Low-rank Anderson Acceleration

Daniel Appelö & Yingda Cheng
GPU contributions by Måns Andersson

Research supported by NSF DMS-2208164, NSF DMS-2436319, U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under Award Number DE-SC0025424, and Virginia Tech.

VIRGINIA TECH

Outline:

- Low-rank Anderson Acceleration
  - Matrix and Tucker tensors applications with PDEs
- CrossDEIM - Cross^2-DEIM
  - Handling nonlinearities when using low rank formats

VIRGINIA TECH

Low-rank numerical methods for matrix /tensor differential equations

- **Linear matrix equations**
  - A well-studied topic in numerical linear algebra, see Simonicini, SIAM review, 2016
- **Nonlinear matrix equations (PDE context)**
  - less studied than full-rank so far
  - Newton + (low-rank)TT-GMRES, Adak et al, 2024, Rogers, Venturi, 2024
  - Riemannian optimization, Sutti, Vandereycken, 2024
  - Dektor collocation, 2024
  - Sparse residual collocation, Naderi, Akhavan, Babaee, 2024
  - Etc… (people in this audience …)

VIRGINIA TECH

Why Anderson Acceleration?

Spatial discretization of a nonlinear (elliptic or stiff) PDE gives a nonlinear matrix or tensor equation

$$G(X) = X, \qquad X \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d},$$

where $G(X)(i_1, i_2, \ldots, i_d)$ is a nonlinear function of X and its grid neighbors.

$u_{xx} + u_{yy} + \lambda e^u = 0$, Bratu,
$u_{xx}u_{yy} - u_{xy}^2 = f$, Elliptic Monge—Ampere,
$u_t = \nu \Delta u + u - u^3$, Allen—Cahn.

The solution is (approximately) low rank for diffusion-dominated problems but nonlinearity is the obstacle to staying in low-rank form.
Sampling the G is helped by X and should be easier to sample than the Jacobian

VIRGINIA TECH

Anderson Acceleration

Algorithm Unconstrained variant of Anderson acceleration in $\mathbb{R}^n$.

Input: $x_0 \in \mathbb{R}^n$, memory parameter $\hat{m} \geq 1$.

Output: $x_k \in \mathbb{R}^n$ as an approximate solution to $x = g(x)$. Fixed point problem

$x_1 = g(x_0)$.

for $k = 1, 2, \ldots$ until convergence **do**

$\hat{m}_k = \min(\hat{m}, k)$.

Multi - secant approximation

Set $D_k = (\Delta f_{k-\hat{m}_k}, \ldots, \Delta f_{k-1})$, where $\Delta f_i = f_{i+1} - f_i$ and $f_i = g(x_i) - x_i$.

Solve $\gamma^{(k)} = \operatorname{argmin}_{v \in \mathbb{R}^{\hat{m}_k}} \|D_k v - f_k\|$, $\gamma^{(k)} = (\gamma_0^{(k)}, \ldots, \gamma_{\hat{m}_k - 1}^{(k)})^T$.

$x_{k+1} = g(x_k) - \sum_{i=0}^{\hat{m}_k - 1} \gamma_i^{(k)} \left[g(x_{k-\hat{m}_k+i+1}) - g(x_{k-\hat{m}_k+i}) \right]$.

end for Least squares approximation gives new iterate

Needs to be changed to make this into a low rank method?

VIRGINIA TECH

Example low-rank Bratu problem

$$\Delta X + \lambda e^X = 0$$

Low rank format

Interested in solving $USV^T = \sum_{k=1}^{r} \sigma_k u_k v_k^T \approx X = G(X)$, where, e.g.

Richardson iteration
With preconditioning

$$X^{k+1}(i,j) = G(i,j;X^k,\alpha) \equiv X^k(i,j) + \alpha M(G_{\mathrm{B}}(i,j;X^k)),$$

Point-wise nonlinearity

$$G_{\mathrm{B}}(i,j;X) = \frac{1}{h_x^2}\left(X(i+1,j) - 2X(i,j) + X(i-1,j)\right)$$
$$+ \frac{1}{h_y^2}\left(X(i,j+1) - 2X(i,j) + X(i,j-1)\right) + \lambda e^{X(i,j)},$$

Standard discretization that must only be sampled a few rows and columns at a time

VIRGINIA TECH

lrAA : low-rank Anderson Acceleration

Algorithm lrAA for nonlinear matrix equation $G(X) = X$.

Input: $X_0 = U_0 S_0 (V_0)^T$, memory parameter $\hat{m} \geq 1$, scheduling parameter $\theta \in (0, 1)$, tolerance TOL.
Output: Approximate solution X_k to the fixed point problem $G(X) = X$ in its SVD form.

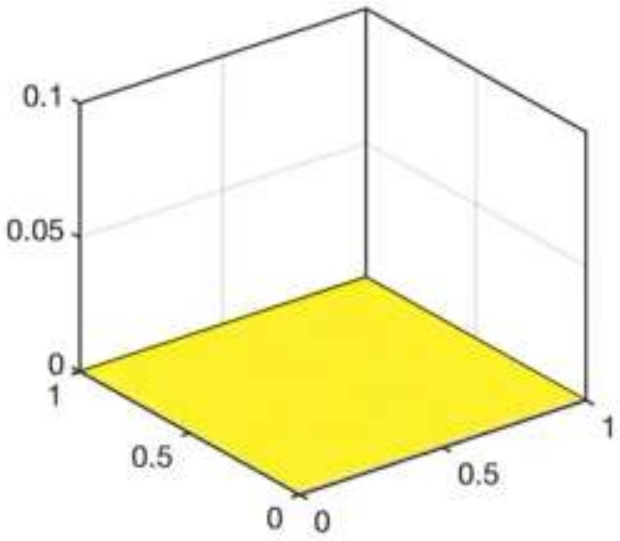
$\epsilon_G = 10^{-2}$ # Choose ϵ_G so that G_0 has low rank.
$X_1 = G_0 = \textbf{Cross-DEIM}(G(X_0), U_0, V_0, \epsilon_G, r_{\max})$.
$\rho_0 = \|X_1 - G_0\|$.
for $k = 1, 2, \ldots$ **do**
  $G_k = \textbf{Cross-DEIM}(G(X_k), U_k, V_k, \epsilon_G, r_{\max})$.
  $\rho_k = \|G_k - X_k\|$.
  $\hat{m}_k = \min(\hat{m}, k)$.
  Solve least square problem to get γ^k
  $X_{k+1} = \textbf{Cross-DEIM}(G_k - \sum_{i=0}^{\hat{m}_k - 1} \gamma_i^{(k)} \left[G_{k-\hat{m}_k+i+1} - G_{k-\hat{m}_k+i} \right], U_k, V_k, \epsilon_G, r_{\max})$.
  Set $\epsilon_G = \theta \rho_k$.
  if $\rho_k <$ TOL **then**
    Exit and return X_{k+1}.
  end if
end for

