Exploring the Potential of the PRIMME Eigensolver

Part III: SVD Problems

Andreas Stathopoulos[1] Eloy Romero Alcalde[1] Lingfei Wu[2]

[1]Computer Science Department, College of William & Mary, USA

[2]IBM Thomas J. Watson Research Center, Yorktown Heights, USA

CSE'17

The problem and how to compute?

Find **k** smallest singular values and associated left and right singular vectors of a large, sparse matrix $A \in \mathbb{R}^{m \times n}$:

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

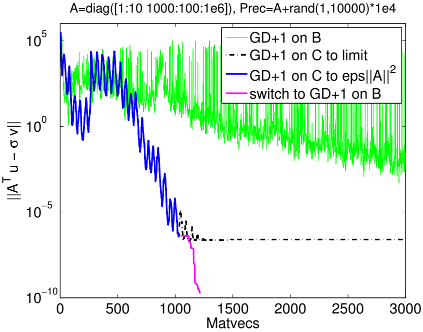
Methods for computing the SVD:

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

$$A = PB_d Q^T \text{ and } B_d = X \Sigma Y^T$$

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

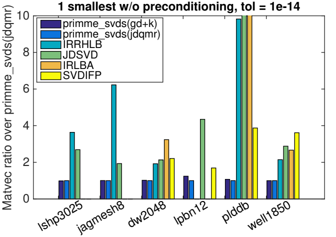
PHSVDS: a Preconditioned Hybrid SVDS method

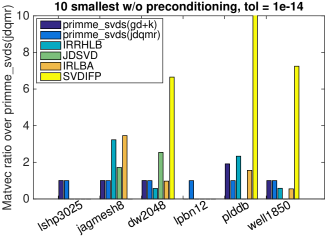


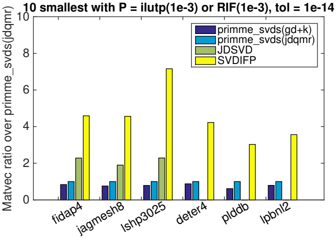
PHSVDS is a preconditioned hybrid two-stage meta-method:

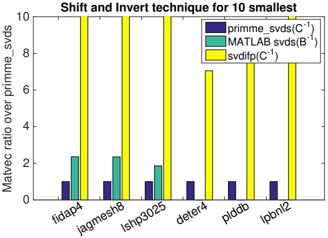
- Stage I: works on C to desired or minimum residual tolerance $\max\left(\sigma_i \delta_{user}\|A\|, \|A\|^2 \epsilon_{mach}\right)$
- Stage II: works on B to improve the approximations from C until user required tolerance $\delta_{user}\|A\|$ is satisfied

