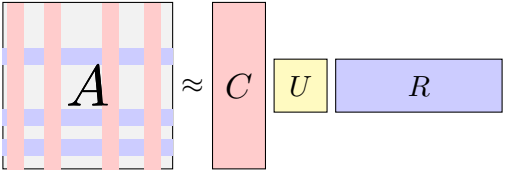
CUR approximation:
basics, computation, and applications



Yuji Nakatsukasa
Oxford University

MORAT Lectures 1+2 of 3

Plan for my lectures

1. Now: CUR basics and computation
2. Later today: Subsampling based algorithms
3. Tomorrow: Stability of (randomized) NLA algorithms

Orthogonal vs. oblique projection for low-rank approx

Goal: Having chosen C $\approx$ Range(A), find $A \approx C\,M$

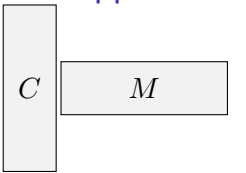
Orthogonal vs. oblique projection for low-rank approx

Goal: Having chosen $C \approx \text{Range}(A)$, find $A \approx C\,M$

Most (randomized) algorithms proceed as

1. Choose C to approximate Range(A),
2. Find M by **projection** onto C, either **orthogonal** or **oblique**

Orthogonal vs. oblique projection for low-rank approx

Goal: Having chosen  $\approx$ Range(A), find $A \approx$ 

Most (randomized) algorithms proceed as

1. Choose C to approximate Range(A),
2. Find M by **projection** onto C, either **orthogonal** or **oblique**

(Review on projections:) $P \in \mathbb{R}^{n\times n}$ is called a **projection** if $P^2 = P$

- P diagonalizable and all eigenvalues 1 or 0
- $\|P\|_2 \geq 1$ and $\|P\|_2 = 1$ iff $P = P^T$; in this case $P = QQ^T$ for Q s.t. $Q^TQ = I_r$, and P called orthogonal projection, otherwise oblique
- $I - P$ also projector, and unless $P = 0$ or $P = I$, $\|I - P\|_2 = \|P\|_2$: Schur form $QPQ^* = \left[\begin{smallmatrix} I & B \\ 0 & 0 \end{smallmatrix}\right]$, $Q(I-P)Q^* = \left[\begin{smallmatrix} 0 & -B \\ 0 & I \end{smallmatrix}\right]$; see [Szyld 2006]

Orthogonal vs. oblique projection for $A \approx CM$

- Orthogonal projection: best in F-norm (and all unitarily invariant)

$$M = \text{argmin}_M \|CM - A\|_F$$

Orthogonal vs. oblique projection for $A \approx CM$

- Orthogonal projection: best in F-norm (and all unitarily invariant)

$$M = \text{argmin}_M \|CM - A\|_F$$

  - least-squares problem with many right-hand sides, soln $M = C^\dagger A = R^{-1}Q^T A$, where $C = QR$
  - so $A \approx CC^\dagger A = QQ^T A$: same form as RandSVD (HMT)

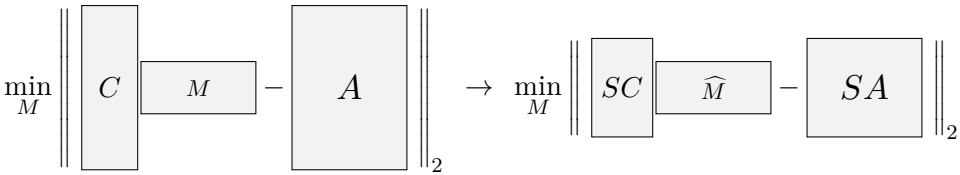
Orthogonal vs. oblique projection for $A \approx CM$

- Orthogonal projection: best in F-norm (and all unitarily invariant)

$$M = \text{argmin}_M \|CM - A\|_F$$

  - least-squares problem with many right-hand sides, soln $M = C^\dagger A = R^{-1}Q^T A$, where $C = QR$
  - so $A \approx CC^\dagger A = QQ^T A$: same form as RandSVD (HMT)

- Oblique projection (sometimes better computationally): consider sketch-and-solve



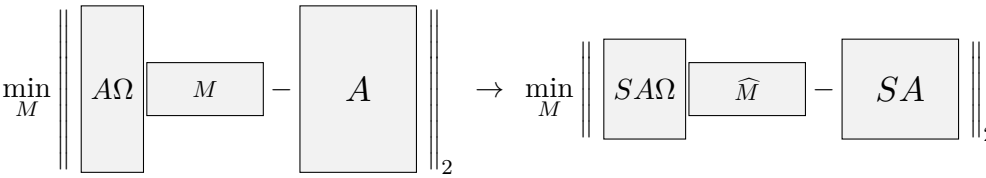
  - soln $\widehat{M} = (SC)^\dagger (SA)$
  - so $A \approx C(SC)^\dagger S A$: same form as gen. Nyström, and **CUR**

(Review) From RandSVD to generalized Nyström

RandSVD (HMT) $A\Omega = QR$, $A \approx QQ^TA$ takes $C = A\Omega$ and orthogonal projection, equivalent to

$$M = \text{argmin}_M \|(A\Omega)M - A\|_F.$$

If we sketch-and-solve this:

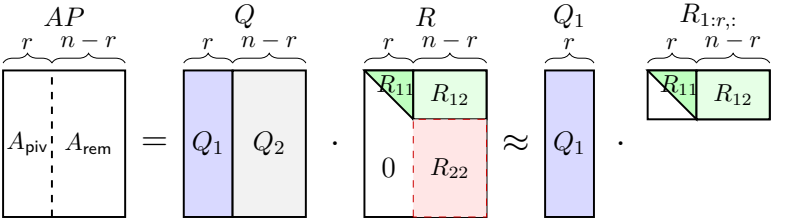


We get $\widehat{M} = (SA\Omega)^\dagger(SA)$, giving approximation
$A \approx (A\Omega)\widehat{M} = (A\Omega)(SA\Omega)^\dagger(SA)$, generalized Nyström!

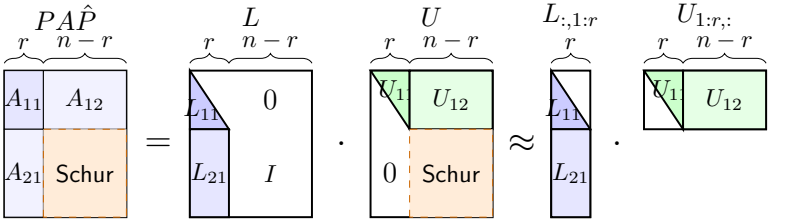
[Woolfe-Liberty-Rokhlin-Tygert 08, Clarkson-Woodruff 17,
Tropp-Yurtsever-Udell-Cevher 17, N. 20 etc]

Review: column-pivoted QR and LU with complete pivoting

(partial) column-pivoted QR: ($P, \hat{P}$: permutation matrices)



(partial) LU with complete pivoting



Orthogonal vs. oblique projection for low-rank approx

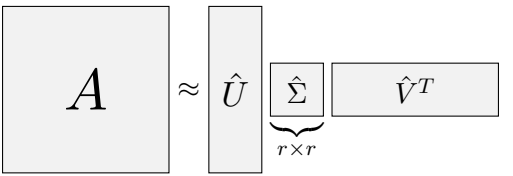
	Orthogonal	Oblique
Classical	QR w/ column pivots	LU complete pivots
$C = A\Omega$	RandSVD $A \approx QQ^T A$	Gen. Nyst $A \approx AX(Y^T AX)^\dagger Y^T A$
C: columns	interpolative decomp. $A \approx CC^\dagger A$	**CUR** $A \approx CUR$

- Oblique can be obtained by 'sketch-and-solve' least-squares problem in Orthogonal (if same C used)
- Big advantages when sketch is **subsampling**: today's main theme
- Note when $A \succeq 0$, Nyström $A \approx CUC^T$ is recommended; this is 'oblique' (suggesting oblique is fundamental!?) but note also

$$CUC^T = A(:,I)A(I,I)^{-1}A(I,:) = A^{1/2}\underbrace{A^{1/2}(:,I)A(I,I)^{-1}A^{1/2}(I,:)}_{\text{orthogonal projection}}A^{1/2}$$

Review: Low-rank approximation via SVD

Given $A \in \mathbb{R}^{m \times n}$, find low-rank (rank r) approximation



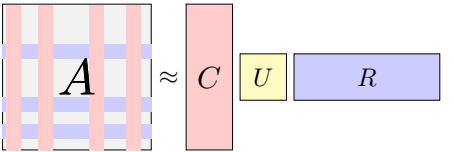
- Optimal solution $A_r = \hat{U}\hat{\Sigma}\hat{V}^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ is truncated SVD, giving [von Neumann 37, Horn-Johnson 85]

$$\|A - A_r\| = \min_{\mathsf{rank}(B)=r} \|A - B\| = \left\| \begin{bmatrix} \sigma_{r+1} & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \right\|$$

- SVD costs $O(mn^2)$; m, n sometimes ∞
- Low-rank matrices in: LoRA (neural networks), recommendation systems (Netflix), smooth functions, structured matrices [Beckermann-Townsend 17]

Low-rank approximation via CUR

[Goreinov-Tyrtyshnikov-Zamarashkin (97), Mahoney-Drineas (09)]

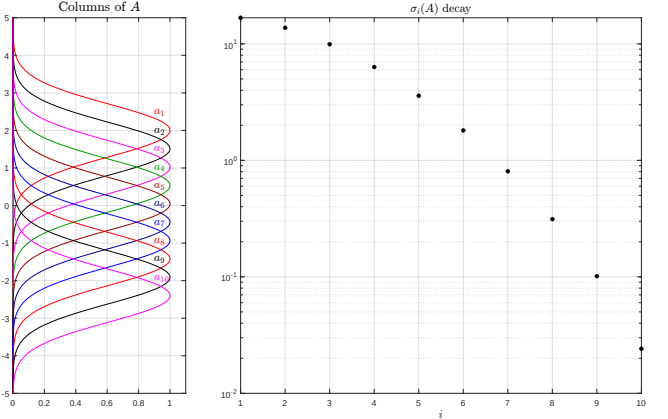


$C = A(:, J)$ column subset, $R = A(I, :)$ row subset, $U = A(I, J)^{-1}$

- Like truncated SVD, but C, R restricted to column/row subset
- Error structure $P_1(A - CUR)P_2 = \left[\begin{smallmatrix} 0 & 0 \\ 0 & * \end{smallmatrix}\right]$. Connections to Schur complement/LU (GE with pivots), ACA, Nyström, Cholesky,...
- Theory [Deshpande-Rademacher-Vempala-Wang 06, Cortinovis-Kressner 20 etc]
- $\exists$ rank-r CUR s.t. $\|A - CU_1R\|_F \leq \sqrt{2(r+1)}\|A - A_r\|_F$
- For any $A \in \mathbb{R}^{m \times n}$, there exists I, J s.t. [Zamarashkin–Osinsky 18]
  $\|A - CUR\|_F \leq (r+1)\|A - A_r\|_F = (r+1)\sqrt{\sum_{i=r+1}^{n} \sigma_i^2}$
- Near-optimal algorithms for computing I, J: deterministic [Osinsky 25], ARP (randomized) [Cortinovis-Kressner 26] ($\exists$ Many other papers)

Low-rank approx example with $A_{i,j} = \exp(-|x_i - \mu_j|^2/2)$

x_i: 500 equispaced pts in $[-5, 5]$, μ_j: equispaced in $[-2.5, 2]$

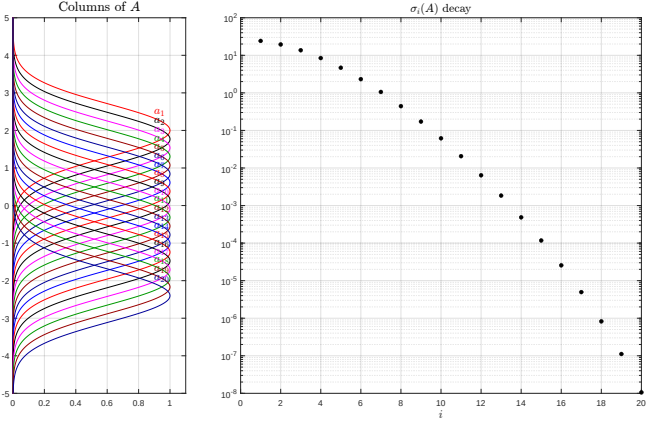


Smooth functions are approximately low rank [Townsend-Trefethen 15]

(proof: Chebyshev interpolation (upcoming))

Low-rank approx example with $A_{i,j} = \exp(-|x_i - \mu_j|^2/2)$

x_i: 500 equispaced pts in $[-5, 5]$, μ_j: equispaced in $[-2.5, 2]$

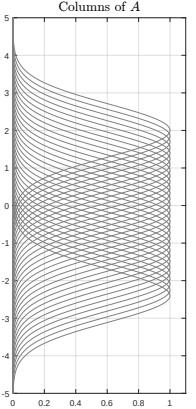


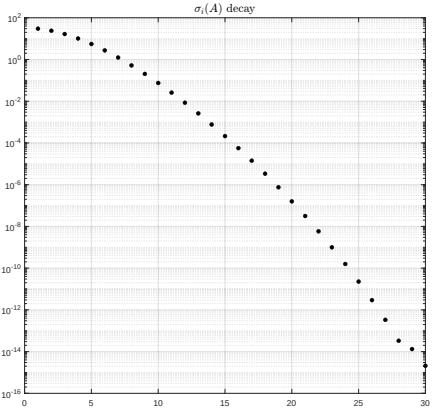
Smooth functions are approximately low rank [Townsend-Trefethen 15]

(proof: Chebyshev interpolation (upcoming))

Low-rank approx example with $A_{i,j} = \exp(-|x_i - \mu_j|^2/2)$

x_i: 500 equispaced pts in $[-5, 5]$, μ_j: equispaced in $[-2.5, 2]$



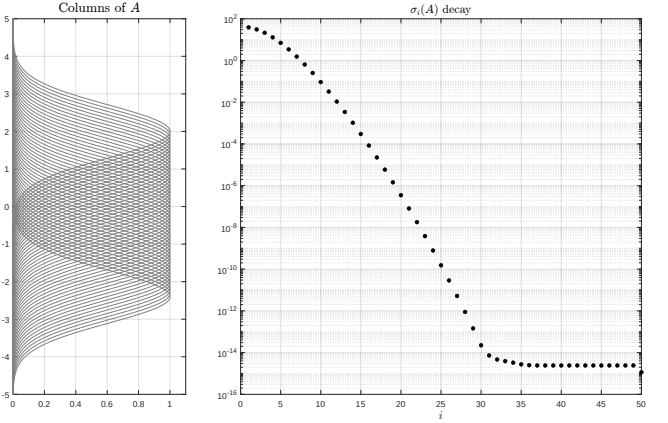


Smooth functions are approximately low rank [Townsend-Trefethen 15]

(proof: Chebyshev interpolation (upcoming))

Low-rank approx example with $A_{i,j} = \exp(-|x_i - \mu_j|^2/2)$

x_i: 500 equispaced pts in $[-5, 5]$, μ_j: equispaced in $[-2.5, 2]$

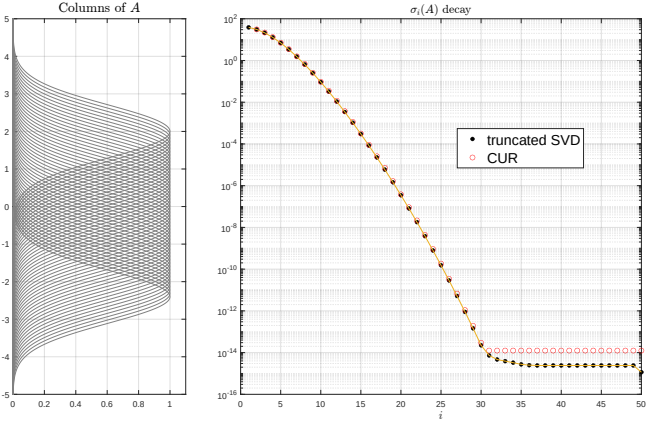


Smooth functions are approximately low rank [Townsend-Trefethen 15]