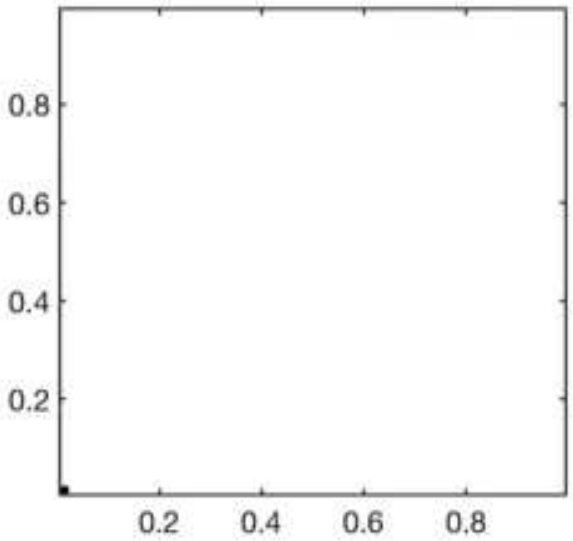
Rank truncating operations

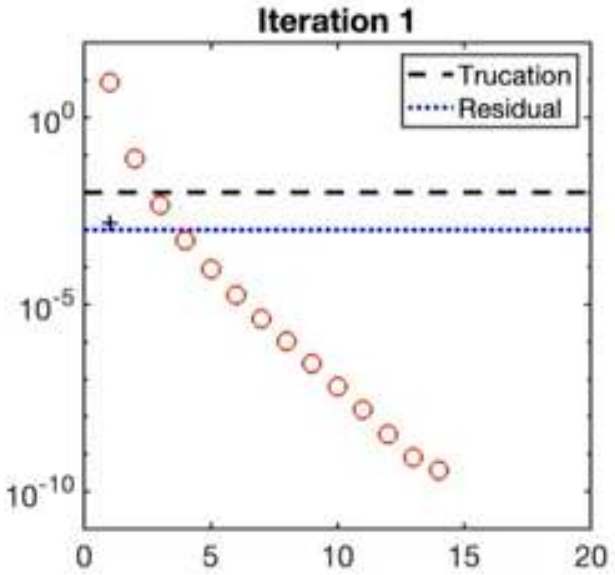
Scheduling the truncation

VIRGINIA TECH

Bratu problem $\Delta X + \lambda e^X = 0$







Observations:

- Rank is gradually increased
- Finds large singular values first

VIRGINIA TECH

Cross approximation

$$G \approx G(:, \mathcal{J}) G(\mathcal{I}, \mathcal{J})^{+} G(\mathcal{I}, :) = Q_1 R_1 G(\mathcal{I}, \mathcal{J})^{+} R_2^T Q_2^T = U S V^T$$

How to chose rows $\mathcal{I}$ and columns $\mathcal{J}$?

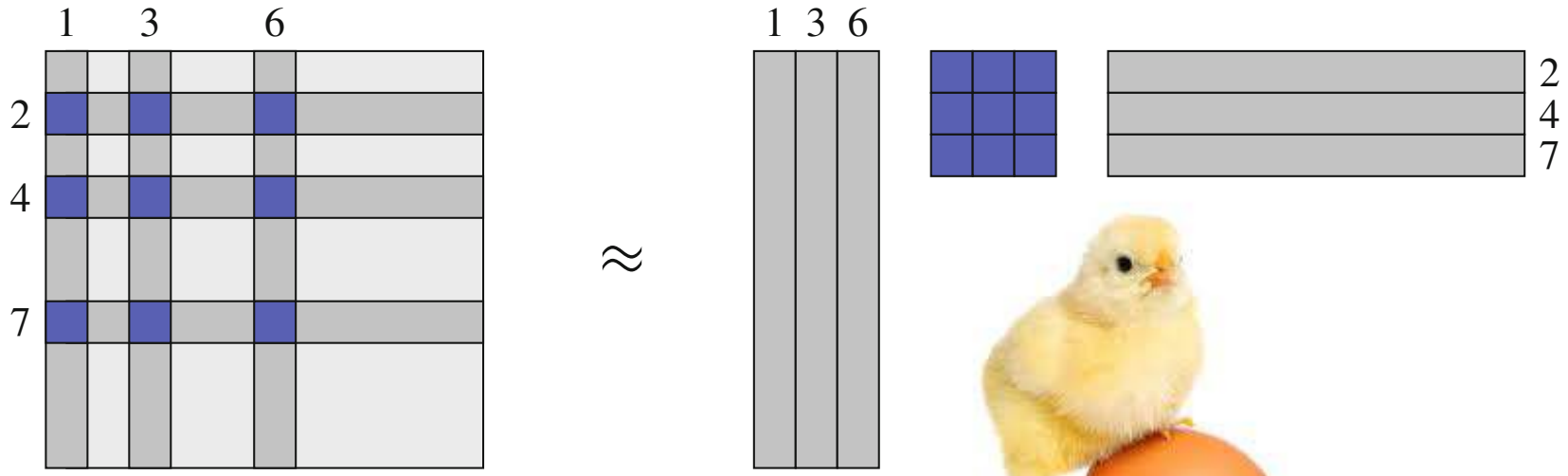
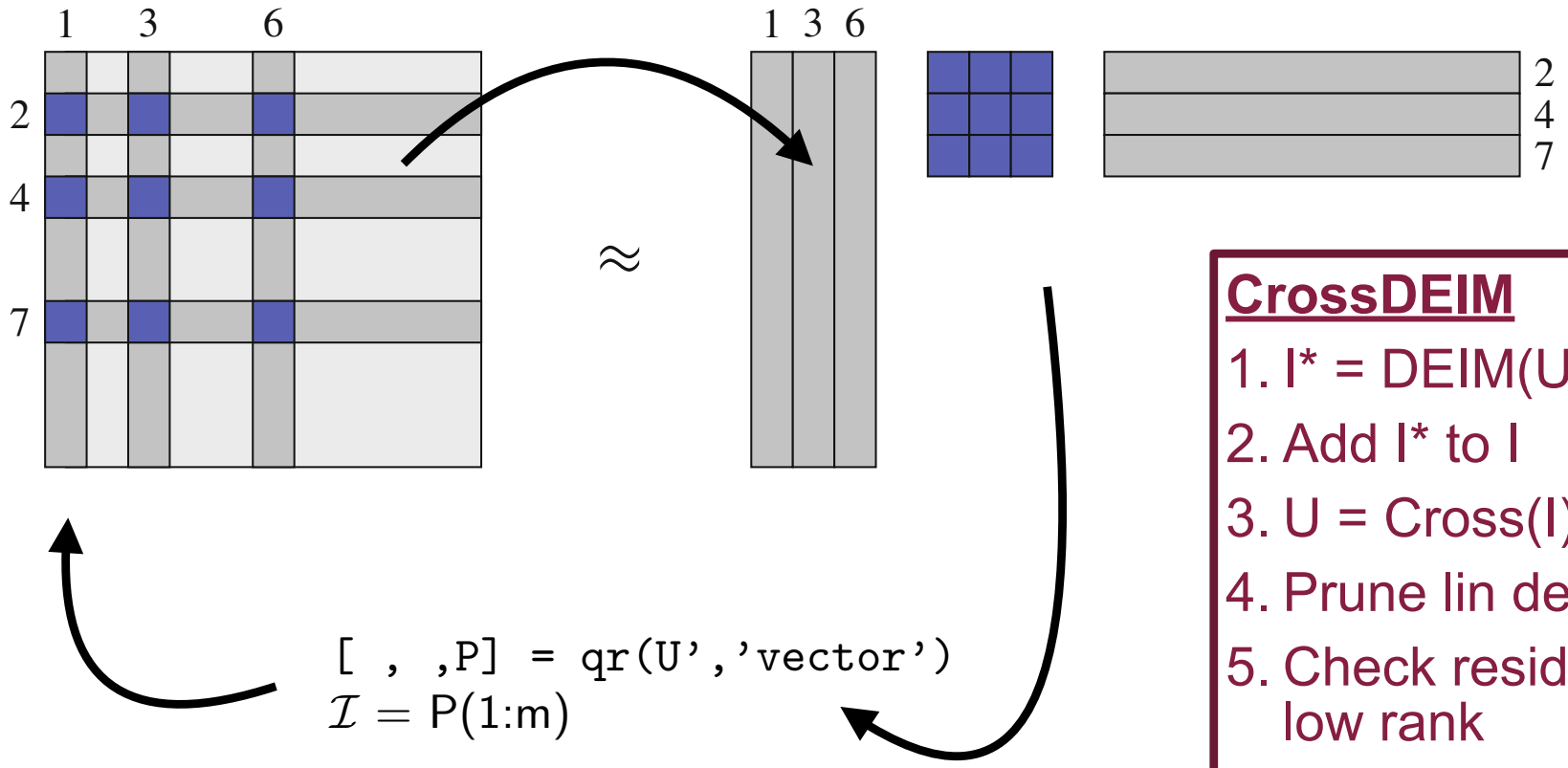


