Accelerating Control Algorithms with Randomized Numerical Linear Algebra

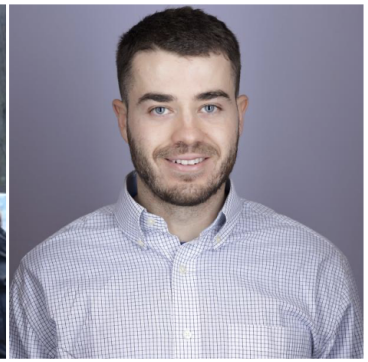
James Anderson

Department of Electrical Engineering
Columbia University

Control Seminar, University of Michigan

February 7, 2025

Acknowledgements

- **Han Wang**, Columbia (now ByteDance)
- **Morgan Jones**, University of Sheffield, UK
- NSF: grants 2144634 & 2231350

The problem?

We have more data than we can compute with

Systems are getting bigger, sensors are getting cheaper & smaller

Phenomenal hardware advances, many work-horse algorithms fail to take advantage

Why Randomization

Trace estimation:

- given an $n \times n$ positive semidefinite matrix A
- provide an estimate of $\mathbf{trace}(A)$ through the primitive $x \mapsto Ax$

Girard's trace estimator

- let $\omega \in \mathbb{C}^n$ be a random vector such that $\mathbb{E}[\omega\omega^*] = I$

$$\text{if } W = \omega^* A w \quad \text{then} \quad \mathbb{E}W = \mathbf{trace}(A)$$

- reduce the variance by averaging independent copies

$$\mathcal{W} = \frac{1}{l}\sum_{i=1}^{l} W_i \quad \Rightarrow \quad \mathbb{E}\mathcal{W} = \mathbf{trace}(A) \quad \texttt{// unbiased estimate}$$

- variance decreases

$$\mathrm{Var}[\mathcal{W}] = \frac{1}{l}\mathrm{Var}[W] \quad \texttt{// decrease proportional to sample size}$$

Why Randomization

Trace estimation:

- given an $n \times n$ positive semidefinite matrix A
- provide an estimate of **trace**(A) through the primitive $x \mapsto Ax$

Choice of distribution:

- $\omega \in \mathcal{N}(0, I)$
- $\omega \in \text{unif}\{\pm 1\}^n$

$$\mathbf{Var}[\mathcal{W}] \leq \frac{2}{l}\|A\|\mathbf{trace}(A)$$

for general matrices, variance may not be related to trace

Outline

1. Randomized SVD
   - Linear system realization
2. Hessian sketching
   - Markov parameter estimation
3. Approximate projections
   - SDP solver

Singular Value Decomposition

Given $A \in \mathbf{C}^{m \times n}$, define $p = \min\{m, n\}$, the SVD of A is given by

$$A = U \Sigma V^*$$

where

- $U \in \mathbf{C}^{m \times m}$ is unitary
- $V \in \mathbf{C}^{n \times n}$ is unitary
- $\Sigma \in \mathbf{R}^{m \times n}$ is diagonal

when $m \neq n$, the matrix Σ takes the form

$$\Sigma = \begin{bmatrix} \widehat{\Sigma} \\ \mathbf{0} \end{bmatrix} \quad \text{or} \quad \Sigma = \begin{bmatrix} \widetilde{\Sigma} & \mathbf{0} \end{bmatrix}$$

where $\widehat{\Sigma} = \mathbf{diag}(\sigma_1, \ldots, \sigma_n)$ and $\widetilde{\Sigma} = \mathbf{diag}(\sigma_1, \ldots, \sigma_m)$, and

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$$

Optimal Low-Rank Approximation

Error bounds

The Eckart-Young theorem tells us that for eack k:

$$\sigma_{k+1} = \begin{array}{ll} \underset{X}{\text{minimize}} & \|A - X\| \\ \text{subject to} & \mathbf{rank}(X) \leq k \end{array}$$

an optimal X can be constructed from the k-dominant singular vectors of A:

$$X^\dagger = \sum_{i=1}^{k} \sigma_i u_i v_i^*$$

Computational cost

- The l-truncated SVD takes $O(mnl)$ flops using classical methods
- for even moderate sized mn, $O(mnl)$ is too large!

