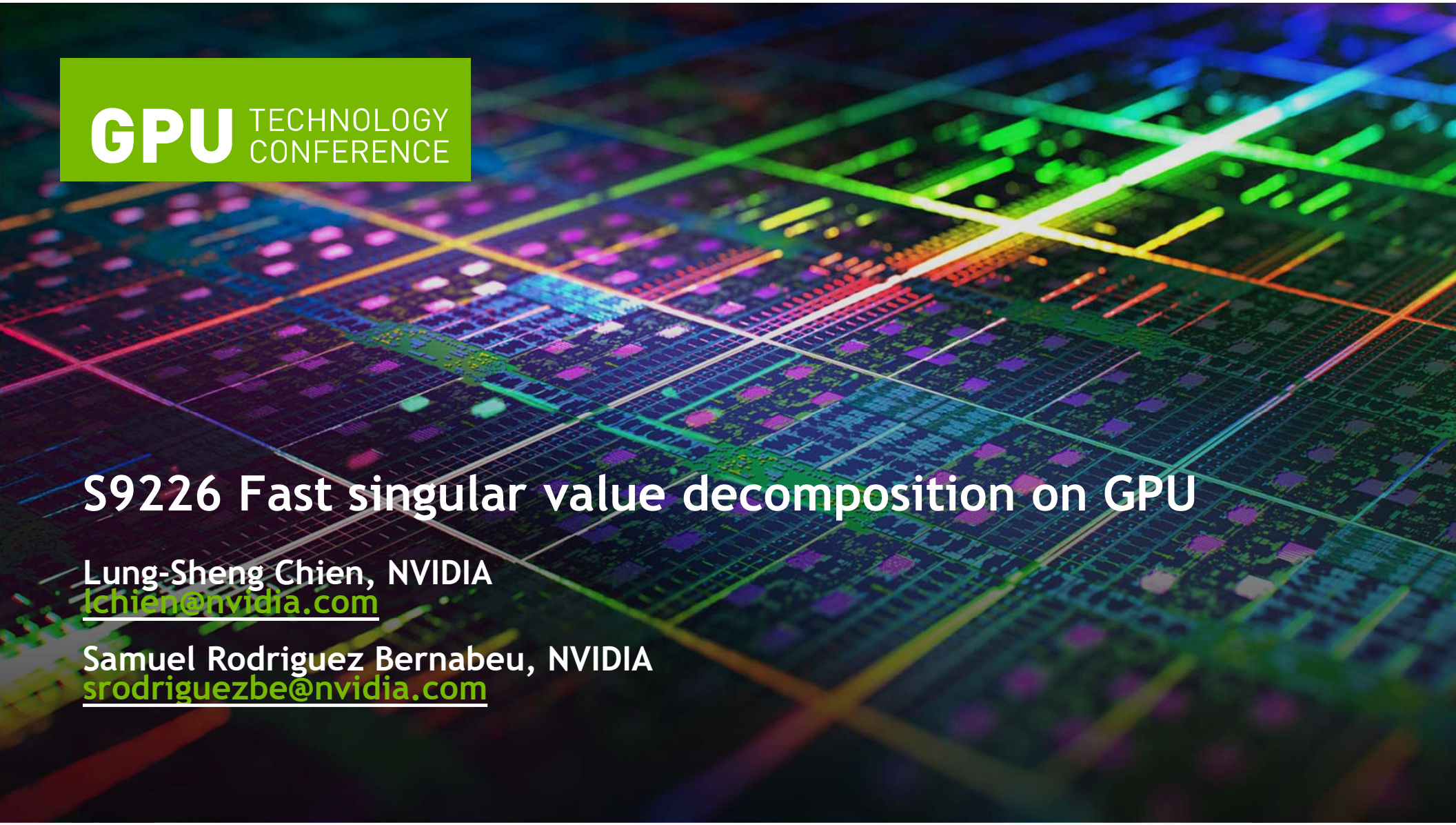


S9226 Fast singular value decomposition on GPU

Lung-Sheng Chien, NVIDIA
lchien@nvidia.com

Samuel Rodriguez Bernabeu, NVIDIA
srodriguezbe@nvidia.com

Outline

- Issues of GESVD
- Approximate SVD
- Randomized SVD
- Conclusions

General Singular Value Decomposition

- The standard form is

$$A = USV^T$$

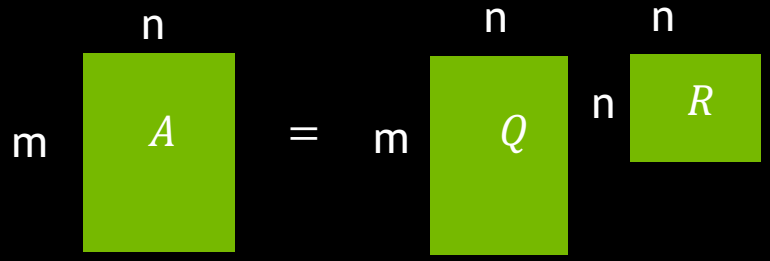
- LAPACK GESVD is most popular routine based on QR iteration

