Low-Rank Structure in Scientific Computing

Hussam Al Daas

Computational Mathematics Theme, STFC-RAL

Two lectures

Today — low-rank structure

- why huge objects defined *implicitly by some problems* turn out to be numerically low rank
- four places where it occurs in scientific computing
- two of them in depth: nonconvex optimisation, and X-ray spectromicroscopy

Almost everything today is about *matrices*.

Next time — going higher-order

- CP, Tucker and Tensor-Train decompositions
- structured random matrices — Kronecker, Khatri-Rao, and matrices whose columns are tensor trains — that make these decompositions fast

The question for today

We keep meeting objects that are far too large to store or analyse:

- the solution X of a matrix equation with n^2 unknowns, $n = 10^6$
- the family of solutions $\{x_\lambda\}$ of a regularized problem, over all λ
- the solutions of a parametric PDE
- a spectral datacube nobody can afford to finish measuring

The question for today

We keep meeting objects that are far too large to store or analyse:

- the solution X of a matrix equation with n^2 unknowns, $n = 10^6$
- the family of solutions $\{x_\lambda\}$ of a regularized problem, over all λ
- the solutions of a parametric PDE
- a spectral datacube nobody can afford to finish measuring

In every case the object is defined *implicitly*, by a problem.

When and why should any of them be close to low rank?

The question for today

We keep meeting objects that are far too large to store or analyse:

- the solution X of a matrix equation with n^2 unknowns, $n = 10^6$
- the family of solutions $\{x_\lambda\}$ of a regularized problem, over all λ
- the solutions of a parametric PDE
- a spectral datacube nobody can afford to finish measuring

In every case the object is defined *implicitly*, by a problem.

When and why should any of them be close to low rank?

And — the part that actually matters — once we know they are, can we compute *with* the low-rank form without ever forming the object?

Two mechanisms

Mechanism 1: smoothness in a parameter

Suppose the columns of X sample a smooth curve: $\mathsf{X} = [\, f(t_1) \;\; f(t_2) \;\; \cdots \;\; f(t_p) \,]$, with f analytic in t.

Truncating a degree-k expansion of f about the centre of the interval writes every column in the same k-dimensional space, up to the remainder:

$$f(t) = \sum_{j=0}^{k-1} c_j \, (t - t_0)^j + r_k(t) \qquad \Longrightarrow \qquad \mathsf{X} = \underbrace{[\, \mathsf{c}_0 \cdots \mathsf{c}_{k-1} \,]}_{n \times k} \mathsf{V}^{\mathsf{T}} + \mathsf{R}_k$$

Analyticity in a neighbourhood $\Longrightarrow \|\mathsf{R}_k\|$ decays *geometrically* in k.

- the rank we need grows like $\log(1/\epsilon)$, not like n or p
- this is the mechanism behind parameter-dependent problems

Mechanism 2: displacement structure

Far sharper, and the one we will use repeatedly. Suppose X satisfies a *Sylvester equation*

$$AX - XB = C, \qquad \operatorname{rank}(C) = \nu$$

with A, B normal, $\Lambda(A) \subset E$, $\Lambda(B) \subset F$, and $E \cap F = \emptyset$.

Beckermann & Townsend (SIMAX 2017)

$$\frac{\sigma_{1+\nu k}(X)}{\sigma_1(X)} \leq Z_k(E, F)$$

where Z_k is the *Zolotarev number* for the pair (E, F).

Read it as: a low-rank right-hand side forces the solution to be numerically low rank, at a rate set purely by how well-separated the two spectra are.

Nothing here depends on n. That is the whole point.

What the Zolotarev number gives you

For real intervals, Z_k decays geometrically, and inverting the bound gives an ϵ-rank estimate of the form

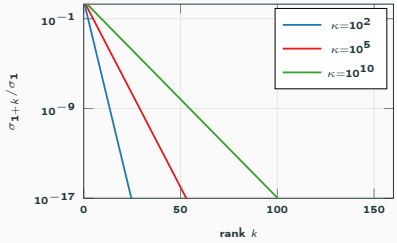
$$\mathrm{rank}_\epsilon(X) \lesssim \nu \cdot \frac{\log(16\gamma)\log(4/\epsilon)}{\pi^2},$$

with γ a *cross ratio* measuring the separation of E and F (for a positive definite problem, essentially the condition number).

Two logarithms is a very forgiving bound:

- $\kappa = 10^{16}$ and $\epsilon = 10^{-16}$ still gives a rank in the *hundreds*
- independent of n — the same rank at $n = 10^4$ and $n = 10^8$
- halving ϵ costs a fixed additive amount of rank

We will meet this exact bound again after the break, in a place where the "matrix" is a set of *optimisation solutions*.



Zolotarev decay, $\nu = 1$

Four examples

Example 1: matrix equations with low-rank right-hand side

The Lyapunov equation

$$AX + XA^T = -BB^T, \qquad A \in \mathbb{R}^{n \times n} \text{ stable}, \quad B \in \mathbb{R}^{n \times \nu}, \quad \nu \ll n$$

Where it comes from:

- **model reduction / control**: X is the controllability Gramian, B has one column per input — a handful
- **Sylvester and Riccati** variants throughout the same applications

Example 1: matrix equations with low-rank right-hand side

The Lyapunov equation

$$AX + XA^T = -BB^T, \qquad A \in \mathbb{R}^{n\times n} \text{ stable}, \quad B \in \mathbb{R}^{n\times \nu}, \quad \nu \ll n$$

Where it comes from:

- **model reduction / control**: X is the controllability Gramian, B has one column per input — a handful
- **Sylvester and Riccati** variants throughout the same applications

$$n = 10^6 \implies X \text{ has } 10^{12} \text{ entries. We are never going to store it.}$$

Example 1: why X is numerically low rank

Rewrite Lyapunov as a Sylvester equation:

$$AX - X\underbrace{(-A^T)}_{B_{\text{syl}}} = \underbrace{-BB^T}_{\text{rank} = \nu}$$

- A stable $\Longrightarrow \Lambda(A)$ in the open left half-plane, $\Lambda(-A^T)$ in the right — disjoint, by construction
- right-hand side has rank exactly ν, the number of inputs

So Mechanism 2 applies verbatim:

$$\frac{\sigma_{1+\nu k}(X)}{\sigma_1(X)} \leq Z_k \qquad \Longrightarrow \qquad X \approx ZZ^T, \quad Z \in \mathbb{R}^{n \times r}, \quad r \ll n$$

Example 1: computing with the factor, never the solution

Keep $X \approx ZZ^T$ in *factored* form throughout. Two standard families:

Low-rank ADI

$$Z_j = \left[Z_{j-1} \quad \sqrt{-2\mu_j}\,(A + \mu_j I)^{-1} v_j \right]$$

- one *shifted solve* per iteration
- factor grows by ν columns at a time
- shifts $\{\mu_j\}$ chosen by the same Zolotarev problem that bounds the rank

Projection onto a subspace

$$X \approx V_k Y_k V_k^T, \qquad V_k^T V_k = I$$

Impose Galerkin orthogonality; Y_k solves a *tiny* $k \times k$ Lyapunov equation.

The good choice of V_k is the extended Krylov subspace

$$\operatorname{span}\{B, A^{-1}B, AB, A^{-2}B, \ldots\}$$

Remember that last box. It comes back after the break, for a completely different problem.

Example 2: regularization in inverse problems

Discrete ill-posed problem: $Kx \approx b$, with $K = U\Sigma V^T$ and singular values decaying to zero with no gap.

Naive inversion amplifies noise:

$$x = \sum_i \frac{u_i^T b}{\sigma_i} v_i$$

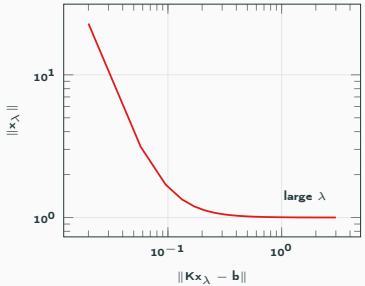
Tikhonov: damp the small σ_i,

$$x_\lambda = \arg\min_x \ \|Kx - b\|^2 + \lambda^2 \|x\|^2$$

$$\iff \quad (K^T K + \lambda^2 I)\, x_\lambda = K^T b$$

The hard part is never one solve. It is choosing λ —

which means computing x_λ for *many* λ (L-curve, GCV, etc).