Image Compression

- 1350×1080 pixels (1,458,000 words)

Image Compression

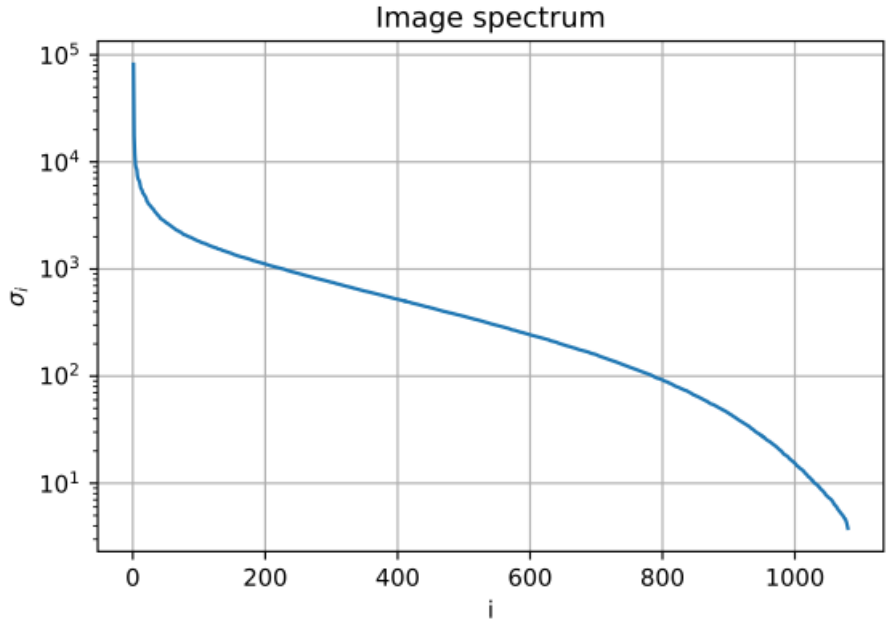
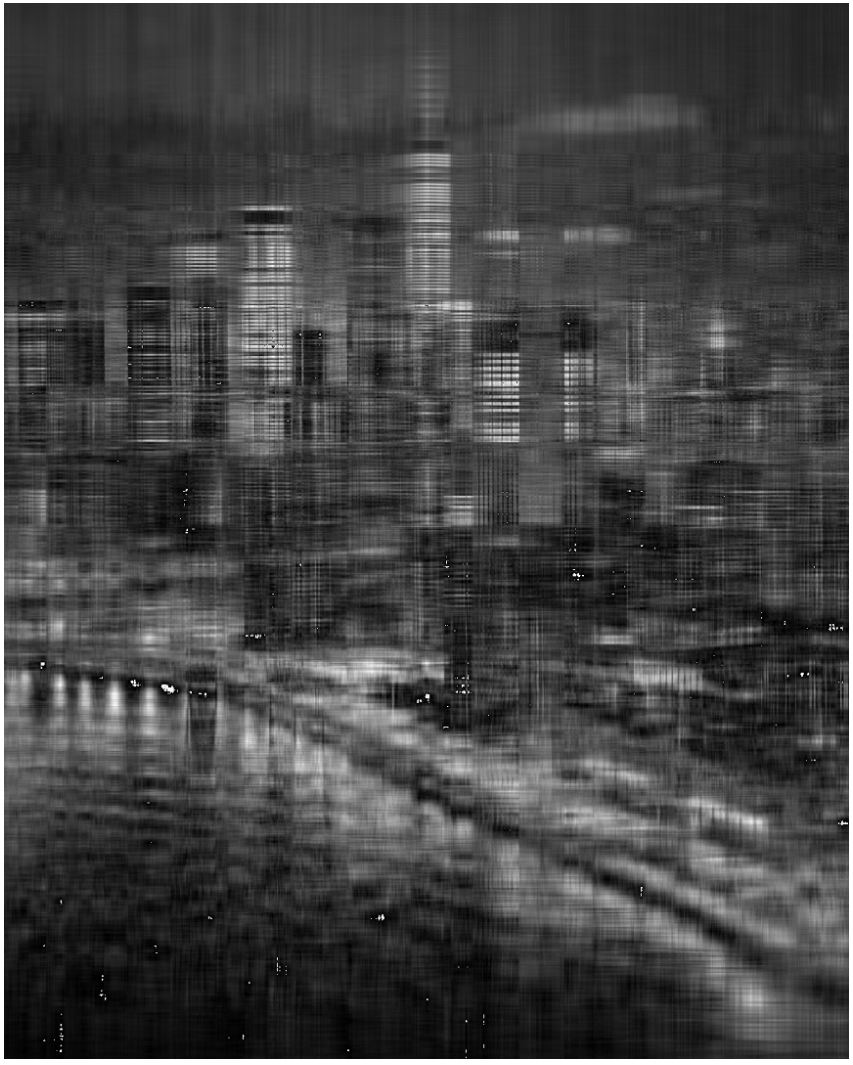


