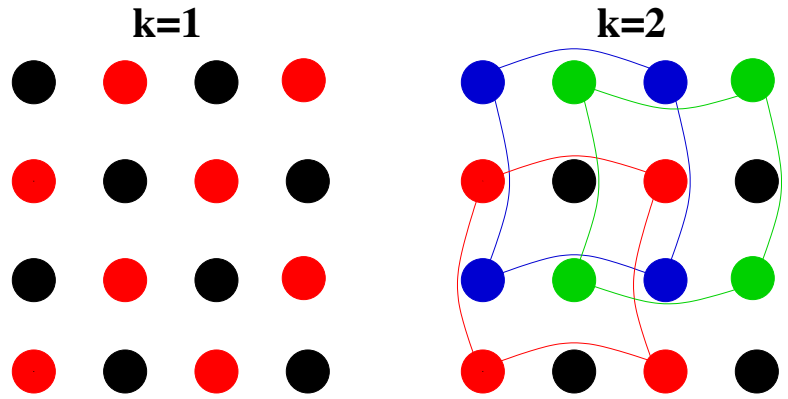
Year	Task	subtopic
1	Update the PRIMME eigenvalue package and provide a Level 3 interface callable from high level LQCD software packages	algorithms
2	Implement deflated Monte Carlo trace estimators	algorithms
3	Explore variance reducing vector samples	algorithms
4	Provide methods and implementations that combine deflation and variance reducing vectors	algorithms
3-5	Integrate the variance reduction mechanisms with effective preconditioners developed by the LQCD community	algorithms

Main theme Variance Reduction (VR) for $\mathbf{Tr}(A^{-1})$

- Extended hierarchical probing (SIAM SISC) to arbitrary lattice sizes
- Use of singular values for VR through eigCG
- Accurate solution of singular values with PRIMME
- Continued VR based on extrapolation techniques



1D:
Doubling k splits grid to 2 1D subgrids
Color R-B each with two new colors

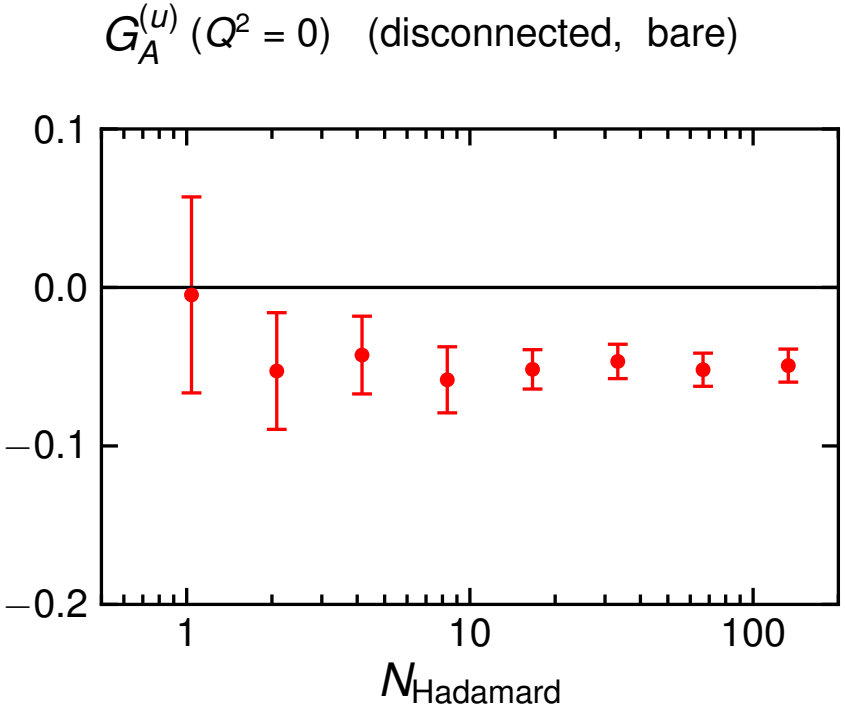


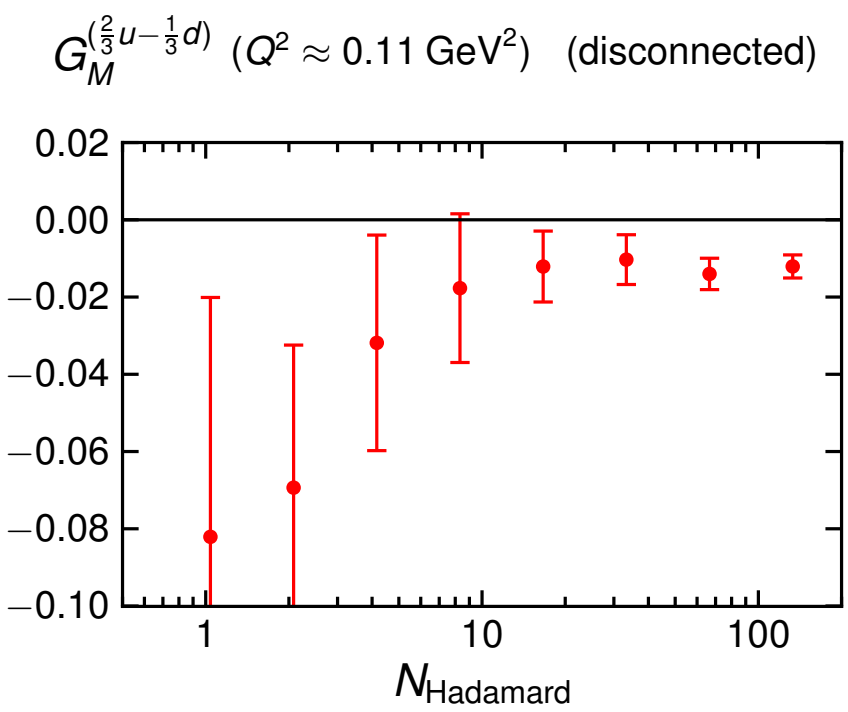
2D:
Doubling k splits grid to 4 2D subgrids
(e.g., Reds split to 4 reds and 4 greens)
Color R-B each with two new colors