Figure from J. Ballani and D. Kressner Matrices with Hierarchical Low-Rank Structures

VIRGINIA TECH

CrossDEIM - iterate Cross-approx and DEIM



CrossDEIM

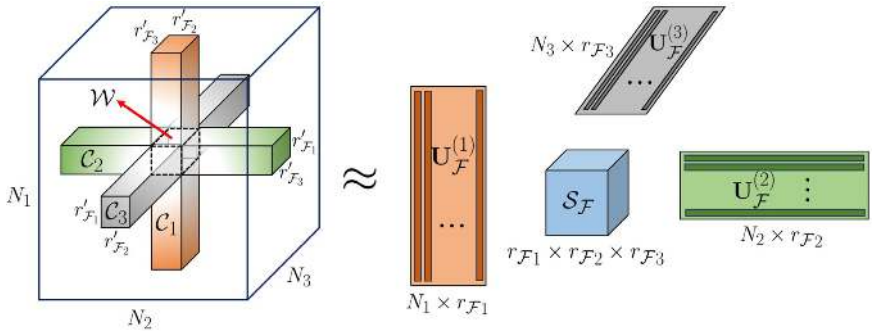
1. I* = DEIM(U)
2. Add I* to I
3. U = Cross(I)
4. Prune lin dep
5. Check residual and low rank

Daniel Appelö & Yingda Cheng, **IrAA: Low-Rank Anderson Acceleration**, arXiv:2503.03909.

VIRGINIA TECH

Tucker tensors

DEIM-FS a FSTD-1 style cross approximation for Tucker tensors



Matrices of size nr^2

Orthogonalization cost nr^4

In d dimensions $nr^{(d-1)}$ and $nr^{2(d-1)}$

Algorithm 1: DEIM-FS Tucker tensor low-rank approximation

Input:
$\tilde{\mathbf{U}}_{\mathcal{F}}^{(i)}$: matrix of exact or approximate left singular vectors of $\mathcal{F}_{(i)}$.
$r_{\mathcal{F}_i}$: target Tucker rank.
$\mathcal{F}$: function handle to compute fibers of the target $\mathcal{F}$
Output: $\mathcal{S}_{\mathcal{F}}$, $\mathbf{U}_{\mathcal{F}}^{(1)}$, $\mathbf{U}_{\mathcal{F}}^{(2)}$, $\mathbf{U}_{\mathcal{F}}^{(3)}$

1 $\mathbf{p}_i = \mathtt{DEIM}(\tilde{\mathbf{U}}_{\mathcal{F}}^{(i)})$,

2 $\mathbf{C}_1 = \Big(\mathcal{F}(:, \mathbf{p}_2, \mathbf{p}_3)\Big)_{(1)}$,
$\mathbf{C}_2 = \Big(\mathcal{F}(\mathbf{p}_1, :, \mathbf{p}_3)\Big)_{(2)}$,
$\mathbf{C}_3 = \Big(\mathcal{F}(\mathbf{p}_1, \mathbf{p}_2, :)\Big)_{(3)}$

3 $[\mathbf{U}_{\mathcal{F}}^{(i)}, \sim, \sim] = \mathtt{SVD}(\mathbf{C}_i, r_{\mathcal{F}_i})$

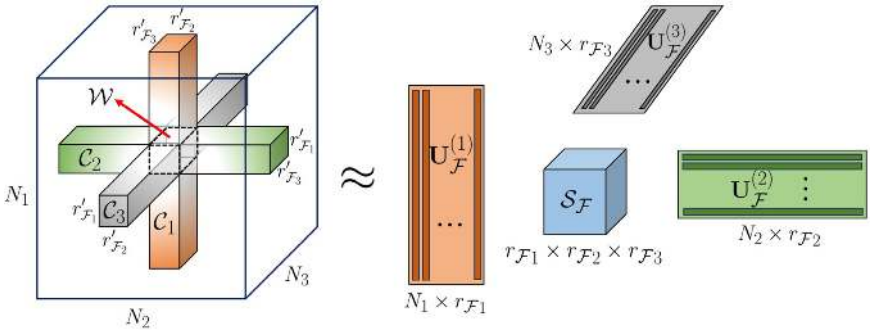
4 $\mathcal{W} = \mathcal{F}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$

5 $\mathcal{S}_{\mathcal{F}} = \mathcal{W} \times_1 \mathbf{U}_{\mathcal{F}}^{(1)}(\mathbf{p}_1, :)^{\dagger} \times_2 \mathbf{U}_{\mathcal{F}}^{(2)}(\mathbf{p}_2, :)^{\dagger} \times_3 \mathbf{U}_{\mathcal{F}}^{(3)}(\mathbf{p}_3, :)^{\dagger}$

Behzad Ghahremani and Hessam Babaee. A DEIM Tucker tensor cross algorithm and its application to dynamical low-rank approximation. *Computer Methods in Applied Mechanics and Engineering*, 423:116879, 2024.

VIRGINIA TECH

Cross²-DEIM an O(nr²) cross approximation for Tucker tensors



Algorithm 2: $[G, U_1^{(1)}, \ldots, U_d^{(1)}]$ = C2Di$(X, U_1^{(0)}, \ldots U_d^{(0)})$
Cross²-DEIM approximation for Tucker tensor (inner algorithm)

Input: Tensor $X \in \mathbb{R}^{n_1 \times n_2 \times \cdots n_d}$ and its approximate orthonormal factor matrices $U_i^{(0)} \in \mathbb{R}^{n_i \times r_i}, i = 1, \ldots d$.

Output: Tucker decomposition $X \approx G \times_1 U_1^{(1)} \times_2 U_2^{(1)} \cdots \times_d U_d^{(1)}$ with $G \in \mathbb{R}^{r_1 \times r_2 \times \cdots r_d}$, and orthonormal factor matrices $U_i^{(1)} \in \mathbb{R}^{n_i \times r_i}, i = 1, \ldots d$.

$r_i = \texttt{size}(U_i^{(0)}, 2),\ i = 1, \ldots, d$

$\mathcal{I}_i = \texttt{QDEIM}(U_i^{(0)}),\ i = 1, \ldots, d$

$\mathcal{I}_i^{\text{os}} = \texttt{gpode}(U_i^{(0)}, r_i + 3), \quad i = 1, \ldots, d$

$W = X(\mathcal{I}_1, \ldots, \mathcal{I}_d)$

$W^{\text{os}} = X(\mathcal{I}_1^{\text{os}}, \ldots, \mathcal{I}_d^{\text{os}})$

for $i = 1, \ldots, d$ **do**

  $W_{(i)} = \texttt{core\_reshape}(W, r_i, R_{\neq i})$

  $[Z_i, \sim, \sim] = \texttt{svd}(W_{(i)}^T, \texttt{"econ"})$

  $\mathcal{J}_{\neq i} = \texttt{QDEIM}(Z_i)$

  $\mathcal{I}_{\neq i} = J(\mathcal{J}_{\neq i})$

  $C_{(i)} = \texttt{reshape}(X([\mathcal{I}_{\neq i}]_1, \ldots, \mathcal{N}_i, \ldots, [\mathcal{I}_{\neq i}]_d), n_i, r_i)$

  $[U_i^{(1)}, \sim, \sim] = \texttt{svd}(C_{(i)}, \texttt{"econ"})$

end for

$G = W^{\text{os}} \times_1 [U_1^{(1)}(\mathcal{I}_1^{\text{os}}, :)]^+ \times_2 [U_1^{(1)}(\mathcal{I}_2^{\text{os}}, :)]^+ \cdots \times_d [U_d^{(1)}(\mathcal{I}_d^{\text{os}}, :)]^+$

Return $G, U_1^{(1)}, \ldots U_d^{(1)}$.

