Accurate computation of smallest singular values using the PRIMME eigensolver

Lingfei Wu
Andreas Stathopoulos

Computer Science Department
College of William and Mary

Acknowledgment: National Science Foundation, DOE Scidac

The ubiquitous SVD

Mostly for largest singular values

- Graph/Data/Text mining
- Image-processing
- Model reduction in PDEs
- Variance reduction in Monte Carlo

The ubiquitous SVD

Mostly for largest singular values

- Graph/Data/Text mining
- Image-processing
- Model reduction in PDEs
- Variance reduction in Monte Carlo

But also for smallest singular values

- similar applications but for functions of matrices (e.g., A^{-1}).
  low rank preconditioning
  variance reduction
- computation of pseudospectrum
- sensitivity analysis

Solving the large, sparse SVD problem

Find k smallest singular values and corresponding left and right singular vectors of large, sparse $A \in \Re^{m \times n}$

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

Solution approaches:

- 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)

  $$A = PB_dQ^T, \text{ where } B_d = X\Sigma Y^T, \; U = PX \text{ and } V = QY$$

- Jacobi-Davidson for SVD (JDSVD)

  uses B but projection based on two subspaces

Theorem. A Krylov method computing σ_n^2 on C is twice as fast asymptotically than for computing σ_n on B

Theorem. A Krylov method computing σ_1^2 on C is always faster asymptotically than for computing σ_1 on B.

Lanczos on C builds:

$$V_k = K_k(A^T A, v_1)$$

LBD builds:

$$V_k = K_k(A^T A, v_1), \quad U_k = K_k(AA^T, Av_1)$$

JDSVD builds:

$$U_k = K_{k/2}(AA^T, u_1) \oplus K_{k/2}(AA^T, Av_1)$$
$$V_k = K_{k/2}(A^T A, v_1) \oplus K_{k/2}(A^T A, A^T u_1)$$

Lanczos on B builds $K_k(B, [v_1; u_1])$. If $u_1 = 0$,

$$\begin{pmatrix} U_k \\ V_k \end{pmatrix} = \begin{pmatrix} 0 \\ K_{k/2}(A^T A, v_1) \end{pmatrix} \oplus \begin{pmatrix} K_{k/2}(AA^T, Av_1) \\ 0 \end{pmatrix}.$$

LBD and Lanczos on C twice the degree of augmented approaches

Squaring $C = A^T A$ (even implicitly) means problems

LBD on A	achieves $\|r\|$ of $O(\|A\|\varepsilon_{mach})$
Eigenmethod on B	achieves $\|r\|$ of $O(\|A\|\varepsilon_{mach})$
Eigenmethod on C	achieves $\|r\|$ only up to $O(\kappa(A)\|A\|\varepsilon_{mach})$

Only LBD seems to be both efficient and accurate...

LBD

Advantages of LBD:

- Same space as Lanczos on C but accurate since working on A
- Lanczos global convergence to many singular triplets

Drawbacks of LBD:

- Orthogonality loss, large memory demands $\Rightarrow$ restart
- Interior spectral info $\Rightarrow$ Harmonic/refined procedures
  $\Rightarrow$ irregular and slow convergence
- No general purpose software for recent advances in LBD
- Cannot use preconditioning

JDSVD

Advantages of JDSVD:

- Can take advantage of preconditioning
- More flexible that eigenmethods on B
- Accurate

Drawbacks of JDSVD:

- Correction equation on B maximally indefinite
- Outer iteration slow
- Only MATLAB implementation

What is missing in real world software

Extremely challenging task for small SVs:

- large sparse matrix $\Rightarrow$ no Shift-Invert
- slow convergence $\Rightarrow$ effective restarting and preconditioning
- Currently only unpreconditioned SVD software:
  - SVDPACK: Lanczos and tracemin on B or C only largest SVs
  - PROPACK: LBD for largest SVs, Shift-Invert for smallest SVs
  - SLEPc: no preconditioning, still in development

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

PRIMME: PReconditioned Iterative MultiMethod Eigensolver

Building an SVD solver on PRIMME

PRIMME: PReconditioned Iterative MultiMethod Eigensolver

- Over 12 eigenmethods including near optimal GD+k and JDQMR methods
- Extensive support for interior eigenvalues