Algorithm in d-dimensions:
1. Recursively split a lattice $\prod_{i=1}^{d} 2^{c_i}$ to 2^d sublattices of size $\prod_{i=1}^{d} 2^{c_i-1}$
2. When no further splits possible, Red-Black each sublattice with unique colors
3. Choose unique colors appropriately to guarantee nesting

This only worked for powers of two sized lattices

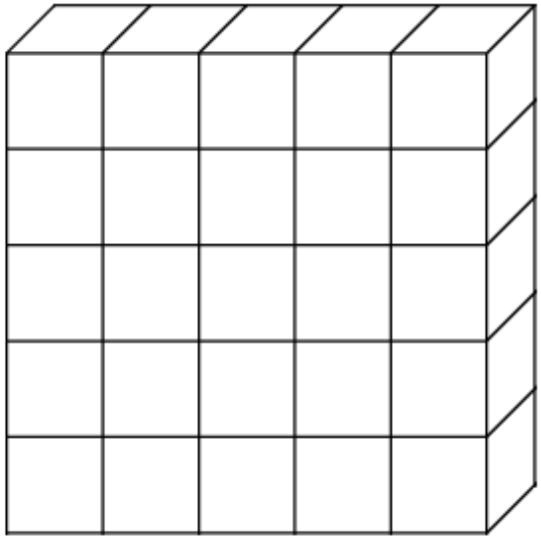
Hierarchical Probing results from Stefan Meinel





Error std ≈ 4, thus speedups of about 16

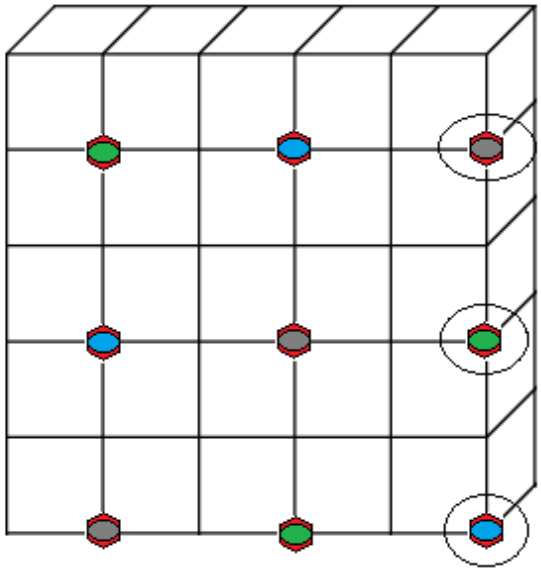
Code available in Chroma



$$l_1 = 6 = 2 \cdot 3$$
$$l_2 = 6 = 2 \cdot 3$$
$$l_3 = 2$$

Consider a d-dimensional lattice with dimension lengths of l_1,l_2,...l_d.
1. Find the common factors (eg. 2, 3 in Figure)
2. Split lattice into sub-lattices. Use common factors to determine which sub-lattice a point lies in
3. Assign each sub-lattice a color
4. Apply recursively until no more common factors
5. Color final sub-lattice with three colors

How to determine which sub-lattice a node lies in



$(5,4,0)$
in mixed radix representation
$< 1,2 >_{2,3} < 0,2 >_{2,3} < 0 >_2$

$(5,2,0)$
in mixed radix representation
$< 1,2 >_{2,3} < 0,1 >_{2,3} < 0 >_2$

$(5,0,0)$
in mixed radix representation
$< 1,2 >_{2,3} < 0,0 >_{2,3} < 0 >_2$

function subLattices = SplitLattice(latticeNodes, SplitLevel)
for $n = 1 : numNodes$
  for $j = 1 : dims$
    $mixedrep(j) =$ dec2mixed(GetCoordOfLatticeNode(j,n))
  $subLattices(mixedrep(1)(1 : SplitLevel), .., mixedrep(d)(1 : SplitLevel)) = latticeNode(n)$
  % Nodes that have the same 1:SplitLevel digits are in the same sublattice
for $j = 1 : numSublattices$
  AssignColorToLattice$(subLattices(j))$
  SplitLattice$(subLattices(j), SplitLevel + 1)$

Creating Probing Vectors

Let c_i be the number of colors needed to color the sublattices of a lattice after i recursive calls

Then we can create the probing matrix as below

$$\begin{aligned}\tilde{Z}^{(1)} &= F_{c(1)},\\ \tilde{Z}^{(i)} &= \left[\tilde{Z}^{(i-1)} \otimes F_{c(i)}(:,1), \ldots, \tilde{Z}^{(i-1)} \otimes F_{c(i)}(:,c(i))\right]\end{aligned}$$

To generate a probing vector (column of $Z^{(i)}$) only two vectors are needed:

one from $\tilde{Z}^{(i-1)}$, $i > 1$

and one from a $F_{c(j)}$.