QDEIM indices from factors

Sample core

Unfold core
Orthogonalize
QDEIM index selection
Map back to original indices

Carry out Cross approximation on the reduced index
The factors are size n_i x r_i

Least squares cross

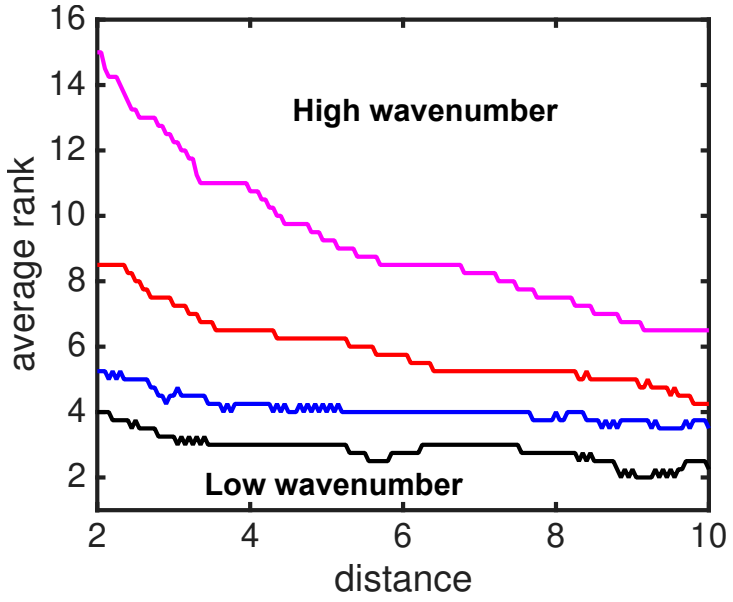
- Daniel Appelö & Yingda Cheng, **A new cross approximation for Tucker tensors and its application in Tucker-Anderson Acceleration**, arxiv.

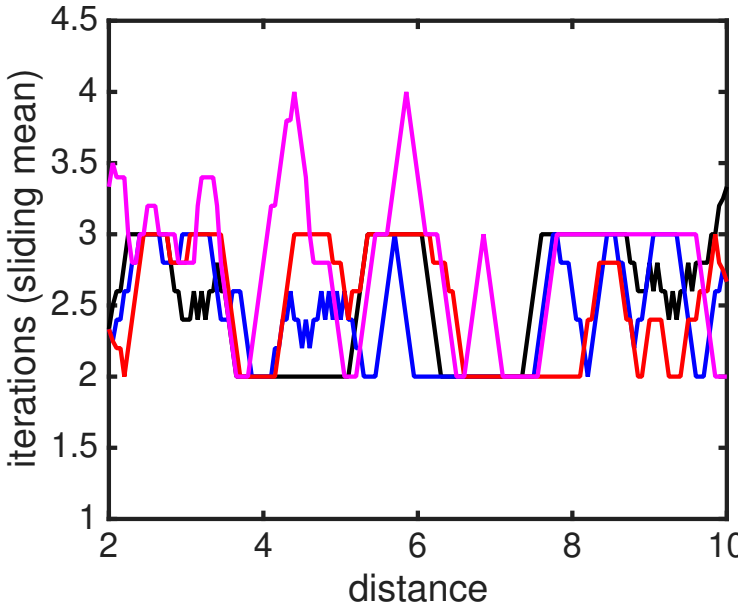
VIRGINIA TECH

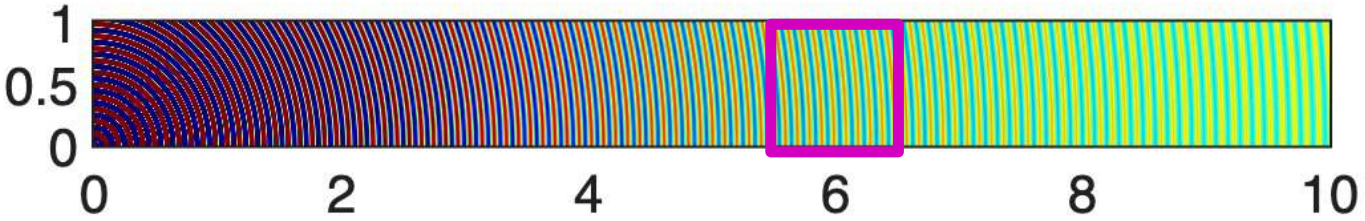
Cross^2-DEIM for function compression 4D

Sampling Helmholtz Green's function in 4D for different wave numbers

Place a 4D unit cube at x = 10 and move to the left







VIRGINIA TECH

Sublinear direct Poisson solver for [1,-2,1]^d stencil

$$\Delta v(x_1, x_2, x_3) = -3e^{-(x_1^2+x_2^2+x_3^2)}, \quad (x_1, x_2, x_3) \in [-2\pi, 2\pi]^3$$

Right hand side

$$F = G_F \times_1 U_1 \times_2 U_2 \times_3 U_3$$

Apply eigenvectors by fast sine transform

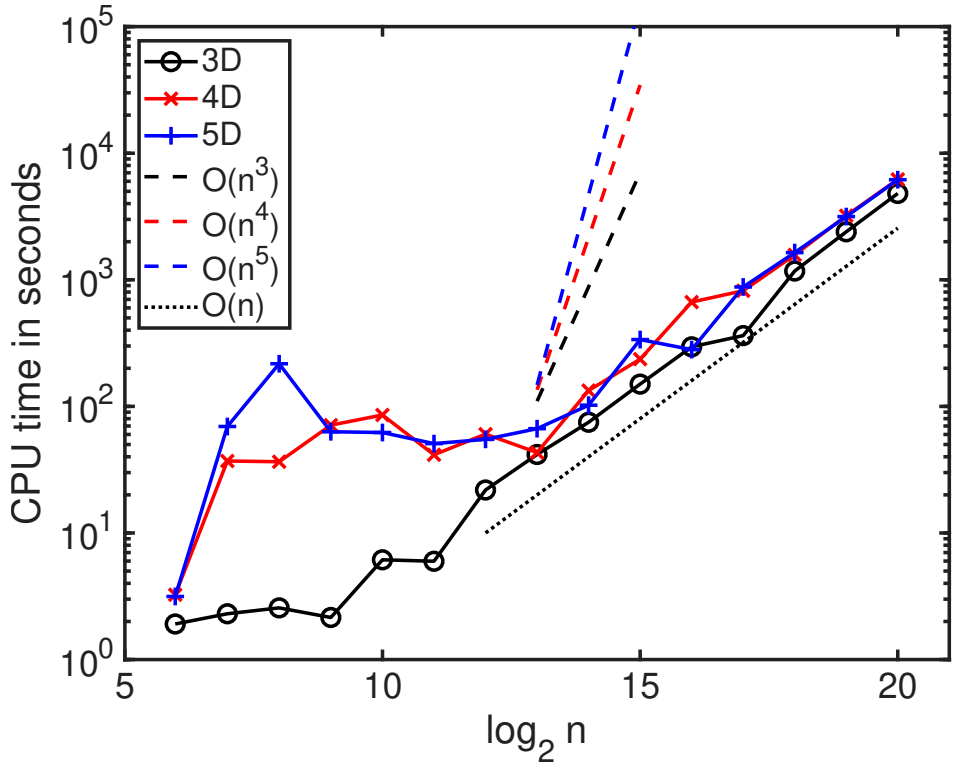
$$\hat{F} = G_F \times_1 \sqrt{\frac{2}{n+1}}\mathtt{dst}(U_1) \times_2 \sqrt{\frac{2}{n+1}}\mathtt{dst}(U_2) \times_3 \sqrt{\frac{2}{n+1}}\mathtt{dst}(U_3)$$

Sample on the transform side with Cross^2-DEIM

$$\hat{V}(i_1, i_2, i_3) = \frac{-h^2\hat{F}(i_1, i_2, i_3)}{\lambda_{i_1} + \lambda_{i_2} + \lambda_{i_3}}. \qquad \lambda_j = 2(1 - \cos(\frac{\pi j}{n+1}))$$

Obtain solution by fast sine transform

$$V = G_V \times_1 \sqrt{\frac{2}{n+1}}\mathtt{dst}(\hat{U}_1) \times_2 \sqrt{\frac{2}{n+1}}\mathtt{dst}(\hat{U}_2) \times_3 \sqrt{\frac{2}{n+1}}\mathtt{dst}(\hat{U}_3).$$



VIRGINIA TECH

Bringing it to the GPU: PyTorch + JIT

- `@torch.compile` = TorchDynamo (graph IR) + TorchInductor (fuse → optimized kernels).
- Wins: less interpreter overhead, kernel **fusion**, fewer launches, CUDA Graphs.
- Gotcha: QDEIM needs a **pivoted QR** (absent in PyTorch) ⇒ called via JAX/MAGMA, wrapped in `@torch.compiler.disable` (graph breaks).
- Size-dependent control flow ⇒ more graph breaks.
- Reproducible timings: fixed RNG seed + warm-up so JIT cost is excluded.

