Computing the SVD for large and sparse matrices

Rasmus Munk Larsen
SCCM & SOI-MDI
Stanford University

Lawrence Berkeley Lab, June 2000

Overview

- Introduction
- Golub-Kahan (Lanczos) bidiagonalization and semiorthogonalization
- PROPACK: Software for sparse SVD and eigenvalue problems
- Numerical experiments – comparisons between PROPACK, LANSO and ARPACK
- One-sided reorthogonalization
- Conclusion

The singular value decomposition (SVD)

Definition: Let A be a rectangular $m \times n$ matrix with $m \geq n$, then the SVD of A is

$$A \;=\; U\,\Sigma\,V^T \;=\; \sum_{i=1}^{n} \sigma_i\, u_i\, v_i^T \;,$$

where the matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal and

$$\Sigma = \begin{array}{c} n \\ m-n \end{array} \left[\begin{array}{c} \Sigma_1 \\ 0 \end{array} \right] ,$$

where $\Sigma_1 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ and

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0 \;,$$

r is the rank of A.

The SVD has numerous applications in, e.g.,

- Information retrieval (LSI)
- Inverse problems (regularization)
- Statistics (PCA)
- Image and signal processing

Equivalent symmetric eigenvalue problems

Let the singular value decomposition of the $m \times n$ matrix A be

$$A = U \, \Sigma \, V^T$$

and assume without loss of generality that $m \geq n$. Then

$$V^T (A^T A) V = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2) \; ,$$

$$U^T (A A^T) U = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2, \underbrace{0, \ldots, 0}_{m-n}) \; .$$

Moreover, if $U = \left[\, U_1 \; U_2 \, \right]$ and

$$Y = \frac{1}{\sqrt{2}} \begin{bmatrix} U_1 & U_1 & \sqrt{2} U_2 \\ V & -V & 0 \end{bmatrix} \; , \quad C = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

then the orthonormal columns of the $(m+n) \times (m+n)$ matrix Y form an eigenvector basis for the 2-cyclic matrix C and

$$Y^T \, C \, Y \; = \; \mathrm{diag}(\sigma_1, \ldots, \sigma_n, -\sigma_1, \ldots, -\sigma_n, \underbrace{0, \ldots, 0}_{m-n}) \; .$$

SVD using sparse symmetric eigensolvers

Many sophisticated software packages exist for the symmetric eigenvalue problem. To mention a few:

- Lehoucq, Sorensen & Yang 1992 – 1997: (P)ARPACK
- Parlett, Simon, Wu et al. 1984 – 1999: (P)LANSO & TRLAN
- Marques 1998: LZPACK & BLZPACK

In several studies (Berry 1992 (SVDPACK), Eldén & Lundström 1996) LANSO and ARPACK have proved highly efficient for computing a few singular triplets of large and sparse or structured matrices.

However, using a symmetric eigensolver as a "black box" for SVD has certain disadvantages.

Using a symmetric eigensolver as a "black box"

Method 0: $A^T A$

- Severe loss of accuracy of small singular values if A is ill-conditioned.
- Fast when $n \ll m$ since only Lanczos vectors of length n need to be stored.

Method 1: $C = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$

- Lanczos vectors have length $m + n \Rightarrow$ Waste of memory and unnecessary work in reorthogonalization.
- Ritz values converge to pairs of $\pm\sigma_i \Rightarrow$ Twice as many iterations are needed.

To (almost) get the best of both worlds: Combine **Lanczos bidiagonalization** (LBD) with the efficient **semi-orthogonalization** schemes developed for the symmetric eigenvalue problem.

Algorithm Bidiag1 **(Paige & Saunders)**

---

1. Choose a starting vector $p_0 \in \mathbb{R}^m$, and let $\beta_1 = \|p_0\|,\ u_1 = p_0/\beta_1$ and $v_0 \equiv 0$
2. **for** $i = 1, 2, \ldots, k$ **do**
   $r_i = A^T u_i - \beta_i v_{i-1}$
   $\alpha_i = \|r_i\|$
   $v_i = r_i/\alpha_i$
   $p_i = A v_i - \alpha_i u_i$
   $\beta_{i+1} = \|p_i\|$
   $u_{i+1} = p_i/\beta_{i+1}$
   end