Vectors of $\tilde{Z}^{(i-1)}$ are obtained recursively

Creating Probing Vectors

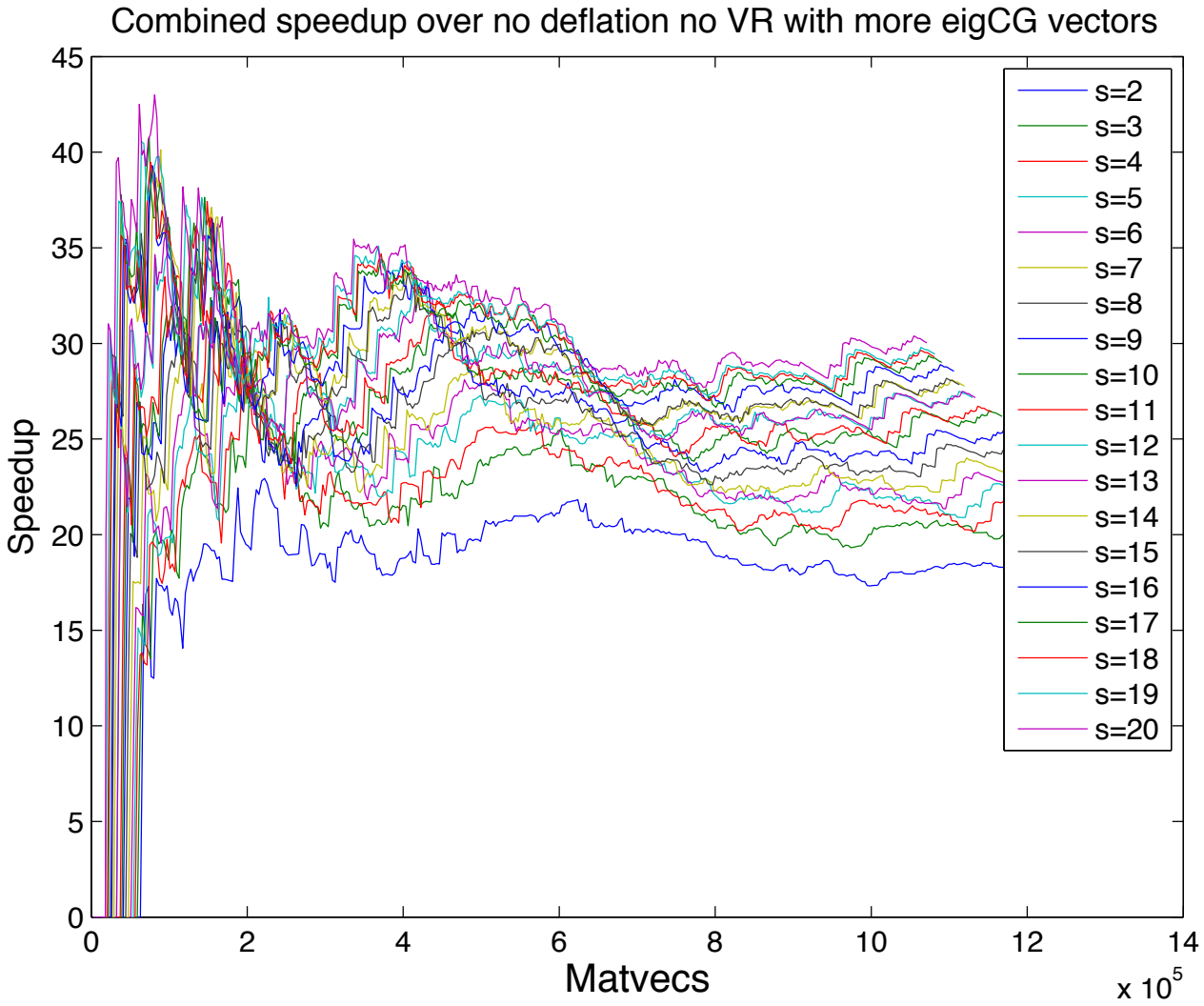
Take the list $c(i)$ containing the number of colors needed per lattice after i splits

Interpert $c(i)$ as a mixed-radix base

For the k-th probing vector, the required vectors of $F_{c_1}, F_{c_2}, \ldots$, are the n digits for the mixed radix representation of k using this mixed-radix base

$$\mathbf{F}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \mathbf{F}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & (-1)^{4/3} & (-1)^{2/3} \\ 1 & (-1)^{2/3} & (-1)^{4/3} \end{pmatrix} \quad (1)$$

If $\boldsymbol{c} = \{2,3\}$, then for k=2 the mixed radix representation of k is $< 0,1 >_{2,3}$, and $F_2(:,1) \otimes F_3(:,2)$ produces the required vector