(proof: Chebyshev interpolation (upcoming))

Low-rank approx example with $A_{i,j} = \exp(-|x_i - \mu_j|^2/2)$

x_i: 500 equispaced pts in $[-5, 5]$, μ_j: equispaced in $[-2.5, 2]$



Smooth functions are approximately low rank [Townsend-Trefethen 15]

(proof: Chebyshev interpolation (upcoming))

Osinsky's alg and sketched variant Osinsky+

Idea: "greedy" sampling with deflation (a la pivoted QR)

$A \in \mathbb{R}^{m\times n}$ and orthonormal $V \in \mathbb{R}^{n\times r}$ s.t. $A \approx AVV^T$

1. Draw a random matrix $G \in \mathbb{R}^{s\times m}$
2. $\tilde{A} := G(A - AVV^T)$ (in Osinsky, $\tilde{A} = A$)
3. For $k = 1$ to r
   - $j = \underset{j\geq k}{\arg\min}\ \|\tilde{A}(1:n, j)\|_2 / \|V(j, k:r)\|_2$ `% greedy selection`
   - $\Pi_C := [\Pi_C \quad j]$
   - Swap columns k and j in $\tilde{A}$ and in V
   - Householder reflector $\nu := V(k:r, k)$, $\nu_1 = \nu_1 + \|\nu\|_2$, $\nu = \nu/\|\nu\|_2$
   - $V(k:r, :) := V(k:r, :) - 2\nu\nu^T V(k:r, :)$ `% deflation`
   - $\tilde{A} := \tilde{A} - \frac{1}{V(k,k)}\tilde{A}(:, k)V(:, k)^T$
4. $C = A(:, \Pi_C)$

Guarantees (up to constant with Osinsky+, in $\mathbb{E}$ with ARP)

$$\|A - CC^\dagger A\|_F \leq \|A - A(:, J)V(:, J)^\dagger V^T\|_F \leq (r+1)\|A - A_r\|_F$$

Osinsky: $O(mnr)$, Osinsky+: $O(nr^2)$

CUR from CSSP [Cortinovis-Kressner 26], [Osinsky 25]

ARP and Osinsky require approx $V \in \mathbb{R}^{n \times r}$ to row-space.

1. Find approx V to row-space (e.g. SVD or sketch ΩA) and obtain columns $C = A(:, J)$ using ARP or Osinsky
2. Use C as approx column-space to find rows $R = A(I, :)$
3. Take $U = A(I, J)^{-1}$.

CUR from CSSP [Cortinovis-Kressner 26], [Osinsky 25]

ARP and Osinsky require approx $V \in \mathbb{R}^{n\times r}$ to row-space.

1. Find approx V to row-space (e.g. SVD or sketch ΩA) and obtain columns $C = A(:, J)$ using ARP or Osinsky
2. Use C as approx column-space to find rows $R = A(I, :)$
3. Take $U = A(I, J)^{-1}$.

- Important to choose R based on C; choosing independently is dangerous with cross approx $U = A(I, J)^{-1}$
- Finds CUR acheiving fundamental theorem! With Osinsky, deterministically

$$\|A - CUR\|_F \leq (r+1)\|A - A_r\|_F$$

and with ARP

$$\mathbb{E}[\|A - CUR\|_F^2] \leq (r+1)\|A - A_r\|_F^2.$$

Above assumes V from SVD. With sketch, incur a modest factor

Proof of $\|A - CUR\|_F \leq (r+1)\|A - A_r\|_F$

Fact: given orthonormal V, one can find J s.t. (e.g. Osinsky)

$$\|A - A(:, J)V(:, J)^{\dagger}V^T\|_F \leq \sqrt{r+1}\|A(I - VV^T)\|_F.$$

Proof of $\|A - CUR\|_F \leq (r+1)\|A - A_r\|_F$

Fact: given orthonormal V, one can find J s.t. (e.g. Osinsky)

$$\|A - A(:,J)V(:,J)^\dagger V^T\|_F \leq \sqrt{r+1}\|A(I - VV^T)\|_F.$$

Proof (of title): based on [Cortinovis-Kressner 26]

Let S be subsampling s.t. $SA = A(I,:)$, and $C = A(:,J) = QR$. Then $A - A(:,J)A(I,J)^{-1}A(I,:) = (I - A(:,J)(SA(:,J))^{-1}S)A = (I - Q(SQ)^{-1}S)A$.

(recall $Q(SQ)^{-1}S$ is a projection). Then

$$\begin{aligned}
\|A - CUR\|_F^2 &= \|(I - Q(SQ)^{-1}S)A\|_F^2 \\
&\leq (r+1)\|(I - QQ^T)A\|_F^2 && \text{(by Fact with } A^T\text{)} \\
&\leq (r+1)\|A - A(:,J)V(J,:)^{-T}V^T\|_F^2 && \text{(orth proj is optimal)} \\
&\leq (r+1)^2\|A - AVV^T\|_F^2. && \text{(by Fact)}
\end{aligned}$$

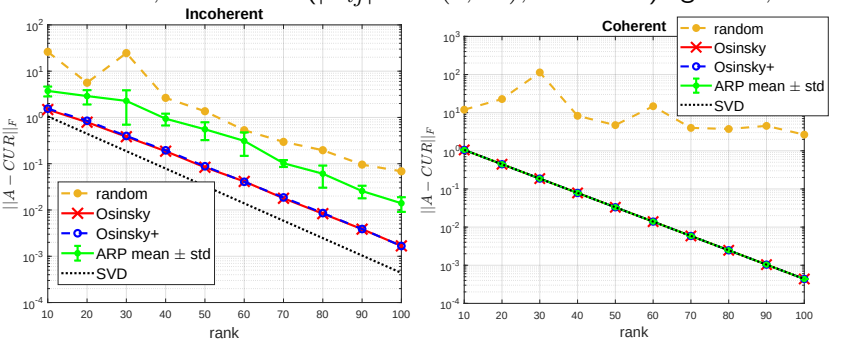
and $\|A - AVV^T\|_F = \|A - A_r\|_F$ if V via SVD.

CUR experiments

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

Incoherent: U, V Haar distributed (arbitrary I, J suffices)

Coherent: $U, V = I + E$ ($|E_{ij}| \sim N(0, \sigma^2), \sigma = 10^{-5}$): good I, J is crucial



- All methods reliable but random
- Coherent problems 'hard' to find good I, J, but reward is enormous, almost as accurate as truncated SVD!

Rank-adaptive, error-controlled CUR computation

Osinsky/ARP are spectacular, but

1. **rank r needed** in advance,
2. **no error control** a priori on $\|A - CUR\|_F$,
3. needs approximant $V \in \mathbb{R}^{n \times r}$ to leading subspace; hence requires at least r **mat-vecs** with A (even with Osinsky+ or Cortinovis-Kressner)

Note: True for most methods, incl. RandSVD and GN.

Rank-adaptive, error-controlled CUR computation

Osinsky/ARP are spectacular, but

1. **rank r needed** in advance,
2. **no error control** a priori on $\|A - CUR\|_F$,
3. needs approximant $V \in \mathbb{R}^{n \times r}$ to leading subspace; hence requires at least r **mat-vecs** with A (even with Osinsky+ or Cortinovis-Kressner)

Note: True for most methods, incl. RandSVD and GN.

IterativeCUR addresses these by incorporating [Pritchard-Park-N.-Martinsson 25]

- Adaptive rank determination+error control $\|A - CUR\|_F \leq \tau$ for user-prescribed τ
- Recycling small $O(1)$ sketch $\boxed{GA}$ to determine pivots.
- Useful practical tool: sketch-and-pivot: pivot with GA

Error estimation via trace estimation

[Martinsson-Tropp Acta Numer. 2020, Martinsson-N. in prep.]

Estimate error $\|A-CUR\|_F =: \|E\|_F$ via trace estimation (Monte Carlo):

$$\|E\|_F^2 = \mathsf{trace}(E^TE) \approx \frac{1}{\ell}\sum_{i=1}^{\ell} g_i^T E^T E g_i = \frac{1}{\ell}\|EG\|_F^2$$

$$= \frac{1}{\ell}\left\| \boxed{A-CUR}\ \boxed{G} \right\|_F^2, \quad G=[g_1,\dots,g_\ell]$$

$g_i \in \mathbb{R}^n$: iid Gaussian $N(0,1)$ or Rademacher [Hutchinson (90), Avron-Toledo (11), Musco-Musco-Woodruff (21), Cortinovis-Kressner (22), Epperly-Tropp-Webber (23) etc. [Gratton+Titley-P (18)] shows:

$$\mathbb{P}[\frac{1}{2}\|E\|^2 \le \frac{1}{\ell}\|EG\|_F^2 \le 2\|E\|^2] \ge 1-2\exp(-C\ell), \quad C = \frac{1}{4}\frac{\mathsf{trace}(E^TE)}{\|E^TE\|_2} \ge \frac{1}{4}$$

V simple and useful! For our purpose, $\ell = O(1)$ enough, say $\ell = 10$

(Update: Kressner has argument $\ell = 2$ enough!)

IterativeCUR: outline

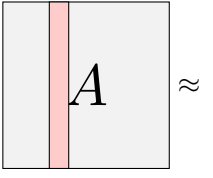
Initialize $\widehat{A} := A$, $|I| = |J| = 0$

1. Find **column indices** $C_{\text{now}} = A(:, J_{\text{now}})$ via sketched pivot on sketch $G\widehat{A}$ with LUPP
2. Find **row indices** I_{now} via pivoting $\widehat{A}(:, J_{\text{now}})$
3. Set $I := I \cup I_{\text{now}}$, $J := J \cup J_{\text{now}}$
4. Form $C = A(:, J)$, $U = A(I, J)^{\dagger}$, $R = A(I, :)$
5. Estimate $\|A - CUR\|_F \approx \|G(A - CUR)\tilde{G}\|_F$, exit if small enough.
6. Return to step 1 with $\widehat{A} := A - CUR$

IterativeCUR: outline

Initialize $\widehat{A} := A$, $|I| = |J| = 0$

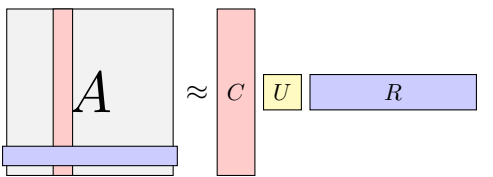
1. Find **column indices** $C_{\text{now}} = A(:, J_{\text{now}})$ via sketched pivot on sketch $G\widehat{A} = G(A - CUR)$ with LUPP
2. Find **row indices** I_{now} via pivoting $\widehat{A}(:, J_{\text{now}})$
3. Set $I := I \cup I_{\text{now}}$, $J := J \cup J_{\text{now}}$
4. Form $C = A(:, J)$, $U = A(I, J)^{\dagger}$, $R = A(I, :)$
5. Estimate $\|A - CUR\|_F \approx \|G(A - CUR)\tilde{G}\|_F$, exit if small enough.
6. Return to step 1 with $\widehat{A} := A - CUR$



IterativeCUR: outline

Initialize $\widehat{A} := A$, $|I| = |J| = 0$

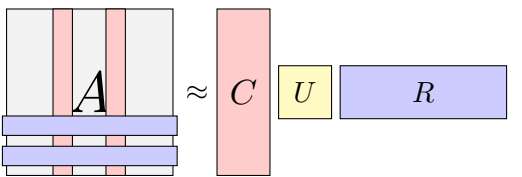
1. Find **column indices** $C_{\text{now}} = A(:, J_{\text{now}})$ via sketched pivot on sketch $G\widehat{A} = G(A - CUR)$ with LUPP
2. Find **row indices** I_{now} via pivoting $\widehat{A}(:, J_{\text{now}})$
3. Set $I := I \cup I_{\text{now}}$, $J := J \cup J_{\text{now}}$
4. Form $C = A(:, J)$, $U = A(I, J)^{\dagger}$, $R = A(I, :)$
5. Estimate $\|A - CUR\|_F \approx \|G(A - CUR)\tilde{G}\|_F$, exit if small enough.
6. Return to step 1 with $\widehat{A} := A - CUR$



IterativeCUR: outline

Initialize $\widehat{A} := A$, $|I| = |J| = 0$

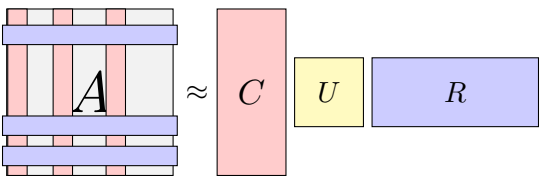
1. Find **column indices** $C_{\text{now}} = A(:, J_{\text{now}})$ via sketched pivot on sketch $G\widehat{A} = G(A - CUR)$ with LUPP
2. Find **row indices** I_{now} via pivoting $\widehat{A}(:, J_{\text{now}})$
3. Set $I := I \cup I_{\text{now}}$, $J := J \cup J_{\text{now}}$
4. Form $C = A(:, J)$, $U = A(I, J)^{\dagger}$, $R = A(I, :)$
5. Estimate $\|A - CUR\|_F \approx \|G(A - CUR)\tilde{G}\|_F$, exit if small enough.
6. Return to step 1 with $\widehat{A} := A - CUR$



IterativeCUR: outline

Initialize $\widehat{A} := A$, $|I| = |J| = 0$

1. Find **column indices** $C_{\text{now}} = A(:, J_{\text{now}})$ via sketched pivot on sketch $G\widehat{A} = G(A - CUR)$ with LUPP
2. Find **row indices** I_{now} via pivoting $\widehat{A}(:, J_{\text{now}})$
3. Set $I := I \cup I_{\text{now}}$, $J := J \cup J_{\text{now}}$
4. Form $C = A(:, J)$, $U = A(I, J)^{\dagger}$, $R = A(I, :)$
5. Estimate $\|A - CUR\|_F \approx \|G(A - CUR)\tilde{G}\|_F$, exit if small enough.
6. Return to step 1 with $\widehat{A} := A - CUR$



IterativeCUR: outline

Initialize $\widehat{A} := A$, $|I| = |J| = 0$

1. Find **column indices** $C_{\text{now}} = A(:, J_{\text{now}})$ via sketched pivot on sketch $G\widehat{A} = G(A - CUR)$ with LUPP: sketch of residual $A - CUR$
2. Find **row indices** I_{now} via pivoting $\widehat{A}(:, J_{\text{now}})$
3. Set $I := I \cup I_{\text{now}}$, $J := J \cup J_{\text{now}}$
4. Form $C = A(:, J)$, $U = A(I, J)^{\dagger}$, $R = A(I, :)$
5. Estimate $\|A - CUR\|_F \approx \|G(A - CUR)\tilde{G}\|_F$, exit if small enough.
6. Return to step 1 with $\widehat{A} := A - CUR$

Upshots:

- Only one small sketch G: small sketch for big approximation
- Updating $G\widehat{A}$ cheap! Only need computing GC_{now}
- Comes with reliable error guarantee $\|A - CUR\|_F < \tau$ for user-defined τ; $\tau = O(u)$ allowed

IterativeCUR: Why does it work?