The L-curve: one point per λ

Example 2: the regularization *path* is the low-rank object

Stack the whole path as columns: $X = [\, x_{\lambda_1} \ \cdots \ x_{\lambda_p} \,] \in \mathbb{R}^{n \times p}$. In the SVD basis of K,

$$X = VG, \qquad g_{ij} = \frac{\sigma_i \, (u_i^T b)}{\sigma_i^2 + \lambda_j^2}$$

Example 2: the regularization *path* is the low-rank object

Stack the whole path as columns: $X = [\, x_{\lambda_1} \ \cdots \ x_{\lambda_p} \,] \in \mathbb{R}^{n \times p}$. In the SVD basis of K,

$$X = VG, \qquad g_{ij} = \frac{\sigma_i \, (u_i^T b)}{\sigma_i^2 + \lambda_j^2}$$

G is a *Cauchy-like* matrix: with $D_1 = \mathrm{diag}(\sigma_i^2)$ and $D_2 = \mathrm{diag}(-\lambda_j^2)$,

$$D_1 G - G D_2 \ = \ d\, e^T, \qquad \nu = 1$$

Mechanism 2 again, with $\nu = 1$: $\mathrm{rank}_\epsilon(X)$ is a couple of dozen, whatever n and p are.

Example 2: the regularization *path* is the low-rank object

Stack the whole path as columns: $X = [\, x_{\lambda_1} \ \cdots \ x_{\lambda_p} \,] \in \mathbb{R}^{n \times p}$. In the SVD basis of K,

$$X = VG, \qquad g_{ij} = \frac{\sigma_i \, (u_i^T b)}{\sigma_i^2 + \lambda_j^2}$$

G is a *Cauchy-like* matrix: with $D_1 = \mathrm{diag}(\sigma_i^2)$ and $D_2 = \mathrm{diag}(-\lambda_j^2)$,

$$D_1 G - G D_2 \ = \ d\, e^T, \qquad \nu = 1$$

Mechanism 2 again, with $\nu = 1$: $\mathrm{rank}_\epsilon(X)$ is a couple of dozen, whatever n and p are.

Consequence worth stating plainly:

The entire regularization path costs about what a handful of solves cost — build a basis for the path once, then every λ is a $k \times k$ problem.

Example 2: the same structure in nonconvex optimisation

Trust-region methods for $\min_x f(x)$ solve, at every iteration,

$$s_k = \arg\min_{s \in \mathbb{R}^n} \; m_k(s) \stackrel{\text{def}}{=} \tfrac{1}{2} s^T H_k s + s^T g_k \;\text{ subject to }\; \|s\| \leq \Delta_k$$

and optimality forces a *shifted linear system*

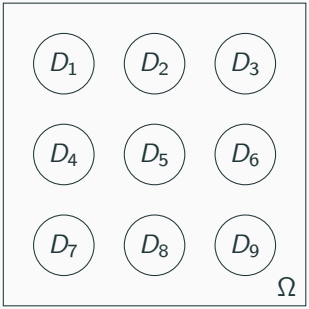
$$(H_k + \sigma I)\, s = -g_k, \qquad \sigma \geq \max(0, -\lambda_1(H_k))$$

Norm-regularization (cubic and friends) replaces the constraint by a penalty $\frac{\rho}{r}\|x\|^r$ — and produces *the same* shifted family.

inverse problems	$(\mathsf{K}^\mathsf{T}\mathsf{K} + \lambda^2\mathsf{I})\mathsf{x}_\lambda = \mathsf{K}^\mathsf{T}\mathsf{b}$
trust region / regularization	$(\mathsf{A} + \sigma\mathsf{I})\mathsf{x}(\sigma) = \mathsf{b}$

One parametrized family of shifted systems, two literatures. That family is the subject of the first deep dive.

Example 3: parameter-dependent problems



A parameter-dependent PDE:

$$\begin{aligned} -\nabla \cdot (\sigma(x, y; \boldsymbol{\rho}) \nabla (u(x, y; \boldsymbol{\rho}))) &= f(x, y) && \text{in } \Omega, \\ u(x, y; \boldsymbol{\rho}) &= 0 && \text{on } \partial\Omega, \end{aligned}$$

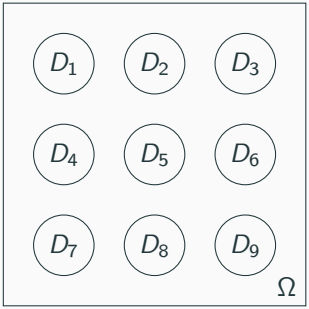
with piecewise-constant conductivity

$$\sigma(x, y; \boldsymbol{\rho}) = \begin{cases} 1 + \rho_i & \text{if } (x, y) \in D_i \\ 1 & \text{elsewhere} \end{cases}$$

the *cookies* problem.

Each ρ_i is a design or uncertainty parameter. We do not want one solve — we want the *solution map* $\boldsymbol{\rho} \mapsto u(\cdot\,; \boldsymbol{\rho})$.

Example 3: parameter-dependent problems



A parameter-dependent PDE:

$$\begin{aligned} -\nabla \cdot (\sigma(x, y; \boldsymbol{\rho}) \nabla (u(x, y; \boldsymbol{\rho}))) &= f(x, y) && \text{in } \Omega, \\ u(x, y; \boldsymbol{\rho}) &= 0 && \text{on } \partial\Omega, \end{aligned}$$

with piecewise-constant conductivity

$$\sigma(x, y; \boldsymbol{\rho}) = \begin{cases} 1 + \rho_i & \text{if } (x, y) \in D_i \\ 1 & \text{elsewhere} \end{cases}$$

the *cookies* problem.

Each ρ_i is a design or uncertainty parameter. We do not want one solve — we want the *solution map* $\boldsymbol{\rho} \mapsto u(\cdot\,; \boldsymbol{\rho})$.

Discretize space *and* all p parameters, and solve simultaneously.

Example 3: where it stops being a matrix

The discrete solution carries one index for space and one per parameter:

$$\mathcal{U}(i, k_1, k_2, \ldots, k_p) \;\approx\; u(x_i;\; \rho_1^{(k_1)},\; \ldots,\; \rho_p^{(k_p)})$$

That is an order-$(1+p)$ array, not a matrix.

- n_x spatial dofs, n samples per parameter $\Longrightarrow$ $n_x\, n^p$ entries
- $n_x = 1781$, $n = 100$, $p = 4$: already 1.8×10^{11}
- Mechanism 1 says it *is* highly compressible (the solution is analytic in $\boldsymbol{\rho}$)

Example 3: where it stops being a matrix

The discrete solution carries one index for space and one per parameter:

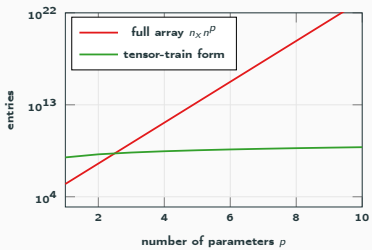
$$\mathcal{U}(i, k_1, k_2, \ldots, k_p) \;\approx\; u(x_i;\; \rho_1^{(k_1)},\; \ldots,\; \rho_p^{(k_p)})$$

That is an order-$(1+p)$ array, not a matrix.

- n_x spatial dofs, n samples per parameter $\Longrightarrow$ $n_x\, n^p$ entries
- $n_x = 1781$, $n = 100$, $p = 4$: already 1.8×10^{11}
- Mechanism 1 says it *is* highly compressible (the solution is analytic in $\boldsymbol{\rho}$)

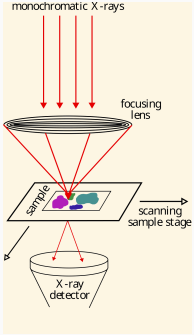
So: highly compressible, but not by any matrix rank.

We will come back and solve exactly this problem next lecture.