Image Compression

- rank-250 approximation, 607,500 words, 3x compression

Image Compression

- rank-20 approximation, 48,600 words, 30x compression



Randomized SVD Algorithm: RSVD

SIAM REVIEW
Vol. 53, No. 2, pp. 217–288

Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*

N. Halko†
P. G. Martinsson†
J. A. Tropp‡

Stage 1

Find a matrix Q such that

$$A \approx QQ^*A, \quad \text{where } Q \text{ has orthonormal columns.}$$

Interpret $\approx$ as meaning that Q satisfies

$$\|(I - QQ^*)A\| \leq \epsilon, \quad \text{for some acceptable } \epsilon > 0$$

//columns of Q form an approximate basis for range(A)

Stage 2

Using Q and your favorite "classical" method, compute an SVD and rearrange

Stage 1: Random Sampling

Objective

Given a matrix $A \in \mathbf{C}^{m \times n}$ compute an approximate basis for $\mathbf{range}(A)$.

Algorithm

1. Sample **range**(A) by generating independent "random" vectors $\omega^{(i)}$ for $i = 1, \ldots, k$:
$$y^{(i)} = A\omega^{(i)} \qquad \Longleftrightarrow \qquad Y = A\Omega$$
2. Orthogonalize the columns of Y

Note

- The matrix Y is a called a **sketch** of A
- $Y \in \mathbf{C}^{m \times k}$ where $k \ll \min\{m, n\}$

Stage 1: Analysis

HMT's "randomized range finder"

Stage 1:

1. Draw a **random matrix** $\Omega \in \mathbf{R}^{n \times (k+l)}$ `// draw` l `extra samples`
2. Form the **sketch** $Y = A\Omega$ `// cost` $O(mn(k+l))$
3. Construct orthogonal basis: $[Q, \sim] = \texttt{QR}(Y, \texttt{''thin''})$ `// cost` $O(n(k+l)^2)$

Stage 1: Analysis

HMT's "randomized range finder"

Stage 1:

1. Draw a **random matrix** $\Omega \in \mathbf{R}^{n \times (k+l)}$ `// draw` l `extra samples`
2. Form the **sketch** $Y = A\Omega$ `// cost` $O(mn(k+l))$
3. Construct orthogonal basis: $[Q, \sim] = \texttt{QR}(Y, \texttt{''thin''})$ `// cost` $O(n(k+l)^2)$

Theorem (Halko, Martinsson, Tropp, 2011)

Given $A \in \mathbf{R}^{m \times n}$, a target rank $k \geq 2$, and parameter $l \geq 2$ such that $k + l \leq \min\{m, n\}$. The algorithm above produces a matrix Q with orthonormal columns that satisfies

$$\mathbf{E} \, \|A - QQ^* A\| \leq \left[1 + \frac{4\sqrt{k+l}}{l-1} \sqrt{\min\{m, n\}} \right] \sigma_{k+1}.$$