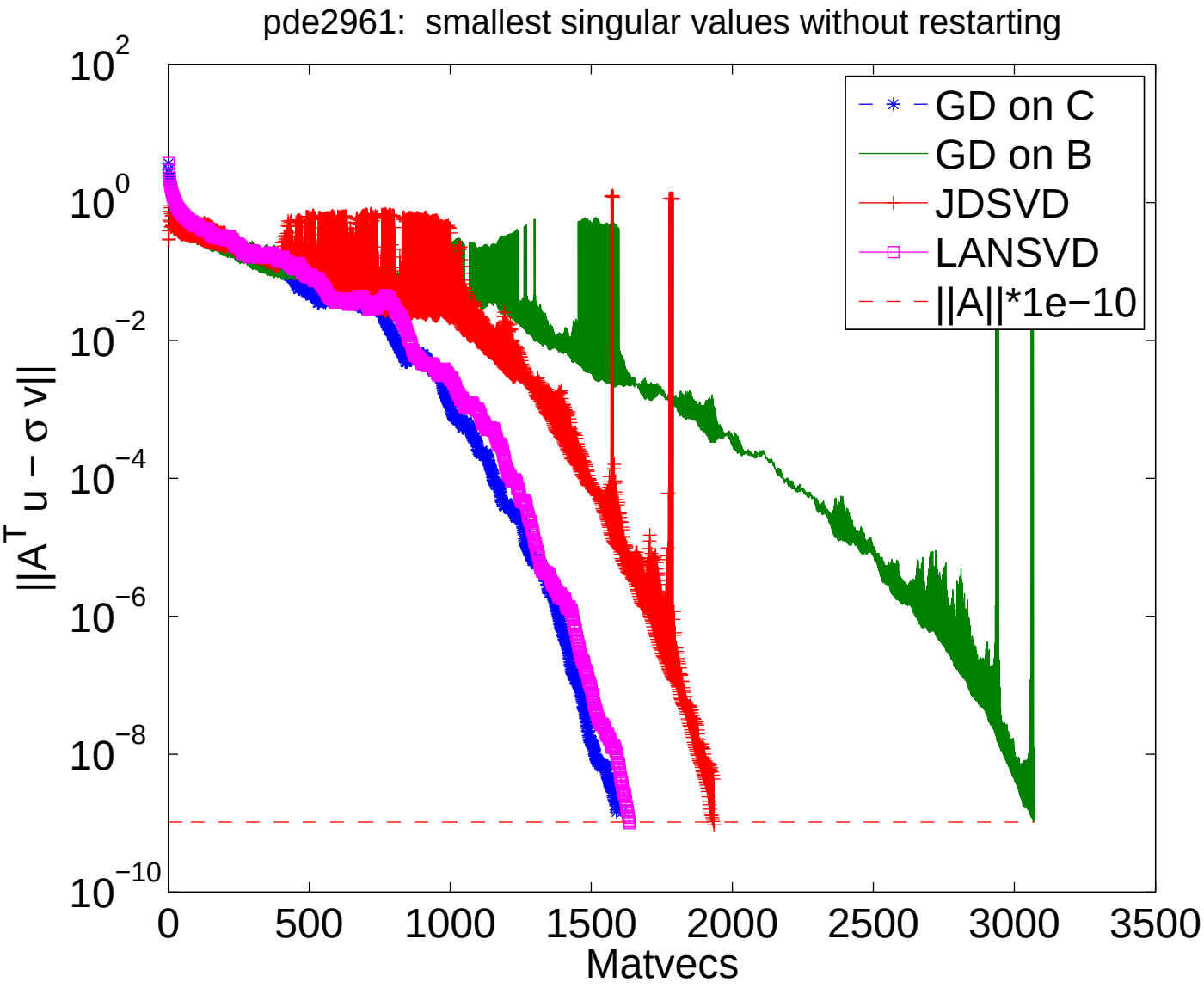
  Accepts initial guesses for all required eigenvectors

  Accepts many shifts and finds the closest eigenvalue to each shift
- Accepts any preconditioner for $M \approx C^{-1}$ or $M \approx B^{-1}$, or