After k steps we have the decomposition:

$$AV_k = U_{k+1} B_{k+1} \ ,$$

where V_j and U_{j+1} have orthonormal columns and

$$B_{k+1} = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix}$$

Semiorthogonality, fundamental result

Simon '84: Let

$$H Q_k = Q_k T_k + \beta_{k+1} q_{k+1} e_k^T$$

be the tridiagonal decomposition computed after k steps of the Lanczos algorithm on the hermitian matrix H. If the columns of Q_k are kept *semiorthogonal*, i.e.,

$$\max_{1 \le i,j \le k} |q_i^T q_j| \le \sqrt{\mathbf{u}/k} \qquad \text{for } i \ne j \ ,$$

then

$$\tilde{Q}_k^T \, H \, \tilde{Q}_k = T_k + E_k \ ,$$

where $Q_k = \tilde{Q}_k \tilde{R}_k$ is the compact QR-decomposition of Q_k and the elements of E_k is of order $O(\mathbf{u} \| H \|)$.

It follows (Wiedlandt-Hoffman) that $\lambda(T_k)$ are Ritz values for H within $O(\mathbf{u} \| H \|)$.

Semiorthogonality in LBD

Bidiag1 is equivalent to performing $2k+1$ steps of symmetric Lanczos with matrix C and starting vector $(u_1, 0)^T \in \mathbb{R}^{m+n}$, thus Simon's result gives

Corollary: Define the levels of orthogonality in Bidiag1 by $\nu_{ij} \equiv v_i^T v_j$ and $\mu_{ij} \equiv u_i^T u_j$. If

$$\max_{1 \le i,j \le k} |\nu_{ij}| \quad \le \quad \sqrt{\mathbf{u}/(2k+1)} \qquad \text{for } i \ne j \ ,$$

$$\max_{1 \le i,j \le k+1} |\mu_{ij}| \quad \le \quad \sqrt{\mathbf{u}/(2k+1)} \qquad \text{for } i \ne j \ ,$$

then

$$\tilde{U}_{k+1}^T A \tilde{V}_k = B_{k+1} + O(\mathbf{u}\|A\|) \ ,$$

where $U_{k+1} = \tilde{U}_{k+1}\tilde{J}_{k+1}$ and $V_k = \tilde{V}_k \tilde{K}_k$ are the compact QR-factorizaton of U_{k+1} and V_k.

Therefore $\sigma(B_{k+1})$ are Ritz values for A within $O(\mathbf{u}\|A\|)$.

The "ω-recurrences" for LBD

In finite precision arithmetic:

$$\begin{aligned} \alpha_j v_j &= A^T u_j - \beta_j v_{j-1} + f_j \\ \beta_{j+1} u_{j+1} &= A\, v_j - \alpha_j u_j + g_j \ , \end{aligned}$$

where f_j and g_j represent round-off errors.

It is simple to show that $\mu_{j+1,i}$ and ν_{ji} satisfy the coupled recurrences:

$$\begin{aligned} \beta_{j+1}\mu_{j+1,i} &= \alpha_i \nu_{ji} + \beta_i \nu_{j,i-1} - \alpha_j \mu_{ji} \\ &\quad + u_i^T g_j - v_j^T f_i \ , \qquad (1) \\ \alpha_j \nu_{ji} &= \beta_{i+1}\mu_{j,i+1} + \alpha_i \mu_{ji} - \beta_j \nu_{j-1,i} \\ &\quad - u_j^T g_i + v_i^T f_j \ , \qquad (2) \end{aligned}$$

where $\mu_{ii} = \nu_{ii} = 1$ and $\mu_{0i} = \nu_{0i} \equiv 0$ for $1 \le i \le j$.

These recurrences were derived independently by Simon & Zha 1997.

Partial reorthogonalization: *Use the recurrences to monitor the size of $\mu_{j+1,i}$ and ν_{ji}. Reorthogonalize only when necessary.*

Bounding the round-off terms

We can bound the size of the round-off term

$$\begin{aligned} |u_i^T g_j - v_j^T f_i| &\leq \|g_j\| + \|f_i\| \\ &\leq 4\,\mathbf{u}\,((\alpha_j^2 + \beta_{j+1}^2)^{1/2} + (\alpha_i^2 + \beta_i^2)^{1/2}) + \epsilon_{MV} \\ &\equiv \tau \end{aligned}$$

