MANCHESTER 1824
The University of Manchester

M\crNA
Manchester Numerical Analysis

Generating large matrices with pre-assigned 2-norm condition number

Massimiliano Fasi[1]

Numerical Linear Algebra Group Meeting
31 October 2019

[1]The University of Manchester, UK

When is a matrix large?

When I...

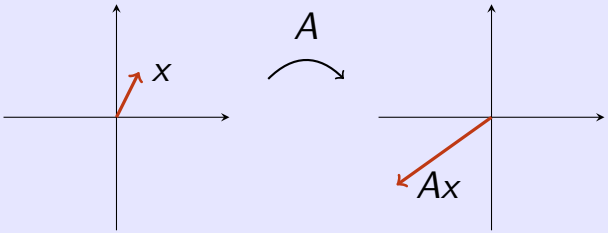
1. ...cannot compute its eigenvalues by hand.
2. ...cannot use $\mathcal{O}(n^3)$ algorithms.
3. ...cannot store it in RAM.
4. ...cannot store it in a HDD-sized RAM.

Matrices of doubles are heavy!

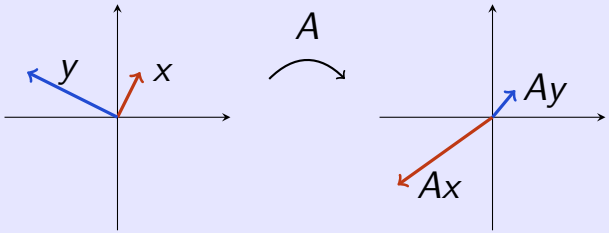
n	size (in bytes)		
10 000	762.94	MiB	Personal computer
100 000	74.51	GiB	UoM HPC Pool (1 node)
1 000 000	7.28	TiB	UoM CSF3
10 000 000	727.60	TiB	Marconi/ARCHER2

Size (in bytes) of a square matrix of order n.

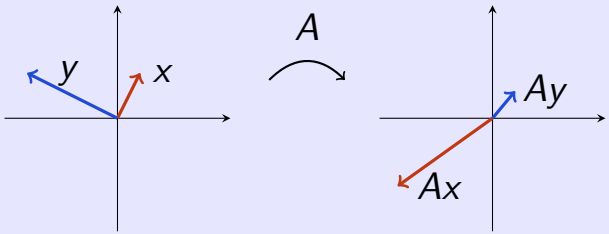
The 2-norm condition number



The 2-norm condition number



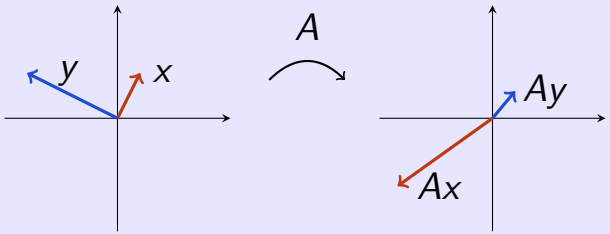
The 2-norm condition number



2-norm condition number

$$\kappa_2(A) := \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \left(\min_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \right)^{-1}$$

The 2-norm condition number

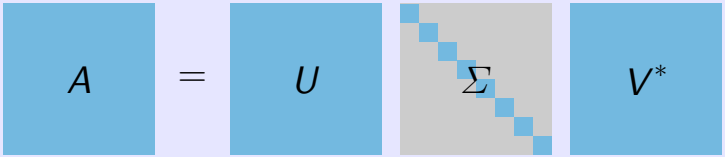


2-norm condition number

$$\kappa_2(A) := \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \gamma^{-1}, \text{ as } \gamma \to \min_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$

Extreme singular values

For any $A \in \mathbb{C}^{m \times m}$ there exist $U, V, \Sigma \in \mathbb{C}^{m \times m}$ such that

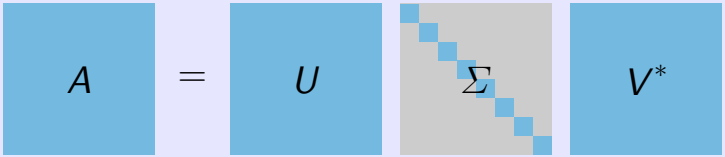


$$U^* U = I_m$$
$$V^* V = I_m$$

where $\Sigma = \mathsf{diag}(\sigma_1, \ldots, \sigma_m)$ is such that $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$.