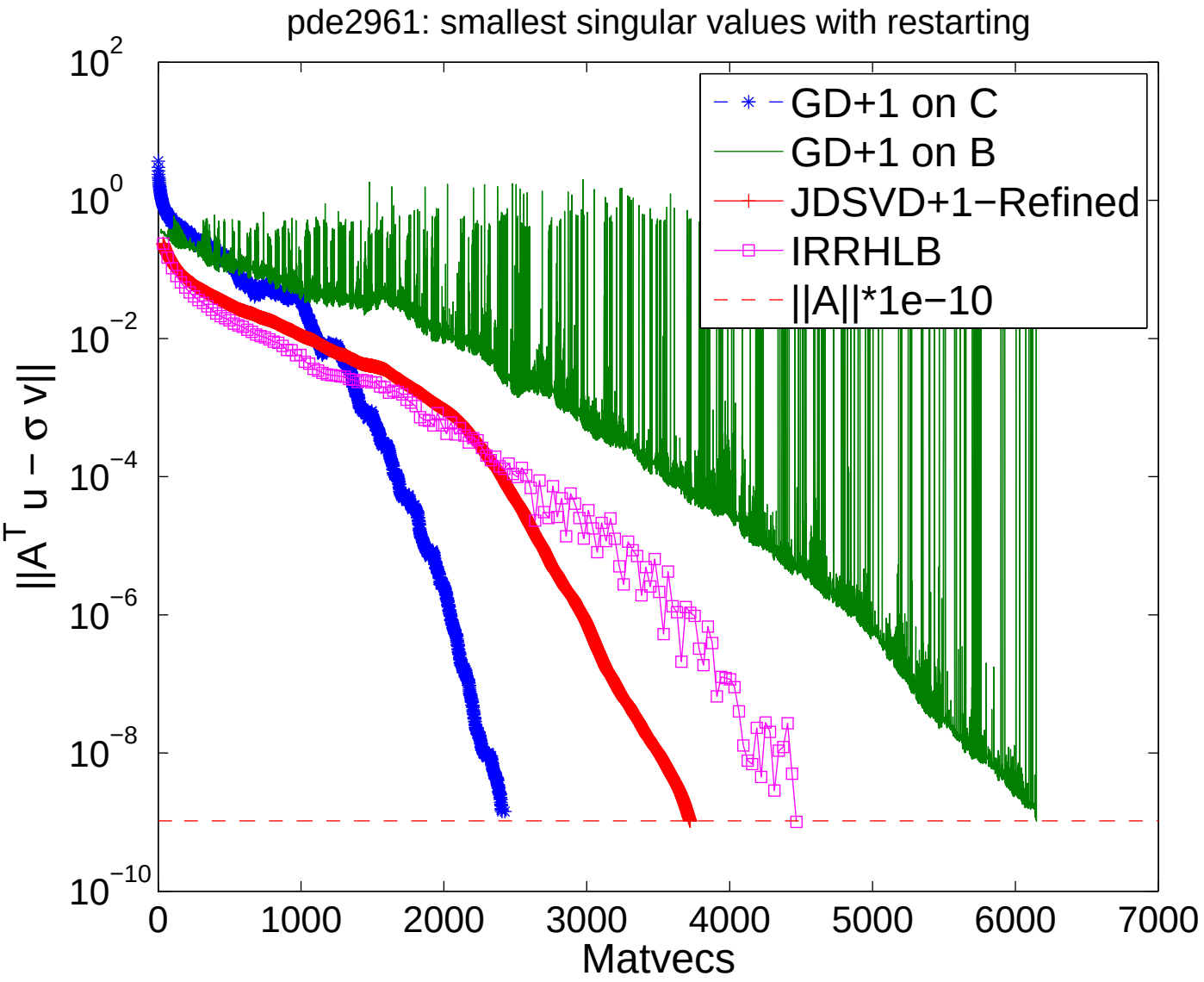
  if $M \approx A^{-1}$, uses $MM^T \approx C^{-1}$ and $\begin{bmatrix} 0 & M \\ M^T & 0 \end{bmatrix} \approx B^{-1}$

  if $M \approx (A^TA)^{-1}$ (e.g., RIF), uses $MM^T \approx C^{-1}$ and $\begin{bmatrix} 0 & AM \\ MA^T & 0 \end{bmatrix} \approx B^{-1}$
- Robust framework: subspace acceleration, locking, block methods
- Parallel, high performance implementation

Methods to use. Why not LBD?