- cuSOLVER library provides two SVD routines
  - GESVD: same algorithm as LAPACK
  - GESVDJ: two-sided Jacobi method

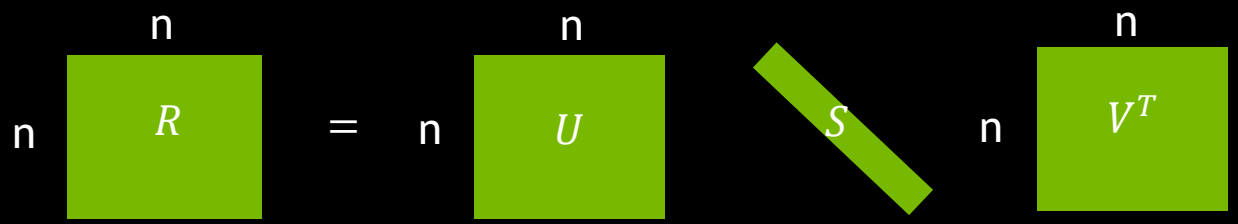
- Tall skinny SVD is a common use case in data analytics
  - singular vectors are required
  - only requires few large singular values
  - typical size: 1.e+6 rows, 100 columns

Strategy of GESVD

- QR factorization: preprocess of tall skinny matrix (m >> n)



- SVD on square matrix (GEBRD + BDSQR + ORGBR)



- GPU does not perform well on tall skinny QR factorization
- GPU does not perform well on QR iteration for small matrix

Review performance on square matrix

- The formula of flops is $2N^3$, same as SGEMM
- The bigger, the faster
- The runtime of SGESVD is about 50x of SGEMM
- Jacobi method (GESVDJ) is faster than QR iteration (GESVD) when matrix size is less than 1024

n	cusolver SGESVD	cusolver SGESVDJ	MKL SGESVD	SGEMM
32	0.04	0.12	0.11	1
64	0.13	0.45	0.12	7
128	0.48	1.63	1.16	74
256	1.31	4.84	2.80	558
512	5.22	13.56	10.26	2,828
1024	18.97	27.73	8.33	8,586
2048	63.15	40.90	19.80	12,514
4096	152.07	58.08	8.03	13,366
8192	264.11	49.19	5.52	13,956

CPU: Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz
MKL: compilers_and_libraries_2018.0.128 with 8 cores

GPU: V100

Review performance on tall skinny matrix

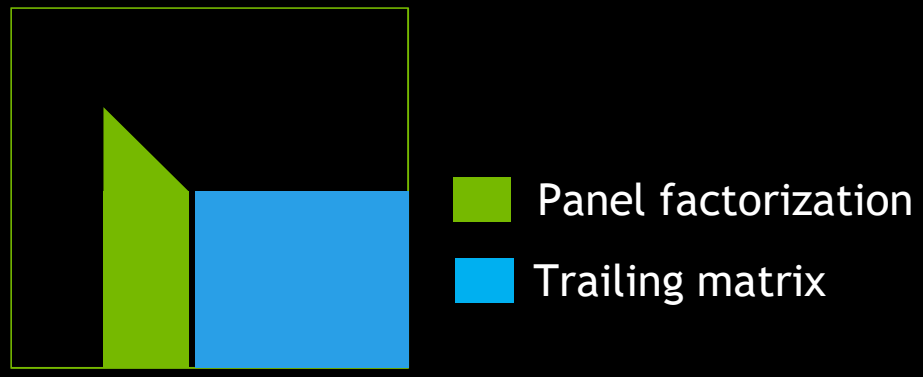
- N = 32, M varies from 1,000 to 1e+6
- SGEQRF (QR factorization) is proportional to M because complexity is $2MN^2$ flops
- QR factorization is the bottleneck in SVD when M becomes bigger and bigger (QR ratio from 0.17 to 0.93)

M	SGEQRF (sec)	SGESVDJ (sec)	QR ratio
1,000	0.00021	0.00128	0.17
10,000	0.00058	0.00147	0.40
100,000	0.00524	0.00654	0.80
1,000,000	0.05897	0.06336	0.93