The gap next lecture is about.

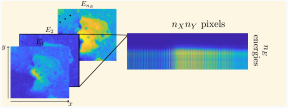
Example 4: spectral imaging



X-ray spectromicroscopy. Scan a beam over a sample; at each beam energy E you record a full spatial image.

Goal: identify the unknown materials in a non-homogeneous sample, *and* map where each one sits.

- raw data is a datacube: $n_X \times n_Y \times n_E$
- measuring all of it takes *hours*
- and the beam damages the sample while you measure



We unfold the cube to a matrix $\mathrm{D} \in \mathbb{R}^{n_E \times n_X n_Y}$: one row per energy.

Example 4: why the datacube is low rank — and what that buys

The sample contains a *small number s of distinct materials*. Beer–Lambert then says the measurement is linear in their abundances:

$$\mathsf{D} = \underbrace{\mathsf{M}}_{n_E \times s} \; \underbrace{\mathsf{T}}_{s \times n_X n_Y} + \mathsf{E}, \qquad s \ll \min(n_E,\ n_X n_Y)$$

- columns of M: the absorption *spectrum* of each material — what it is
- rows of T: the spatial *abundance* of each material — where it is
- so the analysis we want *is* a low-rank factorization

Example 4: why the datacube is low rank — and what that buys

The sample contains a *small number s of distinct materials*. Beer–Lambert then says the measurement is linear in their abundances:

$$\mathrm{D} = \underbrace{\mathrm{M}}_{n_E \times s} \; \underbrace{\mathrm{T}}_{s \times n_X n_Y} + \mathrm{E}, \qquad s \ll \min(n_E,\ n_X n_Y)$$

- columns of M: the absorption *spectrum* of each material — what it is
- rows of T: the spatial *abundance* of each material — where it is
- so the analysis we want *is* a low-rank factorization

Low rank here does not just make the computation cheaper. It means we never have to finish the measurement.

An $n_E \times n_X n_Y$ matrix of rank s has $\sim s(n_E + n_X n_Y)$ degrees of freedom. Sample a fraction of it well, and complete the rest — less time on the beamline, less radiation damage. *Which* entries to sample is the second deep dive.

Where we are

problem	**what is low rank**	**why**	**how it is exploited**
matrix equations, low-rank RHS	the solution X	displacement structure, $\nu = \mathrm{rank}(B)$	factored ADI; projection onto extended Krylov
regularization, inverse problems & optimisation	the *path* $\{x_\lambda\}$, $\{x(\sigma)\}$	Cauchy-like, $\nu = 1$	one subspace serves every parameter value
parameter-dependent PDEs	the solution *map*	analyticity in $\boldsymbol{\rho}$	manipulation of low-rank tensors
spectral imaging	the datacube	few materials $\times$ Beer–Lambert	subsample and complete

Where we are

problem	**what is low rank**	**why**	**how it is exploited**
matrix equations, low-rank RHS	the solution X	displacement structure, $\nu = \mathrm{rank}(B)$	factored ADI; projection onto extended Krylov
regularization, inverse problems & optimisation	the *path* $\{x_\lambda\}$, $\{x(\sigma)\}$	Cauchy-like, $\nu = 1$	one subspace serves every parameter value
parameter-dependent PDEs	the solution *map*	analyticity in ρ	manipulation of low-rank tensors
spectral imaging	the datacube	few materials $\times$ Beer–Lambert	subsample and complete

After the break: Trust-region solvers and subsampling in spectral imaging.

Break

10 minutes

Deep dive I: trust-region and norm-regularization subproblems

Nonconvex optimisation with trust-region

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

- f nonconvex, smooth, preferably C^2, n large, say $O(10^4) - O(10^8)$

Nonconvex optimisation with trust-region

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

- f nonconvex, smooth, preferably C^2, n large, say $O(10^4) - O(10^8)$
- iteration: from x_k, solve the trust-region subproblem

$$s_k = \arg\min_{s \in \mathbb{R}^n} \; m_k(s) \stackrel{\text{def}}{=} \tfrac{1}{2} s^T H_k s + s^T g_k \;\text{ subject to }\; \|s\| \leq \Delta_k$$

where $g_k = \nabla f(x_k)$, symmetric $H_k \approx \nabla^2 f(x_k)$, and radius $\Delta_k > 0$

Nonconvex optimisation with trust-region

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

- f nonconvex, smooth, preferably C^2, n large, say $O(10^4) - O(10^8)$
- iteration: from x_k, solve the trust-region subproblem

$$s_k = \underset{s \in \mathbb{R}^n}{\arg\min} \; m_k(s) \stackrel{\text{def}}{=} \tfrac{1}{2} s^T H_k s + s^T g_k \;\; \text{subject to} \;\; \|s\| \leq \Delta_k$$

  where $g_k = \nabla f(x_k)$, symmetric $H_k \approx \nabla^2 f(x_k)$, and radius $\Delta_k > 0$
- accept $x_{k+1} = x_k + s_k$ and (possibly) increase Δ_{k+1} if $f(x_k + s_k) - f(x_k)$ and $m_k(s_k)$ "agree"
- otherwise reject $x_{k+1} = x_k$ and decrease Δ_{k+1}

Deep dive I: the subproblem

(changing notation — I live in an NLA world)

Given a symmetric $A \in \mathbb{R}^{n \times n}$, a vector b, a radius $\Delta > 0$:

$$x_* = \arg\min_{x \in \mathbb{R}^n} \tfrac{1}{2} x^T A x - x^T b \ \text{ subject to } \ \|x\| \leq \Delta$$

- A is indefinite in general — this is nonconvex optimisation
- n large: $O(10^4)$ to $O(10^8)$
- solved *once per iteration* of the outer trust-region method, so it must be cheap
- and often *re-solved* at a smaller Δ after a rejected step

joint work with Nick Gould (STFC-RAL)

Optimality conditions

The solution $x_* = x(\sigma)$ of the subproblem satisfies the first-order condition

$$(A + \sigma I)x(\sigma) = b,$$

where $\|x(\sigma)\| \leq \Delta$ and the scalar shift

$$\sigma \geq \sigma_{\min} \stackrel{\text{def}}{=} \max(0, -\lambda_1(A))$$

satisfies complementarity $\sigma(\|x(\sigma)\| - \Delta) = 0$ — so in particular $A + \sigma I$ is positive semi-definite.

Optimality conditions

The solution $x_* = x(\sigma)$ of the subproblem satisfies the first-order condition

$$(A + \sigma I)x(\sigma) = b,$$

where $\|x(\sigma)\| \leq \Delta$ and the scalar shift

$$\sigma \geq \sigma_{\min} \stackrel{\text{def}}{=} \max(0, -\lambda_1(A))$$

satisfies complementarity $\sigma(\|x(\sigma)\| - \Delta) = 0$ — so in particular $A + \sigma I$ is positive semi-definite.

Either (interior case), for positive semi-definite A,

$$Ax(0) = b \text{ with } \|x(0)\| \leq \Delta$$

or (boundary case), with unrestricted A,

$$(A + \sigma I)x(\sigma) = b \text{ with } \|x(\sigma)\| = \Delta \text{ for some } \sigma \geq \sigma_{\min}$$

How this is solved today

Focus on the boundary case: $(A + \sigma I)x(\sigma) = b$ with $\|x(\sigma)\| = \Delta$ for some $\sigma \geq \sigma_{\min}$

- **1. Root-find in σ** (Moré–Sorensen, SINUM 1983): pick $\{\sigma_k\}$ so that $\|x(\sigma_k)\| \to \Delta$, safeguarded Newton
  - needs a factorization of $A + \sigma_k I$ for every k — typically 5 to 10 of them
  - globally linear, asymptotically quadratic

How this is solved today

Focus on the boundary case: $(A + \sigma I)x(\sigma) = b$ with $\|x(\sigma)\| = \Delta$ for some $\sigma \geq \sigma_{\min}$