//stage 1: comments & modifications

Slowly decaying spectrum

Can boost accuracy by incorporating **power iterations**. Based on the observation:

$$W := (AA^*)^q A$$

has the same singular vectors as A. But

$$\sigma_j(W) = \sigma_j(A)^{2q+1}, \quad j = 1, 2, \ldots$$

- Replace $Y = A\Omega$ with $Y = W\Omega$
- error bound becomes

$$\left[1 + 4\sqrt{\frac{2\min\{m,n\}}{k-1}}\right]^{\frac{1}{2q+1}}]\sigma_{k+1}$$

Target rank selection

Straight forward to adaptively construct the basis vectors Q until tolerance is met. See **[HMT]** for details.

Stage 2: Building an Approximate SVD

$$A \approx Q \, Q^* \, A$$

$$= Q \, Q^* A$$

$$= Q \, U \, \Sigma \, V^* = Q^* U \, \Sigma \, V^*$$

Cost of Doing Business

Choice of test matrix Ω

- Dominant cost is computing $A\Omega$
- When $A \in \mathbf{C}^{m\times n}$ and $\Omega \in \mathbf{C}^{n\times l}$ cost of $A\Omega$ is $O(mnl)$
- Structured random test matrix, reduce complexity to $O(mn \log l)$
  - *e.g.* subsampled random Fourier Transform (SRFT)

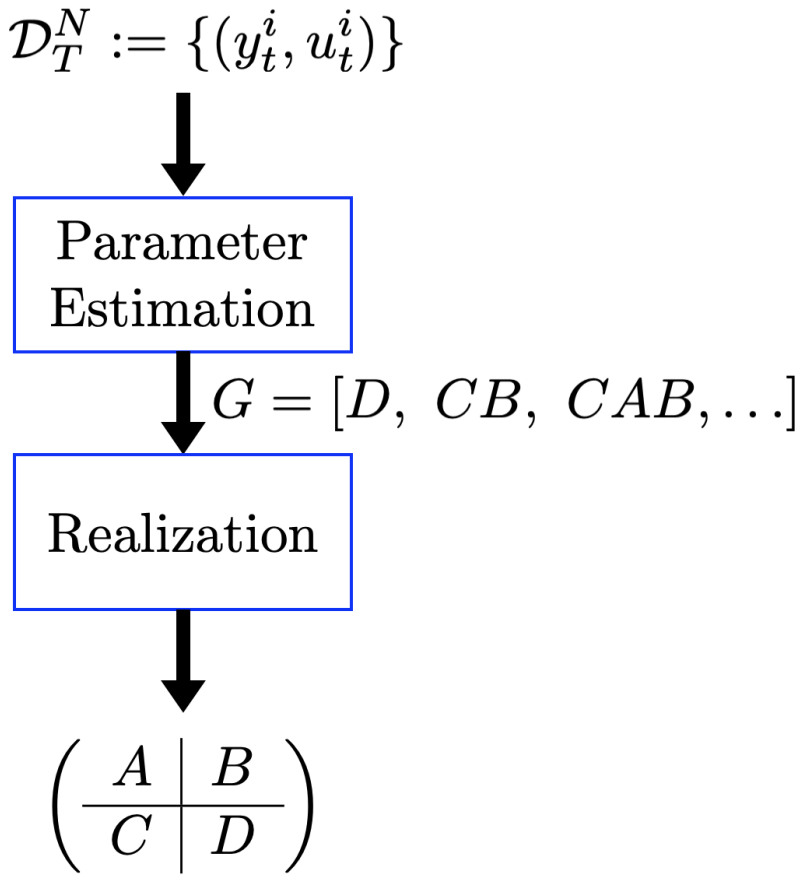
Comparing RSVD with other SVD methods

Method	Time complexity	Space	Passes
RSVD	$\mathcal{O}(mn \log k)$	$\mathcal{O}(mn)$	1
Krylov	$\mathcal{O}(mnk)$	$\mathcal{O}(mn)$	k
Truncated	$\mathcal{O}(mn\text{min}(m,n))$	$\mathcal{O}(mn)$	k