Pivoting on residual $A - CUR$ with temporary C, U, R is like previous work on **adaptive sampling**

- Pivot on $A - CC^{\dagger}A$ to add (block) column indices with probability$\sim$(column norm^2) [Deshpande-Rademacher-Vempala-Wang 06]
- Randomly Pivoted Cholesky [Chen-Epperly-Tropp-Webber 22]
  - contains extensive analysis
  - when $A \succeq 0$, only diagonal elements needed for Cholesky

IterativeCUR: Why does it work?

Pivoting on residual $A - CUR$ with temporary C, U, R is like previous work on **adaptive sampling**

- Pivot on $A - CC^\dagger A$ to add (block) column indices with probability$\sim$(column norm^2) [Deshpande-Rademacher-Vempala-Wang 06]
- Randomly Pivoted Cholesky [Chen-Epperly-Tropp-Webber 22]
  - contains extensive analysis
  - when $A \succeq 0$, only diagonal elements needed for Cholesky

Difference in IterativeCUR:

- We sketch the residual $G(A - CUR)$ to pivot, for efficiency
- Instead of pivoting on $A - CC^\dagger A$ (residual of $\min_M \|CM - A\|_F$), we pivot on $A - CUR$ (residual of **sketch-and-solve** $\min_M \|S(CM - A)\|_F$)

Theory of IterativeCUR (or pivoted LU) remains incomplete

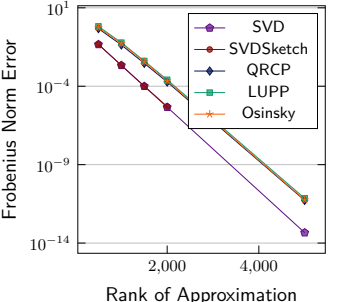
Pivoting strategies

- Classic: QR (QRCP, RRQR, ...), LU (LUPP, LUCP)
- Sketch and classic: QRCP or LUPP on GA or $(GA)^T$ [Martinsson-Voronin 16, Duersch-Gu 17, Dong-Martinsson 23, Grigori-Xue 25, Armstrong-Damle 25 etc]
- Determinantal Point Process [Derezinski-Mahoney 20 etc], twice Ramanujan sparsifier [Batson-Spielman-Srivastava 09], leverage score (later) etc
- "Optimal": Osinsky (23), Cortinovis-Kressner (24)

Pivoting strategies

- Classic: QR (QRCP, RRQR, ...), LU (, LUCP)
- Sketch and classic: QRCP or LUPP on GA or $(GA)^T$ [Martinsson-Voronin 16, Duersch-Gu 17, Dong-Martinsson 23, Grigori-Xue 25, Armstrong-Damle 25 etc]
- Determinantal Point Process [Derezinski-Mahoney 20 etc], twice Ramanujan sparsifier [Batson-Spielman-Srivastava 09], leverage score (later) etc
- "Optimal": Osinsky (23), Cortinovis-Kressner (24)

TL;DR: They all work really well in practice. [Dong-Martinsson 23]



Pivoting strategies

- Classic: QR (QRCP, RRQR, ...), LU (LUPP, LUCP)
- Sketch and classic: QRCP or LUPP on GA or $(GA)^T$ [Martinsson-Voronin 16, Duersch-Gu 17, Dong-Martinsson 23, Grigori-Xue 25, Armstrong-Damle 25 etc]
- Determinantal Point Process [Derezinski-Mahoney 20 etc], twice Ramanujan sparsifier [Batson-Spielman-Srivastava 09], leverage score (later) etc
- "Optimal": Osinsky (23), Cortinovis-Kressner (24)

TL;DR: They all work really well in practice. [Dong-Martinsson 23]

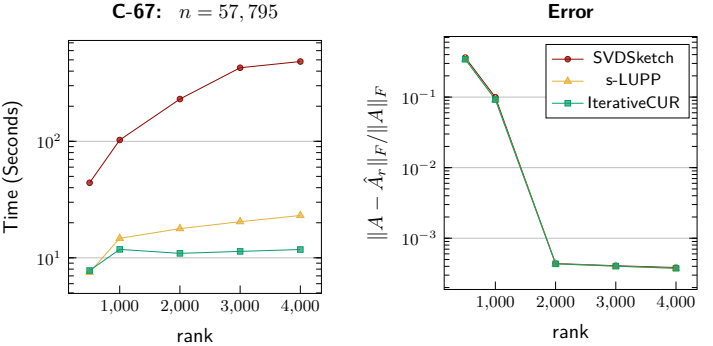
So use fastest: **sketched LUPP**

- Compute sketch $\boxed{GA}$, LUPP on transpose to get C

In IterativeCUR we'll reuse **same sketch** G to find pivots for $G(A - CUR)$ as CUR improves

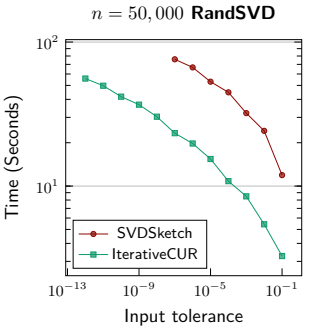
- Bad idea with RandSVD $\hat{A} = AQQ^T$, as then $G(A - AQQ^T) = 0$!
- With CUR, not issue, works great in practice (rigorous theory lacking) [Martinsson-Voronin 16, Duersch-Gu 17]

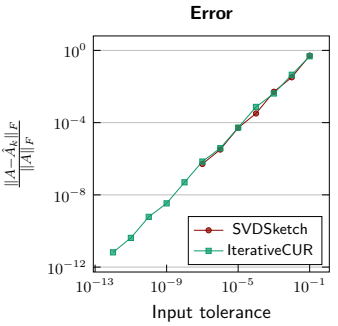
Experiments

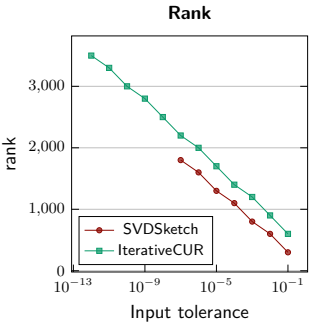


- SVDsketch: MATLAB's built-in [Yu-Yu-Li 18]
  - Roughly 'iterativeHMT' drawing new sketch each time
  - Cannot handle accuracy $\leq u^{1/2} \approx 10^{-8}$
- s-LUPP: big sketch+LUPP
  - to test effect of reusing small sketch G
- IterativeCUR-LUPP: this work

Randsvd test matrix







- SVDSketch fails for tolerance $\tau \leq 10^{-8}$
- Speedup with IterativeCUR esp. for sparse A

Why is CUR so great? Classical answers

Classically,

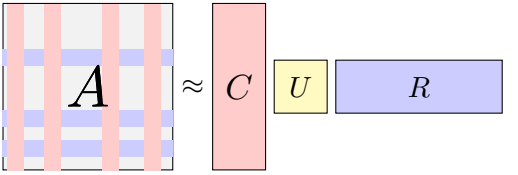
- C, R preserve A's properties, e.g.
  - sparsity,
  - nonnegativity (note U not nonnegative)

Hence CUR is more interpretable; useful e.g. in PCA [Mahoney-Drineas (09), Embree-Sorensen (16)]

Here, we'll focus more on CUR's computational advantages.
Much of remainder: illustrate power of CUR in scientific computing

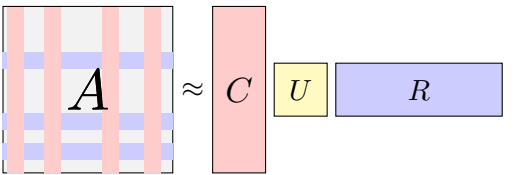
Why is CUR so great? My answer

1. If (big if, but sometimes reasonable) I, J known, CUR without seeing whole A

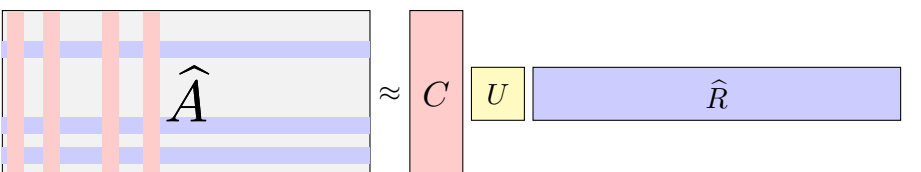


Why is CUR so great? My answer

1. If (big if, but sometimes reasonable) I, J known, CUR without seeing whole A



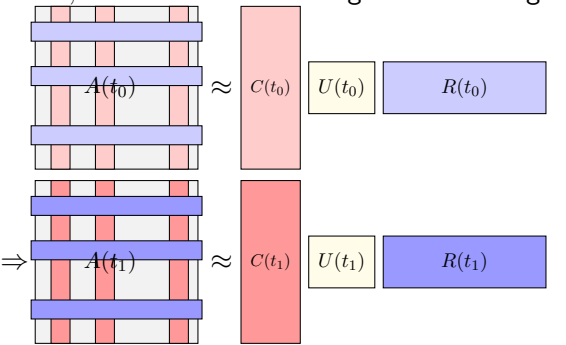
2. New column(/row) predicted by seeing only $|I| \ll m$ entries



CUR app 1:AdaCUR for evolving matrices [Park-N. SISC 25]

$A(t)$: $m \times n$ matrices depending on parameter t (or just many related matrices), e.g. parameter-dependent PDE soln, time series

Idea: CUR indices I, J often need not change for wide range of t



Cost: $O(mnr) \to O(r^3)$! (r: rank)

Error estimator to track error+correct as needed

Algorithms

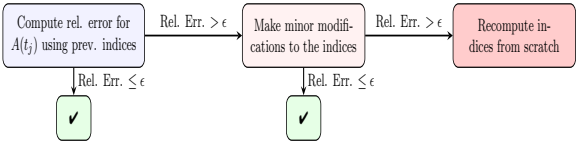
AdaCUR

Input : Parameter-dependent matrix $A(t) \in \mathbb{R}^{m \times n}$, parameters $t_1, ..., t_q$, error tol. ϵ.

Output : CUR factors $C(t_j), U(t_j), R(t_j)$ such that $A(t) \approx C(t_j)U(t_j)^\dagger R(t_j)$.

1. Compute initial set of indices for $A(t_1)$.

❂ For $j = 2, 3, ..., q$,



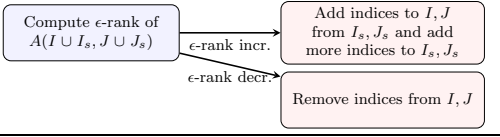
FastAdaCUR

Input : Parameter-dependent matrix $A(t) \in \mathbb{R}^{m \times n}$, parameters $t_1, ..., t_q$, rank tol. ϵ

Output : CUR factors $C(t_j), U(t_j), R(t_j)$ such that $A(t) \approx C(t_j)U(t_j)^\dagger R(t_j)$.

1. Compute initial set of indices I, J and a small set of extra indices I_s, J_s for $A(t_1)$.

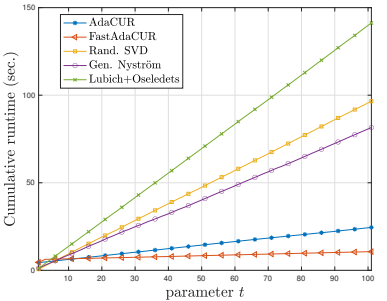
❂ For $j = 2, 3, ..., q$,

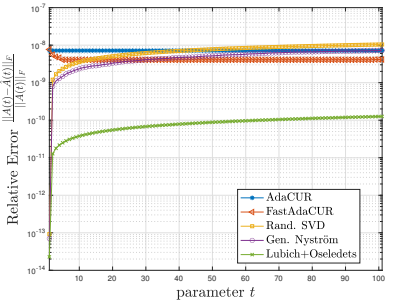


AdaCUR examples

$$A(i) = A + \delta \sum_{j=1}^{i-1} X_j \in \mathbb{R}^{50000\times 5000}, \qquad i \in \{1, 2, ..., 101\}$$

where $\mathrm{rank}(A) = 500$, $\delta = 10^{-12}$, X_j: sparse random $\texttt{sprandn}(\mathtt{m}, \mathtt{n}, 10^{-5})$

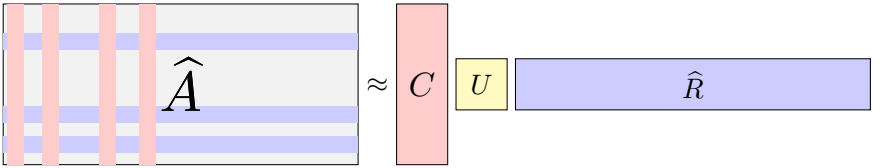




- AdaCUR much faster after first step (precomputation)
- Compared with RandSVD, Dynamical low-rank approx. [Lubich-Oseledets 14], generalized Nyström [Kressner-Lam 23]

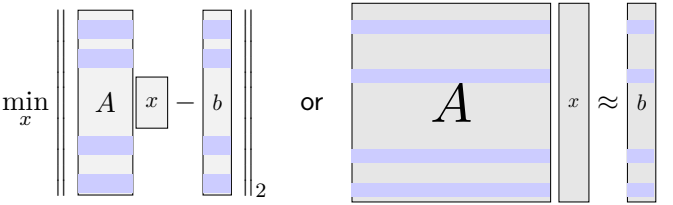
Remainder: Subsampling based computation

Recall 'what makes CUR great 2'



Using this, we'll discuss algorithms based on subsampling

- Does not look at the whole problem!
- Algorithm can be **sublinear** time



'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

Can do via solving a linear system:

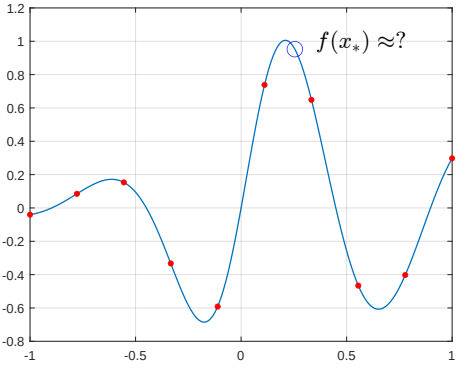
$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \ddots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