Good speedups with only 7-8 rhs of eigCG

Find k smallest singular triplets of a large, sparse matrix $A \in \Re^{m \times n}$

$$Av_i = \sigma_i u_i, \quad \sigma_1 \leq \ldots \leq \sigma_k$$

- A Hermitian eigenvalue problem on
  - Normal equations matrix $C = A^T A$ or $C = AA^T$
  - Augmented matrix $B = \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$
- Lanczos bidiagonalization method (LBD)

$$A = PB_d Q^T$$

$$B_d = X \Sigma Y^T$$

Where $U = PX$ and $V = QY$

Motivation I: difference between methods

- Eigen methods on C
  - fast for largest SVs
  - slow for smallest SVs
  - can only achieve accuracy of $O(\kappa(A)\|A\|\varepsilon_{mach})$
- Eigen methods on B
  - slower for largest SVs
  - extremely slow for smallest SVs
  - can achieve accuracy of $O(\|A\|\varepsilon_{mach})$
- LBD on A
  - fast for largest SVs
  - similar to C but exhibits irregular convergence
  - can achieve accuracy of $O(\|A\|\varepsilon_{mach})$

Motivation II: our goal for an SVD solver

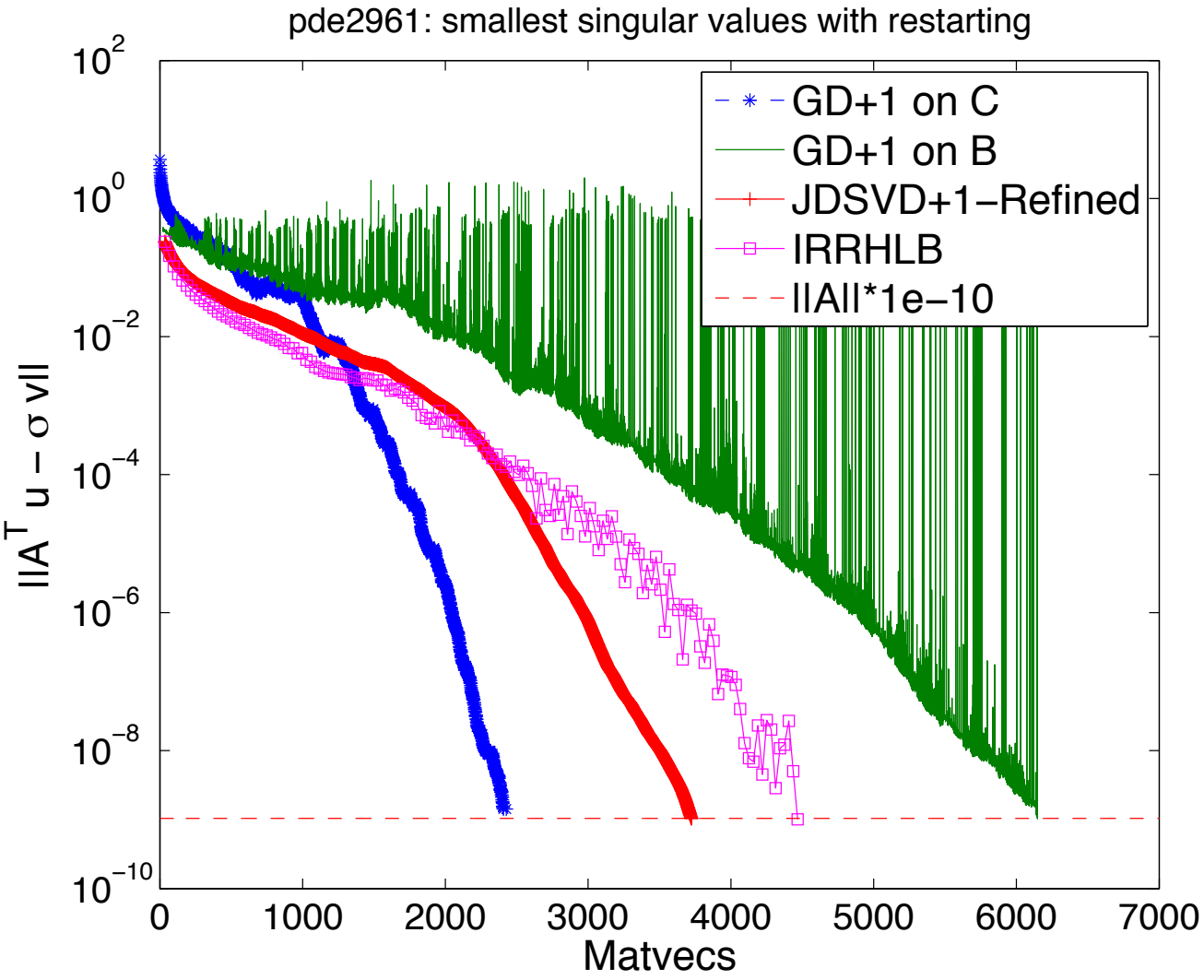
Extremely challenging task for small SVs:

- large sparse matrix $\Rightarrow$ No shift-and-invert
- very slow convergence $\Rightarrow$ restarting and preconditioning
- very few SVD solvers:
  - SVDPACK: Lanczos/trace-min methods on B or C for only largest SVs
  - PROPACK: LBD for largest SVs. Shift-invert for smallest SVs

$\Rightarrow$ calls for full functionality, highly-optimized SVD solver

PRIMME: PReconditioned Iterative MultiMethod Eigensolver

Motivation III: the impact of restarting



primme_svds: a two stage strategy

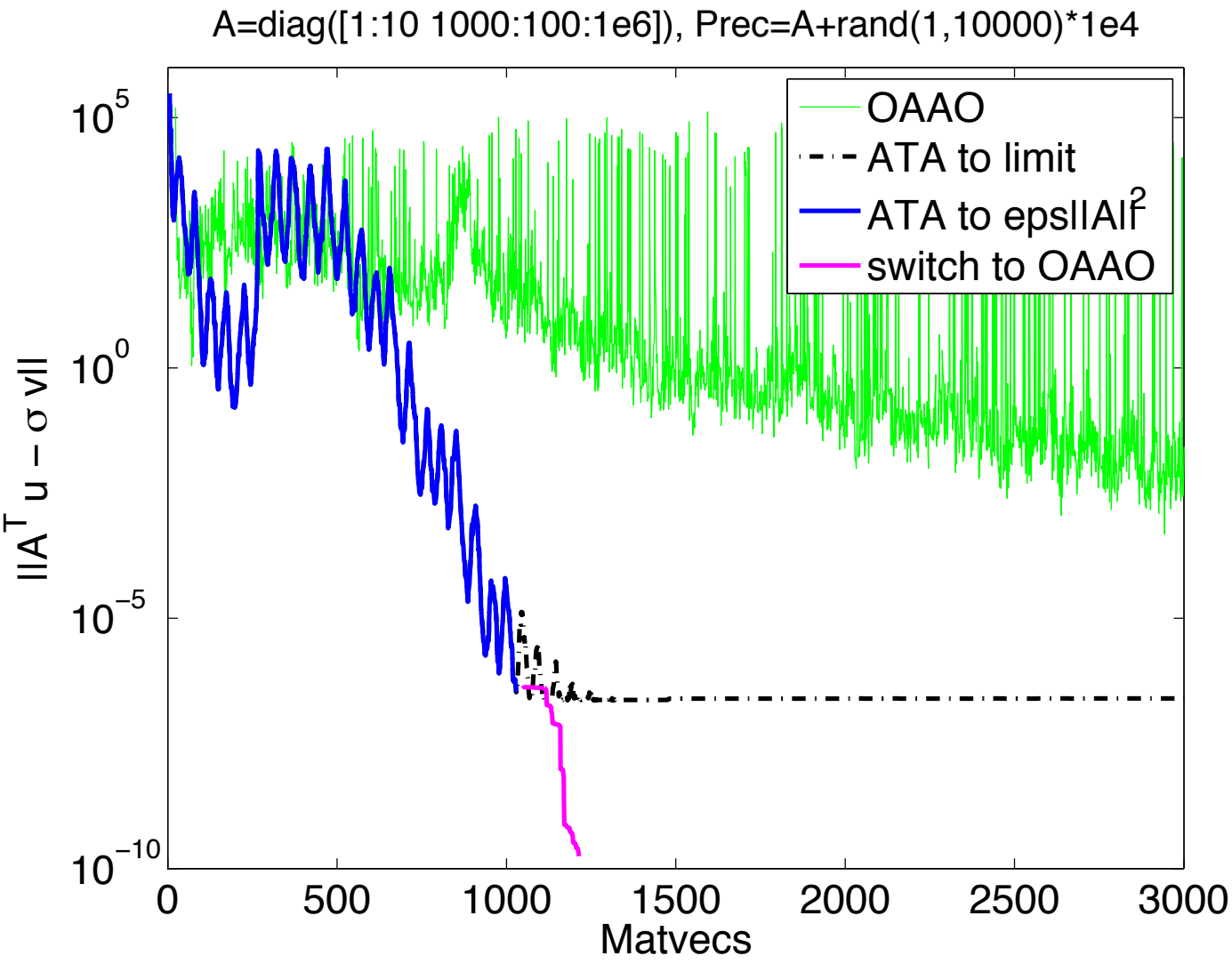
Our solution: a hybrid, two-stage SVD method

- Stage I: works on C to max residual tolerance max $\left(\sigma_i \delta_{user} \|A\|, \|A\|^2 \varepsilon_{mach}\right)$
  - Must dynamically adjust tolerance in PRIMME to meet user tolerance
- Stage II: works on B to improve the approximations from C until user required tolerance $\delta_{user}\|A\|$ is satisfied
  - Inputs from *C*: accurate shifts and good initial guesses
    $\Rightarrow$ Calls for near-optimal method JDQMR in PRIMME
  - Irregular convergence of Rayleigh Ritz (RR) on *B*
    $\Rightarrow$ Enhance PRIMME with refined projection method