Library	Ver.	Role
PyTorch	2.11	arrays + calls
JAX	0.9.2	arrays + calls
cuSOLVE	13.0	GPU LA
MAGMA	2.6.1	GPU LA
MKL	2024.2	CPU LA

Hardware: NVIDIA **A100** (40 GB) GPU; 48-core **AMD EPYC 9454** CPU (ARC@VT).

How to pick rows & columns

① DEIM — pivoted LU

$$[\sim, \sim, P] = \mathrm{lu}(U), \quad \mathcal{I} = P^T I.$$

Indices ordered by importance; LU slightly cheaper than QR.

② QDEIM — pivoted QR

$$[\sim, \sim, p] = \mathrm{qr}(U^T), \quad \mathcal{I} = p.$$

Sharper error bound; preferred in prior (non-GPU) work.

③ Leverage Scores — *randomized*

$$\pi_i = \frac{1}{k} \sum_{\eta=1}^{k} (u_i^\eta)^2$$

$$p_j = \min\{1, c\,\pi_j\},$$

c=O$\left(\frac{k \log k}{\epsilon^2}\right)$ No pivoted factorization $\Rightarrow$ not quadratic in index-set size.

LS picks *worse* indices but **much faster**
Cross-DEIM's adaptive, incremental index growth hides the lower quality.

Full-rank vs. low-rank cost

Full-rank

$$\Lambda\hat{W}+\hat{W}\Lambda = h^2\hat{F}, \qquad \hat{W}(i,j) = \frac{h^2\hat{F}(i,j)}{\lambda_i+\lambda_j}.$$

cost: $\mathcal{O}\big(N_{\mathrm{DOF}} \log N_{\mathrm{DOF}}^{1/2}\big)$.

Low-rank (Cross-DEIM on the divide)

$$\hat{F} = \mathsf{DST}(U_F)\, S_F\, (\mathsf{DST}(V_F))^T \quad [\mathcal{O}(rn\log n)]$$

Apply Cross-DEIM directly to
$\hat{W}(i,j) = h^2\hat{F}(i,j)/(\lambda_i+\lambda_j)$ — never form $\hat{W}$.

$$W = \mathsf{DST}(\hat{U})\, S\, (\mathsf{DST}(\hat{V}))^T.$$

$$\boxed{\mathcal{O}\left(N_{\mathrm{DOF}}^{\frac{1}{2}}\left[\, r^2 + r\log N_{\mathrm{DOF}}^{\frac{1}{2}}\,\right]\right)} \qquad (r<\sqrt{n})$$

Poisson: wall-clock time of the full application

10 seeds; **solid** $=$ **GPU**, **dashed** $=$ **CPU** (mean $\pm$ std). GPU wins for large problems — shorter time-to-solution & better scaling. QDEIM & LS beat DEIM (edge to LS).

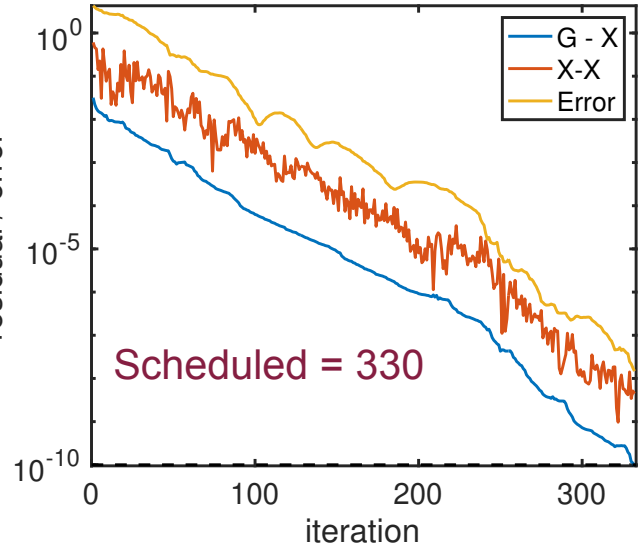
Sub-linear scaling, confirmed on the A100

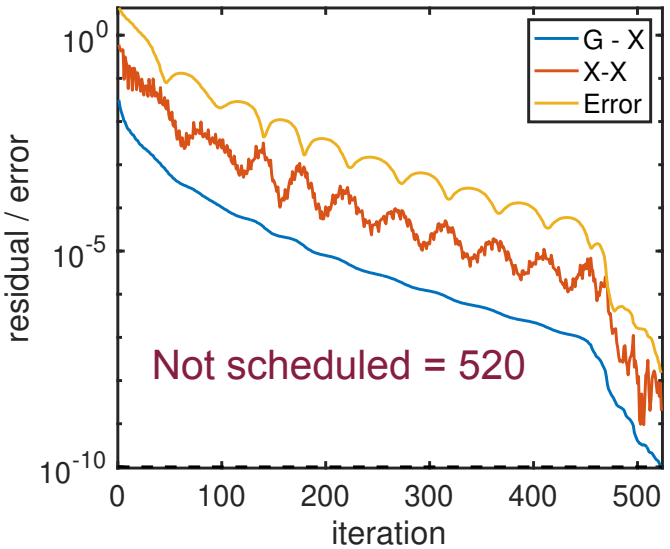
Self-CUDA time (`torch.profiler`). QDEIM & LS show a clean $\mathcal{O}(N_{\mathrm{DOF}}^{1/2})$ trend (matches Eq. 5); DEIM scales worse (LU-bound) and is ruled out for large problems. At the largest sizes LS edges out QDEIM.