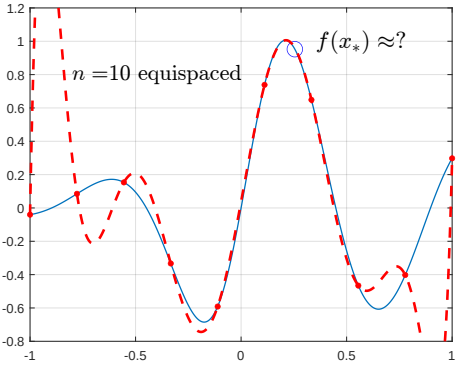


'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

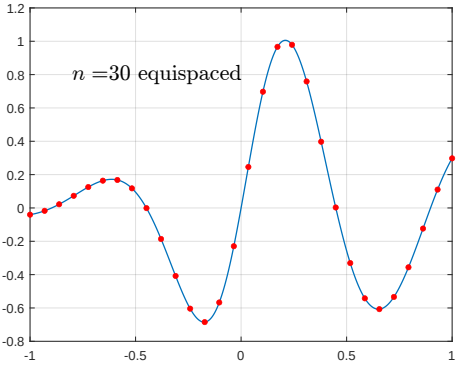


'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

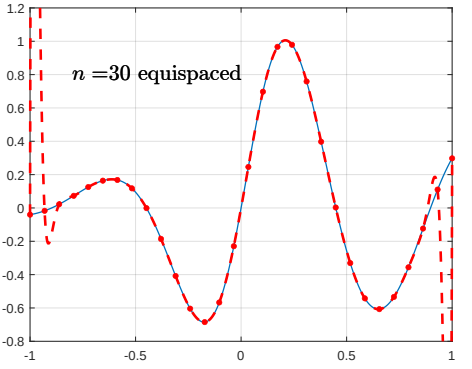


'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

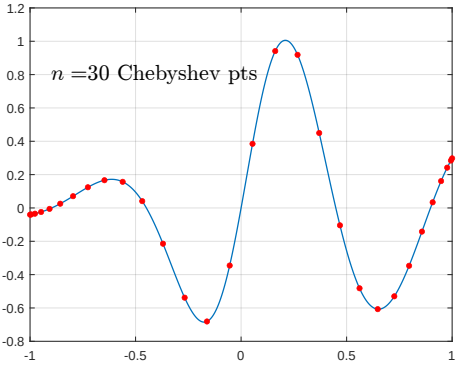


'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

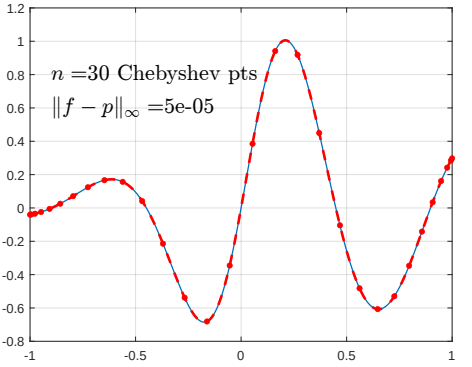


'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$

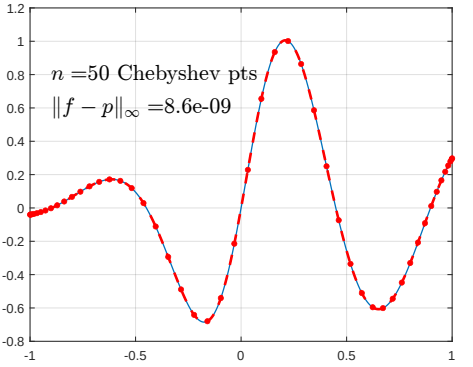


'Sidetracking': Polynomial interpolation/approximation

Goal: given continuous $f : [-1, 1] \to \mathbb{R}$, evaluate/sample f at $\{x_i\}_{i=0}^n$ to find $f(x) \approx p_n(x) = \sum_{i=0}^n c_i x^i$ for $x \in [-1, 1]$

Interpolation: find unique p_n s.t. see [Trefethen ATAP book 2019]

$$f(x_i) = p_n(x_i), \quad i = 0, \ldots, n$$



Lebesgue constant for Chebyshev and equispaced interpolation

$$\|f - p_n\|_\infty \leq (\Lambda_n + 1) \cdot \underbrace{\min_{p \in \Pi_n} \|f - p\|_\infty}_{\to 0 \text{ rapidly if } f \text{ smooth}}$$

where $\Lambda_n := \sup_f \frac{\|p_n\|_\infty}{\|f\|_\infty}$, $\|f\|_\infty := \max_{x \in [-1,1]} |f(x)|$.

Lebesgue constant for Chebyshev and equispaced interpolation

$$\|f - p_n\|_\infty \leq (\Lambda_n + 1) \cdot \underbrace{\min_{p \in \Pi_n} \|f - p\|_\infty}_{\to 0 \text{ rapidly if } f \text{ smooth}}$$

where $\Lambda_n := \sup_f \frac{\|p_n\|_\infty}{\|f\|_\infty}$, $\|f\|_\infty := \max_{x \in [-1,1]} |f(x)|$.

Proof: $\|f - p_n\|_\infty \leq \underbrace{\|f - p_*\|_\infty}_{\|\mathcal{L}(p_n - p_*)\|_\infty \leq \Lambda_n \|p_n - p_*\|_\infty} + \|p_n - p_*\|_\infty$

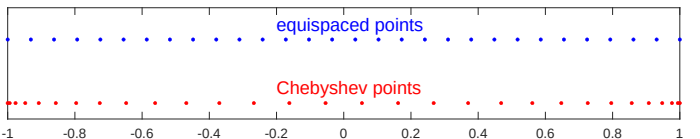
Lebesgue constant for Chebyshev and equispaced interpolation

$$\|f - p_n\|_\infty \le (\Lambda_n + 1) \cdot \underbrace{\min_{p \in \Pi_n} \|f - p\|_\infty}_{\to 0 \text{ rapidly if } f \text{ smooth}}$$

where $\Lambda_n := \sup_f \frac{\|p_n\|_\infty}{\|f\|_\infty}$, $\|f\|_\infty := \max_{x \in [-1,1]} |f(x)|$.

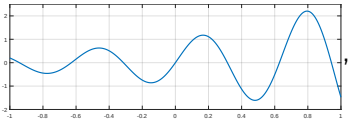
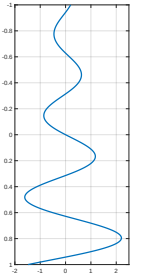
Proof: $\|f - p_n\|_\infty \le \underbrace{\|f - p_*\|_\infty}_{\|\mathcal{L}(p_n - p_*)\|_\infty \le \Lambda_n \|p_n - p_*\|_\infty} + \|p_n - p_*\|_\infty$

- Equispaced points: $\Lambda_n \approx 2^n$
- Chebyshev points $x_i = \cos\theta_i$, $\theta_i = \frac{(2i+1)\pi}{2(n+1)}$: $\Lambda_n \approx \frac{2}{\pi} \log n$



Please get used to two things

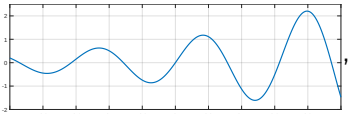
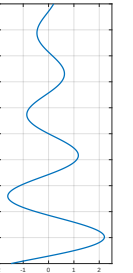
1. I'll plot functions vertically (rotated):

   Instead of  , I'll use 

2. I'll use Chebyshev polynomials:
   $T_k(\cos\theta) = \cos(k\theta)$; $T_k(x) = 2^k x^k + \cdots$
   note: $p(x) = \sum_{k=0}^{n} \hat{c}_k x^k$ then $p(x) = \sum_{k=0}^{n} c_k T_k(x)$

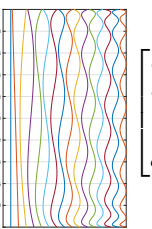
Please get used to two things

1. I'll plot functions vertically (rotated):

   Instead of  , I'll use 

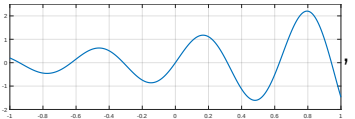
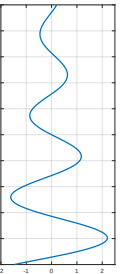
2. I'll use Chebyshev polynomials:
   $T_k(\cos\theta) = \cos(k\theta)$; $T_k(x) = 2^k x^k + \cdots$
   note: $p(x) = \sum_{k=0}^{n} \hat{c}_k x^k$ then $p(x) = \sum_{k=0}^{n} c_k T_k(x)$

Then polynomial interpolation is

$$p_n(x) = \sum_{i=0}^{n} c_i T_i(x) = \begin{bmatrix} T_0(x) & T_1(x) & \dots & T_n(x) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} =$$  $$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Please get used to two things

1. I'll plot functions vertically (rotated):

   Instead of  , I'll use 

2. I'll use Chebyshev polynomials:

   $T_k(\cos\theta) = \cos(k\theta)$; $T_k(x) = 2^k x^k + \cdots$

   note: $p(x) = \sum_{k=0}^{n} \hat{c}_k x^k$ then $p(x) = \sum_{k=0}^{n} c_k T_k(x)$

$$p_n(x) = \quad = \quad \begin{bmatrix} T_0(x_0) & T_1(x_0) & \dots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \dots & T_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(x_n) & T_1(x_n) & \dots & T_n(x_n) \end{bmatrix}^{-1} \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

Polynomial interpolation and CUR: What they have in common

We've covered

- Low-rank approximation via CUR,
- Polynomial interpolation.

Polynomial interpolation and CUR: What they have in common

We've covered

- Low-rank approximation via CUR,
- Polynomial interpolation.

Main message:

Polynomial interpolation is low-rank approximation

Polynomial interpolation and CUR: What they have in common

We've covered

- Low-rank approximation via CUR,
- Polynomial interpolation.

Main message:

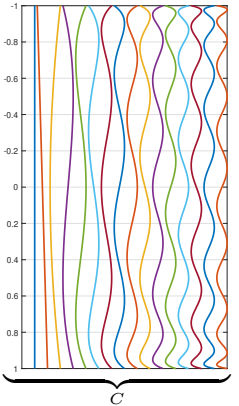
Polynomial interpolation is low-rank approximation

That is, for $f_i : [-1, 1] \to \mathbb{R}$,

$$\underbrace{\boxed{f_1, f_2, \ldots, f_\infty}}_{[-1,1]\times\infty} \approx \underbrace{\boxed{}}_{C,\ [-1,1]\times n} \underbrace{\begin{bmatrix} T_0(x_0) & T_1(x_0) & \ldots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \ldots & T_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(x_n) & T_1(x_n) & \ldots & T_n(x_n) \end{bmatrix}^{-1}}_{U,\ n\times n} \underbrace{\begin{bmatrix} f_1(x_0) & f_2(x_0) & \ldots & \ldots & \ldots & f_\infty(x_0) \\ f_1(x_1) & f_2(x_1) & \ldots & \ldots & \ldots & f_\infty(x_1) \\ \vdots & \vdots & \ldots & \ldots & \ldots & \vdots \\ f_1(x_n) & f_2(x_n) & \ldots & \ldots & \ldots & f_\infty(x_n) \end{bmatrix}}_{R,\ n\times\infty}$$

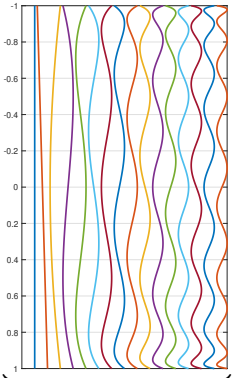
Polynomial (Chebyshev) interpolation as CUR

$$[T_0(x), T_1(x), \ldots, T_n(x)] =$$



Polynomial (Chebyshev) interpolation as CUR

$$[T_0(x), T_1(x), \ldots, T_n(x)] =$$



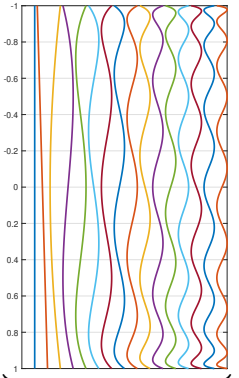
$$\underbrace{}_{C}$$

$$\underbrace{\begin{bmatrix} T_0(x_0) & T_1(x_0) & \ldots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \ldots & T_n(x_1) \\ \vdots & \vdots & \ldots & \vdots \\ T_0(x_n) & T_1(x_n) & \ldots & T_n(x_n) \end{bmatrix}^{-1}}_{U}$$

$$\underbrace{\begin{bmatrix} T_0(x_0) & T_1(x_0) & \ldots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \ldots & T_n(x_1) \\ \vdots & \vdots & \ldots & \vdots \\ T_0(x_n) & T_1(x_n) & \ldots & T_n(x_n) \end{bmatrix}}_{R}$$

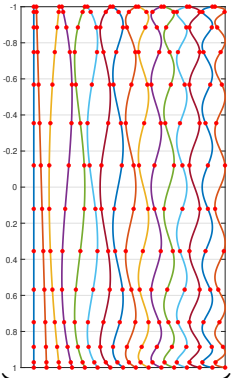
Polynomial (Chebyshev) interpolation as CUR

$$[T_0(x), T_1(x), \ldots, T_n(x)] =$$



C

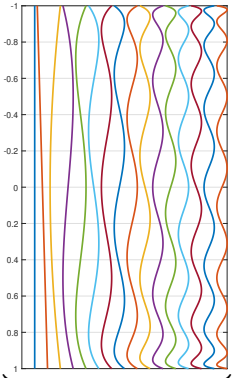
$$\underbrace{\begin{bmatrix} T_0(x_0) & T_1(x_0) & \ldots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \ldots & T_n(x_1) \\ \vdots & \vdots & \ldots & \vdots \\ T_0(x_n) & T_1(x_n) & \ldots & T_n(x_n) \end{bmatrix}^{-1}}_{U}$$



R

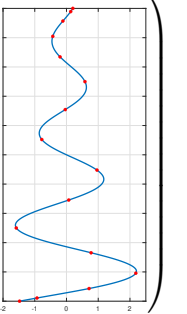
Polynomial (Chebyshev) interpolation as CUR

$$f(x) \approx \sum_{i=0}^{n} c_i T_i(x) =$$



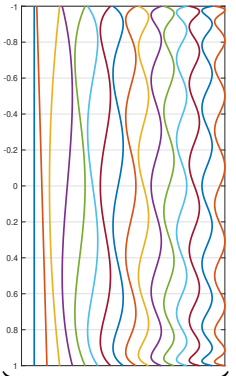
C

$$\underbrace{\begin{bmatrix} T_0(x_0) & T_1(x_0) & \dots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \dots & T_n(x_1) \\ \vdots & \vdots & \dots & \vdots \\ T_0(x_n) & T_1(x_n) & \dots & T_n(x_n) \end{bmatrix}^{-1}}_{U} \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} =$$



Polynomial (Chebyshev) interpolation as CUR

$$[f_1(x), f_2(x), \ldots, f_\infty(x)] \approx [p_1(x), p_2(x), \ldots, p_\infty(x)]$$



C

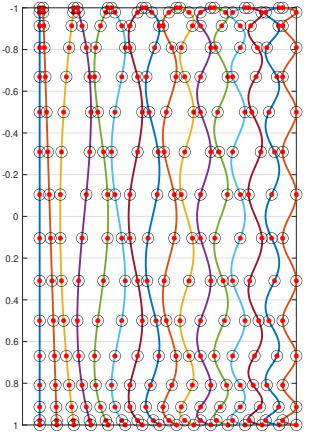
$$\underbrace{\begin{bmatrix} T_0(x_0) & T_1(x_0) & \ldots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \ldots & T_n(x_1) \\ \vdots & \vdots & \ldots & \vdots \\ T_0(x_n) & T_1(x_n) & \ldots & T_n(x_n) \end{bmatrix}^{-1}}_{U} \underbrace{\begin{bmatrix} f_1(x_0) & f_2(x_0) & \ldots & \ldots & f_\infty(x_0) \\ f_1(x_1) & f_2(x_1) & \ldots & \ldots & f_\infty(x_1) \\ \vdots & \vdots & \ldots & \ldots & \vdots \\ f_1(x_n) & f_2(x_n) & \ldots & \ldots & f_\infty(x_n) \end{bmatrix}}_{R}$$

Do same for $f = f_1, f_2, \ldots, f_\infty$;

CUR low-rank approx for "$\infty \times \infty$ matrix"

$f \approx p_n$ via subsampling: only observing f at x_i

Chebyshev interpolation pts vs. GEPP



Red: Chebyshev points
Black circle: Gaussian elimination with partial pivots

- Chebyshev interpolation is pillar of NA
- Points chosen by NLA tools are almost as good! And generalizes to other settings

Generalizing Chebyshev interpolation?

We've interpreted polynomial interpolation as low-rank approximation.
Now attempt to generalize! To approximate vectors $\mathbf{x} \in \mathbb{R}^N$

Key steps in interpolation:

1. Fix subspace (polynomials)
2. Find approximation in subspace by interpolation

Generalizing Chebyshev interpolation?