- **1. Root-find in σ** (Moré–Sorensen, SINUM 1983): pick $\{\sigma_k\}$ so that $\|x(\sigma_k)\| \to \Delta$, safeguarded Newton
  - needs a factorization of $A + \sigma_k I$ for every k — typically 5 to 10 of them
  - globally linear, asymptotically quadratic
- **2. Subspace minimization** (GLTR, SIOPT 1999): pick nested orthonormal V_k, set $x = V_k y$,

$$y_k = \operatorname*{arg\,min}_{y \in \mathbb{R}^k} \ \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \ \text{ subject to } \ \|y\| \leq \Delta$$

  - matrix-free, only factors cheap tridiagonals; V_k from Lanczos on $\mathcal{K}(A, b)$
  - slow if the spectrum is awkward

How this is solved today

Focus on the boundary case: $(A + \sigma I)x(\sigma) = b$ with $\|x(\sigma)\| = \Delta$ for some $\sigma \geq \sigma_{\min}$

- **1. Root-find in σ** (Moré–Sorensen, SINUM 1983): pick $\{\sigma_k\}$ so that $\|x(\sigma_k)\| \to \Delta$, safeguarded Newton
  - needs a factorization of $A + \sigma_k I$ for every k — typically 5 to 10 of them
  - globally linear, asymptotically quadratic
- **2. Subspace minimization** (GLTR, SIOPT 1999): pick nested orthonormal V_k, set $x = V_k y$,

$$y_k = \underset{y \in \mathbb{R}^k}{\arg\min} \ \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \ \text{ subject to } \ \|y\| \leq \Delta$$

  - matrix-free, only factors cheap tridiagonals; V_k from Lanczos on $\mathcal{K}(A, b)$
  - slow if the spectrum is awkward
- **3. Eigenvalue reformulation** (Sorensen et al. 2000; Nakatsukasa et al. 2017)

How this is solved today

Focus on the boundary case: $(A + \sigma I)x(\sigma) = b$ with $\|x(\sigma)\| = \Delta$ for some $\sigma \geq \sigma_{\min}$

- **1. Root-find in σ** (Moré–Sorensen, SINUM 1983): pick $\{\sigma_k\}$ so that $\|x(\sigma_k)\| \to \Delta$, safeguarded Newton
  - needs a factorization of $A + \sigma_k I$ for every k — typically 5 to 10 of them
  - globally linear, asymptotically quadratic
- **2. Subspace minimization** (GLTR, SIOPT 1999): pick nested orthonormal V_k, set $x = V_k y$,

$$y_k = \underset{y \in \mathbb{R}^k}{\arg\min} \ \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \ \text{ subject to } \ \|y\| \leq \Delta$$

  - matrix-free, only factors cheap tridiagonals; V_k from Lanczos on $\mathcal{K}(A, b)$
  - slow if the spectrum is awkward
- **3. Eigenvalue reformulation** (Sorensen et al. 2000; Nakatsukasa et al. 2017)

Is that it?

The object nobody had looked at

Consider the solution manifold — every solution, over every radius at once:

$$\mathcal{M}(A, b) = \{x(\sigma) : (A + \sigma I)x(\sigma) = b \ \text{ for all } \ \sigma \geq \sigma_{\min}\}$$

and the subspace it spans,

$$\mathcal{S}(A, b) = \operatorname{span}\{x(\sigma) : (A + \sigma I)x(\sigma) = b \ \text{ for all } \ \sigma \geq \sigma_{\min}\}$$

The object nobody had looked at

Consider the solution manifold — every solution, over every radius at once:

$$\mathcal{M}(A, b) = \{x(\sigma) : (A + \sigma I)x(\sigma) = b \ \text{ for all } \ \sigma \geq \sigma_{\min}\}$$

and the subspace it spans,

$$\mathcal{S}(A, b) = \text{span}\{x(\sigma) : (A + \sigma I)x(\sigma) = b \ \text{ for all } \ \sigma \geq \sigma_{\min}\}$$

Diagonalize $A = U\Lambda U^T$. Then immediately

$$\mathcal{S}(A, b) = U\text{span}\{y(\sigma) : (\Lambda + \sigma I)y(\sigma) = d \text{ for all } \sigma \geq \sigma_{\min}\} \equiv U\mathcal{S}(\Lambda, d),$$

with $d = U^T b$ — so it suffices to understand $\mathcal{S}(\Lambda, d)$ for *diagonal* Λ.

The object nobody had looked at

Consider the solution manifold — every solution, over every radius at once:

$$\mathcal{M}(A, b) = \{x(\sigma) : (A + \sigma I)x(\sigma) = b \text{ for all } \sigma \geq \sigma_{\min}\}$$

and the subspace it spans,

$$\mathcal{S}(A, b) = \text{span}\{x(\sigma) : (A + \sigma I)x(\sigma) = b \text{ for all } \sigma \geq \sigma_{\min}\}$$

Diagonalize $A = U\Lambda U^T$. Then immediately

$$\mathcal{S}(A, b) = U\text{span}\{y(\sigma) : (\Lambda + \sigma I)y(\sigma) = d \text{ for all } \sigma \geq \sigma_{\min}\} \equiv U\mathcal{S}(\Lambda, d),$$

with $d = U^T b$ — so it suffices to understand $\mathcal{S}(\Lambda, d)$ for *diagonal* Λ.

This is Example 2 again: a one-parameter family of shifted solves, sampled as columns.

Look at it numerically

Take the discrete $n \times p$ subspace matrix Y, with $Y_{i,j} = \dfrac{d_i}{\lambda_i + \sigma_j}$:

- $n = 10^4$, d a vector of ones
- $p = 10n = 10^5$ evaluation points $\sigma_j = j/n$
- eigenvalues $\lambda_i \in (0, 1]$ distributed four different ways:
  - (i) equi-spaced at i/n
  - (ii) clustered low at i^2/n^2
  - (iii) clustered high at $1 - (i-1)^2/n^2$
  - (iv) logarithmically, from 10^{-15} to 1

Look at it numerically

Take the discrete $n \times p$ subspace matrix Y, with $Y_{i,j} = \dfrac{d_i}{\lambda_i + \sigma_j}$:

- $n = 10^4$, d a vector of ones
- $p = 10n = 10^5$ evaluation points $\sigma_j = j/n$
- eigenvalues $\lambda_i \in (0, 1]$ distributed four different ways:
  - (i) equi-spaced at i/n
  - (ii) clustered low at i^2/n^2
  - (iii) clustered high at $1 - (i-1)^2/n^2$
  - (iv) logarithmically, from 10^{-15} to 1

Y is $10^4 \times 10^5$. How many singular values does it actually have?

Largest singular values of the solution subspace

(i): equi-spaced		(ii): clustered low		(iii): clustered high		(iv): logarithmically	
2.53e+4	1.32e-3	1.34e+5	6.33e-3	1.86e+4	4.84e-4	1.09e+6	4.64e-3
1.44e+4	4.99e-4	4.38e+4	2.48e-3	1.09e+4	1.74e-4	7.32e+4	1.81e-3
6.89e+3	1.87e-4	1.92e+4	9.70e-4	5.12e+3	6.23e-5	2.09e+4	7.04e-4
3.07e+3	7.00e-5	8.81e+3	3.78e-4	2.21e+3	2.22e-5	7.75e+3	2.72e-4
1.33e+3	2.60e-5	3.95e+3	1.46e-4	9.19e+2	7.85e-6	3.23e+3	1.05e-4
5.64e+2	9.64e-6	1.73e+3	5.65e-5	3.76e+2	2.77e-6	1.39e+3	4.04e-5
2.36e+2	3.55e-6	7.42e+2	2.17e-5	1.51e+2	9.70e-7	5.96e+2	1.55e-5
9.78e+1	1.30e-6	3.15e+2	8.33e-6	6.03e+1	3.39e-7	2.52e+2	5.90e-6
4.01e+1	4.76e-7	1.33e+2	3.18e-6	2.38e+1	1.18e-7	1.05e+2	2.24e-6
1.63e+1	1.73e-7	5.54e+1	1.21e-6	9.27e+0	4.07e-8	4.35e+1	8.51e-7
6.55e+0	6.29e-8	2.29e+1	4.59e-7	3.58e+0	1.40e-8	1.79e+1	3.21e-7
2.61e+0	2.27e-8	9.44e+0	1.74e-7	1.37e+0	4.80e-9	7.30e+0	1.21e-7
1.04e+0	8.18e-9	3.86e+0	6.54e-8	5.20e-1	1.64e-9	2.96e+0	4.54e-8
4.07e-1	2.93e-9	1.57e+0	2.46e-8	1.96e-1	5.57e-10	1.19e+0	1.70e-8
1.59e-1	1.05e-9	6.34e-1	9.19e-9	7.33e-2	1.89e-10	4.79e-1	6.35e-9
6.18e-2	3.73e-10	2.55e-1	3.43e-9	2.72e-2	6.37e-11	1.91e-1	2.36e-9
2.38e-2	1.32e-10	1.02e-1	1.28e-9	1.00e-2	2.14e-11	7.60e-2	
9.14e-3	4.69e-11	4.05e-2	4.74e-10	3.67e-3		3.01e-2	
3.48e-3		1.61e-2	1.76e-10	1.34e-3		1.18e-2	

Around 35 significant singular values — out of 10^4. In all four cases.