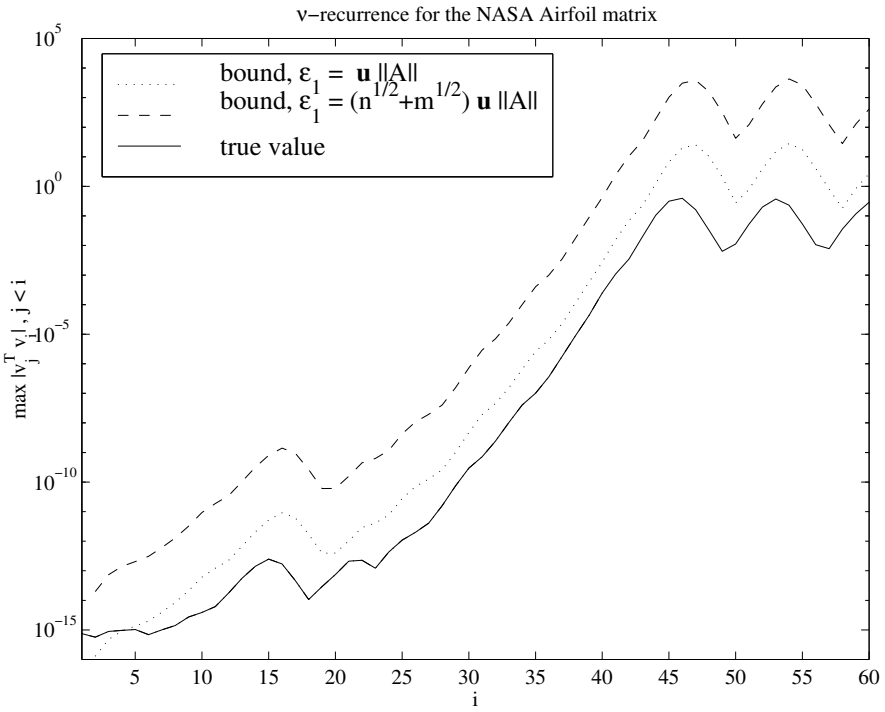
Round-off from matrix-vector multiply ϵ_{MV} is estimated conservatively: $\epsilon_{MV} \leq \mathbf{u}\,(\bar{n} + \bar{m})\,\|A\|$, where $\bar{n}$ $(\bar{m})$ is the maximum number of non-zeros per row (column) in A.

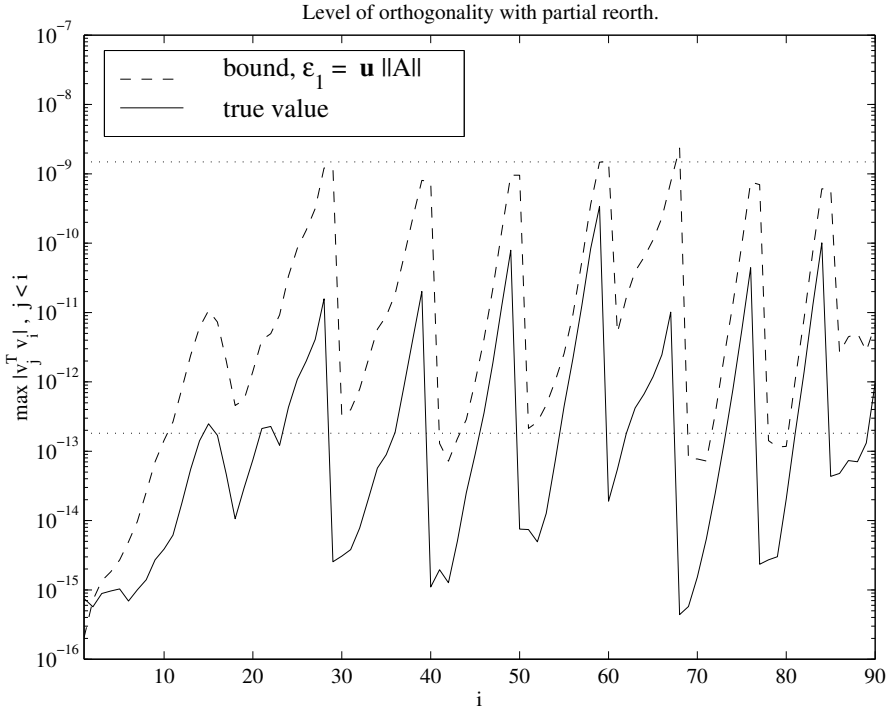
Conservative updating rules $\nu_{j-1,i} \rightarrow \nu_{ji}$ and $\mu_{ji} \rightarrow \mu_{j+1,i}$:

$$\begin{aligned} \nu'_{ji} &= \beta_{i+1}\mu_{j,i+1} + \alpha_i\mu_{ji} - \beta_j\nu_{j-1,i} \\ \nu_{ji} &= (\nu'_{ji} + \mathrm{sign}(\nu'_{ji})\tau)/\alpha_j \end{aligned}$$

$$\begin{aligned} \mu'_{j+1i} &= \alpha_i\nu_{ji} + \beta_i\nu_{j,i-1} - \alpha_j\mu_{ji} \\ \mu_{j+1,i} &= (\mu'_{j+1,i} + \mathrm{sign}(\mu'_{j+1,i})\tau)/\beta_{j+1} \end{aligned}$$

Illustration of recurrences





Partial reorthogonalization (next slide) reduced the work compared to full reorthogonalization from $10100 \longrightarrow 926$ inner products!

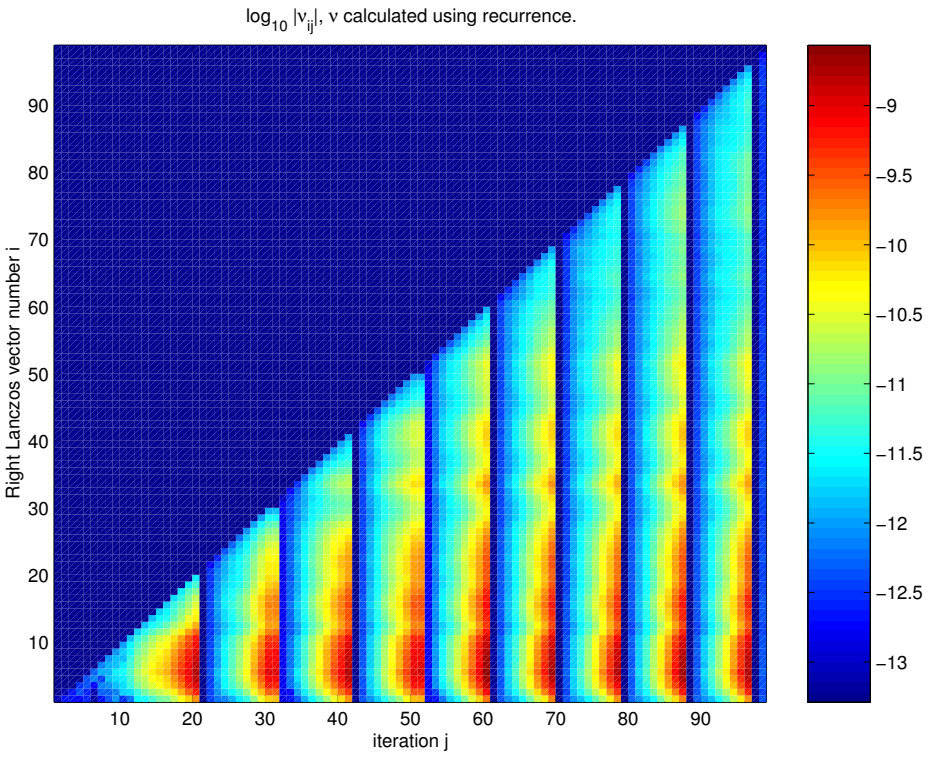
Outline of algorithm

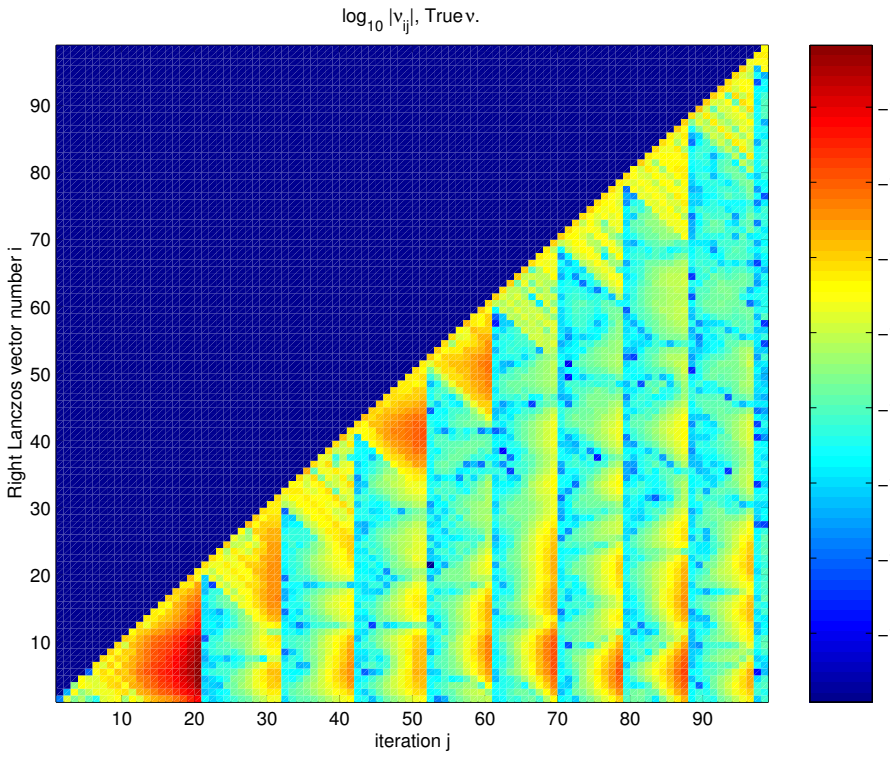
Lanczos bidiagonalization with Partial reorthogonalization:

force = FALSE
for $j = 1, 2, \ldots, k$ **do**
  $\alpha_j v_j = A^T u_j - \beta_j v_{j-1}$
  Update $\nu_{j-1,i} \rightarrow \nu_{ji}$
  if $\max_{1 \le i < j} |\nu_{ji}| > \mathrm{tol}$ **or** force
    Reorthogonalize v_j
    force = $(\max_{1 \le i < j} |\nu_{ji}| > \mathrm{tol})$
  end
  $\beta_{j+1} u_{j+1} = A\, v_j - \alpha_j u_j$
  Update $\mu_{ji} \rightarrow \mu_{j+1,i}$
  if $\max_{1 \le i < j+1} |\mu_{j+1,i}| > \mathrm{tol}$ **or** force
    Reorthogonalize u_{j+1}
    force = $(\max_{1 \le i < j+1} |\mu_{j+1,i}| > \mathrm{tol})$
  end
end

- The variable "force" causes extra reorthogonalizations, which are necessary due to the coupling between ν_{ji} and $\mu_{j+1,i}$.