Compare PHSVDS against state-of-the-art SVD methods







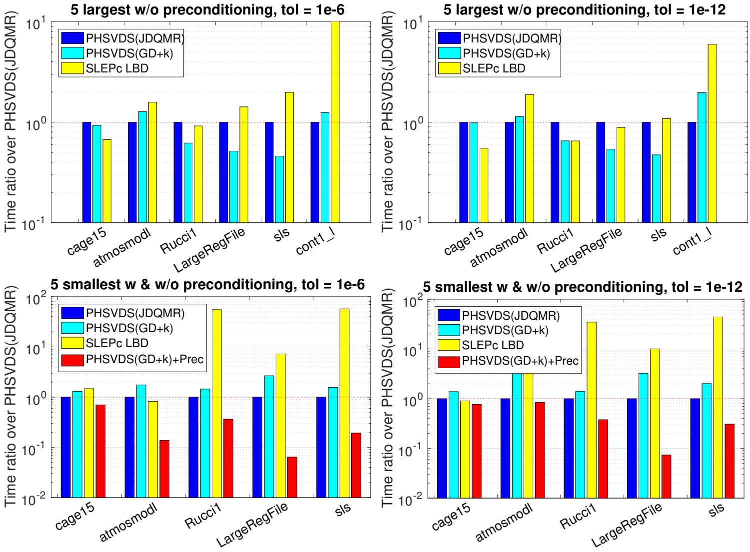


Experiments with PRIMME_SVDS in PRIMME library

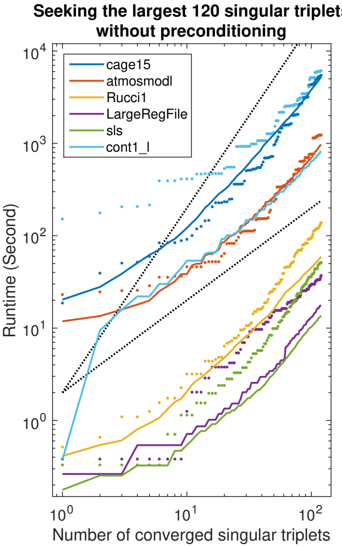
Table: Properties of test matrices: cage15(CAG), atmosmodl(ATM), Rucci1(RUC), LargeRegFile(LRF), sls(SLS), cont1_l(CON), relat9(REL), delaunay_n24(DEL), Laplacian(LAP). Gaps are computed as the smallest value $\frac{\sigma_i - \sigma_{i+1}}{\sigma_{i+1} - \sigma_n}$ for $\sigma_1 < \sigma_2 < \ldots < \sigma_n$.

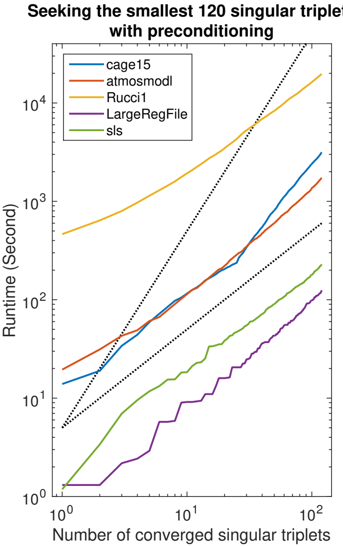
					gap ratios	
Matrix	rows m	cols n	nnz(A)	$\kappa(A)$	largest	smallest
CAG	5,154,859	5,154,859	99,199,551	1.2E+1	6E-4	1E-3
ATM	1,489,752	1,489,752	10,319,760	1.1E+3	5E-5	5E-5
RUC	1,977,885	109,900	7,791,168	6.7E+3	3E-3	5E-5
LRF	2,111,154	801,374	4,944,201	1.1E+4	1.2	3E-7
SLS	1,748,122	62,729	6,804,304	1.3E+3	4E-2	8E-7
CON	1,918,399	1,921,596	7,031,999	2.0E+8	6E-6	5E-8
REL	12,360,060	549,336	7,791,168	∞	3E-3	–
DEL	16,777,216	16,777,216	50,331,601	∞	2E-3	–
LAP	8,000p	8,000p	55,760p	–	–	–

Versus SLEPc on a distributed memory system

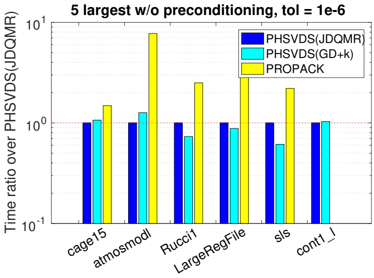


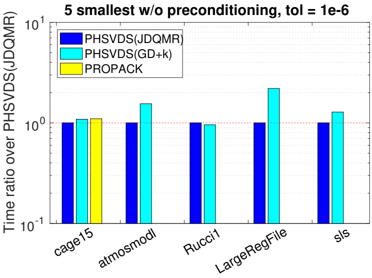
Versus SLEPc on a distributed memory system



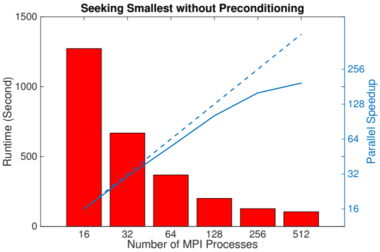


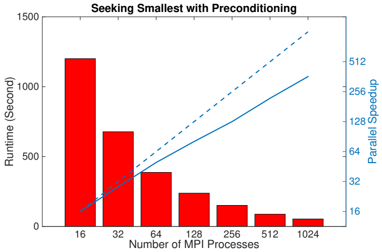
Versus PROPACK on a shared memory system