We've interpreted polynomial interpolation as low-rank approximation.
Now attempt to generalize! To approximate vectors $\mathbf{x} \in \mathbb{R}^N$

Key steps in interpolation:

1. Fix subspace (polynomials)
2. Find approximation in subspace by interpolation

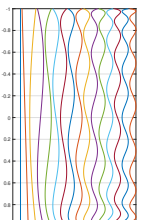
Generalization:

- Function $f \approx p$ polynomial vs. Signal $\mathbf{x} \in \mathbb{R}^N \approx$ vector in subspace $X \subseteq \mathbb{R}^N$
- Subspace: Polynomial vs. learned/snapshot (PCA/SVD)
- Algorithm: interpolating f at $\{x_i\}_{i=0}^n$ vs. Subsample $\min_{\mathbf{c}} \|X\mathbf{c} - \mathbf{x}\|_2$

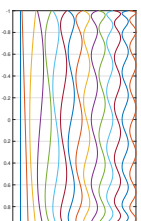
Results in **DEIM** (discrete empirical interpolation method)

[Chaturantabut-Sorensen 2010]

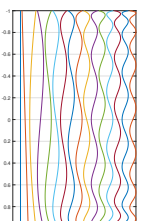
Continuous vs. Discrete interpolation

Aspect	**Continuous**	**Discrete**
object	$f \in C[-1, 1]$	
subspace	Polynomials	
approximation	$f(x) \approx p(x) = \sum_{i=0}^{n} c_i T_i(x)$	
	$p(x) =$  $\mathbf{c}$	
interpolation	$f(x_i) = p(x_i)$	

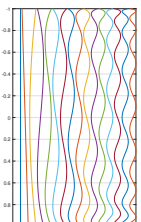
Continuous vs. Discrete interpolation

Aspect	**Continuous**	**Discrete**
object	$f \in C[-1,1]$	$\mathbf{x} \in \mathbb{R}^N$
subspace	Polynomials	
approximation	$f(x) \approx p(x) = \sum_{i=0}^{n} c_i T_i(x)$	
	$p(x) =$  $\mathbf{c}$	
interpolation	$f(x_i) = p(x_i)$	

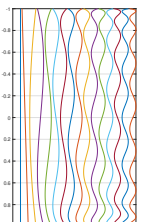
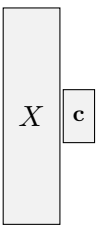
Continuous vs. Discrete interpolation

Aspect	**Continuous**	**Discrete**
object	$f \in C[-1, 1]$	$\mathbf{x} \in \mathbb{R}^N$
subspace	Polynomials	Subspace $X \subset \mathbb{R}^N$, e.g. PCA
approximation	$f(x) \approx p(x) = \sum_{i=0}^{n} c_i T_i(x)$	
	$p(x) =$  $\mathbf{c}$	
interpolation	$f(x_i) = p(x_i)$	

Continuous vs. Discrete interpolation

Aspect	Continuous	Discrete
object	$f \in C[-1, 1]$	$\mathbf{x} \in \mathbb{R}^N$
subspace	Polynomials	Subspace $X \subset \mathbb{R}^N$, e.g. PCA
approximation	$f(x) \approx p(x) = \sum_{i=0}^{n} c_i T_i(x)$	$\mathbf{x} \approx \hat{\mathbf{x}} = X\mathbf{c}$
	$p(x) =$  $\mathbf{c}$	$\hat{\mathbf{x}} =$ X $\mathbf{c}$
interpolation	$f(x_i) = p(x_i)$	

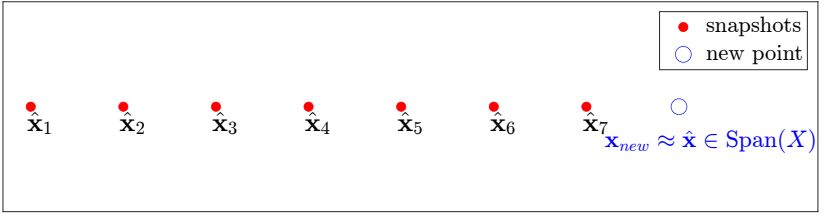
Continuous vs. Discrete interpolation

Aspect	Continuous	Discrete
object	$f \in C[-1,1]$	$\mathbf{x} \in \mathbb{R}^N$
subspace	Polynomials	Subspace $X \subset \mathbb{R}^N$, e.g. PCA
approximation	$f(x) \approx p(x) = \sum_{i=0}^{n} c_i T_i(x)$	$\mathbf{x} \approx \hat{\mathbf{x}} = X\mathbf{c}$
	$p(x) =$  $\mathbf{c}$	$\hat{\mathbf{x}} =$ 
interpolation	$f(x_i) = p(x_i)$	$X\mathbf{c} = \mathbf{x}$ at r coordinates
		(or $\min_c \|X\mathbf{c} - \mathbf{x}\|_2$)

Subspace X via snapshot+PCA

We'll take $X = [\mathbf{x}_1, \ldots, \mathbf{x}_\ell]$: **snapshot**, or representative signals

(think $\mathbf{x}_t$ signal at time t; then expect $\mathbf{x}_{new} \approx Xc$).



Subspace X via snapshot+PCA

We'll take $X = [\mathbf{x}_1, \ldots, \mathbf{x}_\ell]$: **snapshot**, or representative signals

(think $\mathbf{x}_t$ signal at time t; then expect $\mathbf{x}_{new} \approx Xc$).

- PCA: find most active directions via SVD

$$X = \sum_{i=1}^{\ell} \sigma_i u_i v_i^T, \quad X \leftarrow [u_1, u_2, \ldots, u_r]$$

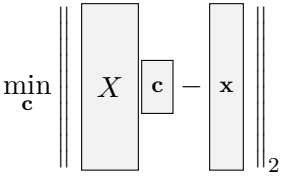
- Under smoothness/analyticity assumptions,

[Kressner-Tobler 11, Cohen-DeVore 16]

$$\min_c \|X\mathbf{c} - \mathbf{x}_{new}\|_2 = O(\exp(-Cr))$$

Approximation in subspace via subsampling

Given snapshot X, approximate new signal $\mathbf{x} \approx X\mathbf{c}$ via least squares



Approximation in subspace via subsampling

Given snapshot X, approximate new signal $\mathbf{x} \approx X\mathbf{c}$ via least squares

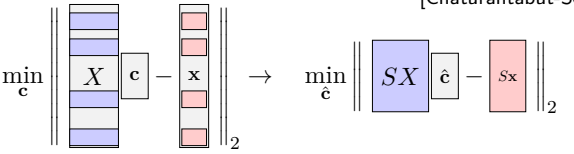
$$\min_{\mathbf{c}} \left\| X\,\mathbf{c} - \mathbf{x} \right\|_2$$

We'll solve via subsampling: $S = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$

$$\min_{\mathbf{c}} \left\| X\,\mathbf{c} - \mathbf{x} \right\|_2 \quad \rightarrow \quad \min_{\hat{\mathbf{c}}} \left\| SX\,\hat{\mathbf{c}} - S\mathbf{x} \right\|_2 .$$

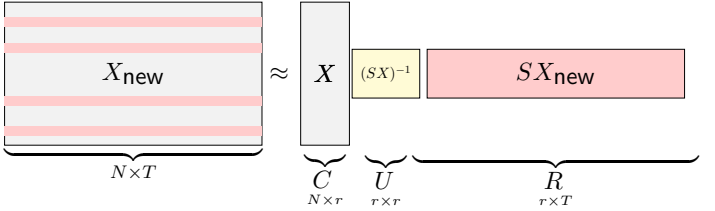
DEIM: Approximation in subspace via subsampling=CUR

[Chaturantabut-Sorensen 2010]



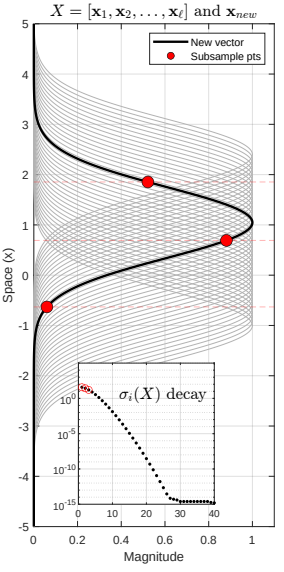
- Only $r \ll N$ evaluations $S\mathbf{x}$ needed to get $\mathbf{x} \approx X\hat{\mathbf{c}}$, speedup $O(\frac{N}{r})$

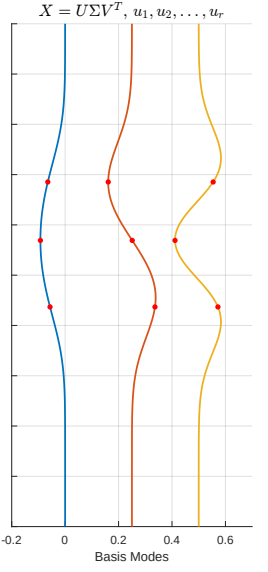
  aka active learning, gappy POD etc [Everson-Sirovich 95, Willcox 06]
- Variants wrt choice of S: Q-DEIM [Drmac-Gugercin 16], use of [Gu-Eisenstat 96] etc
- Applications: PDEs, recommendation systems,...
- **DEIM is CUR**: For new signals $X_{\text{new}} := [\mathbf{x}_{\ell+1}, \ldots, \mathbf{x}_{\ell+T}]$,