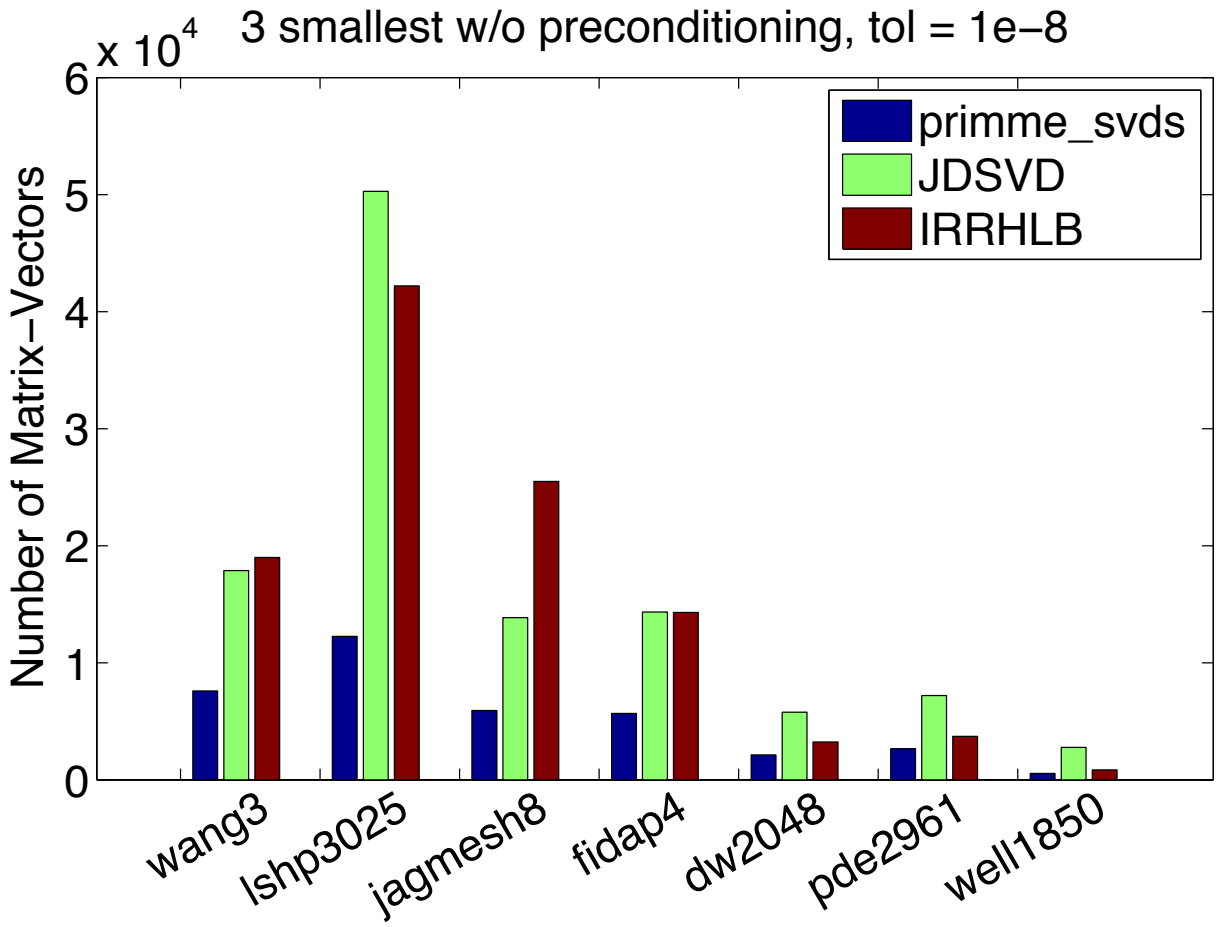
Refined projection minimizes the residual $\|BVy - \tilde{\sigma}Vy\|$ where V search space for a given user shift $\tilde{\sigma}$

- QR factorization on $BV - \tilde{\sigma}V$ only after restart
- one column updating for Q and R during iteration
- computational cost similar with the RR method