Table: Comparison of complexity

System Identification Pipeline

Linear System Identification Pipeline



System Identification

Problem formulation

Collect data from the LTI system

$$x_{t+1} = Ax_t + Bu_t + w_t$$
$$y_t = Cx_t + Du_t + v_t$$

The data

- We have $N < \infty$ experiments over finite time-horizon T
- Observe: $\{y_t^i\}_{t=0}^T$, $\{u_t^i\}_{t=0}^T$ for $i = 1, \ldots, N$
- Assume: $u_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_u^2 I)$, $w_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_w^2 I)$, $v_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_v^2 I)$

Objective

Find matrices $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ that "best fit" observed data

Realization

The process of obtaining a state-space model from the Markov matrix

$$G = \begin{bmatrix} D & CB & CAB & \dots & CA^{T-2}B \end{bmatrix} \in \mathbf{R}^{m \times Tp}.$$

- When $w_t, v_t \equiv 0$ have access to G, otherwise must solve an optimization problem to obtain an estimate $\widehat{G}$
- The realization problem is to determine the state-space matrices from $\widehat{G}$, *i.e.*, a mapping

$$\widehat{G} \mapsto \left(\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{C} & \hat{D} \end{array} \right)$$

Ho-Kalman Algorithm

Assumptions

- (A, B, C) is minimal
- $n = \mathbf{rank}(\mathcal{H}) \leq \min\{T_1, T_2\}$

Algorithm

1. From G construct the Hankel matrix

$$\mathcal{H} = \left[\begin{array}{ccc|c} CB & CAB & \dots & CA^{T_2}B \\ CAB & CA^2B & \dots & CA^{T_2+1}B \\ CA^2B & CA^3B & \dots & CA^{T_2+2}B \\ \vdots & \vdots & \vdots & \vdots \\ CA^{T_1-1}B & CA^{T_1}B & \dots & CA^{T_1+T_2-1}B \end{array}\right] \in \mathbf{R}^{pT_1 \times m(T_1+1)}$$

where $T = T_1 + T_2 + 1$.

By assumption $\mathcal{H}$ and $\mathcal{H}^-$ are full rank

Ho-Kalman Algorithm

Assumptions

- (A, B, C) is minimal
- $n = \mathbf{rank}(\mathcal{H}) \leq \min\{T_1, T_2\}$

Step 2: Factorization

$\mathcal{H}^-$ is full rank and so it can be factored as

$$\mathcal{H}^- = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{T-1} \end{bmatrix} \begin{bmatrix} B & AB & \dots & A^{T_2-1}B \end{bmatrix}$$
$$= \mathcal{O}\mathcal{Q}$$

Ho-Kalman Algorithm

Algorithm

1. From G construct the Hankel matrix

$$\mathcal{H} = \left[\begin{array}{ccc|c} CB & CAB & \dots & CA^{T_2}B \\ CAB & CA^2B & \dots & CA^{T_2+1}B \\ CA^2B & CA^3B & \dots & CA^{T_2+2}B \\ \vdots & \vdots & \vdots & \vdots \\ CA^{T_1-1}B & CA^{T_1}B & \dots & CA^{T_1+T_2-1}B \end{array}\right] \in \mathbf{R}^{pT_1 \times m(T_1+1)}$$

   where $T = T_1 + T_2 + 1$.

2. Compute a n-**truncated SVD** of $\mathcal{H}^-$: $\mathcal{H}^- \approx U_n \Sigma_n V_n^*$

3. $\mathcal{O} = U_n \Sigma_n^{\frac{1}{2}}$, $\mathcal{Q} = \Sigma_n^{\frac{1}{2}} V_n^*$

4. $\hat{A} = \mathcal{O}^\dagger \mathcal{H}^+ \mathcal{Q}^\dagger$

Comments

- Realization is a non-convex problem
- Ho-Kalman algorithm provides realizations unique up to a similarity transform

$$(A, B, C, D) \mapsto (TAT^{-1}, TB, CT^{-1}, D)$$

- Computation cost: at least $O(pmnT_1T_2)$ flops from the SVD step.
- Robustness guarantee [Oymak & Ozay, 2019]:

$$\begin{aligned}\max \left\{ \|\hat{A} - T^{-1}AT\|, \|\hat{B} - T^{-1}B\|, \|\hat{C} - CT\| \right\} \\ \leq c\sqrt{\|G - \hat{G}\|} = O\left(\frac{1}{N^4}\right).\end{aligned}$$

Stochastic Ho-Kalman Algorithm

Idea

Replace the truncated SVD with a randomized SVD!