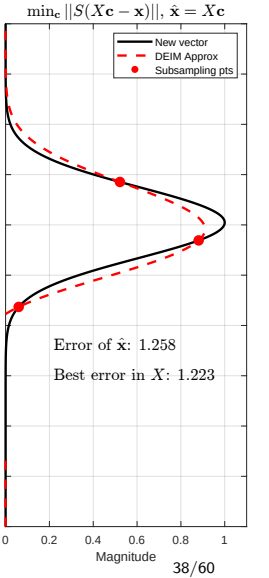


DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 3 Samples

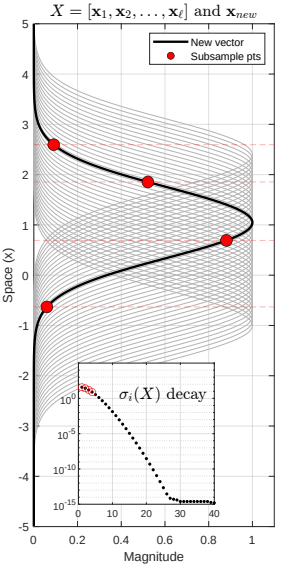


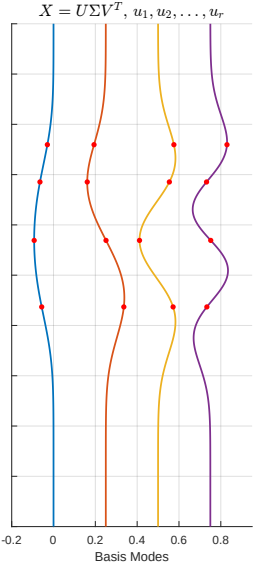


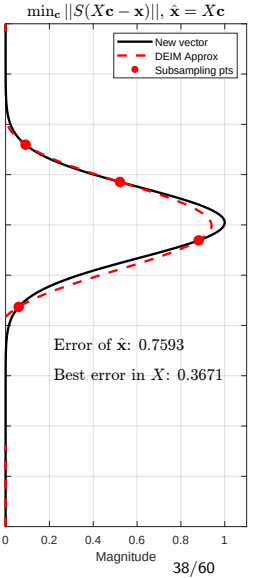


DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 4 Samples

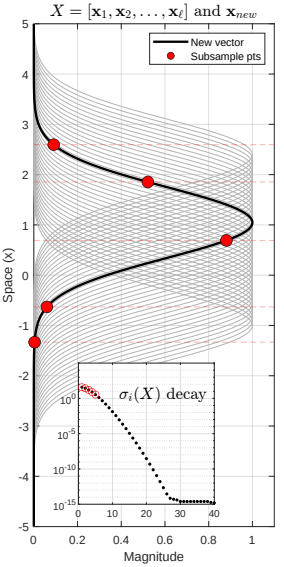


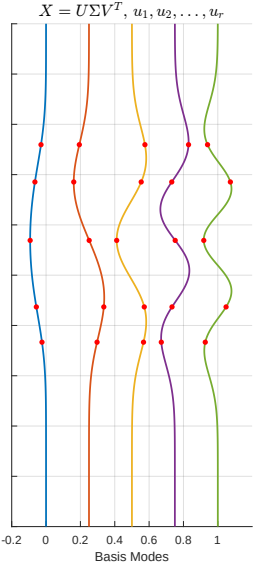


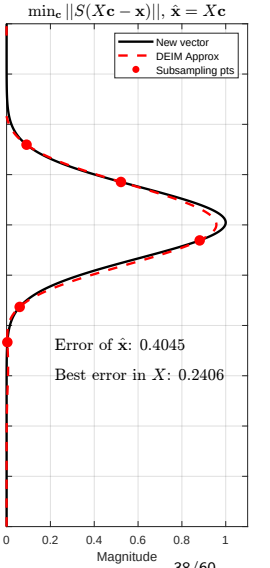


DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 5 Samples

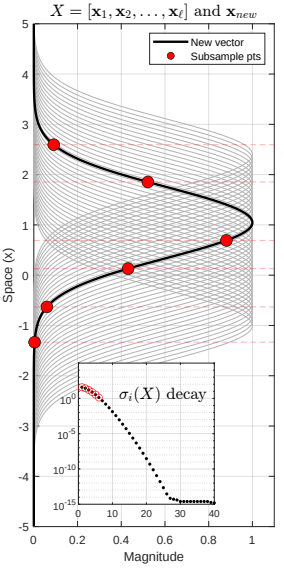


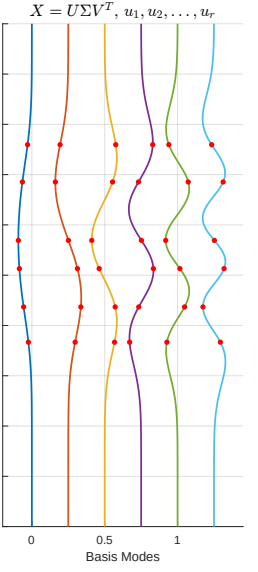


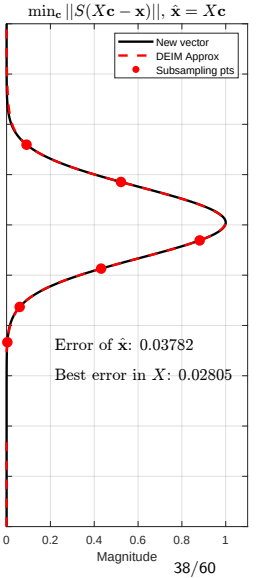


DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 6 Samples

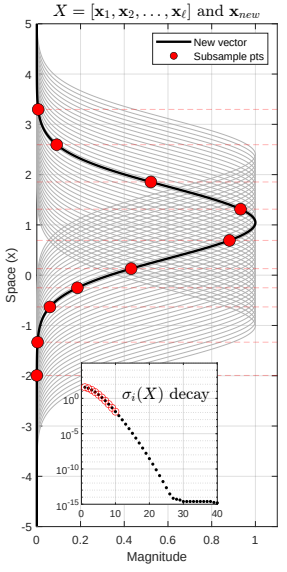


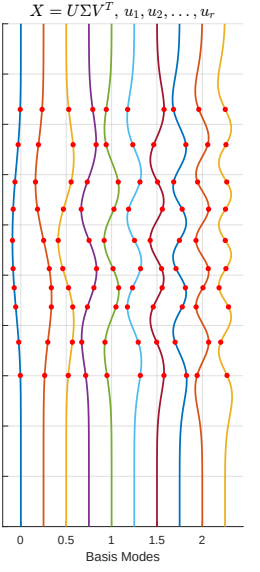


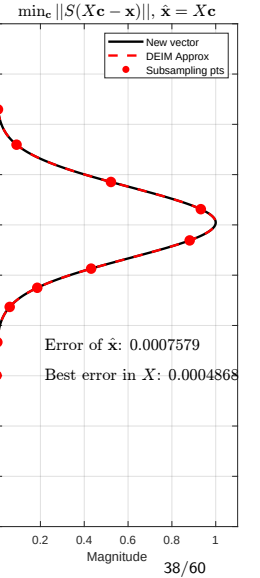


DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 10 Samples

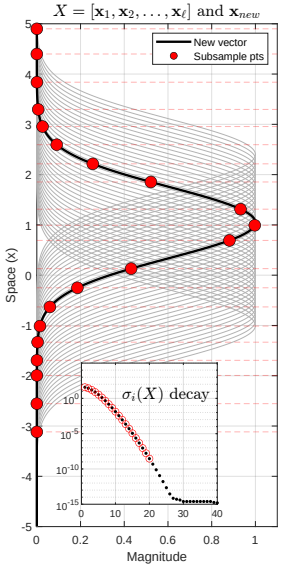


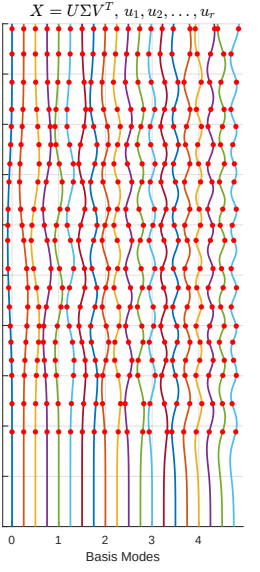


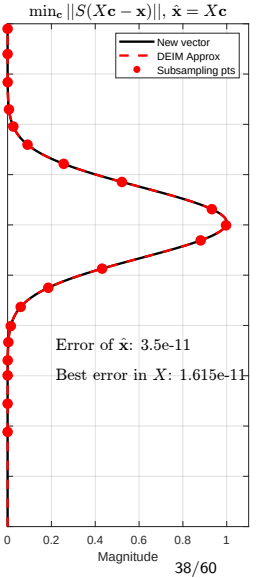


DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 20 Samples

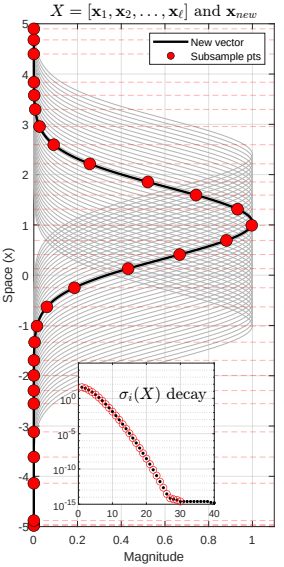


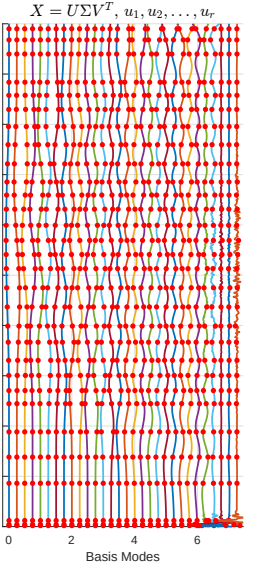




DEIM Illustration $f(x,\mu) = \exp((x-\mu)^2/2)$

DEIM: Reconstructing a new vector from 30 Samples





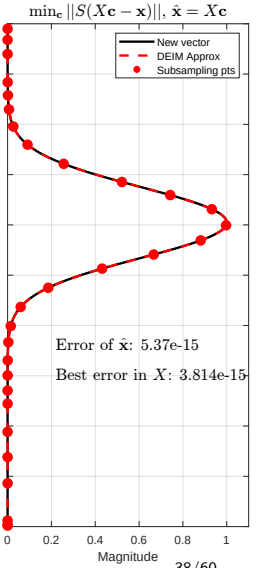
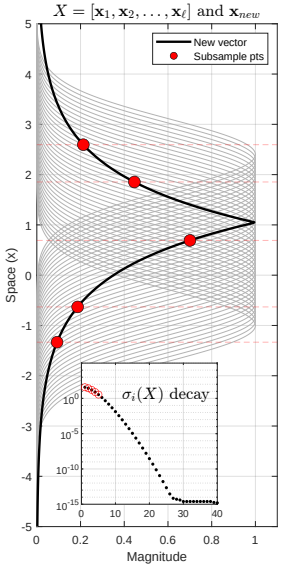
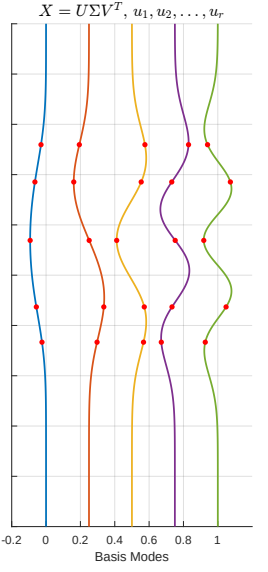


Illustration PCA+subsampling w/ $f(x, \mu) = \exp(-|x - \mu|)$

DEIM: Reconstructing a new vector from 5 Samples





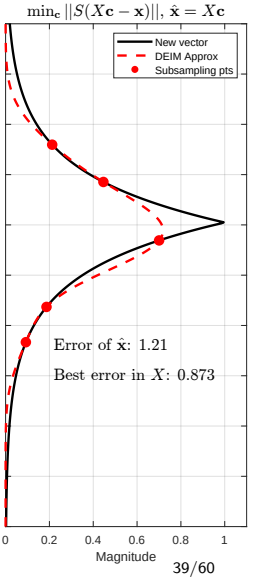
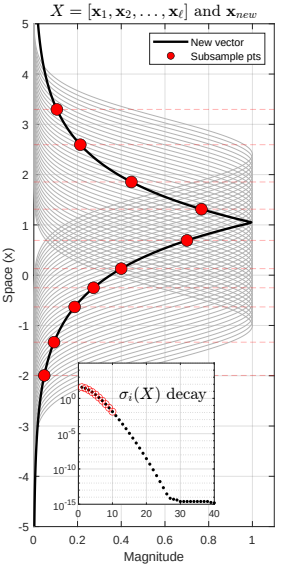
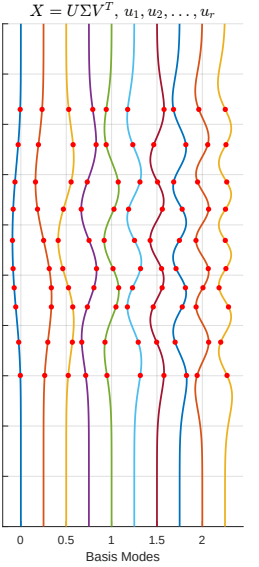


Illustration PCA+subsampling w/ $f(x, \mu) = \exp(-|x - \mu|)$

DEIM: Reconstructing a new vector from 10 Samples





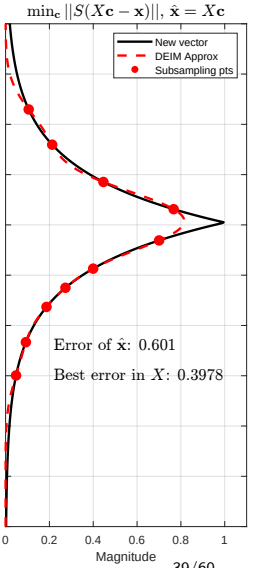
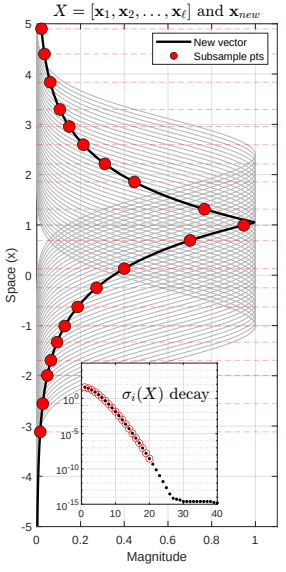
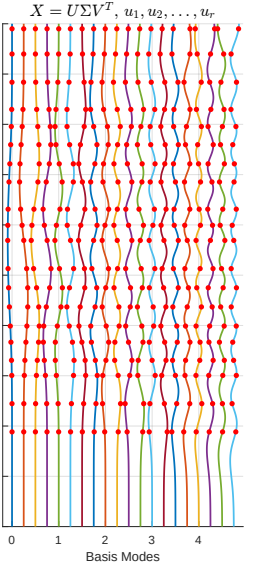


Illustration PCA+subsampling w/ $f(x, \mu) = \exp(-|x - \mu|)$

DEIM: Reconstructing a new vector from 20 Samples





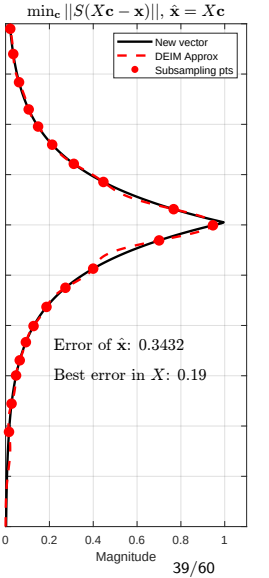
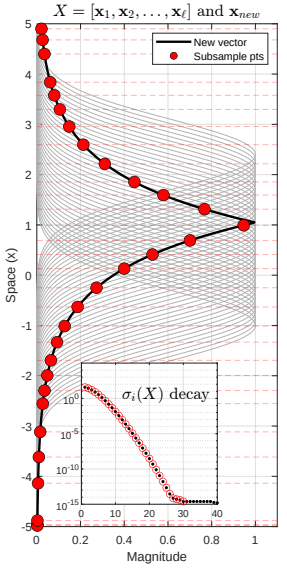
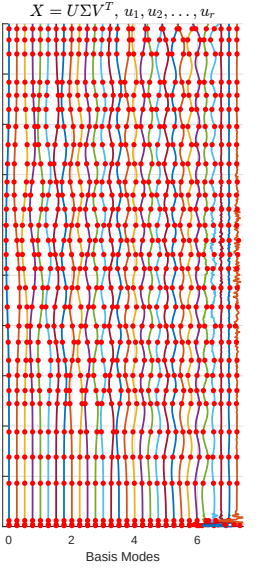
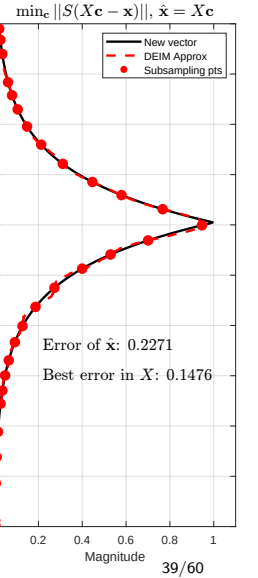


Illustration PCA+subsampling w/ $f(x, \mu) = \exp(-|x - \mu|)$

DEIM: Reconstructing a new vector from 30 Samples







Where do tall-skinny problems come from?

We'll spend some time discussing

$$\min_x \left\| \boxed{A}\,\boxed{x} - \boxed{b} \right\|_2$$

- Naturally arises in DEIM, as we've seen
- Also when linear system $Mx = b$ is solved for $M \in \mathbb{R}^{n\times n}$, but a **subspace** method is used. $x \in \mathsf{Span}(X) \Leftrightarrow x = \boxed{X}\,c$,

  $\min_{x\in\mathsf{Span}(X)} \|Mx - b\|_2 = \min_{x\in\mathsf{Span}(X)} \|MXc - b\|_2$
  e.g. X: Krylov subspace, snapshot (coming)
- Massively oversampled problems (e.g. for denoising)

Sketch-and-solve for tall-skinny least-squares problems

For A: $N \times r$, $N \gg r$

$$\min_x \left\| A\,x - b \right\|_2 \quad \Rightarrow \quad \min_{\hat{x}} \left\| SA\,\hat{x} - Sb \right\|_2$$

With **sketch** $S \in \mathbb{R}^{s \times N}$ $(s \geq r)$ oblivious subspace embedding,

$$(1-\epsilon)\|Av - b\|_2 \leq \|S(Av - b)\|_2 \leq (1+\epsilon)\|Av - b\|_2,$$

for some ϵ (not small, e.g. $\epsilon = \frac{1}{2}$) Hence the sketched solution $\hat{x}$ satisfies

$$\|A\hat{x} - b\|_2 \leq \frac{1+\epsilon}{1-\epsilon}\|Ax_* - b\|_2.$$

- if $\|Ax_* - b\|_2$ is small, $\hat{x}$ is a great solution! (x_*: exact soln)

Residual bounds for sketched least squares

Let $A \in \mathbb{R}^{N\times r}, b \in \mathbb{R}^N$ with $N > r$, $x_*, \hat{x}$: soln for $\min_x \|Ax - b\|_2$ and $\min_{\hat{x}} \|S(Ax - b)\|_2$, where $S \in \mathbb{R}^{s\times N}$ with $s \geq r$. Then

- b-oblivious bound, holds for all b

$$\frac{\|A\hat{x} - b\|_2}{\|Ax_* - b\|_2} \leq \frac{\|S\|_2}{\sigma_{\min}(SQ)},$$

where $A = QR$ is the thin QR.

PROOF: Kressner's lecture day 1. (or next slide).

Residual bounds for sketched least squares

Let $A \in \mathbb{R}^{N\times r}, b \in \mathbb{R}^N$ with $N > r$, $x_*, \hat{x}$: soln for $\min_x \|Ax - b\|_2$ and $\min_{\hat{x}} \|S(Ax - b)\|_2$, where $S \in \mathbb{R}^{s\times N}$ with $s \geq r$. Then

- b-oblivious bound, holds for all b

$$\frac{\|A\hat{x} - b\|_2}{\|Ax_* - b\|_2} \leq \frac{\|S\|_2}{\sigma_{\min}(SQ)},$$

where $A = QR$ is the thin QR.

PROOF: Kressner's lecture day 1. (or next slide).

- b-dependent bound

$$\frac{\|A\hat{x} - b\|_2}{\|Ax_* - b\|_2} \leq \kappa_2(S\tilde{Q})$$

where $\tilde{Q}\tilde{R} = [A\ b] \in \mathbb{R}^{n\times(r+1)}$.

Usually better than oblivious, but b often unknown

Lemma on sketched (block) least squares proof

Write $B = QQ^TB + (I - QQ^T)B =: B_1 + B_2$. As $X = A^\dagger B = R^{-1}Q^TB$,

$$\|AX - B\|_2 = \|QR(R^{-1}Q^TB) - B_1 - B_2\|_2 = \|QQ^TB - B_1 - B_2\|_2 = \|B_2\|_2.$$

The solution of $\min_{\hat{X}} \|S(A\hat{X} - B)\|_F$ is

$$\begin{aligned}\hat{X} &= (SA)^\dagger(SB) = (SA)^\dagger S(B_1 + B_2) = (SQR)^\dagger S(QQ^TB + B_2) \\ &= R^{-1}(SQ)^\dagger(SQ)Q^TB + R^{-1}(SQ)^\dagger SB_2 = R^{-1}Q^TB + R^{-1}(SQ)^\dagger SB_2.\end{aligned}$$

Therefore

$$A\hat{X} = QR(R^{-1}Q^TB + R^{-1}(SQ)^\dagger SB_2) = QQ^TB + Q(SQ)^\dagger SB_2 = B_1 + Q(SQ)^\dagger SB_2,$$

$$\begin{aligned}\|A\hat{X} - B\|_F &= \|Q(SQ)^\dagger SB_2 - B_2\|_F = \|(Q(SQ)^\dagger S - I)B_2\|_F \le \|Q(SQ)^\dagger S - I\|_2\|B_2\|_F \\ &= \|Q(SQ)^\dagger S\|_2\|B_2\|_F \qquad \text{as } Q(SQ)^\dagger S \text{ is an (oblique) projection} \\ &= \|(SQ)^\dagger S\|_2\|B_2\|_F \le \|(SQ)^\dagger\|_2\|S\|_2\|B_2\|_F \\ &= \frac{\|S\|_2}{\sigma_{\min}(SQ)}\|AX - B\|_F.\end{aligned}$$

Proof of b-dependent bound

Let $[A, b] = QR$. Note $S[A, b] = (SQ)R$, $\|S[A, b]v\|_2 = \|(SQ)Rv\|_2$, and by Courant-Fischer $\sigma_n(SQ)\|Rv\|_2 \le \|(SQ)Rv\|_2 \le \sigma_1(SQ)\|Rv\|_2$.