And this is exactly Mechanism 2

$Y_{i,j} = d_i/(\lambda_i + \sigma_j)$ is Cauchy-like: $\mathsf{D}_\lambda Y - Y\mathsf{D}_{-\sigma} = \mathbf{d}\mathbf{e}^\mathsf{T}$, so Beckermann & Townsend [SIMAX, 2017] give the ϵ-rank

$$\mathsf{rank}_\epsilon(Y) \leq \left\lceil \log(16\gamma)\log(4/\epsilon)/\pi^2 \right\rceil,$$

with the cross ratio

$$\gamma = \left| \frac{(\sigma_{\max} + \lambda_1)(\sigma^+_{\min} + \lambda_n)}{(\sigma_{\max} + \lambda_n)(\sigma^+_{\min} + \lambda_1)} \right|$$

- bounded above provided $\sigma^+_{\min} + \lambda_1 \geq \delta > 0$
- for positive definite A, as $\sigma_{\max} \to \infty$ this is just $\kappa(A)$

And this is exactly Mechanism 2

$Y_{i,j} = d_i/(\lambda_i + \sigma_j)$ is Cauchy-like: $\mathrm{D}_\lambda Y - Y\mathrm{D}_{-\sigma} = \mathrm{de}^\mathsf{T}$, so Beckermann & Townsend [SIMAX, 2017] give the ϵ-rank

$$\mathrm{rank}_\epsilon(Y) \le \left\lceil \log(16\gamma)\log(4/\epsilon)/\pi^2 \right\rceil,$$

with the cross ratio

$$\gamma = \left| \frac{(\sigma_{\max} + \lambda_1)(\sigma^+_{\min} + \lambda_n)}{(\sigma_{\max} + \lambda_n)(\sigma^+_{\min} + \lambda_1)} \right|$$

- bounded above provided $\sigma^+_{\min} + \lambda_1 \ge \delta > 0$
- for positive definite A, as $\sigma_{\max} \to \infty$ this is just $\kappa(A)$

For the four examples the bound is 44. With $\kappa(A) = 10^{16}$ and $\epsilon = 10^{-16}$ it is 154 — *regardless of n.*

The double logarithm, and the independence from n, are the whole story.

So build a basis for that subspace

Back to subspace minimization: choose V_k so that it approximately spans $\mathcal{S}(A, b)$, and solve

$$y_k = \underset{y \in \mathbb{R}^k}{\arg\min} \ \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \ \text{ subject to } \ \|y\| \leq \Delta$$

So build a basis for that subspace

Back to subspace minimization: choose V_k so that it approximately spans $\mathcal{S}(A, b)$, and solve

$$y_k = \arg\min_{y \in \mathbb{R}^k} \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \text{ subject to } \|y\| \leq \Delta$$

Two obvious candidates, both bad:

- pick k shifts $\{\sigma_k\}$, solve $(A + \sigma_k I)x(\sigma_k) = b$ and orthogonalize
  - k factorizations — this is Moré–Sorensen's cost, or worse

So build a basis for that subspace

Back to subspace minimization: choose V_k so that it approximately spans $\mathcal{S}(A, b)$, and solve

$$y_k = \arg\min_{y \in \mathbb{R}^k} \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \text{ subject to } \|y\| \leq \Delta$$

Two obvious candidates, both bad:

- pick k shifts $\{\sigma_k\}$, solve $(A + \sigma_k I)x(\sigma_k) = b$ and orthogonalize
  - k factorizations — this is Moré–Sorensen's cost, or worse
- use Lanczos vectors from $\mathcal{K}(A, b)$, as GLTR does
  - too slow: polynomials in A carry almost no information about the *low* end of the spectrum, which is exactly where $(A + \sigma I)^{-1}$ lives

So build a basis for that subspace

Back to subspace minimization: choose V_k so that it approximately spans $\mathcal{S}(A, b)$, and solve

$$y_k = \arg\min_{y \in \mathbb{R}^k} \tfrac{1}{2} y^T (V_k^T A V_k) y - y^T (V_k^T b) \text{ subject to } \|y\| \leq \Delta$$

Two obvious candidates, both bad:

- pick k shifts $\{\sigma_k\}$, solve $(A + \sigma_k I)x(\sigma_k) = b$ and orthogonalize
  - k factorizations — this is Moré–Sorensen's cost, or worse
- use Lanczos vectors from $\mathcal{K}(A, b)$, as GLTR does
  - too slow: polynomials in A carry almost no information about the *low* end of the spectrum, which is exactly where $(A + \sigma I)^{-1}$ lives

We want the accuracy of the first and the cost of the second.

What the two ends of the family look like

$$x(\sigma) = (A + \sigma I)^{-1} b = Uy(\sigma), \quad \text{where } y_i(\sigma) = \frac{d_i}{\lambda_i + \sigma} \text{ and } d = U^T b.$$

Two extremes:

- $|\sigma| > |\lambda_i|$ for all i — Laurent expansion

$$y_i(\sigma) = \frac{d_i}{\sigma} \sum_{j=0}^{\infty} \left(\frac{\lambda_i}{-\sigma} \right)^j, \quad y(\sigma) \approx \sigma^{-1} \sum_{j=0}^{O(1)} (-\sigma)^{-j} A^j d$$

  so $x(\sigma)$ is well approximated by a polynomial of moderate degree in A applied to b

What the two ends of the family look like

$$x(\sigma) = (A + \sigma I)^{-1} b = U y(\sigma), \quad \text{where } y_i(\sigma) = \frac{d_i}{\lambda_i + \sigma} \text{ and } d = U^T b.$$

Two extremes:

- $|\sigma| > |\lambda_i|$ for all i — Laurent expansion

$$y_i(\sigma) = \frac{d_i}{\sigma} \sum_{j=0}^{\infty} \left(\frac{\lambda_i}{-\sigma} \right)^j, \quad y(\sigma) \approx \sigma^{-1} \sum_{j=0}^{O(1)} (-\sigma)^{-j} A^j d$$

  so $x(\sigma)$ is well approximated by a polynomial of moderate degree in A applied to b

- $|\sigma| < |\lambda_i|$ for all i — Taylor expansion

$$y_i(\sigma) = \frac{d_i}{\lambda_i} \sum_{j=0}^{\infty} \left(\frac{-\sigma}{\lambda_i} \right)^j, \quad y(\sigma) \approx \sum_{j=0}^{O(1)} (-\sigma)^j A^{-j-1} d$$

  so $x(\sigma)$ is well approximated by a polynomial of moderate degree in A^{-1} applied to b

Krylov gives us the first. We need *both*.