- Measurements contain noise, so full accuracy isn't necessary anyway
- The deterministic algorithm struggles with modest systems sizes

Main Result (Informal)

Theorem

The stochastic Ho-Kalman Algorithm reduces the computational complexity of the realization problem from $O(pmn^3)$ to $O(pmn^2 \log n)$ when $T_1 = T_2 = n$. The achievable robustness is the same as deterministic algorithm.

Full details in Wang and Anderson, ACC'22

Numerical Experiments

Scalability and Approximation Error

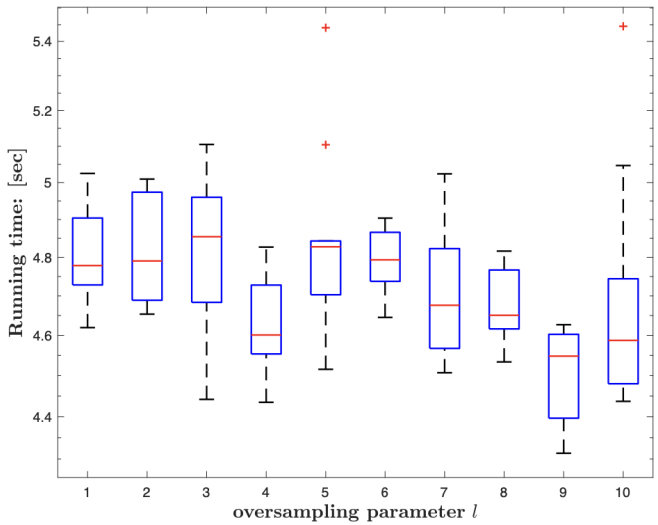
Eg	(n, m, p, T)	$\dim(\hat{H}^-)$	**Running Time [s]**		**Realization Error**	
			deterministic	**stochastic**	**deterministic**	**stochastic**
1	(30,20,10,90)	450×880	0.1079	0.0156	7.64e-04	7.70e-04
2	(40,30,20,100)	2000×2970	5.7456	0.0897	6.67e-04	1.19e-03
3	(60,50,40,360)	7200×8950	227.0116	0.9323	8.27e-04	1.75e-03
4	(100,80,50,500)	12500×19920	922.8428	4.4581	6.53e-04	1.66e-03
5	(120,110,90,600)	27000×32890	Inf	17.6603	N/A	1.96e-03
6	(200,150,100,600)	30000×44850	Inf	52.1762	N/A	1.45e-03

- $(n, m, p, T) =$ (state, input, output, horizon)
- No parallelization used with the randomized SVD
- No power iterations
- Oversampling parameter: $p = 10$
- Relative error:

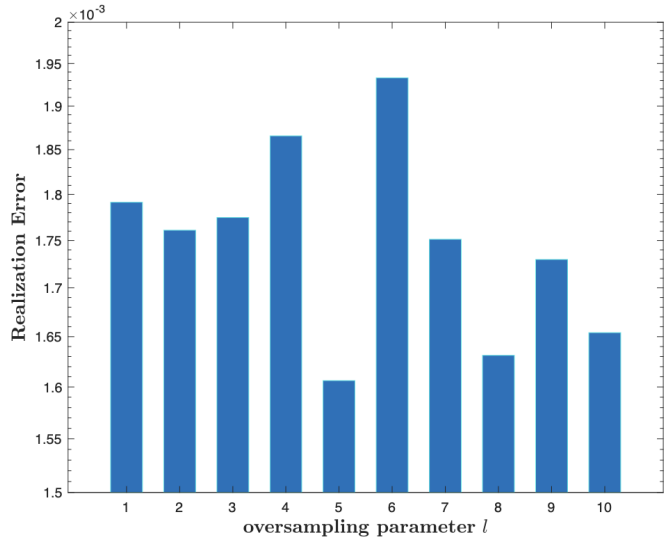
$$\frac{\|\mathcal{G} - \hat{\mathcal{G}}\|_{\mathcal{H}_\infty}}{\|\mathcal{G}\|_{\mathcal{H}_\infty}}$$