N is fixed to 32

Weakness of QR factorization

- M = 8192, N varies from 32 to 4096
- Complexity of SGEQRF is $2MN^2$ flops, however the runtime is proportional to N
- Only trailing matrix uses BLAS-3, it is negligible on tall skinny matrix, the runtime is dominated by panel factorization, which is mainly BLAS-1



N	SGEQRF (Gflops)
32	32.6
64	66.3
128	116.6
256	159.5
512	338.1
1024	627.6
2048	990.8
4096	2487.6

M is fixed to 8192

Goal and strategy

Goal

- Tall skinny matrix
  - Up to millions of rows
  - Up to hundreds of columns
- Few large singular values
- Left and right singular vectors

Pros and Cons

- QR factorization is not good on tall skinny matrix
- Jacobi method is faster than QR iteration on small matrix
- SGEMM is much faster

$A = USV^T$

QR factorization

strategy

$A^TA = VS^2V^T$

GEMM + EIG

Technical issues

- Rounding error
  $A^T A$ has rounding error proportional to $\|A\|^2$, so high precision GEMM is used to control rounding error

- Performance
  $A^T A$ is N-by-N, a small matrix compared to tall skinny A
  regular GEMM does not perform well
  Need special GEMM to improve the performance

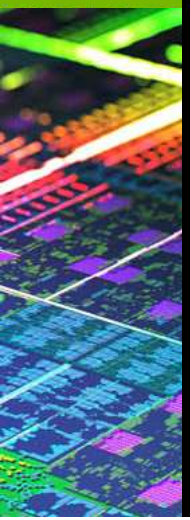
  Jacobi method to compute eigen-pair of ($A^T A$) because it is faster than QR iteration on small matrix

GESVDA: approximate SVD

- B = $A^T A$ by DGEMM
  DGEMM can reduce rounding errors

- (S,V) = eig(B) by DSYEVJ (Jacobi method)
  adjust stopping criteria to improve the performance

- $U = AVS^{-1}$ by DGEMM and scaling
  left singular vector is not accurate when singular value is small



Quality of solution

- right singular vector is always accurate up to 1.e-6
- Singular values and left singular vectors depend on M and N

$$N \leq 100$$

$$M \leq 176{,}000 * N$$

For those numerical singular values bounded below by

$$S_l(A) \geq 2.65 \times 10^{-3} \|A\|_F$$

The singular value and singular vectors are accurate up to 1.e-6

Example: Largest singular value $S_l(A) \geq \frac{1}{\sqrt{N}} \|A\|_F$ is accurate up to 1.e-6

Performance of GESVDA

- N = 32, M varies from 1,000 to 1.e+6
- Runtime of eigenvalue solver is independent of M
- Runtime of GESVDA is determined by GEMM
  it is up to 16x faster than GESVDJ
- The speedup comes from replacing QR factorization by GEMM

M	SGEQRF (sec)	SGESVDJ (sec)	SGESVDA (sec)	QR ratio	speedup
1,000	0.00021	0.00128	0.00077	0.17	1.67
10,000	0.00058	0.00147	0.00078	0.40	1.89
100,000	0.00524	0.00654	0.00118	0.80	5.55
1,000,000	0.05897	0.06336	0.00376	0.93	16.84

N is fixed to 32

GESVDA Breakdown

- DGEMM is $2MN^2$, DSYEVJ is $O(N^3)$, others are $O(MN)$
- DGEMM is very efficient, only 40% of total time
- The cost of DSYEVJ is independent of M, so it decreases as M increases
- "compute U" is slower than "Residual" because it requires 'double precision"

Ratio of each component in GESVDA

M	DGEMM (sec)	DSYEVJ (sec)	Compute U (sec)	Residual (sec)
1,000	0.13	0.78	0.03	0.10
10,000	0.15	0.74	0.03	0.10
100,000	0.33	0.46	0.13	0.10
1,000,000	0.40	0.16	0.35	0.11

N is fixed to 32

Performance of batched GESVDA

- SGESVDJ is performed by OpenMP
- SGESDVA is performed by multi-stream
- OpenMP can run “batchSize” GEQRF in parallel (GEQRF is 40%+ of GESVD), so speedup of batchSize 32 is not significant

M	batchSize	SGESVDJ (sec)	SGESVDA (sec)	speedup
16,384	1	0.0022	0.0012	1.77
16,384	32	0.0562	0.0076	7.41
65,536	1	0.0051	0.0015	3.41
65,536	32	0.0977	0.0141	6.94
1,048,576	1	0.0770	0.0062	12.45
1,048,576	16	0.5329	0.0775	6.87