Extreme singular values

For any $A \in \mathbb{C}^{m \times m}$ there exist $U, V, \Sigma \in \mathbb{C}^{m \times m}$ such that



$$U^* U = I_m$$
$$V^* V = I_m$$

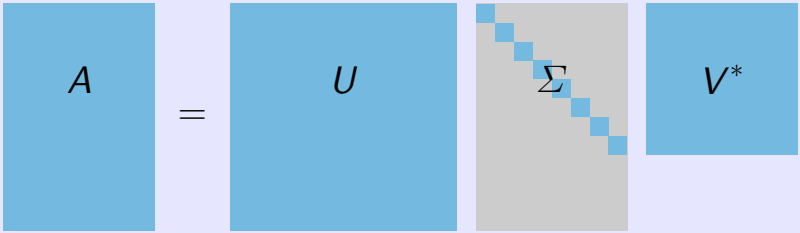
where $\Sigma = \mathsf{diag}(\sigma_1, \ldots, \sigma_m)$ is such that $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$.

Theorem

$$\kappa_2(A) = \lim_{\gamma \to \sigma_m} \sigma_1 \gamma^{-1}$$

The SVD decomposition: a more general problem

If $m > n$, we can write $A \in \mathbb{C}^{m \times n}$ as



$$U^*U = I_m$$

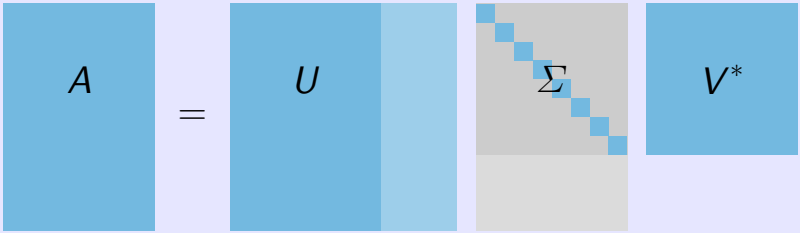
$$V^*V = I_n$$

If $m < n$, take transpose.

New problem: generating matrix with pre-assigned singular values.

The SVD decomposition: a more general problem

If $m > n$, we can write $A \in \mathbb{C}^{m \times n}$ as



$$U^*U = I_n$$

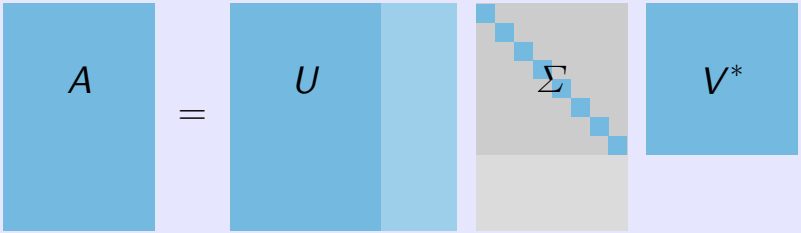
$$V^*V = I_n$$

If $m < n$, take transpose.

New problem: generating matrix with pre-assigned singular values.

The SVD decomposition: a more general problem

If $m > n$, we can write $A \in \mathbb{C}^{m \times n}$ as



$$U^*U = I_n$$

$$V^*V = I_n$$

If $m < n$, take transpose.

New problem: generating matrix with pre-assigned singular values.

Solution: using the SVD decomposition.

The RandSVD algorithm

The SVD decomposition

$$A = U\Sigma V^*, \quad U \in \mathbb{C}^{m\times m}, \Sigma \in \mathbb{C}^{m\times n}, V \in \mathbb{C}^{n\times n}$$

Produce m-by-n matrix with singular values $\sigma_1 \geq \cdots \geq \sigma_{\min\{m,n\}} \geq 0$:

1. Generate $\Sigma = \mathsf{diag}(\sigma_1, \ldots, \sigma_{\min\{m,n\}}) \in \mathbb{C}^{m\times n}$.
2. Compute $A \leftarrow (V\Sigma^T)^*$, for some random unitary $V \in \mathbb{C}^{n\times n}$.
3. Compute $A \leftarrow UA$, for some random unitary $U \in \mathbb{C}^{m\times m}$.