Additional Numerical Experiments

Oversampling parameter



(a) Running time of stochastic Ho-Kalman Algorithm using `RSVD` with oversampling parameter l.

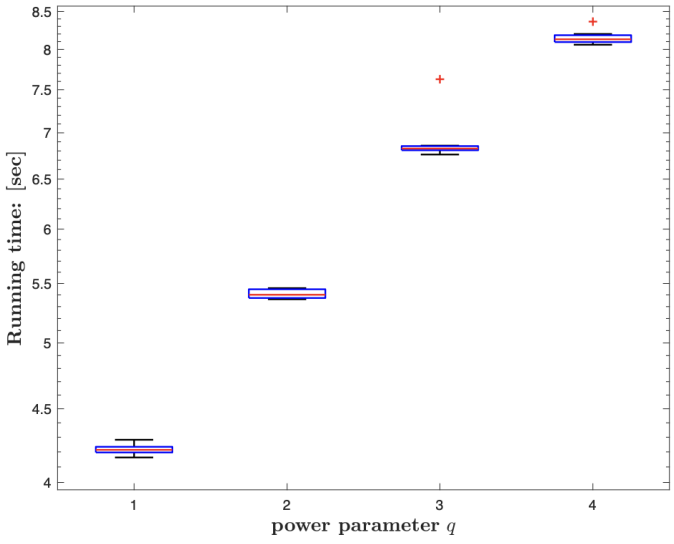


(b) Realization error of stochastic Ho-Kalman Algorithm using `RSVD` with oversampling parameter l.

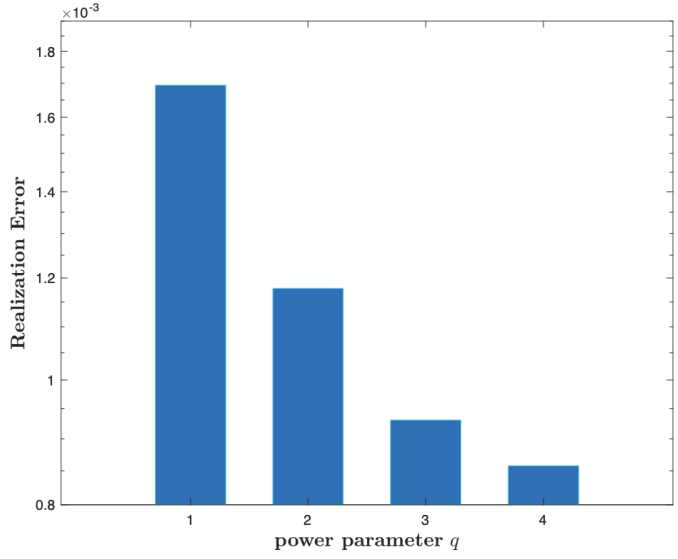
- Example 4: $(n, m, p, T) = (100, 80, 50, 500)$
- No power iterations

Additional Numerical Experiments

Power Iterations



(c) Running time of stochastic Ho-Kalman Algorithm with varying power parameter q. The oversampling parameter l is 10.



(d) Realization error of the stochastic Ho-Kalman Algorithm with varying power parameter q. The oversampling parameter l is 10.