Extended Krylov subspaces

Extremes of $\mathcal{S}(A, b)$ are well approximated by a few terms of $\{b, Ab, A^2b, \dots\}$ *and* of $\{b, A^{-1}b, A^{-2}b, \dots\}$ for nonsingular A. Consider then the extended Krylov subspaces

$$\mathcal{K}_{2k}(A, b) = \text{span}\{A^{-k}b, \dots, A^{-1}b, b, Ab, \dots, A^{k-1}b\} \quad \text{and} \quad \mathcal{K}_{2k+1}(A, b) = \text{span}\{\mathcal{K}_{2k}(A, b), A^k b\},$$

$\mathcal{K}_0(A, b) = \{b\}$

Extended Krylov subspaces

Extremes of $\mathcal{S}(A, b)$ are well approximated by a few terms of $\{b, Ab, A^2b, \dots\}$ *and* of $\{b, A^{-1}b, A^{-2}b, \dots\}$ for nonsingular A. Consider then the extended Krylov subspaces

$$\mathcal{K}_{2k}(A, b) = \text{span}\{A^{-k}b, \dots, A^{-1}b, b, Ab, \dots, A^{k-1}b\} \quad \text{and} \quad \mathcal{K}_{2k+1}(A, b) = \text{span}\{\mathcal{K}_{2k}(A, b), A^k b\},$$
$$\mathcal{K}_0(A, b) = \{b\}$$

Convergence (essentially Knizhnerman & Simoncini, Num. Math. 2011)

Suppose A is positive definite and $\sigma > -\lambda_1(A) + \delta$ for some tiny $\delta > 0$. Then there is an $x_i(\sigma) \in \mathcal{K}_i(A, b)$ with

$$\|x_i(\sigma) - x(\sigma)\| \le c\, \mu^{2i}(\sigma), \qquad \mu(\sigma) \le \frac{\kappa^{1/4}(A) - 1}{\kappa^{1/4}(A) + 1} < 1$$

Note the fourth root of the condition number. And building the space costs one factorization, reused for every A^{-1} application.

A bonus: resolves are nearly free

Given V_k, we often must solve again with $\Delta_{\text{new}} < \Delta$ (so $\sigma_{\text{new}} > \sigma$), with A and b *unchanged* — this is exactly what happens after a rejected trust-region step.

- the convergence of V_k to a basis for $\mathcal{S}(A, b)$ accelerates as σ increases
- so the space we already built is *better* for the new, harder problem
- in practice we almost never need to increase k

The outer optimisation method gets its rejected steps at almost no cost — which is the opposite of the usual situation.

Numerical experiments

Compare GALAHAD codes

- `GLTR` — standard Krylov-based, matrix-free
- `TRS` — multi-factorization, Newton-based
- `TREK` — extended-Krylov-based, one factorization (max $k = 100$)

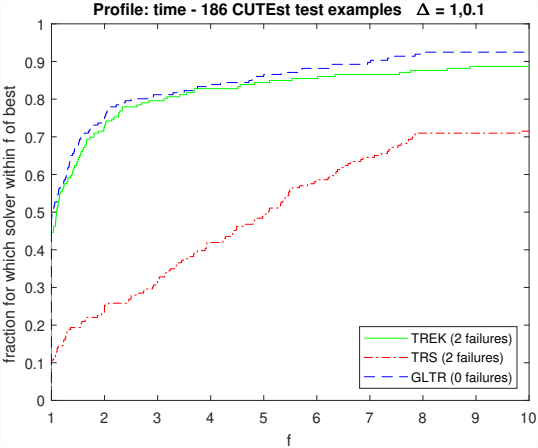
All 93 CUTEst unconstrained examples with $n \geq 1000$; sparse factorization.

Two cases:

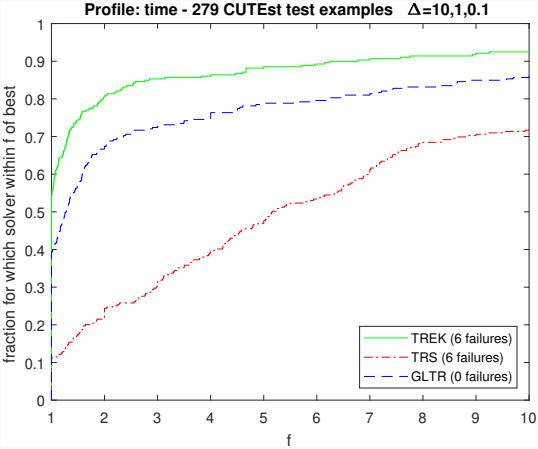
I: two radii each — start with $\Delta = 1$, restart with 0.1

II: three radii each — start with $\Delta = 10$, restart with 1 and 0.1

Results I — two radii



Results II — three radii



The more resolves, the further ahead TREK gets — as predicted.

The same machinery, more problems

- easy to generalise to the scaled subproblem

$$\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2} x^T A x - x^T b \ \text{ subject to } \ \|x\|_S \leq \Delta,$$

where $\|x\|_S^2 \stackrel{\text{def}}{=} x^T S x$, so long as S is diagonally dominant and easily invertible

The same machinery, more problems

- easy to generalise to the scaled subproblem

$$\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2} x^T A x - x^T b \ \text{ subject to } \ \|x\|_S \leq \Delta,$$

  where $\|x\|_S^2 \stackrel{\text{def}}{=} x^T S x$, so long as S is diagonally dominant and easily invertible

- the same EKS approach applies to the norm-regularization subproblem

$$x_* = \ \arg\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2} x^T A x - b^T x + \frac{\rho}{r} \|x\|_S^r$$

  with weight $\rho > 0$ and power $r \geq 2$ — cubic regularization is $r = 3$

  - only difference: solve the small diagonal regularization problem

$$z_k = \arg\min_{z \in \mathbb{R}^k} \ \tfrac{1}{2} z^T \Lambda_k z - d_k^T z + \frac{\rho}{r} \|z\|_S^r$$

  - requires the rightmost root of the secular equation $\|x(\sigma)\| = (\sigma/\rho)^{\frac{1}{r-2}}$
  - GALAHAD package `NREK`

Deep dive I: what actually happened

- trust-region and norm-regularization solutions live in an approximately very low-dimensional subspace
- we can approximate that subspace very well with an extended Krylov method and a single matrix factorization
- resolves at smaller radii are nearly free
- competitive with, and often faster than, both existing families of solver
- software in GALAHAD (`TREK`, `NREK`): Fortran, C, Python, Julia, Matlab

Deep dive I: what actually happened

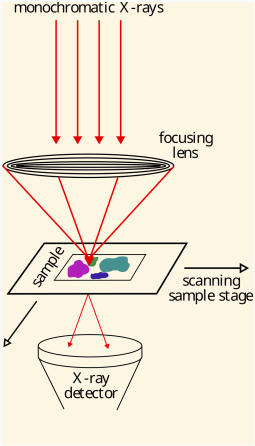
- trust-region and norm-regularization solutions live in an approximately very low-dimensional subspace
- we can approximate that subspace very well with an extended Krylov method and a single matrix factorization
- resolves at smaller radii are nearly free
- competitive with, and often faster than, both existing families of solver
- software in GALAHAD (`TREK`, `NREK`): Fortran, C, Python, Julia, Matlab

The low-rank object was never in the data.

A and b have no low-rank structure at all. What was low rank was the *solution manifold of a parametrized family* — an object that only exists because we asked for many answers at once.

Deep dive II: subsampling for spectromicroscopy

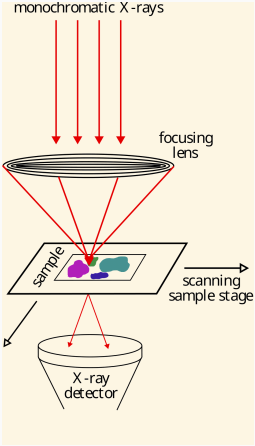
Deep dive II: the experiment, and its two sub-tasks



Overall goal: determine the identity *and* spatial distribution of unknown materials in a non-homogeneous sample

1. (Sub)sampling strategy
2. Analysis

Deep dive II: the experiment, and its two sub-tasks

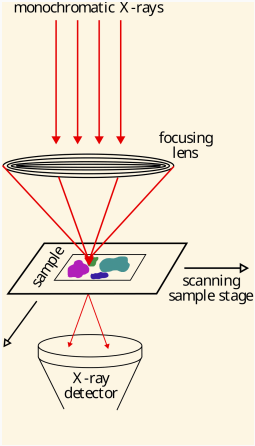


Overall goal: determine the identity *and* spatial distribution of unknown materials in a non-homogeneous sample

1. (Sub)sampling strategy 2. Analysis

2. Analysis, given a completed D: SVD $\rightarrow$ eigenspectra and eigenimages; cluster the eigenimages to group pixels with similar spectra; average each cluster's measured data to get its absorption spectrum.