Restarting changes the game



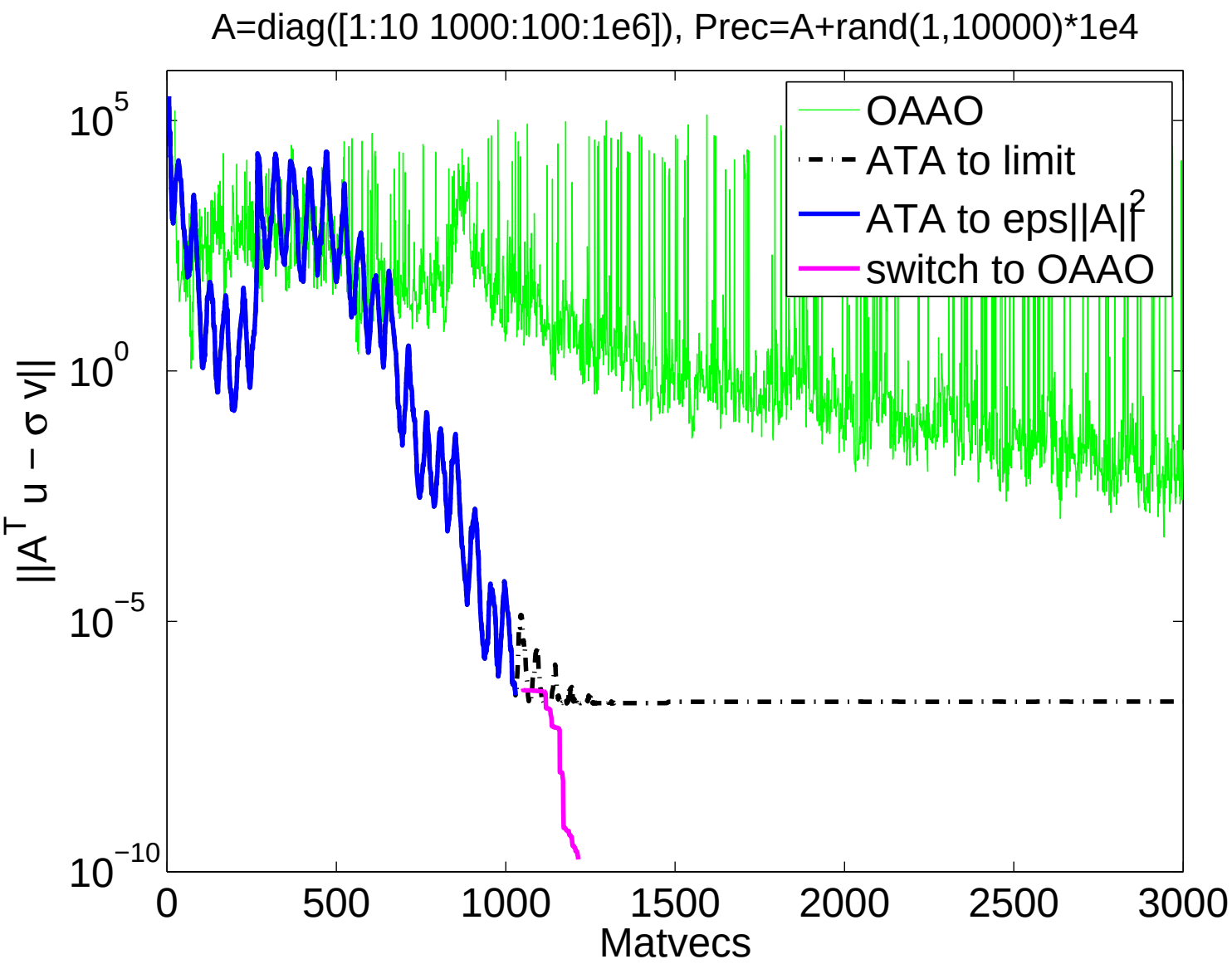
primme_svds: A two-stage strategy

- GD+k on C extremal evs and near optimal restart best for a few smallest SVs
- or JDQMR on C if A is very sparse
- Limited by accuracy

Our solution: a hybrid, two-stage singular value method

- Stage I: work on C to residual tolerance $\max\left(\sigma_i \delta_{user} \|A\|, \|A\|^2 \varepsilon_{mach}\right)$
- Stage II: work on B to improve the approximations from C to $\delta_{user}\|A\|$

primme_svds: an example



Stage I of primme_svds (C)

- Flexible stopping criterion

Let $(\tilde{\sigma}, \tilde{u}, \tilde{v})$ be a targeted singular triplet of A

$$r_v = A\tilde{v} - \tilde{\sigma}\tilde{u},\ \ r_u = A^T\tilde{u} - \tilde{\sigma}\tilde{v},\ \ r_C = C\tilde{v} - \tilde{\sigma}^2\tilde{v},\ \ r_B = B\begin{bmatrix}\tilde{v}\\ \tilde{u}\end{bmatrix} - \tilde{\sigma}\begin{bmatrix}\tilde{v}\\ \tilde{u}\end{bmatrix}.$$