[Stewart, 1980], [Demmel and McKenney, 1989]

Applying a unitary transformation implicitly

Householder transformations

$$H_u = I_m - \frac{2}{u^* u} u u^* \in \mathbb{C}^{m \times m}, \qquad \text{for any } u \in \mathbb{C}^m \setminus \{0\}$$

- H_u is Hermitian ($H_u^* = H_u$) and unitary ($H_u^* H_u = H_u H_u^* = I_m$).
- H_u has at most m parameters.
- For $A \in \mathbb{C}^{m \times n}$, computing $H_u A$ requires $\sim 4mn$ flops.
- RandSVD applies $m - 1$ Householder transformation of increasing size.

Beyond Householder transformations

Assume $m > n$, and for any $\theta \in \mathbb{R}$, $u \in \mathbb{C}^n \setminus \{0\}$, $v \in \mathbb{C}^{m-n}$, define

$$W_{u,v} := \begin{bmatrix} I_n + \alpha u u^* \\ \alpha v u^* \end{bmatrix} \in \mathbb{C}^{m \times n}, \qquad \alpha = -\frac{e^{i\theta} + 1}{\|u\|_2^2 + \|v\|_2^2}.$$

- $W_{u,v}$ is rectangular.
- $W_{u,v}$ has orthonormal columns (i.e. $W_{u,v}^* W_{u,v} = I_n$).
- For $A \in \mathbb{C}^{n \times p}$, computing $W_{u,v} A$ requires $mp + 3np$ flops.

The RandSVD approach at scale

Thin SVD decomposition (m > n)

$$A = U\Sigma V^*, \quad U^*U = V^*V = I_n,$$

where $U \in \mathbb{C}^{m\times n}, \Sigma \in \mathbb{C}^{n\times n}, V \in \mathbb{C}^{n\times n}$.

1. Pick $\theta \in \mathbb{R}, u \in \mathbb{C}^n, v \in \mathbb{C}^{m-n}$.
2. Generate a matrix $Q \in \mathbb{C}^{m\times n}$ s.t. $Q^*Q = I_n$.
3. Compute $\widetilde{Q} \leftarrow Q\Sigma$.
4. Compute $y \leftarrow \widetilde{Q}u$.
5. Return $Q\Sigma W^*_{u,v} = \left[\widetilde{Q} + \overline{\alpha}yu^* \quad \overline{\alpha}yv^*\right]$.

Forward

The RandSVD approach at scale

Thin SVD decomposition (m > n)

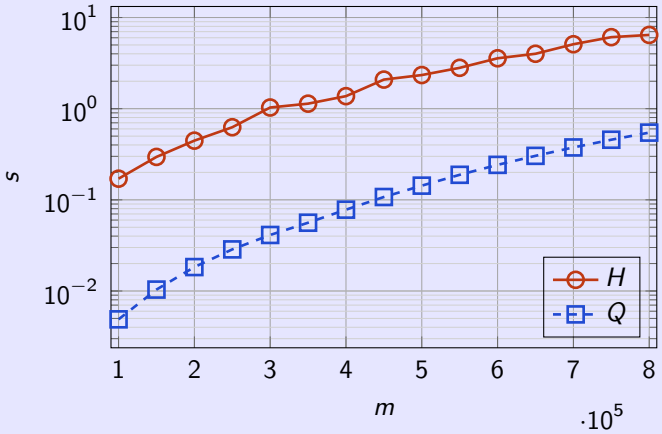
$$A = U\Sigma V^*, \quad U^*U = V^*V = I_n,$$

where $U \in \mathbb{C}^{m\times n}, \Sigma \in \mathbb{C}^{n\times n}, V \in \mathbb{C}^{n\times n}$.

1. Pick $\theta \in \mathbb{R}, u \in \mathbb{C}^n, v \in \mathbb{C}^{m-n}$.
2. Generate a unitary matrix $Q \in \mathbb{C}^{n\times n}$.
3. Compute $\widetilde{Q} \leftarrow Q\Sigma$.
4. Compute $y \leftarrow \widetilde{Q}u$.
5. Return $W_{u,v}\Sigma Q^* = \begin{bmatrix} \widetilde{Q}+\alpha uy^* \\ \alpha vy^* \end{bmatrix}$.