N is fixed to 35

Batched GESVDA breakdown

- DGEMM is $2MN^2$, DSYEVJ is $O(N^3)$, others are $O(MN)$
- DGEMM is less than 40% of total time
- The cost of DSYEVJ shrinks to 3% because of inhouse batched design it is no longer kernel launch limited
- "compute U" becomes bottleneck

Ratio of each component in batched GESVDA

M	batchSize	DGEMM (sec)	DSYEVJ (sec)	Compute U (sec)	Residual (sec)
16,384	32	0.21	0.50	0.13	0.15
65,536	32	0.32	0.22	0.29	0.17
1,048,576	16	0.36	0.03	0.43	0.17

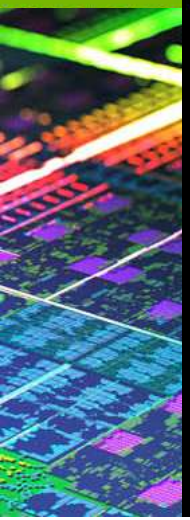
N is fixed to 35

double-double (fp128) GEMM ?

- Question: low-rank SVD with accuracy up to 1.e-14 ?
- N = 32, M varies from 1,000 to 1.e+6
- QD package (http://crd-legacy.lbl.gov/~dhbailey/mpdist/)
- LGEMM: C (dd) += A (d) * B (dd)
- LGEMM is only useful when M > 100,000

M	DGEQRF (sec)	DGEMM / QR	LGEMM / QR
1,000	0.00023	1.49	0.38
10,000	0.00077	7.39	0.96
100,000	0.00822	26.19	1.88
1,000,000	0.08447	14.53	5.34

N is fixed to 32

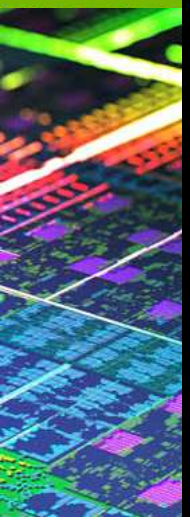


Conclusions

- GESVDA replaces SGEQRF by DGEMM to gain the speedup up to 16x
- GESVDA has inhouse batched eigenvalue solver to avoid limitation of kernel launches
- GESVDA can deliver good quality of singular values and singular vectors in common use case
- GESVDA is delivered in cuda 10.1 with batched API

Low-rank approximation of A

- Low-rank approximation of a given matrix A.
  - *Fixed precision.*
  - *Fixed rank.*
- Truncation of SVD gives best rank-k approximation [Eckart-Young-Mirsky]
  - Huge cost. Time complexity is O(n^3)



Randomized SVD

- Compute top-k eigenpairs to ***sufficient*** accuracy.
  - Data analytics, PCA, clustering: 1e-2 could be enough
  - Physics simulations: 1e-2 may be useless
- Highlights:
  - Reduced time and space complexity.
  - Preserve sparsity of A
  - One-pass or streaming algorithms (touch A once)

Core idea of Randomized LA

Data: A, k
Result: Q,B such $A \approx QB$

1 $\Omega = \text{SketchingMatrix}(\text{ A}, \mathcal{O}(k)\ )$
2 $C = A\Omega$
3 $Q = orth(C)$
4 $B = Q^T A$

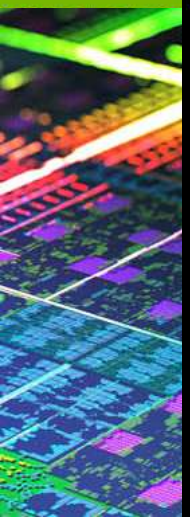
- C is a n-by-O(k) matrix that ensures with high probability:

$$||A - QQ^H A|| \leq \lambda_{k+1} + \left(1 + 4\sqrt{\frac{2m}{k-1}}\right)^{\left(\frac{1}{2q+1}\right)} \lambda_{k+1}$$

Halko et Al "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM review* 53.2 (2011)

Time complexity

Data: A, k	
Result: Q,B such $A \approx QB$	
1 $\Omega = \text{SketchingMatrix}(\ \text{A},\ \mathcal{O}(k)\ )$	$\mathcal{O}(m \times k)$
2 $C = A\Omega$	$\mathcal{O}(m \times n \times k)$
3 $Q = orth(C)$	$\mathcal{O}(m \times k^2)$
4 $B = Q^T A$	$\mathcal{O}(m \times n \times k)$