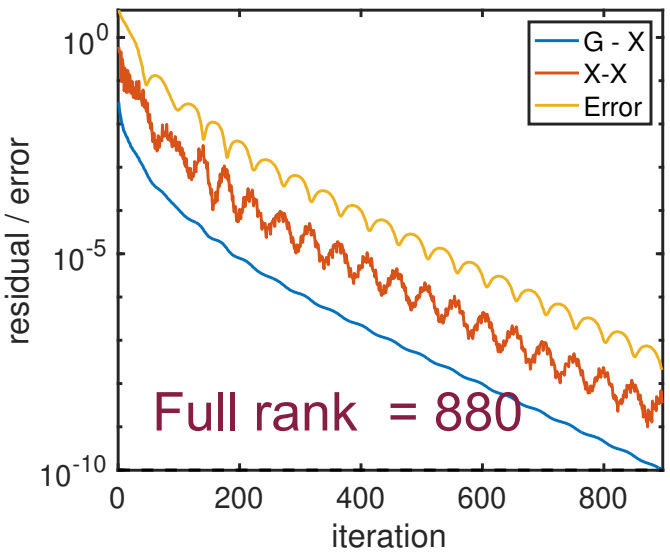
Anderson Acceleration results

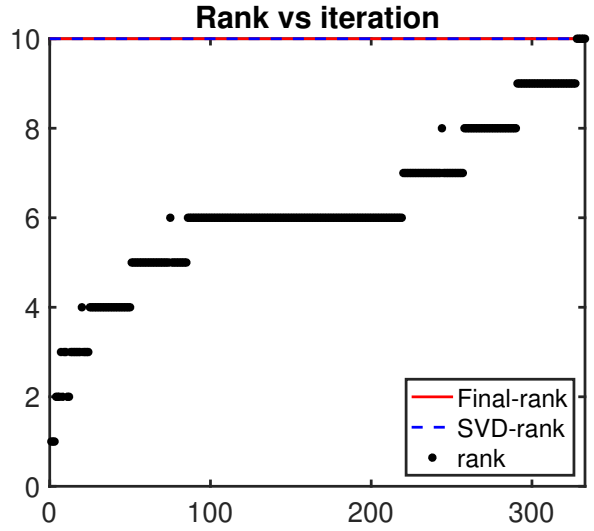
VIRGINIA TECH

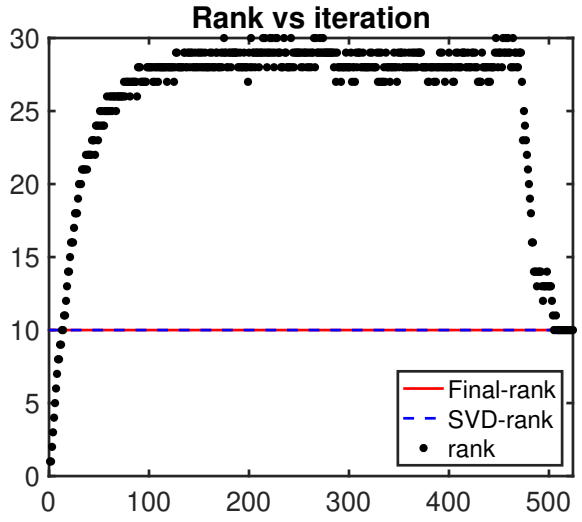
Poisson's equation by lrAA - Matrix Poisson











Scheduling keeps the rank controlled

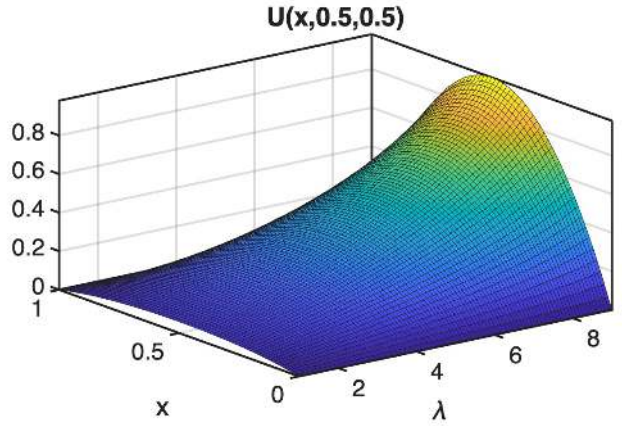
often monotonic

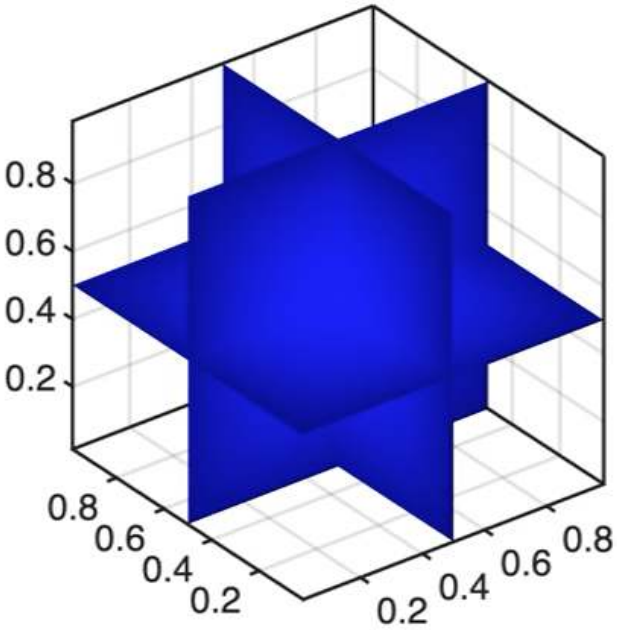
VIRGINIA TECH

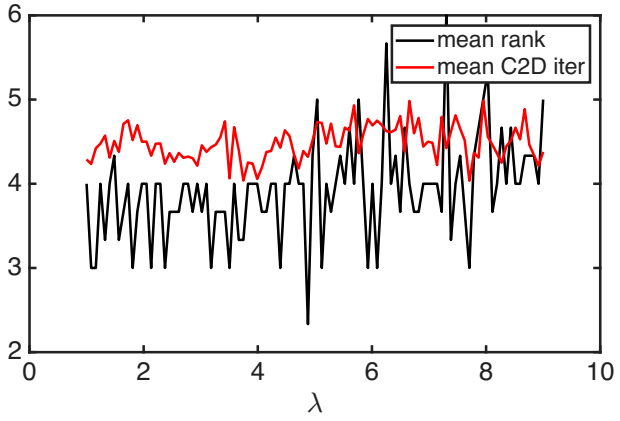
Bratu in 3D for λ from 1 to 9

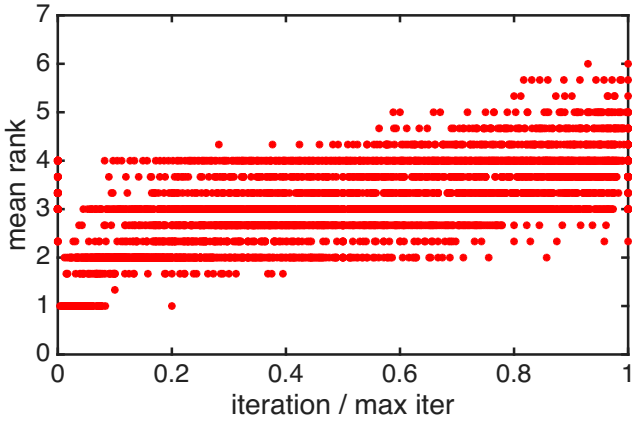
$$\Delta u + \lambda e^u = 0, \quad (x, y, z) \in [0, 1]^3$$

- Using the fast Poisson solver as preconditioner
- The majority of the compute is spent indexing (Matlab illness…)
- Rank increases gradually
- Average iteration count ~ 40







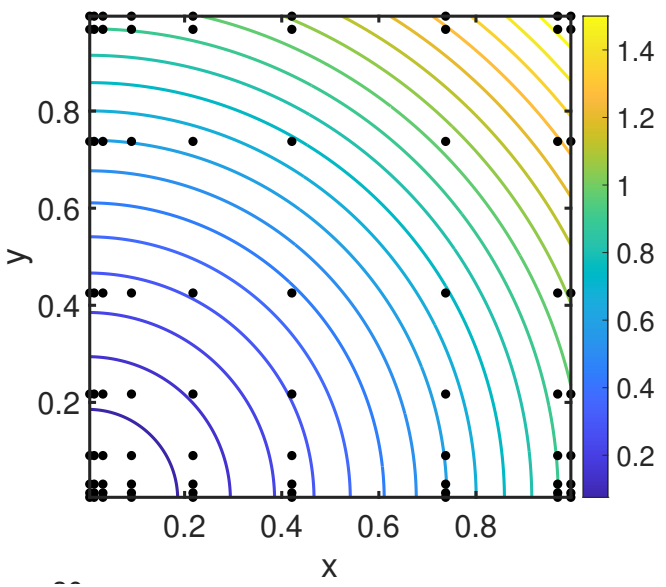


VIRGINIA TECH

Monge-Ampere

Discretized with “method 1” from [6]

$$\left(\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial x^2} - \left(\frac{\partial^2 u}{\partial x \partial x}\right)^2\right) = f(x,y) \qquad u = \frac{2\sqrt{2}}{3}(x^2+y^2)^{\frac{3}{4}}$$



$n = m$	lrAA iterations	iterations reported in[6]	final rank
21	109 (7)	1083	13 (4)
61	287 (8)	8967	18 (7)
101	443 (13)	23849	20 (7)
221	675 (11)	107388	26 (8)

Again the scheduling gives much lower number of iterations.