Backward

Generating a unitary matrix locally is faster



$$q_{ij} = \frac{2}{\sqrt{2m+1}} \sin\left(\frac{2ij\pi}{2m+1}\right)$$

$$h_{ij} = \delta_{ij} - \frac{2}{\|u\|_2} u_i u_j$$

Time to generate matrices of size m on 32 nodes (1024 cores).

The RandSVD method for the condition number

Thin SVD decomposition ($m > n$)

$$A = U\Sigma V^*, \quad U^*U = V^*V = I_n\,,$$

where $U \in \mathbb{C}^{m\times n}$, $\Sigma \in \mathbb{C}^{n\times n}$, $V \in \mathbb{C}^{n\times n}$.

1. Pick $\theta \in \mathbb{R}$, $u \in \mathbb{C}^n$, $v \in \mathbb{C}^{m-n}$.
2. Generate a matrix $Q \in \mathbb{C}^{m\times n}$ s.t. $Q^*Q = I_n$.
3. Compute $\widetilde{Q} \leftarrow Q\Sigma$.
4. Compute $y \leftarrow \widetilde{Q}u$.
5. Return $Q\Sigma W^*_{u,v} = \left[\widetilde{Q} + \overline{\alpha} y u^* \quad \overline{\alpha} y v^*\right]$.

Forward

The RandSVD method for the condition number

Thin SVD decomposition (square matrices)

$$A = U\Sigma V^*, \quad U^*U = V^*V = I_m,$$

where $U \in \mathbb{C}^{m\times m}$, $\Sigma \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{m\times m}$.

1. Pick $\theta \in \mathbb{R}$, $u \in \mathbb{C}^m$, ~~$v \in \mathbb{C}^{m-n}$~~.
2. Generate a unitary matrix $Q \in \mathbb{C}^{m\times m}$.
3. Compute $\widetilde{Q} \leftarrow Q\Sigma$.
4. Compute $y \leftarrow \widetilde{Q}u$.
5. Return $Q\Sigma W^*_{u,v} = \left[\widetilde{Q} + \overline{\alpha} y u^* \quad ~~\overline{\alpha} y v^*~~\right]$.

Forward

The RandSVD method for the condition number

Thin SVD decomposition (square matrices)

$$A = U\Sigma V^*, \quad U^*U = V^*V = I_m,$$

where $U \in \mathbb{C}^{m\times m}$, $\Sigma \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{m\times m}$.

1. Pick $\theta \in \mathbb{R}$, $u \in \mathbb{C}^m$, ~~$v \in \mathbb{C}^{m-n}$~~.
2. Generate a matrix $Q \in \mathbb{C}^{m\times m}$.
3. Compute $\widetilde{Q} \leftarrow Q\Sigma$.
4. Compute $y \leftarrow \widetilde{Q}u$.
5. Return $Q\Sigma W^*_{u,v} = \left[\widetilde{Q} + \overline{\alpha} y u^* \quad ~~\overline{\alpha} y v^*~~\right]$.

Forward

Further reducing the computational cost ($m = n$)

Assumptions:

- $Q \in \mathbb{C}^{m \times m}$ is unitary ($QQ^* = Q^*Q = I_m$) and Hermitian ($Q^* = Q$)
- u is the ℓth column of Q (i.e., $Qu = e_\ell$)

then

$$y_i = (Q\Sigma u)_i = \sum_{k=1}^{m} q_{ki}(\sigma_k - 1)q_{k\ell} + \delta_{i\ell}$$

Further reducing the computational cost ($m = n$)

Assumptions:

- $Q \in \mathbb{C}^{m \times m}$ is unitary ($QQ^* = Q^*Q = I_m$) and Hermitian ($Q^* = Q$)
- u is the ℓth column of Q (i.e., $Qu = e_\ell$)
- $\sigma_2 = \cdots = \sigma_{m-2} = 1$

then

$$y_i = (Q\Sigma u)_i = q_{1i}\,(\sigma_1 - 1)\,q_{1\ell} + q_{mi}\,(\sigma_m - 1)\,q_{m\ell} + \delta_{i\ell}$$

Further reducing the computational cost ($m = n$)

Assumptions:

- $Q \in \mathbb{C}^{m \times m}$ is unitary ($QQ^* = Q^*Q = I_m$) and Hermitian ($Q^* = Q$)
- u is the ℓth column of Q (i.e., $Qu = e_\ell$)
- $\sigma_2 = \cdots = \sigma_{m-2} = 1$
- $\sigma_1 = \sqrt{\kappa}$, $\sigma_m = \sqrt{\kappa^{-1}}$

then

$$y_i = (Q\Sigma u)_i = q_{1i}\left(\sqrt{\kappa} - 1\right) q_{1\ell} + q_{mi}\left(\sqrt{\kappa^{-1}} - 1\right) q_{m\ell} + \delta_{i\ell}$$

A faster algorithm (forward variant)

Algorithm: Generate m-by-m matrix with condition number κ.

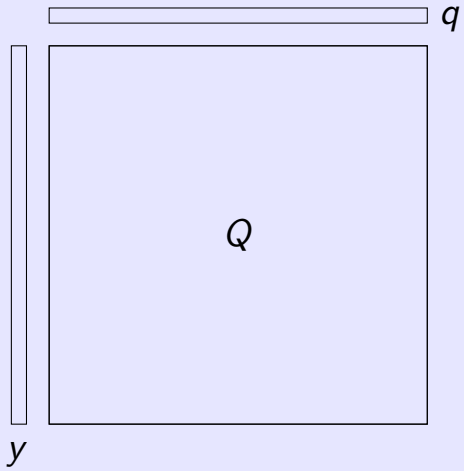
1. Initialize Q to an m-by-m unitary Hermitian matrix.
2. $\alpha \leftarrow -(\mathrm{e}^{\mathrm{i}\theta} + 1)$, for some $\theta \in \mathbb{R} \setminus \{(2k+1)\pi : k \in \mathbb{Z}\}$.
3. Choose ℓ between 1 and m.
4. **for** $i \leftarrow 1$ **to** m **do**
5. $\quad y_i \leftarrow q_{1i} q_{1\ell} (\sqrt{\kappa} - 1) + q_{mi} q_{m\ell} (\sqrt{\kappa^{-1}} - 1) + \delta_{i\ell}$
6. Multiply first column of Q by $\sqrt{\kappa}$.
7. Multiply last column of Q by $\sqrt{\kappa^{-1}}$.
8. **return** $Q + \overline{\alpha} y \mathbf{q}_\ell^*$

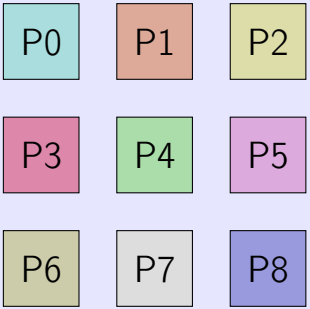
A faster algorithm (forward variant)

Algorithm: Generate m-by-m matrix with condition number κ.

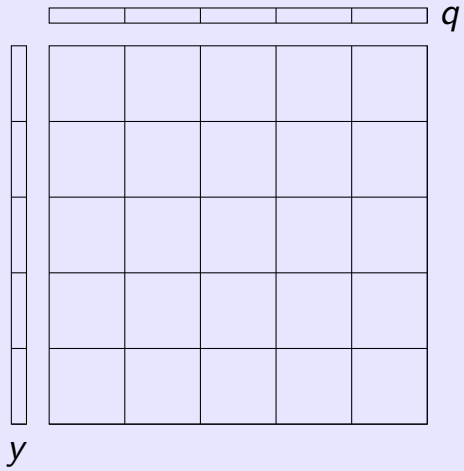
1. Initialize Q to an m-by-m unitary Hermitian matrix.
2. $\alpha \leftarrow -(\mathrm{e}^{\mathrm{i}\theta} + 1)$, for some $\theta \in \mathbb{R} \setminus \{(2k+1)\pi : k \in \mathbb{Z}\}$.
3. Choose ℓ between 1 and m.
4. **for** $i \leftarrow 1$ **to** m **do**
5. $\quad y_i \leftarrow q_{1i} q_{1\ell}(\sqrt{\kappa} - 1) + q_{mi} q_{m\ell}(\sqrt{\kappa^{-1}} - 1) + \delta_{i\ell}$
6. Multiply first column of Q by $\sqrt{\kappa}$.
7. Multiply last column of Q by $\sqrt{\kappa^{-1}}$.
8. **return** $Q + \overline{\alpha} y \mathbf{q}_\ell^*$