primme_svds: an example

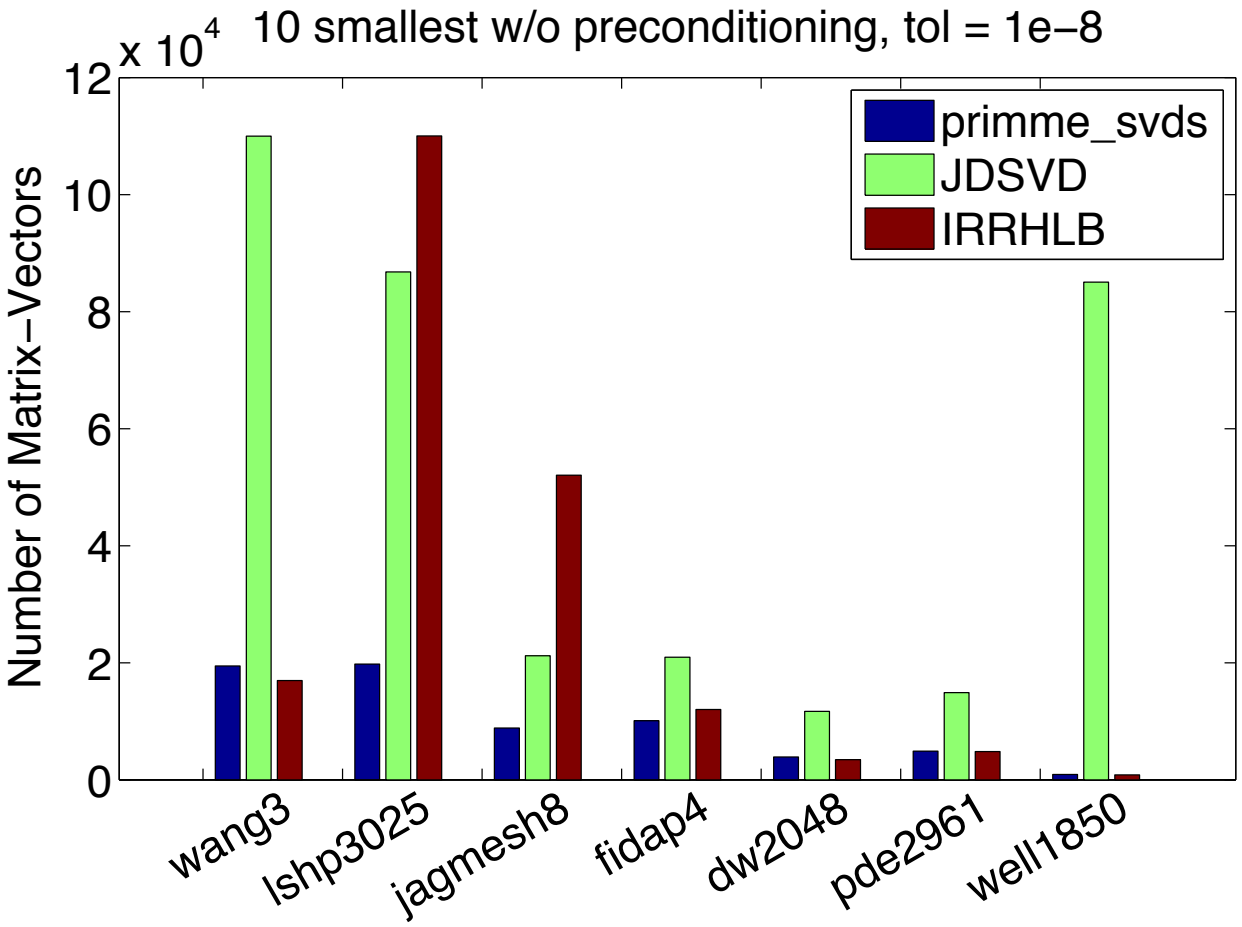


Evaluation: Without preconditioning



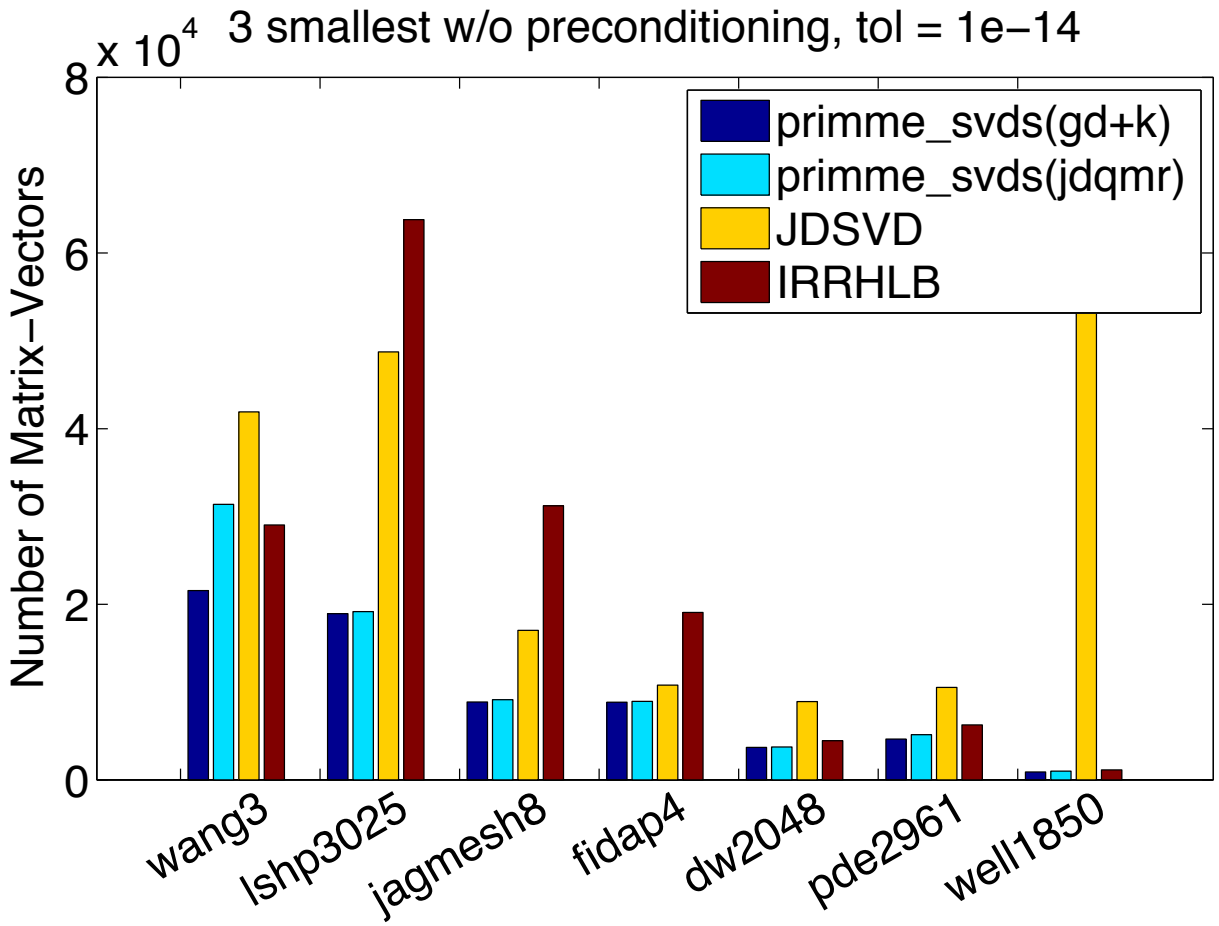
primme_svds (only first stage) is the fastest method