- **It is not correct simply to replace "Reorthogonalize u_{j+1}" with "Reorthogonalize u_j and u_{j+1}" and "Reorthogonalize v_j" with "Reorthogonalize v_{j-1} and v_j".**

Estimated level of orthogonality...and the truth





Software

PROPACK: Software package written in Matlab.

Main components:

`lanpro`	:	Hermitian Lanczos with PRO
`lanbpro`	:	Lanczos bidiagonalization with PRO
`lansvd`	:	Singular value decomposition
`laneig`	:	Hermitian eigensolver (≈LANSO)

Important implementation details:

— respecting coupling between μ and ν
— extended local reorthogonalization
— iterated Gram-Schmidt reorth. (DGKS, BLAS-2)
— recovery from near zero α_i or β_i
— proper estimation of $\|A\|$
— interface similar to IRAM routines `eigs` and `svds`

URL: `http://soi.stanford.edu/~rmunk/PROPACK`

Fortran 77 versions of `lanbpro` and `lansvd` are also available upon request.

Numerical Experiments

The algorithms were tested by computing the first 10 singular triplets for the following real non-symmetric matrices from Matrix Market:

Name	m	n	nnz(A)
WELL1850	1850	712	8758
ILLC1850	1850	712	8758
TOLS4000	4000	4000	8784
MHD4800A	4800	4800	102252
AF23560	23560	23560	460598
BCSSTK32	90449	90449	1921955

Software:

Algorithm	Matlab	Fortran
Lanczos bidiagonalization with PRO	`lansvd`	LANSVD
Lanczos with PRO on A^TA	`laneig`	LANSO
Lanczos with PRO on C	`laneig`	LANSO
IRAM on A^TA	`eigs`	ARPACK
IRAM on C	`svds`	ARPACK

Experimental setup: PC workstation with 600 MHz Pentium III CPU, 512MB memory, IEEE arithmetic, running RedHat Linux 6.2.