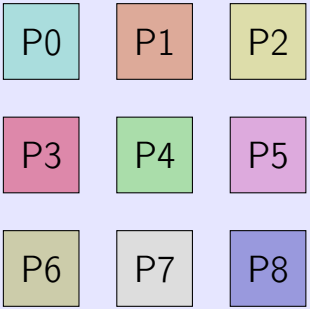
Avoiding communication at (sm)all costs



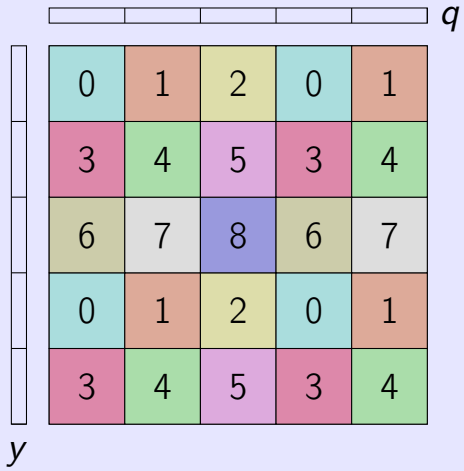


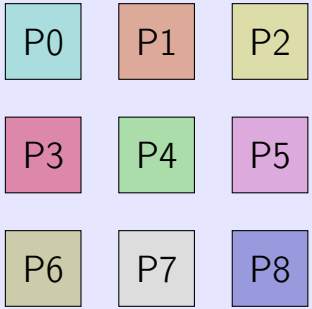
Avoiding communication at (sm)all costs



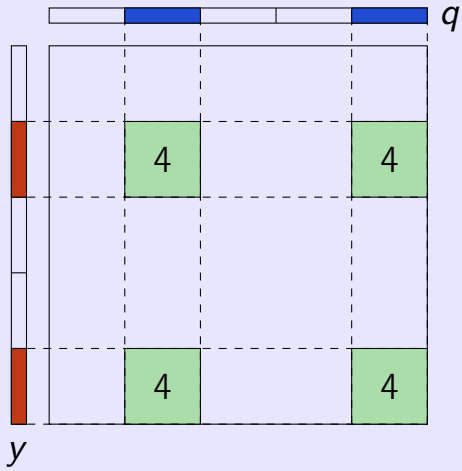


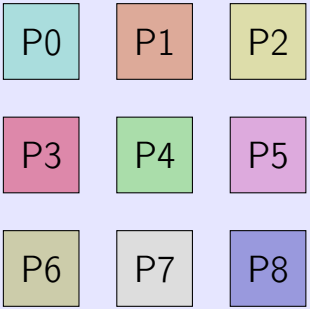
Avoiding communication at (sm)all costs





Avoiding communication at (sm)all costs





Numerical experiments

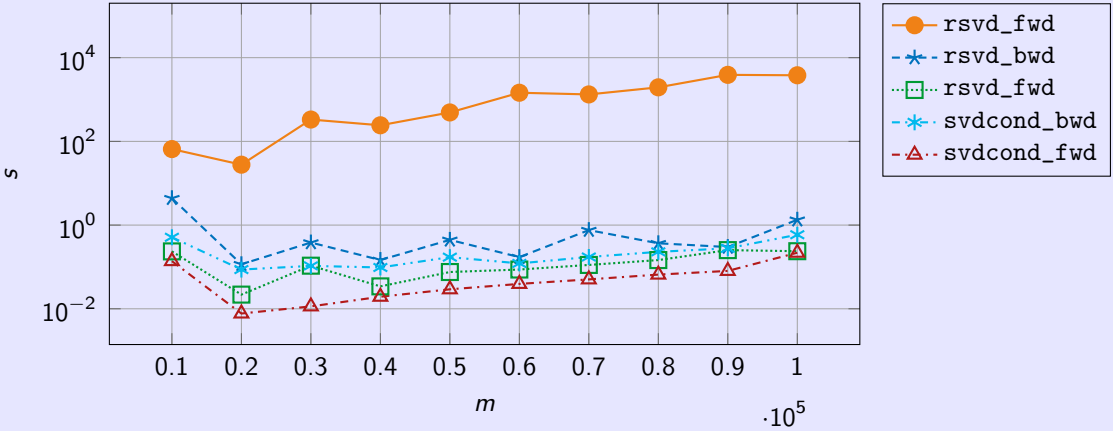
We compare five algorithms:

- `rsvd`, our implementation of ScaLAPACK's `PDLAGGE`
- `rsvd_fwd`, forward variant of large-scale RandSVD approach
- `rsvd_bwd`, backward variant of large-scale RandSVD approach
- `svdcond_fwd`, communication-avoiding forward variant
- `svdcond_bwd`, communication-avoiding backward variant

Libraries: Intel MKL Library 18.0.3, Open MPI 3.1.4

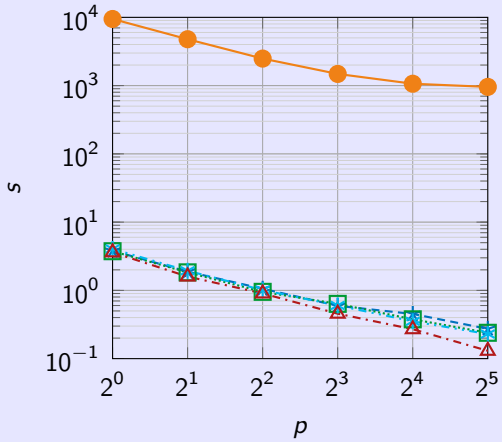
Nodes: 2 $\times$ 16-core Intel Xeon Gold 6130 @ 2.10GHz, 187.3GiB of RAM

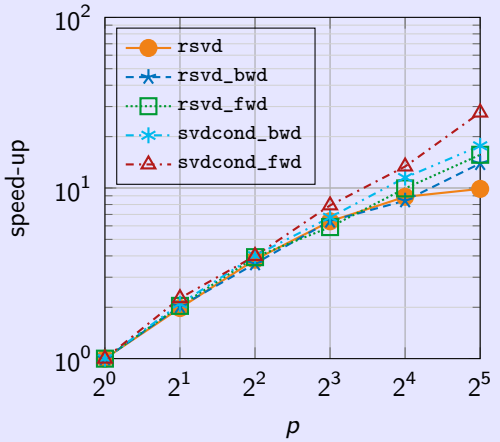
New algorithms are faster



Execution time to generate matrices of increasing size m on 1024 cores.

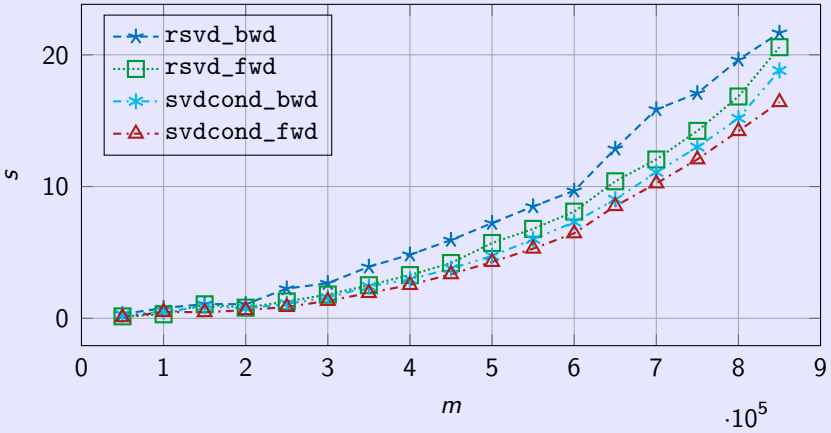
New algorithms are better-suited to parallel settings





Execution time (left) and speedup (right) to generate matrices of size $5 \cdot 10^5$ on p cores.

New algorithms scale nicely



Execution time (in seconds) to generate matrices of increasing size m on 1024 cores.

Conlusions

- The RandSVD algorithm is ill-suited to HPC environments.
- By tweaking the idea, we obtain algorithms that:
  - do not require any communication
  - are much faster in **distributed** as well as **shared** memory environments
  - scale to supercomputer-sized problems