Evaluation: Without preconditioning



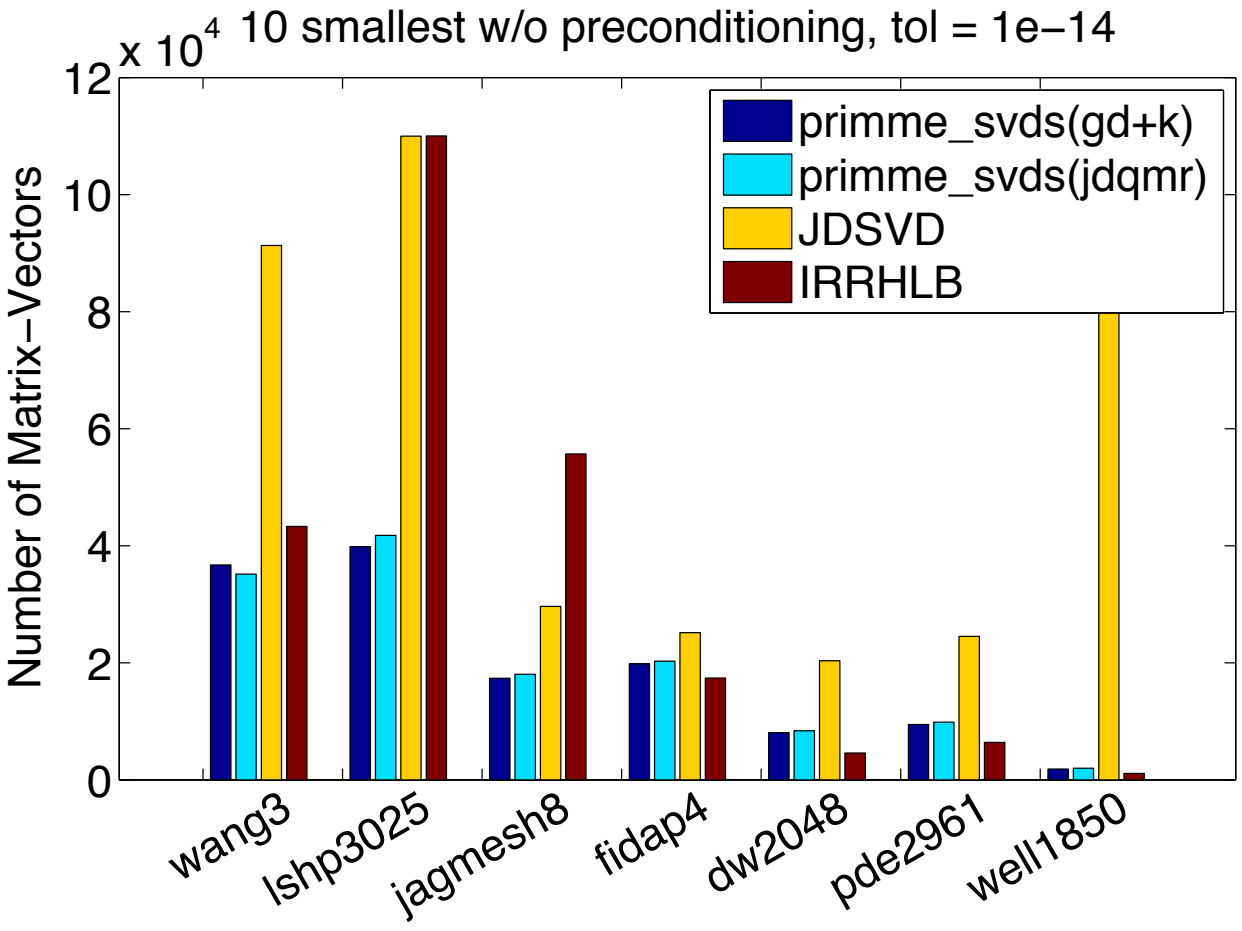
primme_svds (only first stage) is much faster in hard cases

Evaluation: Without preconditioning



primme_svds (two stage) is superior for a few singular triplets

Evaluation: Without preconditioning



primme_svds (two stage) is much faster in hard cases