Deep dive II: the experiment, and its two sub-tasks



Overall goal: determine the identity *and* spatial distribution of unknown materials in a non-homogeneous sample

1. (Sub)sampling strategy 2. Analysis

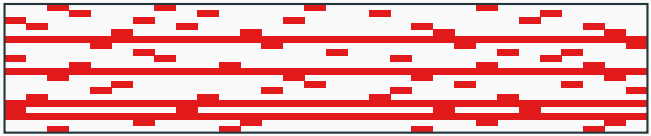
2. Analysis, given a completed D: SVD $\rightarrow$ eigenspectra and eigenimages; cluster the eigenimages to group pixels with similar spectra; average each cluster's measured data to get its absorption spectrum.

Part 2 is standard. The hard part is task 1

with M. Meier, L. Lazzarino, B. Shustin and P. Quinn

The constraint on measurements

We want to measure only a fraction of $D = MT + E \in \mathbb{R}^{n_E \times n_X n_Y}$ and complete the rest.



- a row of D = a full spatial image at one beam energy
- the beam rasters: it cannot measure a single pixel without scanning its whole spatial row.

Now: *which* entries should we sample?

Importance sampling, and the chicken-and-egg

Uniform raster sampling wastes measurements on uninformative energies and regions. The right measure is leverage scores:

$$\ell_i(\mathrm{D})^2 := \|\mathrm{U}_{\mathsf{dominant}}(i,:)\|_2^2$$

— how much row i matters to the dominant subspace.

Problem: computing $\ell(\mathrm{D})$ needs D — which is the thing we are trying to avoid measuring.

In principle $\ell(\mathrm{D}) \approx \ell(\mathrm{M})$: the leverage of an energy is a property of the *materials*, not of this particular sample.

Importance sampling, and the chicken-and-egg

Uniform raster sampling wastes measurements on uninformative energies and regions. The right measure is leverage scores:

$$\ell_i(\mathrm{D})^2 := \|\mathrm{U}_{\mathsf{dominant}}(i,:)\|_2^2$$

— how much row i matters to the dominant subspace.

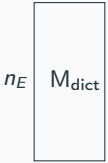
Problem: computing $\ell(\mathrm{D})$ needs D — which is the thing we are trying to avoid measuring.

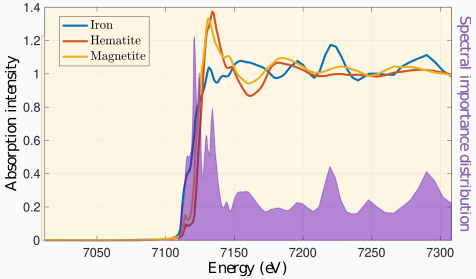
In principle $\ell(\mathrm{D}) \approx \ell(\mathrm{M})$: the leverage of an energy is a property of the *materials*, not of this particular sample.

Our answer: a data-driven surrogate. Use a dictionary $\mathrm{M}_{\mathsf{dict}}$ of reference spectra — we know roughly what materials might be there, even if we do not know where they are or in what proportion.

Spectral importance from a dictionary

Compute leverage scores of the (small, $n_E \times |\text{dict}|$) matrix M_{dict} and sample s_E energies from them.

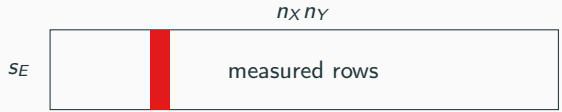




resulting energy sampling distribution

- cheap: the dictionary is tiny
- and it never has to be right about *this* sample, only about the physics

Spatial importance: a distribution over *blocks*



Having measured s_E full energy scans, we hold a short-and-wide matrix F — a few rows of D, all of their columns.

We now need *spatial* importance. Two complications:

- we cannot pick individual pixels, only whole spatial rows — so we need a distribution over *blocks of columns*, not columns
- nearby pixels are highly correlated, so the top-k scoring blocks would all tell us the same thing

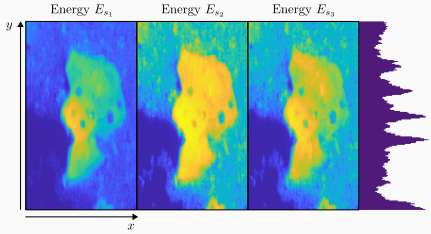
> Sampling independently from a fixed score vector is the wrong thing to do here.

Adaptive Randomized Pivoting

ARP (Cortinovis & Kressner, 2024) computes leverage-type scores *adaptively*: after each pick, the information it captured is removed before the next score is computed.

1. compute $\bar{U} := U_{\text{dominant}}$ of F, and its leverage scores
2. select an index i_1 accordingly
3. update $\bar{U} \leftarrow \bar{U}$ with $\bar{U}(:, i_1)$ reduced to zero (via Householder reflectors)
4. recompute the leverage scores of the updated $\bar{U}$
5. repeat until $\bar{U}$ is fully reduced

Step 3 is the whole idea: it is what stops us from buying the same measurement twice.



the resulting spatial importance distribution

The scheme so far

Algorithm 1: importance distributions

1. compute leverage scores of M_{dict}
2. sample s_E energies accordingly
3. change unfolding $\rightarrow \mathsf{F}$
4. use ARP on F

Obtained: a *spectral* and a *spatial* importance distribution.

The scheme so far

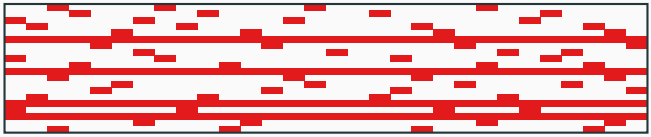
Algorithm 1: importance distributions

1. compute leverage scores of $\mathsf{M}_{\mathsf{dict}}$
2. sample s_E energies accordingly
3. change unfolding $\rightarrow$ F
4. use ARP on F

Obtained: a *spectral* and a *spatial* importance distribution.

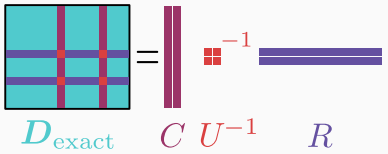
Now we have to turn two distributions into an actual experiment.

RISS: raster importance sampling



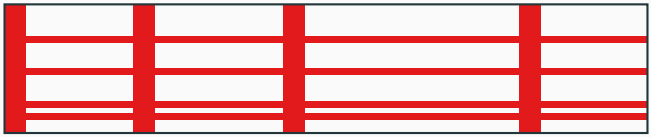
- use Algorithm 1 to measure a few full energy scans and get a spatial distribution
- **for** each non-measured energy E:
  - set the beam energy to E
  - sample s_R spatial rows from the spatial distribution — freshly each time
  - measure those rows
- complete the measured dataset using loopedASD

CUR: sample rows and columns, get the completion for free



- importance distributions come with *theoretical guarantees* here — leverage-score sampling is exactly what CUR theory asks for
- a natural fit for experiment design: CUR wants whole rows and whole columns, and that is all the instrument can give us anyway
- and it *is* a completion: D ≈ CUR interpolates the unmeasured entries, no separate solver needed