If $\|v_i\| = 1$, $\|u_i\| = \|Av_i/\sigma_i\| = 1$, then $r_v = 0$ and

$$\|r_u\| = \frac{\|r_C\|}{\tilde{\sigma}} = \|r_B\|\sqrt{2}$$

Thus, the stopping criterion for the methods on C becomes,

$$\delta_C = \max\left(\delta_{user}\ \tilde{\sigma}/\|A\|, \varepsilon_{mach}\right)$$

- Explicit Rayleigh-Ritz of converged eigenpairs

  Improves directions of individual eigenvectors

Stage II of primme_svds (B)

Excellent initial guesses (shifts, eigenvectors) from C

$\Rightarrow$ Use JDQMR

Subspace acceleration and optimal stopping in JDQMR

$\Rightarrow$ Better than iterative refinment

Irregular convergence of Rayleigh Ritz (RR) on B

$\Rightarrow$ Enhance PRIMME with Refined Projection ($\min \|BVy - \tilde{\sigma}Vy\|$)
QR only column update per step
QR factorization only at restart

Implementation of primme_svds

- Developed PRIMME MEX, a MATLAB interface for PRIMME
- UI extends MATLAB's eigs() and svds() to PRIMME's full functionality
- External stopping criterion, refined projection implemented in PRIMME

Evaluation: Test matrices

Square

Matrix	dw2048	fidap4	jagmesh8	lshp3025
order	2048	1601	1141	3025
$\kappa(A)$	5.3e3	5.2e3	5.9e4	2.2e5
$\|A\|_2$	1.0e0	1.6e0	6.8e0	7.0e0