$\|A\hat{x} - b\|_2 = \|[A, b]\begin{bmatrix}\hat{x}\\ -1\end{bmatrix}\|_2$; let $v = \begin{bmatrix}\hat{x}\\ -1\end{bmatrix}$ and use $\|Rv\|_2 = \|[A, b]v\|_2$:

$$\|S(A\hat{x} - b)\|_2 = \|S[A, b]v\|_2 \ge \sigma_n(SQ)\|A\hat{x} - b\|_2. \tag{1}$$

Similarly, by taking $v = \begin{bmatrix}x_*\\ -1\end{bmatrix}$ we obtain

$$\|S(Ax_* - b)\|_2 = \|S[A, b]v\|_2 \le \sigma_1(SQ)\|Ax_* - b\|_2. \tag{2}$$

$$\begin{aligned}
\|A\hat{x} - b\|_2 &\le \frac{1}{\sigma_n(SQ)}\|S(A\hat{x} - b)\|_2 && \text{(by (1))}\\
&\le \frac{1}{\sigma_n(SQ)}\|S(Ax_* - b)\|_2 && \text{(by optimality of } \hat{x} \text{ for } \min_x \|S(Ax - b)\|_2)\\
&\le \frac{\sigma_1(SQ)}{\sigma_n(SQ)}\|Ax_* - b\|_2 && \text{(by (2))}\\
&= \kappa_2(SQ)\|Ax_* - b\|_2.
\end{aligned}$$

Leverage scores: Definition and properties

Leverage score sampling is one (but not only) way to do sketched least squares by **subsampling** in the b-oblivious case.

Definition: Let $A = QR$. Then $\tau_i(A) = \|q_i\|^2$ (q_i: ith row of Q)

Properties:

1. Bounds: $0 \leq \tau_i(A) \leq 1$.
2. Total Leverage (Rank sum): $\sum_{i=1}^{N} \tau_i(A) = \text{rank}(A) = r$.
3. Invariance: $\tau_i(AB) = \tau_i(A)$ for any full-rank $B \in \mathbb{R}^{r \times r}$.

Subspace embedding via Leverage score sampling

Sampling Strategy:

- Define row sampling probabilities: $p_i = \frac{\tau_i(A)}{r}$ for $i \in \{1, \ldots, N\}$.
- Sample k indices $s_1, \ldots, s_s \in \{1, \ldots, N\}$ i.i.d. according to $\{p_i\}$.
- Construct sample-and-rescale matrix $S \in \mathbb{R}^{s \times N}$ where row j has a single non-zero entry $\frac{1}{\sqrt{sp_{s_j}}}$ at index s_j.

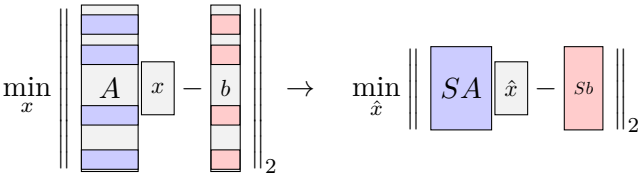
Theorem (Subspace Embedding Guarantee): Fix $\varepsilon, \delta > 0$. If $k = O\left(\frac{r \log(r/\delta)}{\varepsilon^2}\right)$, then with probability at least $1 - \delta$:

$$(1-\varepsilon)\|Ax\|_2^2 \leq \|SAx\|_2^2 \leq (1+\varepsilon)\|Ax\|_2^2 \quad \forall x \in \mathbb{R}^r$$

Proof by e.g. matrix Chernoff [Tropp 12]; note $(SQ)^T SQ = \sum_{j=1}^{s} \underbrace{\frac{1}{s\|q_{s_j}\|^2} q_{s_j} q_{s_j}^T}_{\text{equal norm}}$

With orthonormal $Q \in \mathbb{R}^{N \times r}$, equivalent to $\|Q^T S^T SQ - I\|_2 \leq \varepsilon$.

Subsampled least squares (aka active learning) via leverage score



Relative Error Approximation: If $s = \Omega\left(d\log(d/\delta) + \frac{d}{\delta\varepsilon}\right)$, then with probability $\geq 1-\delta$: [Clarkson-Woodruff 17, R. Meyer's blog]

$$\|A\tilde{x} - b\|_2 \leq (1+\varepsilon)\min_{x\in\mathbb{R}^d}\|Ax - b\|_2$$

(Randomized) leverage score subsampling achieves excellent b-oblivious residual bound with good probability.

Rand vs. deterministic, probabilistic vs. worst-case

$$\|A\tilde{x} - b\|_2 \leq (1+\varepsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2 \tag{3}$$

Holds with good (not V high) probability. What is the worst case wrt b? It is back to

$$\frac{\|A\hat{x} - b\|_2}{\|Ax_* - b\|_2} \leq \frac{\|S\|_2}{\sigma_{\min}(SQ)}. \tag{4}$$

Problem:

- $1/\sigma_{\min}(SQ)$ usually depends on N, like $\sqrt{N}$ (when Q Haar)
- N large, possibly $N = \infty$ in function (quasimatrix) regression
- This is UNAVOIDABLE: $\exists b$ s.t. (4) sharp.
  Deterministic+worst-case+$N \to \infty \Rightarrow$ worst-case bound $\to \infty$!

Rand vs. deterministic, probabilistic vs. worst-case

$$\|A\tilde{x} - b\|_2 \leq (1+\varepsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2 \tag{3}$$

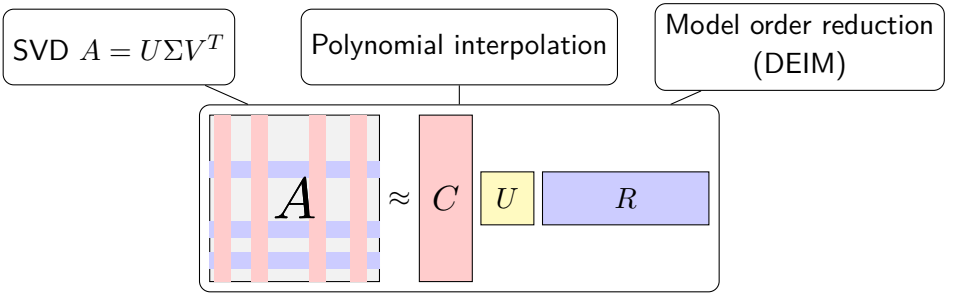
Holds with good (not V high) probability. What is the worst case wrt b? It is back to

$$\frac{\|A\hat{x} - b\|_2}{\|Ax_* - b\|_2} \leq \frac{\|S\|_2}{\sigma_{\min}(SQ)}. \tag{4}$$

Problem:

- $1/\sigma_{\min}(SQ)$ usually depends on N, like $\sqrt{N}$ (when Q Haar)
- N large, possibly $N = \infty$ in function (quasimatrix) regression
- This is UNAVOIDABLE: $\exists b$ s.t. (4) sharp.
  Deterministic+worst-case+$N \to \infty \Rightarrow$ worst-case bound $\to \infty$!
- Randomized algorithms allow us to get (3) by avoiding pathological example with good probability. [Shimizu-Cheng-Musco-Weare 24 etc]

'Summary' so far



Having reinvented two wheels, we'll next do something new the insight!

CUR app 3: Parameter-dependent linear systems

Increasingly important+common: solving **many** related linear systems.

$$\underbrace{A(p)}_{N\times N}\underbrace{x(p)}_{N\times 1} = \underbrace{b(p)}_{N\times 1} \tag{5}$$

where $p \in \Omega \subset \mathbb{R}^d$ is a parameter, or just many related $A_i x_i = b_i$.

Goal: solve (5) for many values of p

CUR app 3: Parameter-dependent linear systems

Increasingly important+common: solving **many** related linear systems.

$$\underbrace{A(p)}_{N\times N}\underbrace{x(p)}_{N\times 1} = \underbrace{b(p)}_{N\times 1} \tag{5}$$

where $p \in \Omega \subset \mathbb{R}^d$ is a parameter, or just many related $A_i x_i = b_i$.
Goal: solve (5) for many values of p

Arise in

- PDEs (e.g. time-dependent or parameter-dependent)
- Model order reduction
- Hyper-parameter tuning (kernel ridge regression etc)
- Nonlinear eigenvalue problems
- Newton's method, modified Laplacian system, etc.

CUR app 3: Parameter-dependent linear systems

Increasingly important+common: solving **many** related linear systems.

$$\underbrace{A(p)}_{N\times N}\underbrace{x(p)}_{N\times 1} = \underbrace{b(p)}_{N\times 1} \tag{5}$$

where $p \in \Omega \subset \mathbb{R}^d$ is a parameter, or just many related $A_i x_i = b_i$.
Goal: solve (5) for many values of p

Our algorithm, SubApSnap (in online phase) [Jones-N arXiv 25]:

- Has **sublinear** complexity $O(N)$.
- Needs to look only at $O(1)$ rows of $A(p)$ and $b(p)$.
- Achieves orders of magnitude speedup in experiments.

Snapshot/PCA: finding solution in subspace

Snapshot/PCA Choose snapshot points $p_1, \ldots, p_r$, and solve $A(p_i)x(p_i) = b(p_i)$, $i = 1, \ldots, r$. Then Snapshot is

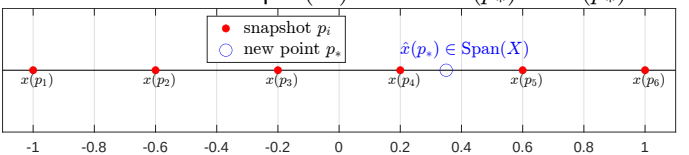
$$\boxed{X} = [x(p_1), \ldots, x(p_r)].$$

Snapshot/PCA: finding solution in subspace

Snapshot/PCA Choose snapshot points $p_1, \ldots, p_r$, and solve $A(p_i)x(p_i) = b(p_i)$, $i = 1, \ldots, r$. Then Snapshot is

$$X = [x(p_1), \ldots, x(p_r)].$$

Idea: find soln in span(X) for new $A(p_*)x = b(p_*)$

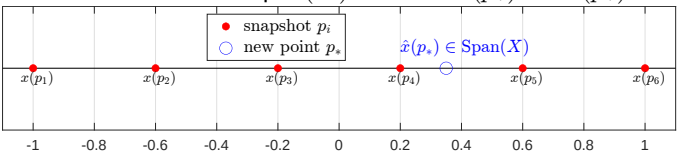


Snapshot/PCA: finding solution in subspace

Snapshot/PCA Choose snapshot points $p_1, \ldots, p_r$, and solve $A(p_i)x(p_i) = b(p_i)$, $i = 1, \ldots, r$. Then Snapshot is

$$X = [x(p_1), \ldots, x(p_r)].$$

Idea: find soln in span(X) for new $A(p_*)x = b(p_*)$



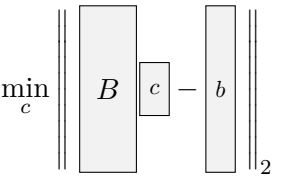
In PCA, take $\hat{r} \geq r$ points and dominant subspace via SVD:

$$X = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T + \sum_{i=r+1}^{\hat{r}} \sigma_i u_i v_i^T, \qquad \text{and } X \leftarrow [u_1, \ldots, u_r].$$

SubApSnap: Subsampling+Snapshot

ApSnap: with $B = A(p)X$, where X is snapshot, solve and set $x = Xc$:

[Markovinovic-Jansen 06, Guido-Kressner-Ricci 24]



SubApSnap: Subsampling+Snapshot

ApSnap: with $B = A(p)X$, where X is snapshot, solve and set $x = Xc$:

[Markovinovic-Jansen 06, Guido-Kressner-Ricci 24]

$$\min_c \left\| B\,c - b \right\|_2$$

SubApSnap:

SubApSnap: Subsampling+Snapshot

ApSnap: with $B = A(p)X$, where X is snapshot, solve and set $x = Xc$:

[Markovinovic-Jansen 06, Guido-Kressner-Ricci 24]

$$\min_c \left\| B\,c - b \right\|_2$$

SubApSnap: **Subsample** $S = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$

$$\min_c \left\| B\,c - b \right\|_2 \quad \to \quad \min_{\hat{c}} \left\| SB\,\hat{c} - Sb \right\|_2 .$$

and set $x = X\hat{c}$.

SubApSnap: Subsampling+Snapshot

ApSnap: with $B = A(p)X$, where X is snapshot, solve and set $x = Xc$:

[Markovinovic-Jansen 06, Guido-Kressner-Ricci 24]

$$\min_c \left\| B\,c - b \right\|_2$$

SubApSnap: **Subsample** $S = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$

$$\min_c \left\| B\,c - b \right\|_2 \quad \to \quad \min_{\hat{c}} \left\| SB\,\hat{c} - Sb \right\|_2 .$$

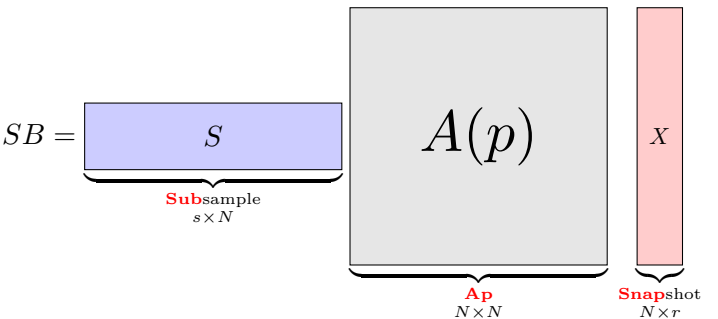
and set $x = X\hat{c}$. Important: we'll use the same S for all p_*.