Performance of Matlab functions

The Matlab implementations were tested using Matlab 5.3 with LAPACK numerics library. The table shows execution time in seconds:

Function	`lansvd`	`laneig`		`eigs`	`svds`
Matrix	A	A^TA	C	A^TA	C
WELL1850	1.00	0.63	1.77	2.20	13.31
ILLC1850	0.57	0.37	1.75	1.39	8.91
TOLS4000	2.73	1.76	6.76	16.84	73.71
MHD4800A	1.52	1.41	3.31	4.38	20.80
AF23560	30.57	20.92	61.54	50.51	181.61
BCSSTK32	85.72	76.34	192.66	179.76	628.69

- `laneig`(A^TA) wins on speed
- `lansvd` is a factor of 2 faster than `laneig`(C)
- PROPACK routines up to an order of magnitude faster than `svds`!

Performance, Fortran implementations

The Fortran version of `lansvd` was compared with the LANSO and ARPACK codes. The table shows execution (CPU) time in seconds:

Program	LANSVD	LANSO		ARPACK	
Matrix	A	A^TA	C	A^TA	C
WELL1850	0.30	0.12	1.00	0.20	1.44
ILLC1850	0.19	0.12	0.61	0.14	1.07
TOLS4000	1.04	0.99	6.32	2.66	9.97
MHD4800A	0.51	0.38	1.22	0.73	4.70
AF23560	11.11	4.49	15.42	8.73	28.10
BCSSTK32	28.22	29.09	94.77	107.91	194.32

- LANSO(A^TA) wins on speed
- LANSO consistently faster than ARPACK on same problem
- LANSVD significantly faster than any other backwards stable variants.

Experimental setup: EGCS (GNU) 2.95.2 compiler suite, ASCI Red BLAS by Greg Henry, LAPACK 3.0 compiled locally.

Tuning LANSO

The results for LANSO were improved by changing the strategy for expanding Lanczos basis.
Before:

```
IF (NEIG.EQ.0) THEN
   LAST = FIRST+8
ELSE
   LAST = FIRST+MAX(2,((J-6)*(MAXPRS-NEIG))/NEIG)
ENDIF
```

After:

```
IF (NEIG.EQ.0) THEN
   LAST = FIRST+max(8,FIRST/2)
ELSE
   LAST = FIRST+MAX(2,((J-6)*(MAXPRS-NEIG))/(2*NEIG+1))
ENDIF
```

	Before		After	
Matrix	A^TA	C	A^TA	C
WELL1850	0.62	0.65	0.12	1.00
ILLC1850	1.28	0.73	0.12	0.61
TOLS4000	0.28	1.52	0.99	6.32
MHD4800A	0.46	1.45	0.38	1.22
AF23560	20.85	41.16	4.49	15.42
BCSSTK32	53.64	93.02	29.09	94.77

One-sided reorthogonalization

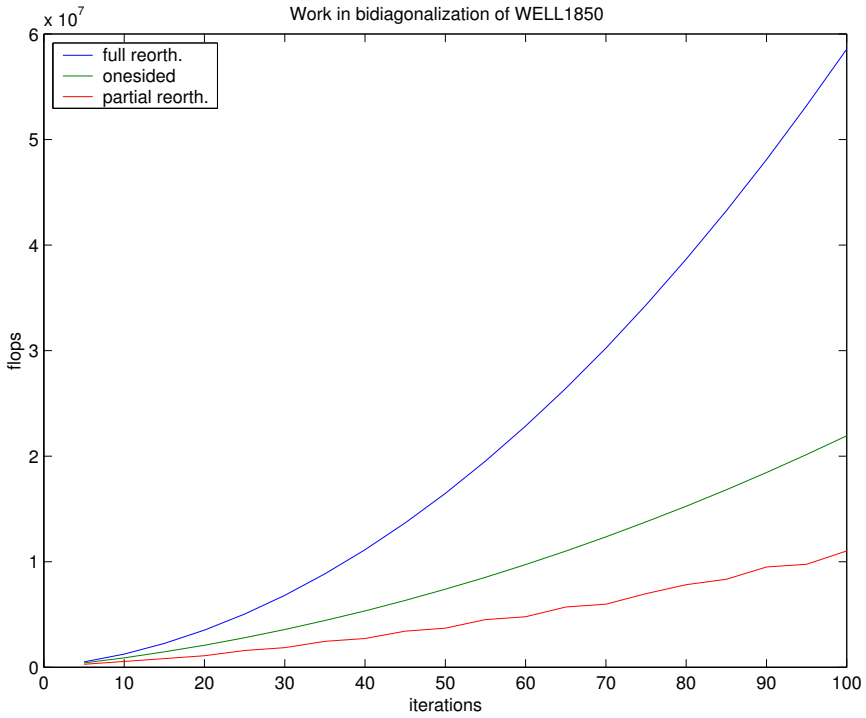
- Simon & Zha (1997): It is sufficient to keep *either* U_{k+1} *or* V_k orthogonal to compute accurate low rank approximations.