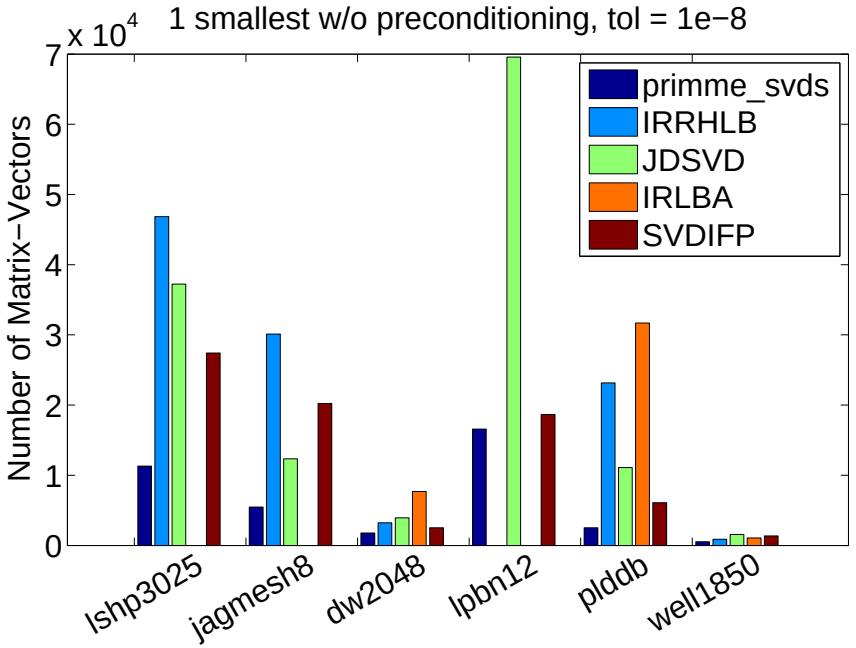
Rectangular

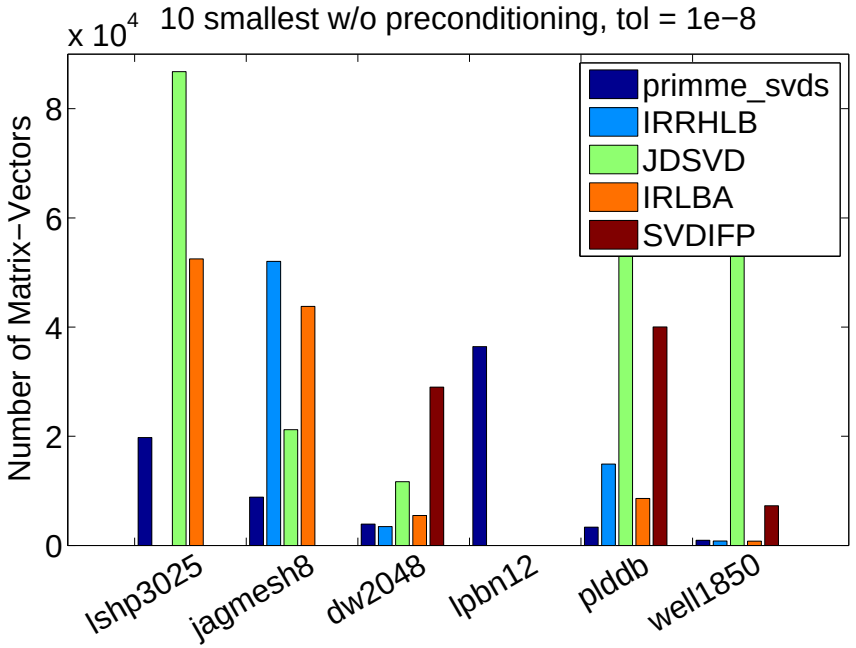
Matrix	well1850	plddb	lp_bnl2
rows m:	1850	3049	2324
cols n:	712	5069	4486
$\kappa(A)$	1.1e2	1.2e4	7.8e3
$\|A\|_2$	1.8e0	1.4e2	2.1e2

We compare against:

- JDSVD (Hochstenbach, 2001)
- IRRHLB (Jia, 2010)
- SVDIFP (Ye, 2014)
- IRLBA (Baglama, 2005)
- MATLAB svds() (ARPACK, 1998)

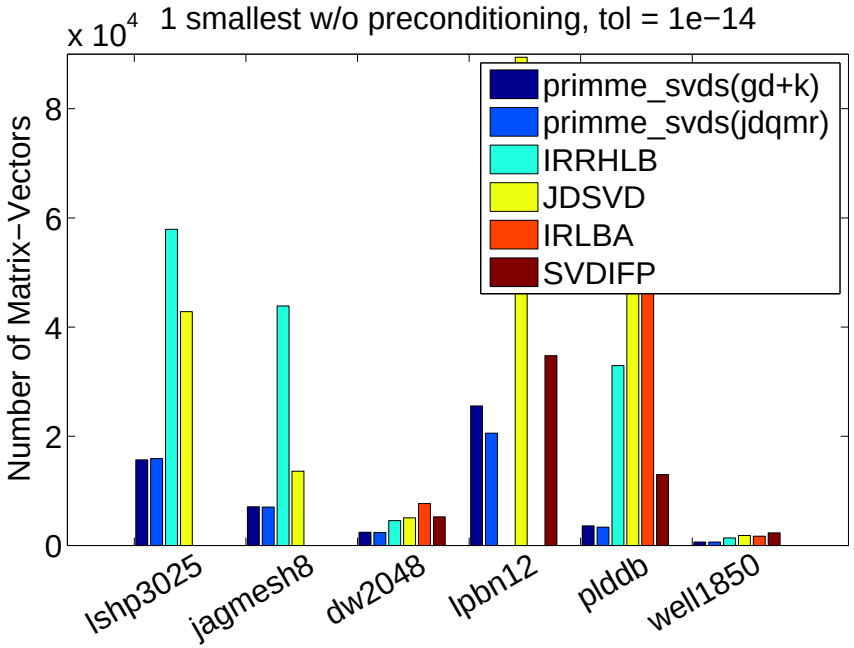
No preconditioning — first stage only — 1 or 10 smallest triplets