Sketching matrix

- m-by-order(k) matrix which:
  - Captures column space of A
  - Easy to construct
  - Easy to apply.
- Some options:
  - Gaussian projection
  - Subsampled Randomized Hadamard Transform
  - Count sketch
  - Leverage-score subsampling (sparse cases)

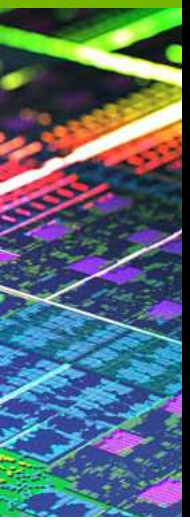
SVDR, Randomized SVD

Data: A, k
Result: $\hat{U}$,Σ_k, V_k^H

1 $\Omega = \text{SketchingMatrix}(\text{ A, } \mathcal{O}(k)\text{ })$
2 $C = A\Omega$
3 $[Q, R] = qr(C)$
4 $\left[\tilde{U}, \tilde{\Sigma}, \tilde{V}\right] = svd(Q^H A)$
5 $\hat{U} = Q\tilde{U}$
6 $A_k = \hat{U}\Sigma_k V_k^H$

- Provided rounding error on SVD:

$$||A - \tilde{A}_k|| = ||A - \tilde{Q}\tilde{\Sigma}_k\tilde{V}^H|| \leq \lambda_{k+1} + \left(1 + 4\sqrt{\frac{2m}{k-1}}\right)^{\left(\frac{1}{2q+1}\right)} \lambda_{k+1}$$



Numerical experiments. Error metric

- Do not look at

$$||U - \tilde{U}_k|| < \epsilon$$

$$||V - \tilde{V}_k|| < \epsilon$$

- But instead

$$||A - \tilde{A}_k|| = (1 + \eta)\,\lambda_{k+1}$$

Spectral norm estimator

$$p_j(A) = \sqrt{\frac{||(A^H A)^j \tilde{\omega}||_2}{||(A^H A)^{j-1} \tilde{\omega}||_2}}$$

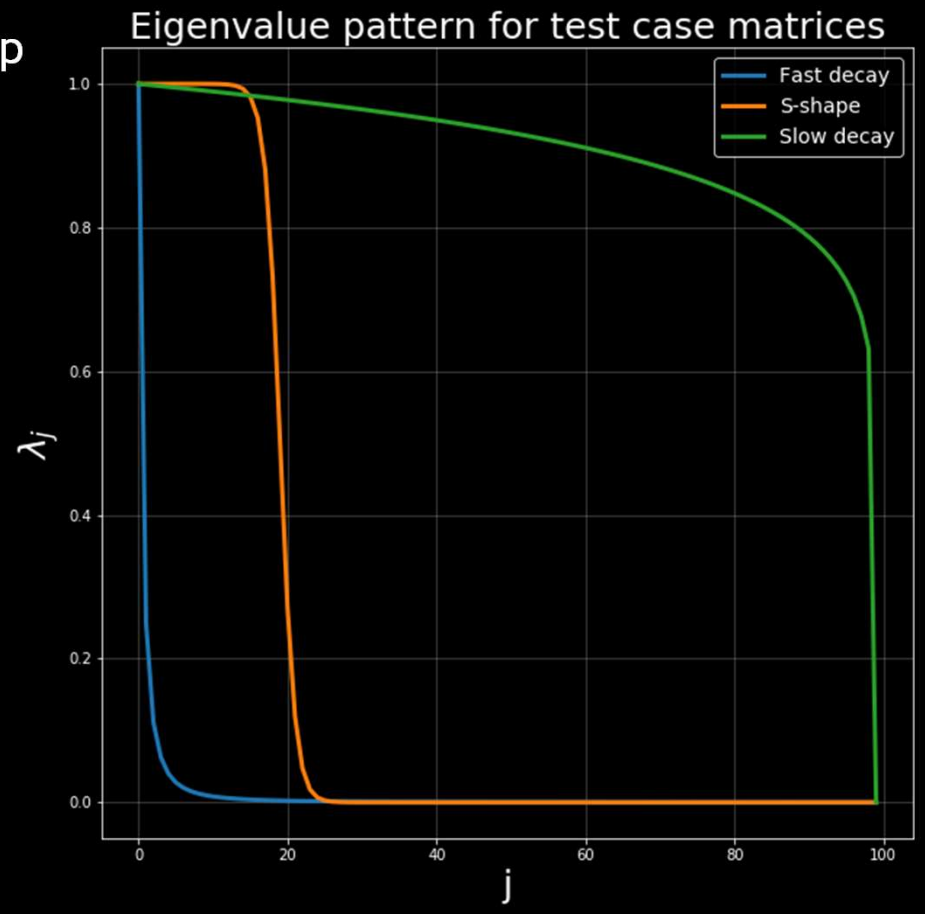
$$\mathbb{E}\left| p_j(A) \geq \frac{||A||_2}{10} \right| > 1 - 4\sqrt{\frac{n}{j-1}} 100^{-j}$$

- Bottom line: 6 iters estimate norm of A within a factor of 10.