It is highly beneficial to incorporate local truncation error in rounding / truncation

[6] Benamou, Jean-David, Brittany D. Froese, and Adam M. Oberman. "Two Numerical Methods for the elliptic Monge-Ampère equation." *ESAIM: Mathematical Modelling and Numerical Analysis* 44, no. 4 (2010): 737-758.

VIRGINIA TECH

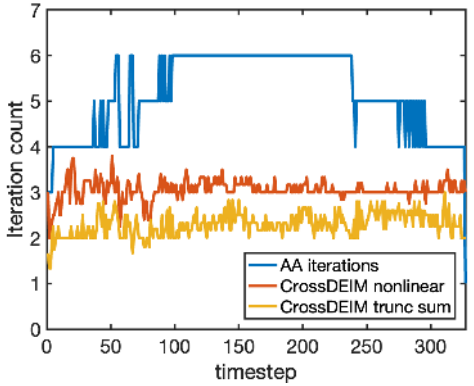
Allen-Cahn

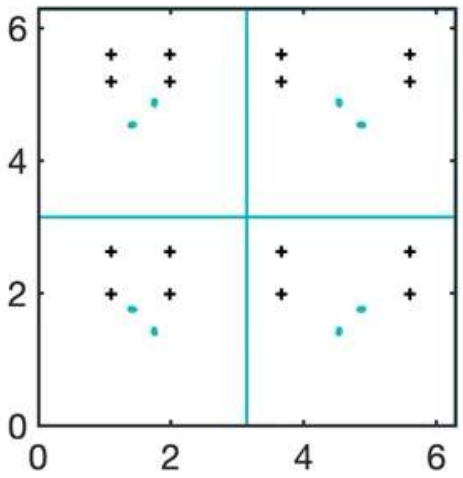
Here we solve Allen-Cahn $u_t = \frac{1}{100}\Delta u + u - u^3, \qquad X(i,j) \approx u(x_i, y_j).$

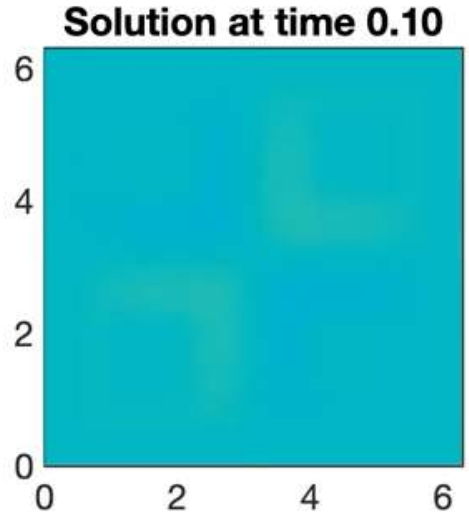
We use a 1024×1024 mesh but only store $2 \times 5 \times 1024 \sim 60,000$ doubles.

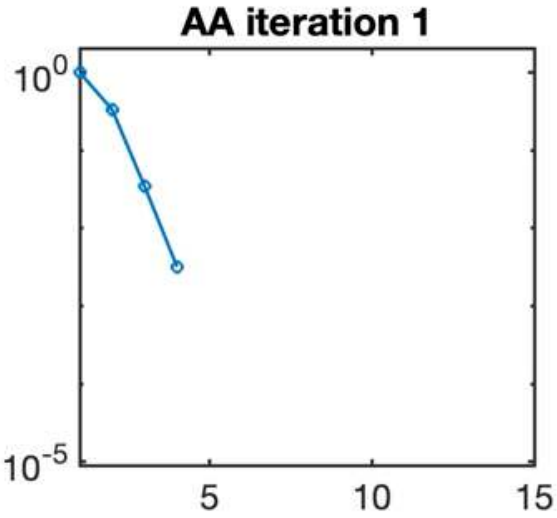
$$u^{n+1} - \Delta t\left(\Delta u^{n+1} + u^{n+1} - (u^{n+1})^3\right) = u^n$$

Exponential sum preconditioner







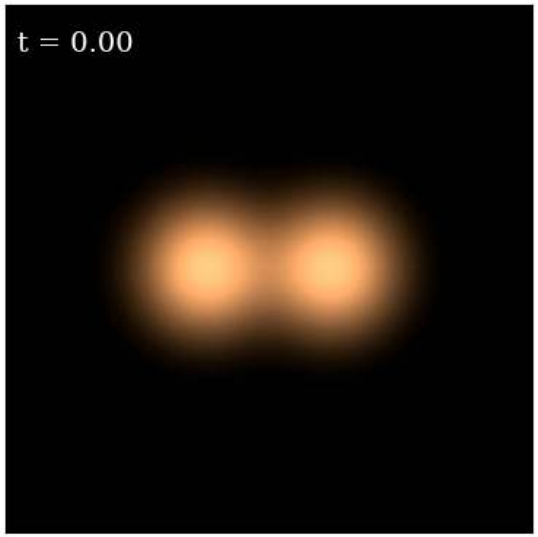


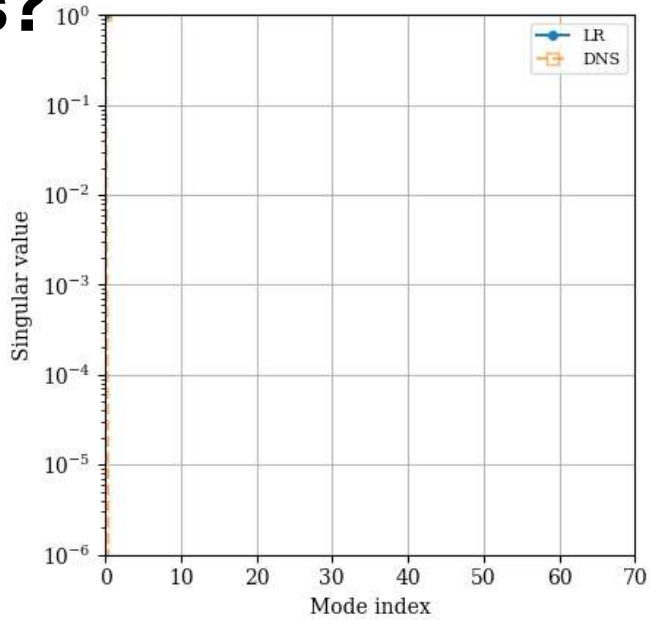
Summary:

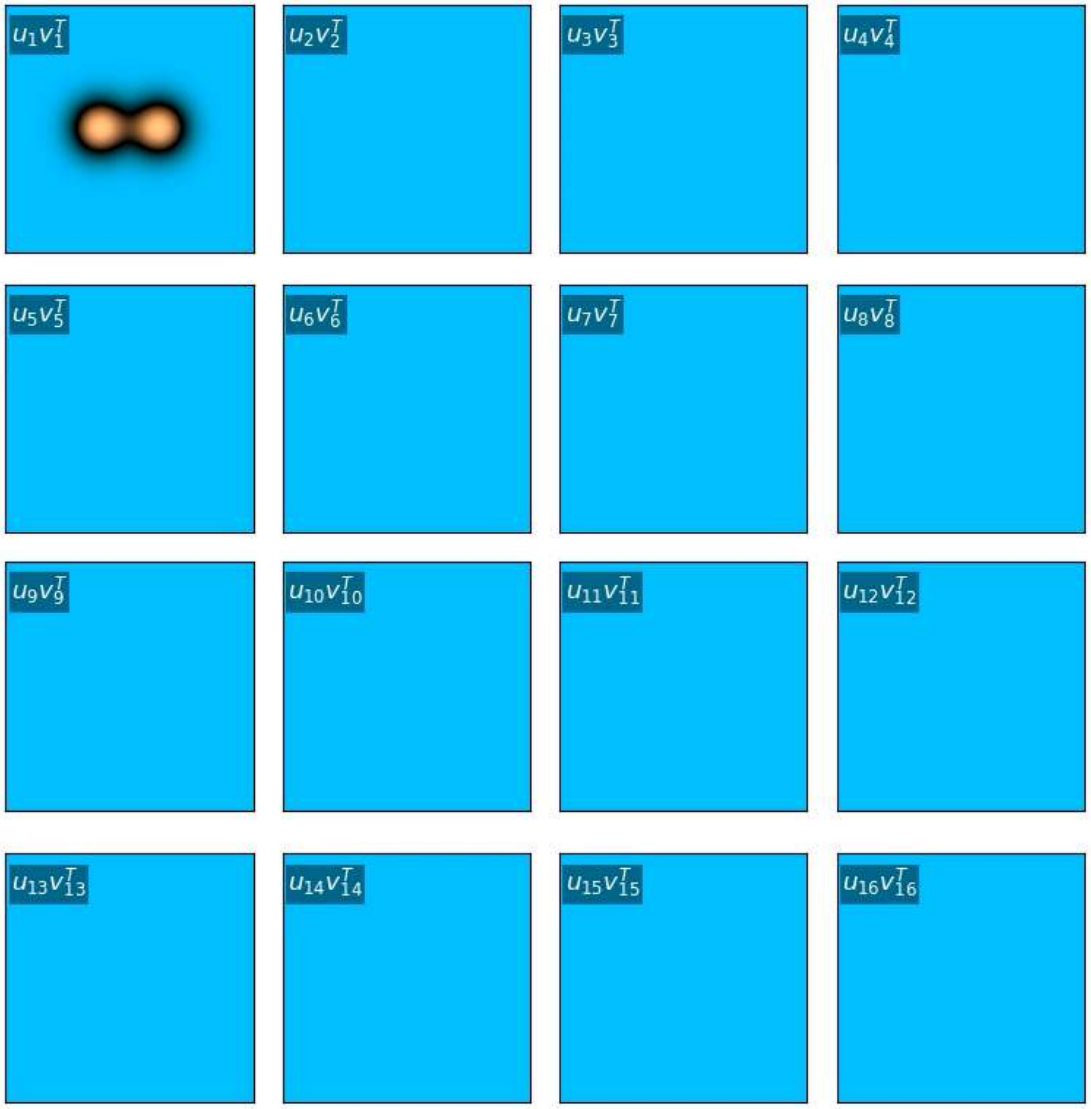
- lrAA (low-rank Anderson acceleration): A sublinear method for computing low-rank solutions to nonlinear matrix equations.
- Cross-DEIM & Cross^2-DEIM - Adaptive iterative cross approximation with a warm-start strategy.
- LS works better with iteration and is fast on GPU

VIRGINIA TECH

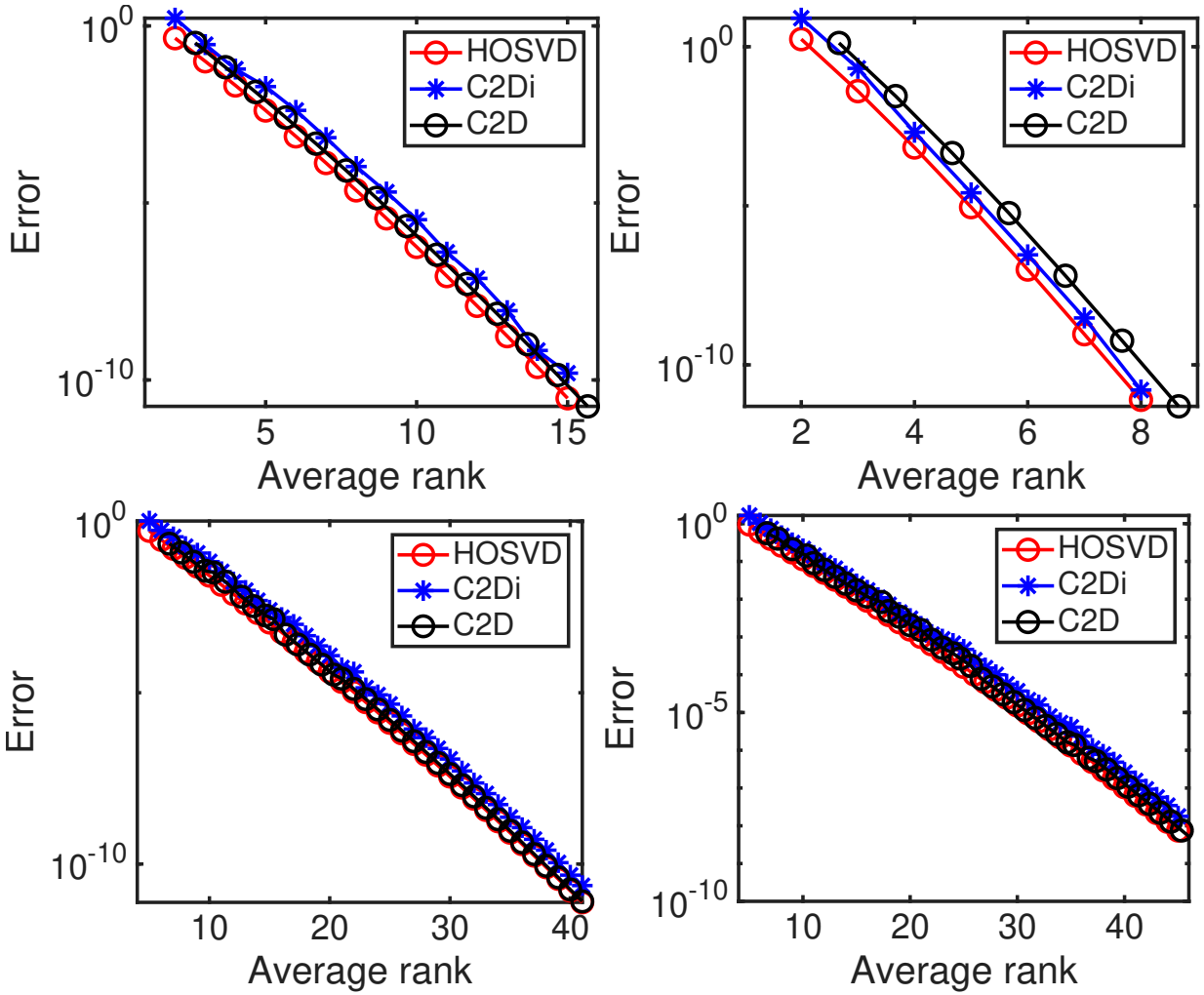
Questions?


t = 0.00


LR
DNS
Singular value
Mode index
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
10
20
30
40
50
60
70


$u_1v_1^T$
$u_2v_2^T$
$u_3v_3^T$
$u_4v_4^T$
$u_5v_5^T$
$u_6v_6^T$
$u_7v_7^T$
$u_8v_8^T$
$u_9v_9^T$
$u_{10}v_{10}^T$
$u_{11}v_{11}^T$
$u_{12}v_{12}^T$
$u_{13}v_{13}^T$
$u_{14}v_{14}^T$
$u_{15}v_{15}^T$
$u_{16}v_{16}^T$

Cross^2-DEIM for function compression 3D



$$X_1(i_1, i_2, i_3) = \frac{1}{i_1 + i_2 + i_3},$$

$$X_2(i_1, i_2, i_3) = e^{-(x_1(i_1)x_2(i_2)x_3(i_3))^2},$$

$$X_3(i_1, i_2, i_3) = (i_1^3 + i_2^3 + i_3^3)^{-\frac{1}{3}},$$

$$X_4(i_1, i_2, i_3) = (i_1^5 + i_2^5 + i_3^5)^{-\frac{1}{5}}.$$

VIRGINIA TECH