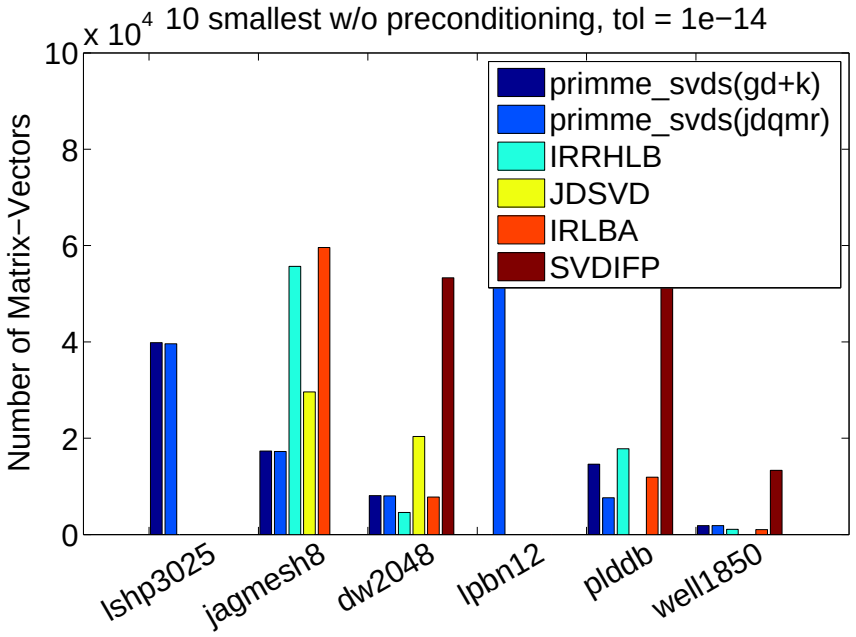




primme_svds: fastest and most robust

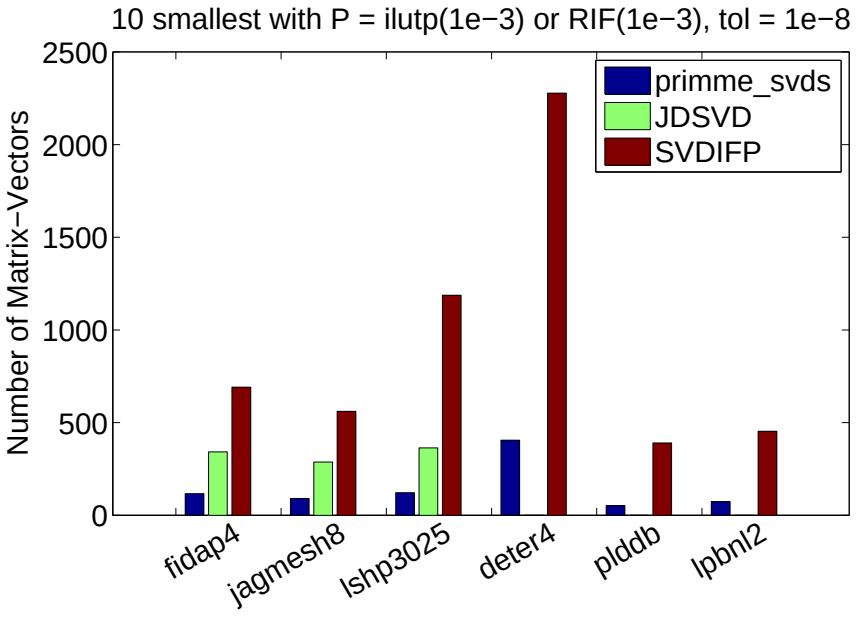
No preconditioning — 1e-14 accuracy — 1 or 10 smallest triplets