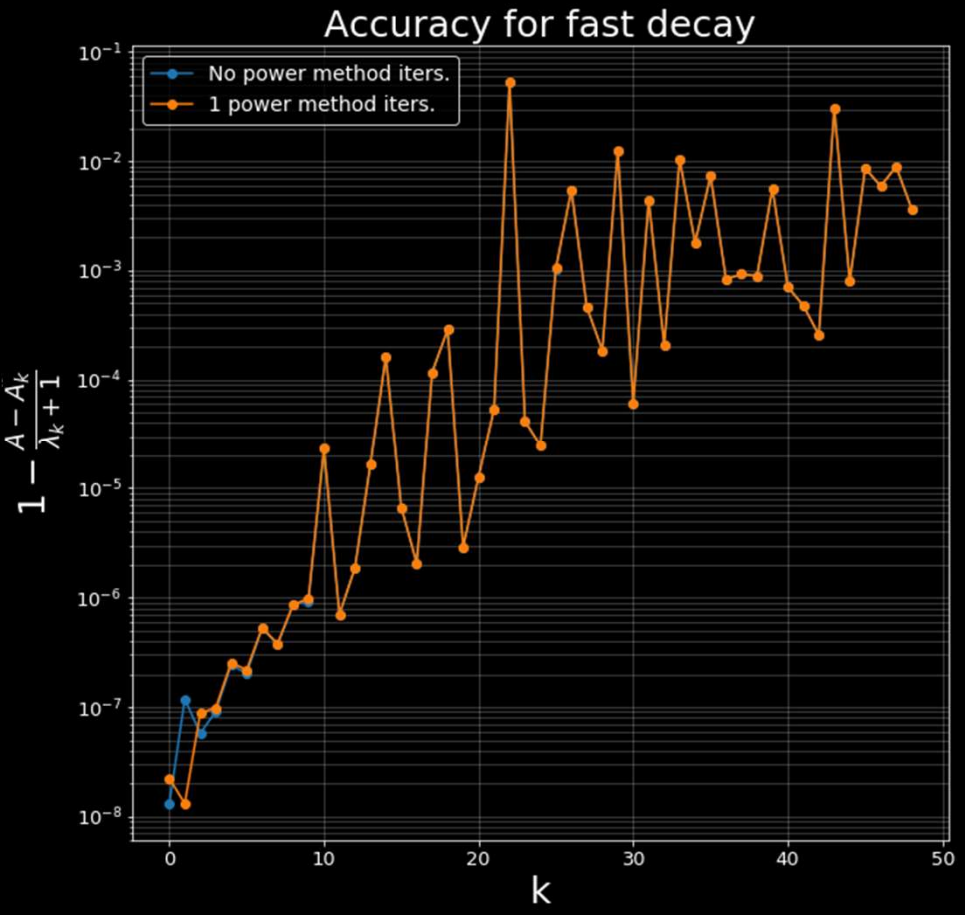
Magdon-Ismail, Malik. "A Note On Estimating the Spectral Norm of A Matrix Efficiently." *arXiv preprint arXiv:1104.2076* (2011).

Numerical experiments. Test cases.