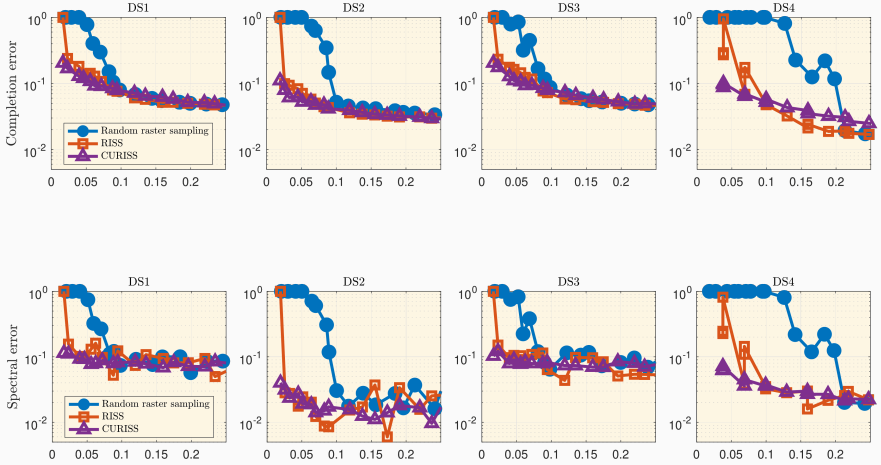
CURISS: CUR importance sampling



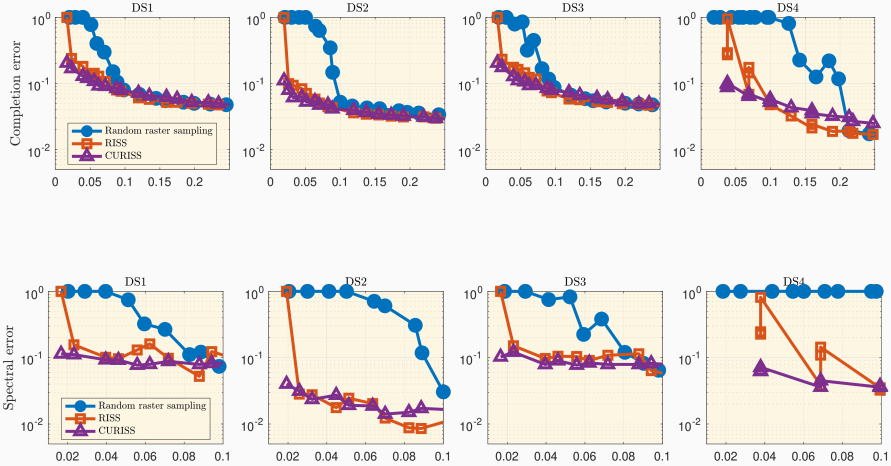
- use Algorithm 1 to measure a few full energy scans and get a spatial distribution
- sample s_R spatial rows once
- **for** each non-measured energy E: set the beam to E, measure the same sampled rows
- complete the measured dataset using CUR

The one change from RISS — reusing the same rows at every energy — is what turns the sample into a genuine CUR cross.

Results


DS1
DS2
DS3
DS4
Completion error
Spectral error
Random raster sampling
RISS
CURISS
10^0
10^{-1}
10^{-2}
0
0.05
0.1
0.15
0.2

Results

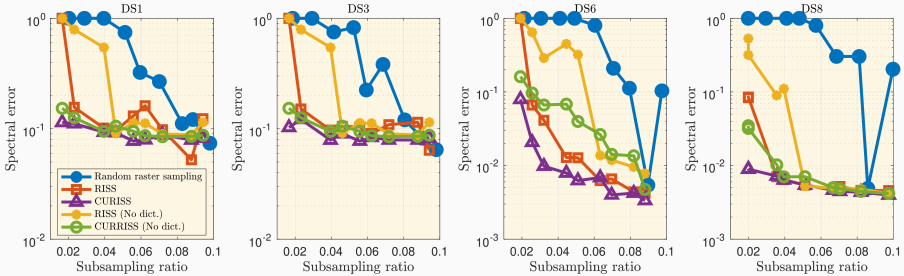

DS1
DS2
DS3
DS4
Completion error
Spectral error
Random raster sampling
RISS
CURISS

Results: what if there is no dictionary?

What if M_{dict} is not available? $\rightarrow$ uniform sampling for the energy scans

Results: what if there is no dictionary?

What if M_{dict} is not available? $\rightarrow$ uniform sampling for the energy scans



Still works — the spatial half of the scheme (ARP) carries most of the benefit.

Results: reconstructed spectra

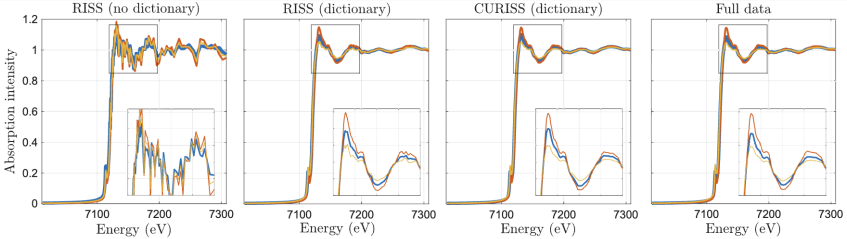


Figure 13: Spectra obtained from DS6 at 6% subsampling for different sampling strategies. The right figure shows the spectra obtained from the full data. The spectral errors are from left-to-right 0.048, 0.006, and 0.004.

ACURISS: making it adaptive

Goal: refine the subsampling adaptively, starting from CURISS with an initial subsampling ratio p_0

Refinement

- How do we update the importance distributions?

Stopping criterion

- How do we judge the accuracy of the refined CUR, and know when to stop?

ACURISS: making it adaptive

Goal: refine the subsampling adaptively, starting from CURISS with an initial subsampling ratio p_0

Refinement

- How do we update the importance distributions?

1. initial CUR by CURISS with p_0
2. alternate adding a full energy scan (a row of D) and a spatial row scan (a block of columns of D)

For a new energy scan: leverage scores of M_{dict} are already computed — just zero out the energies already measured.

Stopping criterion

- How do we judge the accuracy of the refined CUR, and know when to stop?

ACURISS: making it adaptive

Goal: refine the subsampling adaptively, starting from CURISS with an initial subsampling ratio p_0

Refinement

- How do we update the importance distributions?

1. initial CUR by CURISS with p_0
2. alternate adding a full energy scan (a row of D) and a spatial row scan (a block of columns of D)

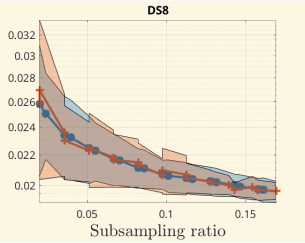
For a new energy scan: leverage scores of M_{dict} are already computed — just zero out the energies already measured.

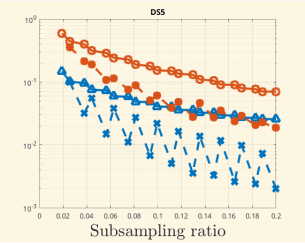
Stopping criterion

- How do we judge the accuracy of the refined CUR, and know when to stop?

On a beamline there is no reference to compare against, and every extra measurement costs time and does damage. The stopping rule is the deliverable, not a detail.

ACURISS vs CURISS





Adapting lets us spend the measurement budget where the data turns out to need it, rather than where the dictionary guessed it would.

Deep dive II: what actually happened

- the datacube is low rank because the sample contains few materials
- leverage scores say which rows and columns carry that rank — and a dictionary lets us estimate them *before* measuring
- ARP handles the constraint that we can only choose whole blocks
- CUR then matches the sampling pattern the instrument is capable of, and supplies the completion at the same time

Deep dive II: what actually happened

- the datacube is low rank because the sample contains few materials
- leverage scores say which rows and columns carry that rank — and a dictionary lets us estimate them *before* measuring
- ARP handles the constraint that we can only choose whole blocks
- CUR then matches the sampling pattern the instrument is capable of, and supplies the completion at the same time

Low rank became an experiment design.

We did not speed up a computation on data we already had. The decomposition told the instrument what to measure — less beamtime, less radiation damage, same answer.

Wrapping up

Closing — and the question left open

Today, both deep dives exploited low rank in a *matrix*:

- a solution manifold that we compressed into a k-dimensional subspace
- a datacube that we flattened into $\mathrm{D} \in \mathbb{R}^{n_E \times n_X n_Y}$ and subsampled

Closing — and the question left open

Today, both deep dives exploited low rank in a *matrix*:

- a solution manifold that we compressed into a k-dimensional subspace
- a datacube that we flattened into $\mathrm{D} \in \mathbb{R}^{n_E \times n_X n_Y}$ and subsampled

Next lecture: what replaces the SVD when the object has more than two modes, and how randomization makes it fast.