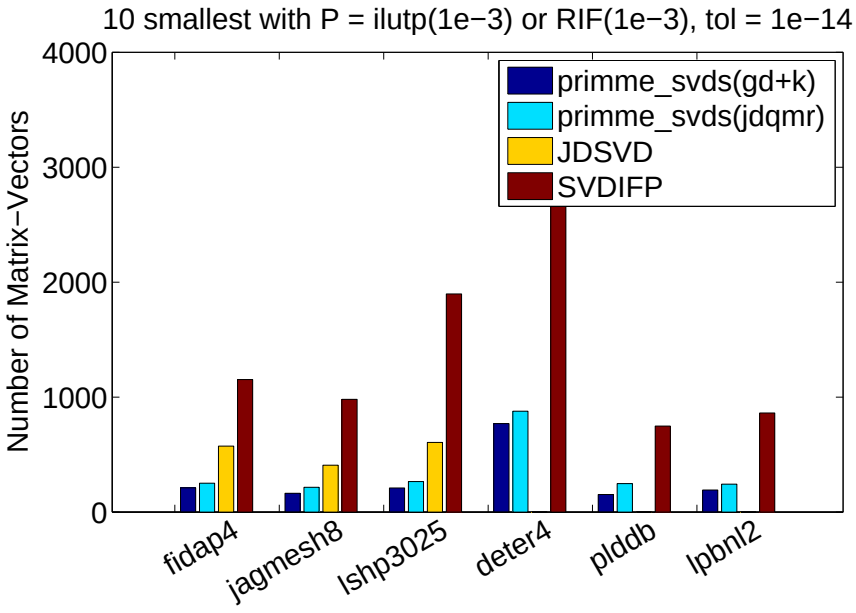




primme_svds: fastest and most robust at high accuracy

With preconditioning — 1e-8/1e-14 — 1 smallest triplets

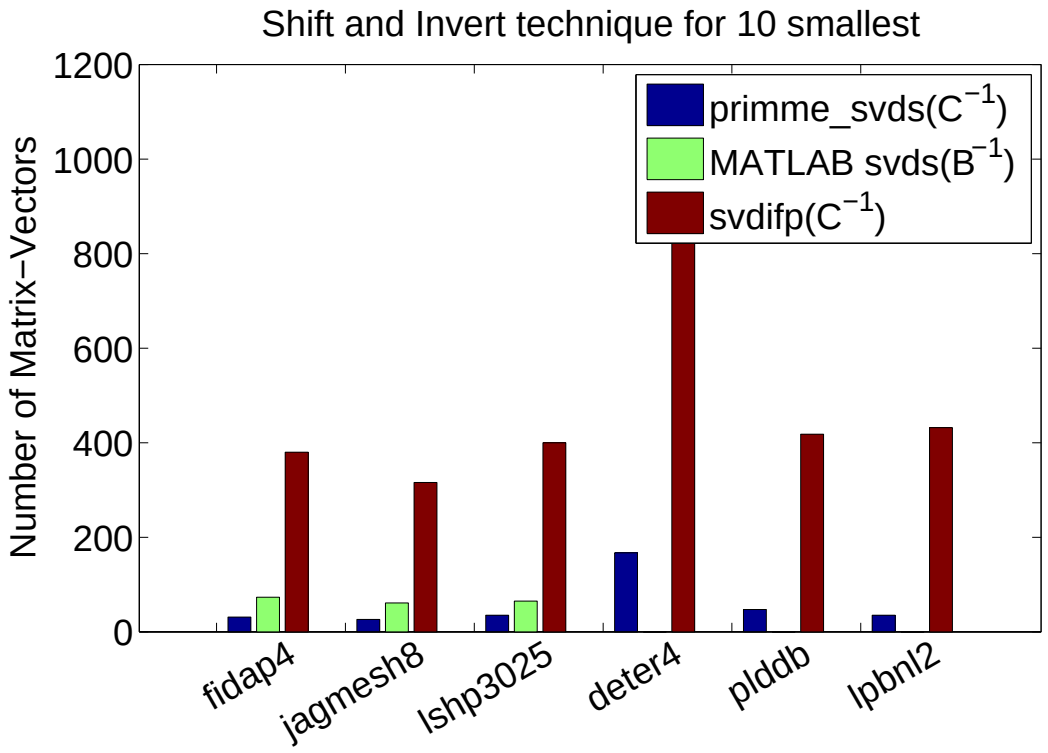




primme_svds: significantly more robust and efficient

Note JDSVD's problem with rectangular matrices

With shift and invert operator



primme_svds on C is faster than MATLAB svds

Smallest triplet of very large matrices in low tolerance

Matrix	debr	cage14	thermal2	sls	Rucci1
rows m:	1048576	1505785	1228045	1748122	1977885
cols n:	1048576	1505785	1228045	62729	109900
σ_1	1.11E-20	9.52E-02	1.61E-06	9.99E-1	1.04E-03
$\kappa(A)$	3.6E+20	1.01E+1	7.48E+6	1.30E+3	6.74E+3
Preconditioner	No	ILU(0)	ILU(1e-3)	RIF(1e-3)	RIF(1e-3)

δ = 1e-12		primme_svds		SVDIFP		JDSVD	
Matrix	Precond.time	MV	Sec	MV	Sec	MV	Sec
debr	—	539	84.3	*	*	1971	474.6
cage14	2e0	19	11	33	28	111	185
thermal2	3.69e+3	419	506	–	–	309	535
sls	3.29e+3	1779	170	*	*	–	–
Rucci1	6.92e+4	4728	1087	4649	6464	–	–

primme_svds far more robust and more efficient

Conclusions

- primme_svds: computes a few singular triplets based on PRIMME
- Key idea: a two-stage strategy
  - take advantage of faster convergence on normal equations matrix
  - resolve remaining accuracy by exploiting power of PRIMME on augmented matrix
  - Many existing or new methods could be applied at each stage
- Robust and efficient both with and without preconditioning
- A highly optimized production software!

PRIMME

PRIMME: PReconditioned Iterative MultiMethod Eigensolver

- primme_svds and PRIMME's MATLAB interface will be available soon
- C implementation of primme_svds will be released with next version of PRIMME

Download: `www.cs.wm.edu/~andreas/software`