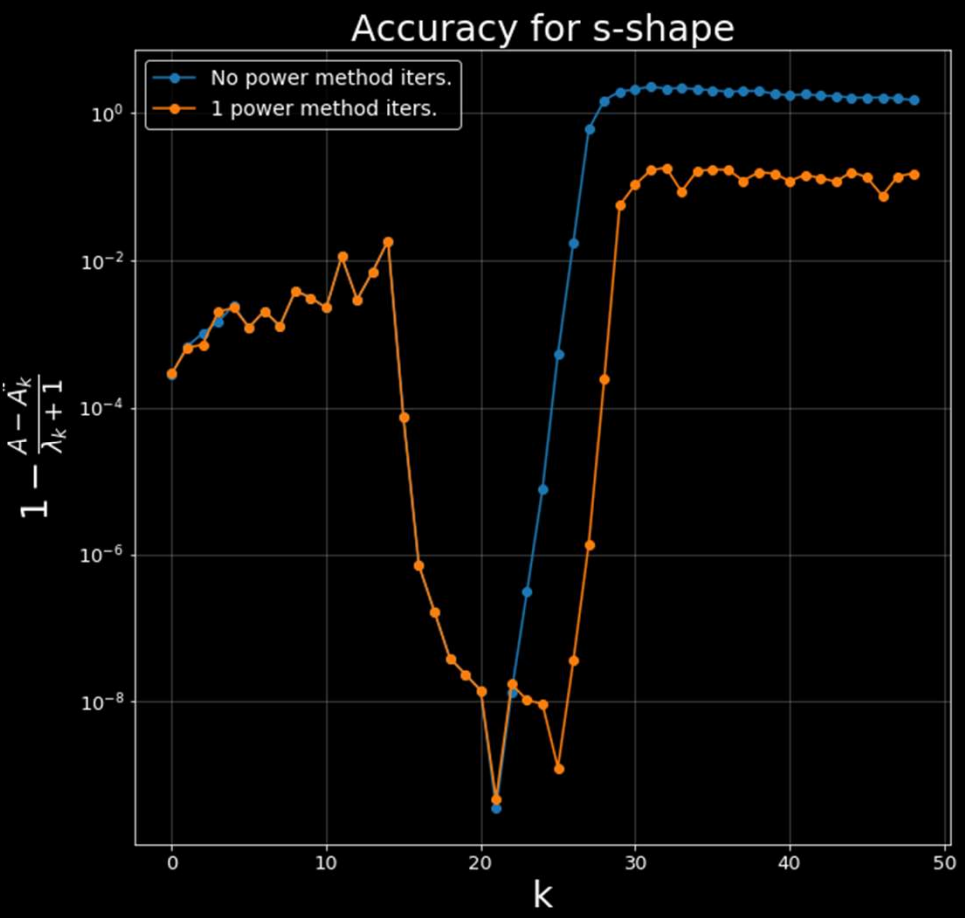
- Accuracy depends on spectral gap
- Three test cases:
  - Fast decay
  - S-shape
  - Slow decay



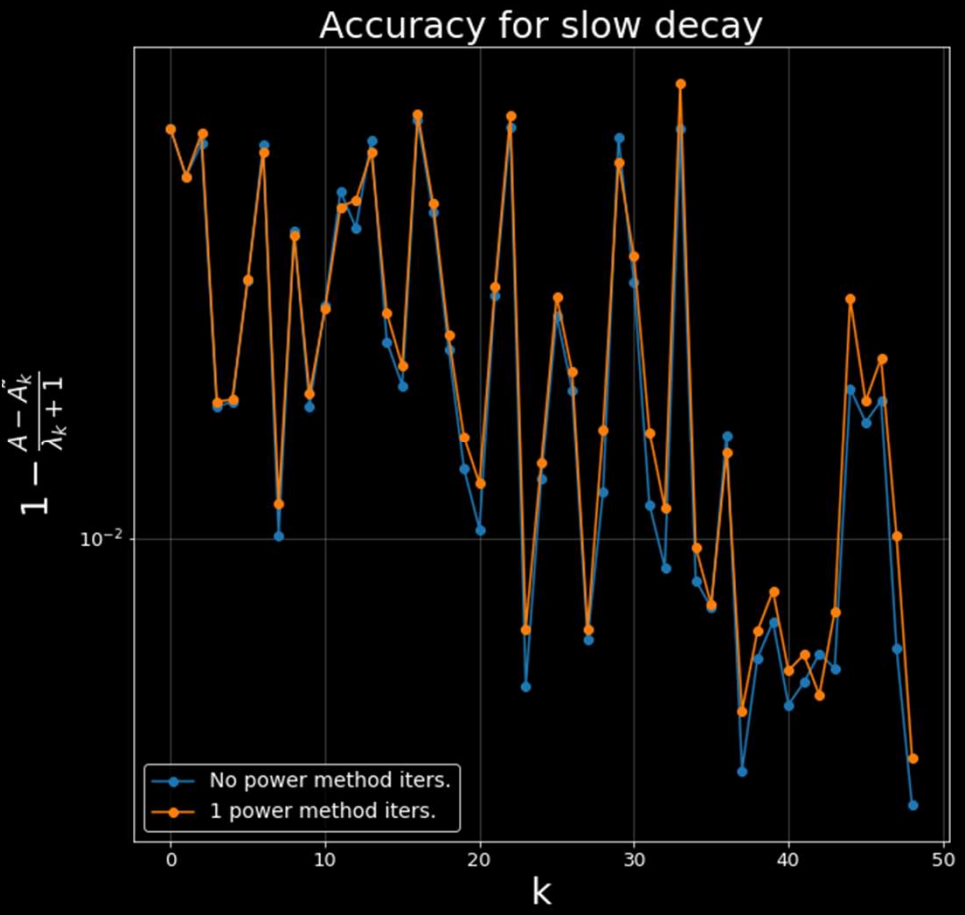
Numerical experiments. Accuracy.