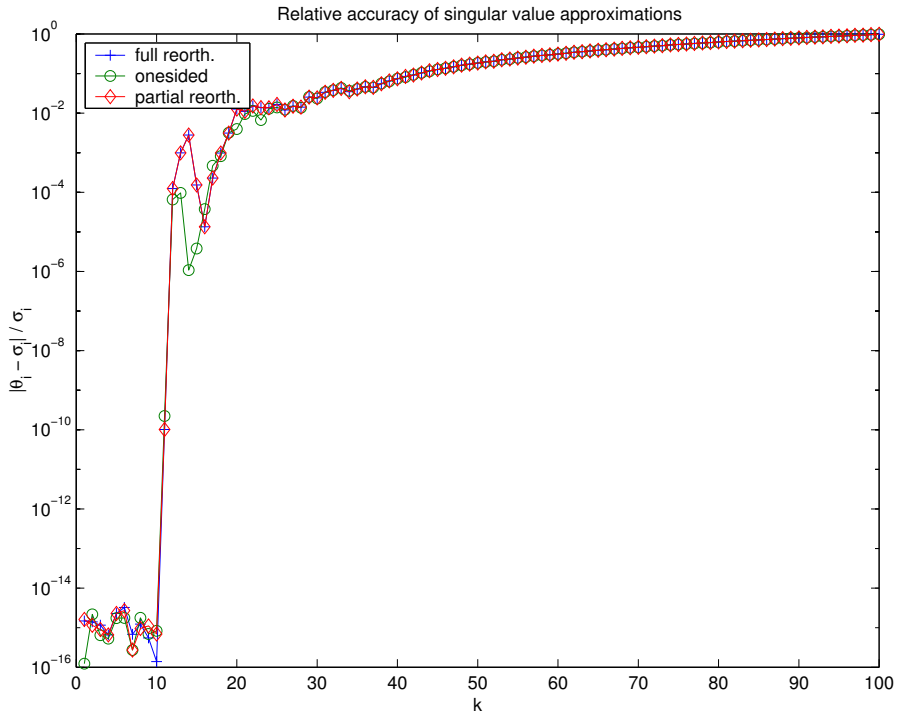
$$A \approx U_{k+1} B_k V_k^T$$

- It is also sufficient to compute accurate singular values of A.
- Leads to efficient algorithms for "skinny" matrices ($m \gg n$) $\Rightarrow$ only the short vectors v_i need to be reorthogonalized.
- Only the current long vector u_i need not be stored $\Rightarrow$ Storage requirements are low.

One-sided reorthogonalization

Singular values are just as accurate as with full or partial reorthogonalization. Example matrix not skinny enough to beat partial reorthogonalization in term of flops.





Conclusion

Experiments with matrices from different applications show that:

- PROPACK provides efficient and robust replacements for `eigs` and `svds`
- LANSVD is almost as fast as LANSO(A^TA) (ratio $(m+n)/(2n)$)
- LANSVD is 2-4 times as fast LANSO(C) with same higher accuracy
- LANSVD and LANSO generally outperform ARPACK. Up to a factor of 2-3 on large examples.
  Caveats: No restarts = more memory!

Future work:

- Restarting (e.g. Thick Restarts as in TRLAN)
- Finish parallel implementation of LANSVD
- Add solver based on one-sided reorth. to PROPACK