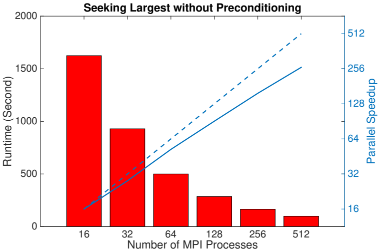


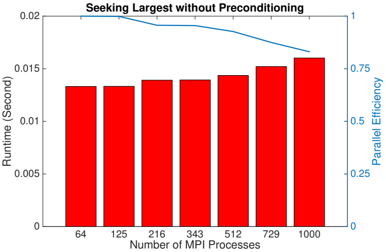


Strong and weak scaling









PRIMME_SVDS: a high-performance preconditioned SVD solver in PRIMME Library

PRIMME_SVDS is built on top of PRIMME_Eigs for solving large-scale SVD problems

Methods PRIMME_SVDS implements:

- primme_svds_hybrid: PHSVDS on both C and B
- primme_svds_normalequations: eigenmethod on C
- primme_svds_augmented: eigenmethod on B

Problems PRIMME_SVDS solves:

- primme_svds_largest: seeking for largest singular triplets
- primme_svds_smallest: seeking for smallest singular triplets
- primme_svds_closest_abs: seeking for interior singular triplets

Interfaces PRIMME_SVDS provides:

- C/C++, Fortran, Matlab/Octave, Python, R

How to use PRIMME_SVDS

Preparation steps (see more details in Tutorial part II): 1) download PRIMME; 2) specify C compiler in Make_flags; 3) make lib

A simple example in C:

```
#include "primme.h" // PRIMME_SVDS header file
double *svals, *rnorms, *svecs; // Arrays for singular triplets
primme_svds_params primme_svds; // Declare PRIMME_SVDS struct
primme_svds_initialize(&primme_svds); // Initialization
// Set user's Matvec function that implements both A*x and A^T*x
primme_svds.matrixMatvec = matrixMatvecSVDS;
primme_svds.m = 1E6, primme_svds.n = 1E4; // Set the size of A^{m×n}
primme_svds.numSvals = 5; // Number of singular triplets wanted
primme_svds.eps = 1e-12; // ||r|| <= eps * ||A||
primme_svds.target = primme_svds_smallest;
svals = (double*)malloc(primme_svds.numSvals*sizeof(double));
svecs = (double*)malloc((primme_svds.n+primme_svds.m)
        *primme_svds.numSvals*sizeof(double));
rnorms = (double*)malloc(primme_svds.numSvals*sizeof(double));
ret = dprimme_svds(svals, svecs, rnorms, &primme_svds);
```

User's Matrix-vector product for PRIMME_SVDS

```
void matrixMatvecSVDS(                       // Do y = A * x and y = A^T * x
  void *x, PRIMME_INT *ldx,                  // Input vectors and lead. dim.
  void *y, PRIMME_INT *ldy,                  // Output vectors and lead. dim.
  int *blockSize,                            // Number of columns of a block
  int *transpose,                            // Matrix transpose flag
  primme_svds_params *primme_svds,           // PRIMME_SVDS configuration
  int *err                                   // Output flag error
) {
   if (*transpose == 0) { // Do y = A * x
      for (int i=0; i<*blockSize; i++)
        Do y_i = A * x_i
   }
   else { // Do y = A^T * x
      for (int i=0; i<*blockSize; i++)
        Do y_i = A' * x_i
   }

   *err = 0; // All went ok
}
```

How to use PRIMME_SVDS in Matlab

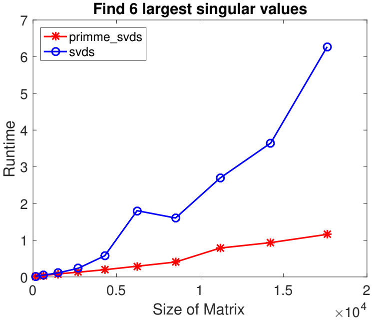
PRIMME_SVDS can be called as easily as SVDS in Matlab:

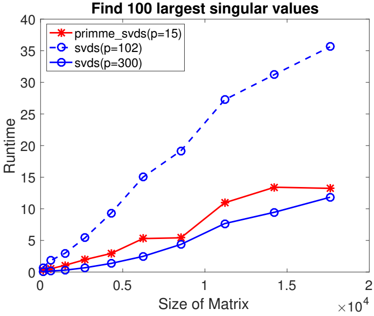
```
Input: [A, K, SIGMA, OPTIONS, P]

Output: [U, S, V, R, STATS]

Function call:
 primme_svds(A)
 primme_svds(A, K)
 primme_svds(A, K, SIGMA)
 primme_svds(A, K, SIGMA, OPTIONS)
 primme_svds(A, K, SIGMA, OPTIONS, P)
 primme_svds(A, K, SIGMA, OPTIONS, P1, P2)
 primme_svds(A, K, SIGMA, OPTIONS, Pfun)
 primme_svds(Afun, M, N,...)
```

To learn more parameters in options, type "help primme_svds"

Largest singular values

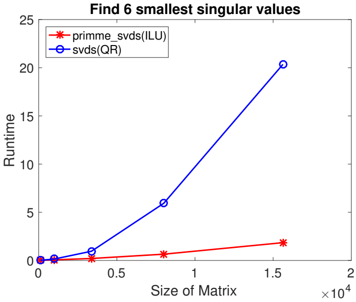
```
A = delsq(numgrid('C',15));
s = primme_svds(A)
s' =
    7.8666    7.7324    7.6531    7.5213    7.4480    7.3517
s = primme_svds(A,3)
s' =
    7.9987    7.9975    7.9967
```

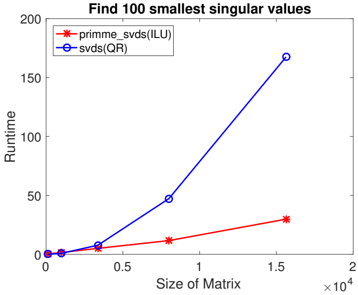




Smallest singular values

```
A = laplacian([20, 20, 20]);
[u,s,v,rnorm,stats] = primme_svds(A,6,'S');
diag(s)' = 0.0670  0.1335  0.1335  0.1335  0.2000  0.2000
stats.numMatvecs = 6050
opts.tol = 1E-14;
[u,s,v,rnorm,stats] = primme_svds(A,6,'S',opts);
stats.numMatvecs = 10794
[L,U] = ilu(A,struct('type','ilutp','droptol',1e-3));
[u,s,v,rnorm,stats] = primme_svds(A,5,'S',opts,L,U);
stats.numMatvecs = 440
```





Find singular values with Afun and Pfun

```
function y = Afun(x,tflag)
    global A; % User's matrix (e.g. deter4 in Florida Sparse)
    if strcmp(tflag,'notransp') y = A*x;
    else y = A'*x;

opts.tol = 1E-6;
opts.method = 'primme_svds_normalequations';
[u,s,v,rnorm,stats] = primme_svds(@(x,tflag)Afun(x,tflag),
3235, 9133, 5, 'S', opts);
diag(s)' = 0.0275    0.1328    0.1331    0.1336    0.1337
stats.numMatvecs = 8420

function y = Pfun(x,model)
    global LL; % Use RIF to generate a preconditioner for AA'
    if strcmp(model, 'AAH') % AA' = LL*LL'
        y = LL'\(LL\x); % y = LL^-T * LL^-1 * x

[u,s,v,rnorm,stats] = primme_svds(@(x,tflag)Afun(x,tflag),
3235, 9133, 5, 'S', opts, @(x,model)Pfun(x,model));
diag(s)' = 0.0275    0.1328    0.1331    0.1336    0.1337
stats.numMatvecs = 1164
```

References

For more details in PHSVDS method and PRIMME_SVDS sovler, please refer to the following papers and PRIMME website `http://www.cs.wm.edu/~andreas/software/doc/readme.html`:

- Lingfei Wu, Eloy Romero, and Andreas Stathopoulos, "PRIMME_SVDS: A High-Performance Preconditioned SVD Solver for Accurate Large-scale Computations", SIAM Journal on Scientific Computing (2017), `https://arxiv.org/abs/1607.01404`.
- Lingfei Wu and Andreas Stathopoulos, "A Preconditioned Hybrid SVD Method for Computing Accurately Singular Triplets of Large Matrices", SIAM Journal on Scientific Computing 37-5 (2015), pp. S365-S388.