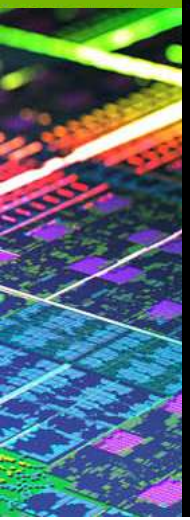
Numerical experiments. Accuracy.



Numerical experiments. Accuracy.



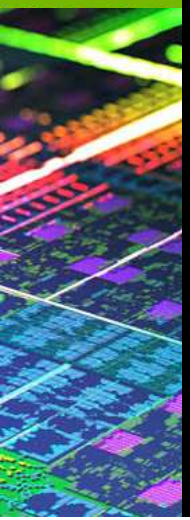
Numerical experiments. Rank-10



Speed up for RSVD over GESVD

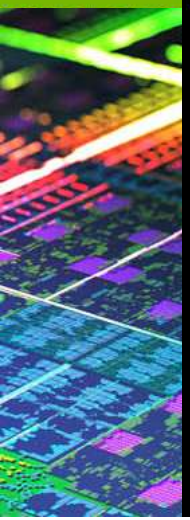
Number of rows of A \ Number of columns of A	256	512	768	1024	1280	1536	1792	2048
256	2.5	2.4	2.3	2.6	2.6	2.5	2.6	2.5
512	2.4	5.9	5.6	6.1	6.1	5.7	5.9	6.0
768	2.3	5.6	10	11	10	11	11	11
1024	2.6	6.1	11	24	22	20	22	21
1280	2.6	6.1	10	22	35	35	36	34
1536	2.5	5.7	11	20	35	53	51	50
1792	2.6	5.9	11	22	36	51	77	70
2048	2.5	6.0	11	21	34	50	70	103
2304	2.5	6.1	10	21	34	51	74	92
2560	2.6	5.6	10	21	34	47	70	103
2816	2.6	6.3	10	20	35	49	69	98
3072	2.6	5.8	11	22	34	47	70	99
3328	2.7	5.9	11	22	34	48	64	96
3584	2.6	5.8	10	22	33	48	70	94
3840	2.6	5.7	10	21	33	46	63	92
4096	2.6	6.4	10	21	31	47	65	82

Numerical experiments. Rank-20



Speed up for RSVD over GESVD

Number of rows of A \ Number of columns of A	256	512	768	1024	1280	1536	1792	2048
256	1.2	1.2	1.2	1.2	1.2	1.1	1.2	1.1
512	1.2	3.1	2.8	2.9	2.8	2.6	2.8	3.3
768	1.2	2.8	5.5	5.3	5.1	5.2	5.0	5.1
1024	1.2	2.9	5.3	10	9.8	9.1	9.6	9.3
1280	1.2	2.8	5.1	9.8	16	15	15	14
1536	1.1	2.6	5.2	9.1	15	24	21	21
1792	1.2	2.8	5.0	9.6	15	21	34	30
2048	1.1	3.3	5.1	9.3	14	21	30	43
2304	1.1	2.7	4.8	9.3	14	21	30	38
2560	1.2	2.6	4.9	9.1	14	19	29	43
2816	1.2	2.6	5.0	8.7	15	20	29	42
3072	1.2	2.7	5.3	9.7	14	19	30	41
3328	1.2	2.6	5.0	9.3	14	20	27	40
3584	1.1	2.7	5.0	9.3	13	20	28	40
3840	1.2	2.7	5.0	9.4	14	18	26	39
4096	1.2	2.9	4.7	9.1	13	19	28	34



Conclusions SVDR

- **Concern**: lack of error estimator for accuracy of spectrum
  - Bounds very pessimistic in practice.
- SVDR provides good accuracy for top-k eigenvalues.
  - Works out of the box for k<10 in practice
- Good alternative PCA, not substitute of GESVD.
  - Can get impressive performance if you know your data
- Internal research project, not included in CUDA 10.1
  - We'd love feedback from you!