Also recall subsampled least squares$\subsetneq$CUR

SubApSnap: Etymology

$$\min_{\hat{c}} \left\| SB\,\hat{c} - Sb \right\|_2,$$

where

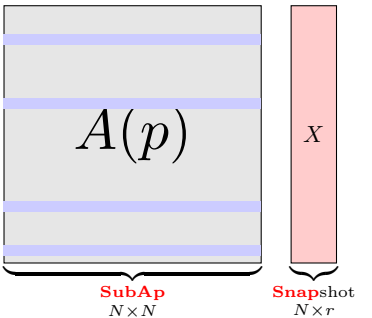


SubApSnap: Etymology

$$\min_{\hat{c}} \left\| SB\,\hat{c} - Sb \right\|_2,$$

where

$SB =$

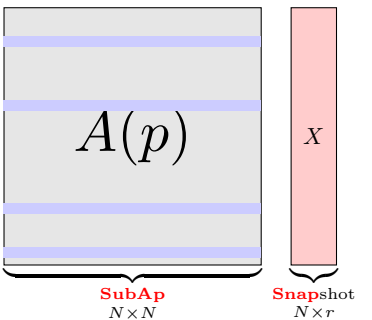


SubApSnap: Etymology

$$\min_{\hat{c}} \left\| SB\,\hat{c} - Sb \right\|_2,$$

where

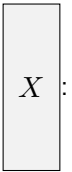
$SB =$



Important: $SA(p)$ is row-**submatrix** of $A(p)$; only need to see $s = O(r)$ rows for SubApSnap solution! Same for $Sb(p)$

DEIM=SubSnap vs SubApSnap

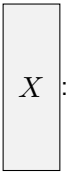
[Chaturantabut-Sorensen 2010]

DEIM: nonlinear $y(p_*) \approx Xc$,  : snapshot

$$\underset{\text{(Snap)}}{\min_c \|Xc - y(p_*)\|_2} \quad \rightarrow \quad \underset{\text{(SubSnap=DEIM)}}{\min_c \|S(Xc - y(p_*))\|_2}$$

DEIM=SubSnap vs SubApSnap

[Chaturantabut-Sorensen 2010]

DEIM: nonlinear $y(p_*) \approx Xc$,  : snapshot

$$\underset{\text{(Snap)}}{\min_c \|Xc - y(p_*)\|_2} \quad \rightarrow \quad \underset{\text{(SubSnap=DEIM)}}{\min_c \|S(Xc - y(p_*))\|_2}$$

$$\underset{\text{(ApSnap)}}{\min_c \|A(p_*)Xc - b(p_*)\|_2} \quad \rightarrow \quad \underset{\text{(SubApSnap)}}{\min_c \|S(A(p_*)Xc - b(p_*))\|_2}.$$

DEIM=SubSnap, 'special case' of SubApSnap where $A(p) = I_{54/60}$

Complexity

Assume $A(p)$ is $N \times N$ dense, r snapshot points. Cost for ℓ values of p:

	Complexity
Full	$O(N^3\ell)$
ApSnap	$O(N^3r + N^2\ell r)$
SubApSnap	$O(N^3r + N\ell r^2)$

Complexity

Assume $A(p)$ is $N \times N$ dense, r snapshot points. Cost for ℓ values of p:

	Complexity
Full	$O(N^3\ell)$
ApSnap	$O(N^3r + N^2\ell r)$
SubApSnap	$O(N^3r + N\ell r^2)$

For each new value of p,

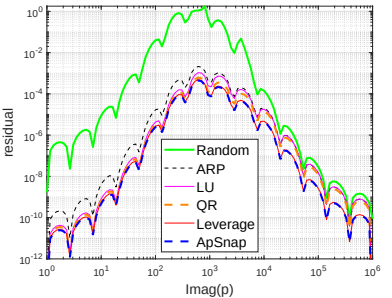
	Complexity
Full	$O(N^3)$
ApSnap	$O(N^2r)$
SubApSnap	$O(Nr^2)$

SubApSnap can be even faster: e.g. $O(r^3)$ when $A(p)$ sparse

Subsampling strategies and estimating residual

LUPP, CPQR, leverage scores, Osinsky 2023, DPP,
ARP [Cortinovis-Kressner 2024], ... , applied to $A(p_i)X$ (blue: randomized)

- LU and QR: deterministic, classical NLA tools. Often V good.
- Leverage scores need to oversample $s > r$, but usually has smaller residual and residual can be estimated.

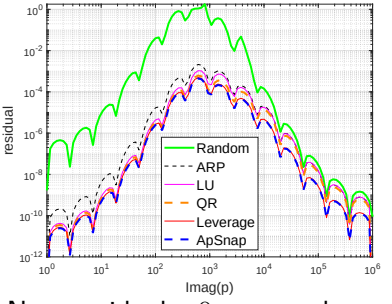


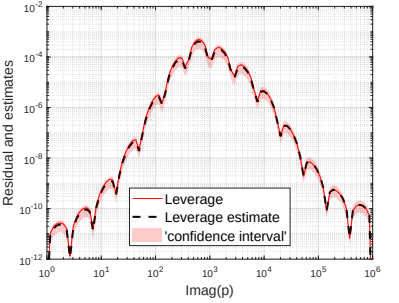
Note residual$= 0$ at snapshot points

Subsampling strategies and estimating residual

LUPP, CPQR, leverage scores, Osinsky 2023, DPP,
ARP [Cortinovis-Kressner 2024], ... , applied to $A(p_i)X$ (blue: randomized)

- LU and QR: deterministic, classical NLA tools. Often V good.
- Leverage scores need to oversample $s > r$, but usually has smaller residual and residual can be estimated.

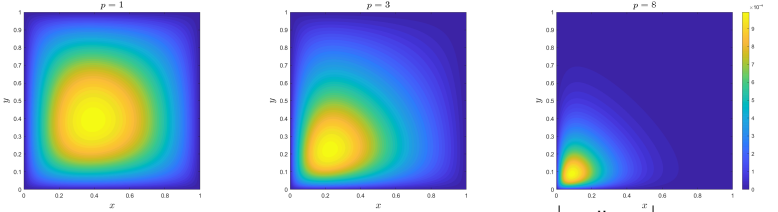


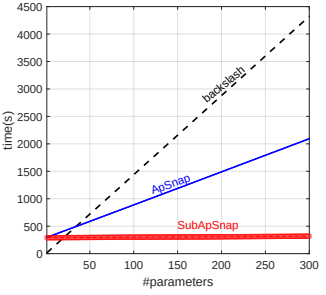


Note residual$= 0$ at snapshot points

PDE Example: variable-coefficient heat equation

$$-\nabla \cdot (\kappa(\mathbf{x}, p)\nabla u(\mathbf{x}, p)) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega = [0,1]^2, \quad \kappa(\mathbf{x}, p) = e^{p(\mathbf{x}_1 + \mathbf{x}_2)}$$



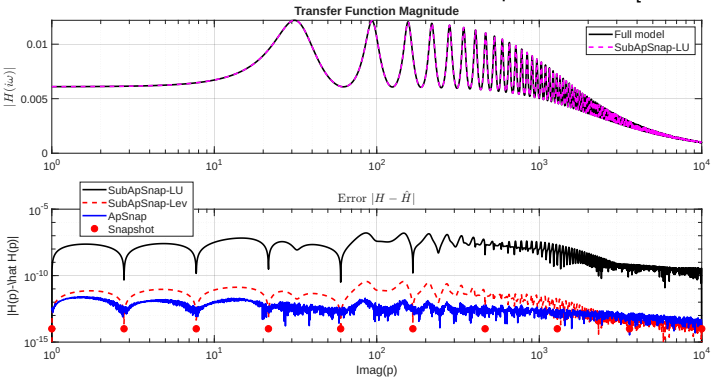


	all p	new p_*
Snap	290.0	
SubApSnap	379.8	0.0898
assembly SA	77.8	0.0778
ApSnap	6323.9	6.034
assembly A	5757	5.757
matvec AX	187.8	0.1878
backslash	14402	14.40

max error: 5×10^{-8}

MOR example $H(p) = c^T(pI - A_0 - e^{\tau p}A_1)^{-1}b$, A_0, A_1 : tridiag, $n = 10^7$

problem from [Beattie-Gugercin 09]



	time(s)	error $\max_p \lvert H(p) - \hat{H}(p)\rvert$	$\max_p \lVert A(p)x(p) - b\rVert_2 / \lVert b\rVert_2$
Full	9806.9		
Snap	47.3		
ApSnap	14386.1	2.6e-12	4.8e-06
SubApSnap-LU	0.302	1.6e-07	1.88e-05
SubApSnap-Lev	0.469	3.6e-11	5.8e-06

$> 20,000\times$ speedup with minimal loss of accuracy!

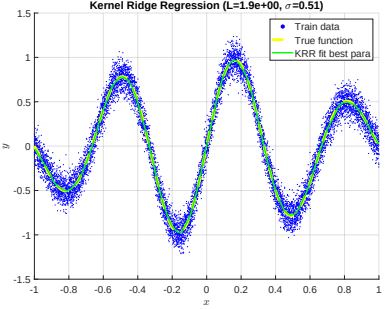
Example: Kernel Ridge Regression parameter tuning

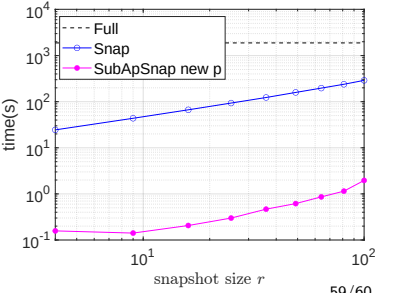
Given data $y_i = \sin(3t_i)\exp(-t_i^2) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, 0.1^2)$, KRR finds $y \approx f(t)$ via

$$(K + \lambda I)x = y_{\text{train}},$$

where $K_{ij} = k(t_i, t_j) = \exp(-\frac{|t_i - t_j|}{\sigma})$, for hyperparameters (λ, σ): need tuning! Do by 30×30 grid search + cross validation

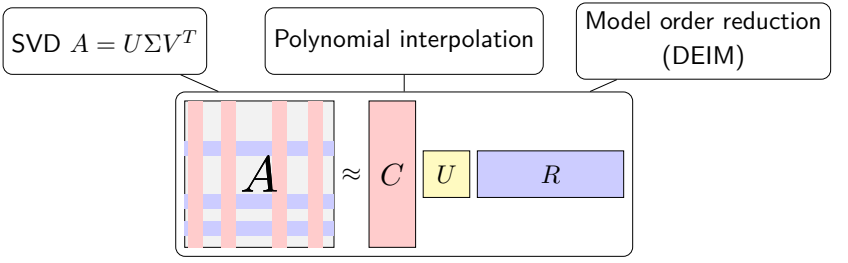
$\rightarrow 900$ linear systems. SubApSnap!





Summary of lectures 1+2

- Orthogonal vs. oblique projection
- CUR: only $r+1$ times worse than truncated SVD, often much better
- Subsampled (tall-skinny) least-squares problem



CUR allows for computation with subsampling

Applications include

- Function approximation (2D, complicated domains, high-dim)
- DEIM, PDEs, statistics (see Irina Haas's poster)
- (Randomized) NLA (e.g. parameter-dependent problems), add yours!

References I

R. Armstrong and A. Damle.
Collect, commit, expand: Efficient CPQR-based column selection for extremely wide matrices.
SIAM J. Sci. Comput., 48(4):A1951–A1978, 2026.

H. Avron and S. Toledo.
Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix.
J. ACM, 58(2):8, 2011.

J. D. Batson, D. A. Spielman, and N. Srivastava.
Twice-Ramanujan sparsifiers.
In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 255–262, 2009.

C. Beattie and S. Gugercin.
Interpolatory projection methods for structure-preserving model reduction.
Systems & Control Letters, 58(3):225–232, 2009.

B. Beckermann and A. Townsend.
On the singular values of matrices with displacement structure.
SIAM J. Matrix Anal. Appl., 38(4):1227–1248, 2017.

References II

S. Chaturantabut and D. C. Sorensen.
Nonlinear model reduction via discrete empirical interpolation.
SIAM J. Sci. Comput., 32(5):2737–2764, 2010.

Y. Chen, E. N. Epperly, J. A. Tropp, and R. J. Webber.
Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations.
Comm. Pure Appl. Math., 78(5):995–1041, 2025.

K. L. Clarkson and D. P. Woodruff.
Low-rank approximation and regression in input sparsity time.
Journal of the ACM, 63(6):54, 2017.

A. Cortinovis and D. Kressner.
Low-rank approximation in the frobenius norm by column and row subset selection.
SIAM J. Matrix Anal. Appl., 41(4):1651–1673, 2020.

A. Cortinovis and D. Kressner.
Adaptive randomized pivoting for column subset selection, DEIM, and low-rank approximation.
SIAM J. Matrix Anal. Appl., 47(1):25–47, 2026.

References III

M. Derezinski and M. W. Mahoney.
Determinantal point processes in randomized numerical linear algebra.
Notices of the American Mathematical Society, 68(1):34–45, 2021.

A. Deshpande, L. Rademacher, S. S. Vempala, and G. Wang.
Matrix approximation and projective clustering via volume sampling.
Theory of Computing, 2(1):225–247, 2006.

Y. Dong and P.-G. Martinsson.
Simpler is better: a comparative study of randomized pivoting algorithms for CUR and interpolative decompositions.
Adv. in Comput. Math., 49:66(4), 2023.

Z. Drmac and S. Güğercin.
A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions.
SIAM J. Sci. Comput., 38(2):A631–A648, 2016.

J. A. Duersch and M. Gu.
Randomized projection for rank-revealing matrix factorizations and low-rank approximations.
SIAM Rev., 62(3):661–682, 2020.

References IV

S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin.
A theory of pseudoskeleton approximations.
Linear Algebra Appl., 261(1-3):1–21, 1997.

L. Grigori and Z. Xue.
Randomized strong rank-revealing qr for column subset selection and low-rank matrix approximation.
arXiv preprint arXiv:2503.18496, 2025.

M. Guido, D. Kressner, and P. Ricci.
Subspace acceleration for a sequence of linear systems and application to plasma simulation.
J. Sci. Comput., 99(3):68, 2024.

E. Jones and Y. Nakatsukasa.
Subapsnap: Solving parameter-dependent linear systems with a snapshot and subsampling.
arXiv preprint arXiv:2510.04825, 2025.

M. W. Mahoney and P. Drineas.
CUR matrix decompositions for improved data analysis.
Proc. Natl. Acad. Sci., 106(3):697–702, 2009.

References V

R. Markovinović and J. D. Jansen.
Accelerating iterative solution methods using reduced-order models as solution predictors.
Internat. J. Numer. Methods Eng., 68(5):525–541, 2006.

P.-G. Martinsson.
Randomized methods for matrix computations.
arXiv preprint 1607.01649, 2016.

P.-G. Martinsson and J. A. Tropp.
Randomized numerical linear algebra: Foundations and algorithms.
Acta Numer., pages 403—572, 2020.

Y. Nakatsukasa.
Fast and stable randomized low-rank matrix approximation.
arXiv preprint arXiv:2009.11392, 2020.

A. Osinsky.
Close to optimal column approximations with a single SVD.
Linear Algebra Appl., 725:359–377, 2025.

References VI

T. Park and Y. Nakatsukasa.
Low-rank approximation of parameter-dependent matrices via CUR decomposition.
SIAM J. Sci. Comput., 47(3):A1858–A1887, 2025.

N. Pritchard, T. Park, Y. Nakatsukasa, and P.-G. Martinsson.
Fast rank adaptive CUR via a recycled small sketch.
arXiv preprint arXiv:2509.21963, 2025.

A. Shimizu, X. Cheng, C. Musco, and J. Weare.
Improved active learning via dependent leverage score sampling.
arXiv preprint arXiv:2310.04966, 2023.

D. B. Szyld.
The many proofs of an identity on the norm of oblique projections.
Numerical Algorithms, 42(3-4):309–323, 2006.

A. Townsend and L. N. Trefethen.
Continuous analogues of matrix factorizations.
Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471(2173):20140585, 2015.

References VII

L. N. Trefethen.
Approximation Theory and Approximation Practice, Extended Edition.
SIAM, 2019.

J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher.
Fixed-rank approximation of a positive-semidefinite matrix from streaming data.
In *Advances in Neural Information Processing Systems*, pages 1225–1234, 2017.

D. P. Woodruff.
Sketching as a tool for numerical linear algebra.
Foundations and Trends® in Theoretical Computer Science, 10(1–2):1–157, 2014.

F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert.
A fast randomized algorithm for the approximation of matrices.
Appl. Comput. Harmon. Anal., 25(3):335–366, 2008.

W. Yu, Y. Gu, and Y. Li.
Efficient randomized algorithms for the fixed-precision low-rank matrix approximation.
SIAM J. Matrix Anal. Appl., 39(3):1339–1359, 2018.

References VIII

N. L. Zamarashkin and A. I. Osinsky.
On the existence of a nearly optimal skeleton approximation of a matrix in the Frobenius norm.
In *Doklady Mathematics*, volume 97(2), pages 164–166. Springer, 2018.