- Example 4: $(n, m, p, T) = (100, 80, 50, 500)$

Stochastic Realization Algorithm: Conclusions

Methodology

- Performance degradation due to randomization almost negligible
- Sample complexity bounds remain intact
- Order of magnitude gain in computation time (for large instances)
- Not yet exploited parallel computing

Algorithm tuning

- Algorithm performance can be boosted by including power iterations
  - This does impact running time
- Algorithm performance not sensitive to oversampling rate
  - Doesn't appear to impact running time

Sketch and Solve

Sketching as a Tool for Numerical Linear Algebra

now
the essence of knowledge
Boston — Delft

David P. Woodruff
IBM Research Almaden
dpwoodru@us.ibm.com

Sketched Least squares Given $A \in \mathbf{R}^{n \times d}$ and $S \in \mathbf{R}^{m \times n}, m \ll n$

don't solve $x^\star \in \arg\min_{x \in \mathcal{C}} \underbrace{\frac{1}{2n}\|Ax - y\|^2}_{f(x)}$, instead, solve $x^\sharp \in \arg\min_{x \in \mathcal{C}} \frac{1}{2n}\|S(Ax - y)\|^2$

- aim for ϵ approximate solutions $f(x^\star) \leq f(x^\sharp) \leq (1+\epsilon)^2 f(x^\star)$
- m depends on ϵ^{-2}
- Pilanci and Wainwright (JMLR'17) concretely show why this is a bad idea

Iterative Hessian Sketch

Bartan & Pilanci **[BP]** propose the equivalent LS problem

$$\underset{x \in \mathcal{C}}{\text{minimize}} \ \|Ax\|^2 - \langle x, A^T y \rangle.$$

Newton's method produces updates

$$x_{t+1} = x_t - \alpha (A^T A)^{-1} A^T (Ax_t - b).$$

If we sketch A in the norm only (and keep track of residuals), we get

$$x_{t+1} = x_t - \alpha (A^T S_t^T S_t A)^{-1} A^T (Ax_t - b).$$

Distribute this over q nodes:

$$x_{t+1} = x_t - \alpha \frac{1}{q} \sum_{k=1}^{q} (A^T S_{t,k}^T S_{t,k} A)^{-1} A^T (Ax_t - b).$$

Distributed Iterative Hessian Sketch

Proposed by Bartan and Pilanci **[BP]**

Generalized sketching and refined the analysis [Wang & Anderson 2022]

Algorithm 1 Distributed Iterative Hessian Sketch

Input: Number of iterations T, step size μ.
for $t = 1$ **to** T **do**
  for workers $k = 1$ **to** q in parallel **do**
    Sample $S_{t,k} \in \mathbb{R}^{m \times n}$.
    Sketch the data $S_{t,k}A$.
    Compute gradient $g_t = A^T(Ax_t - b)$.
    Solve $\hat{\Delta}_{t,k} = \arg\min_\Delta \frac{1}{2}\|S_{t,k}A\Delta\|_2^2 + g_t^T\Delta$ and send to master.
  end for
  Master: Update $x_{t+1} = x_t + \mu\frac{1}{q}\sum_{k=1}^{q}\hat{\Delta}_{t,k}$ and send x_{t+1} to workers.
end for
return x_T

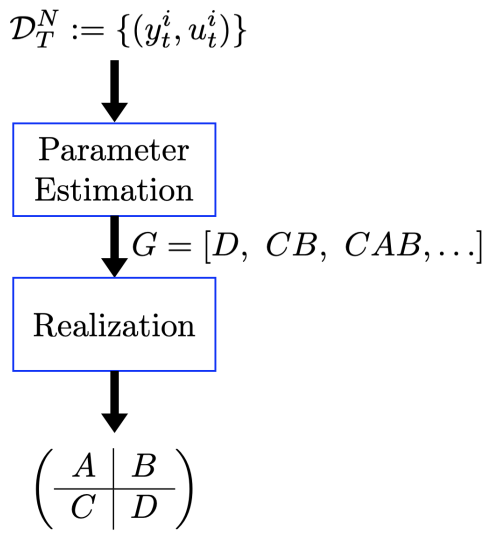
System Identification

Parameter estimation

Given observed data $\mathcal{D}$ believed to have been generated by

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + Du_t + v_t, \end{aligned}$$

estimate the Markov parameters.



Learning Markov Parameters

OLS formulation

An estimate $\hat{G}$ of the Markov matrix is obtained by solving

$$\underset{X}{\text{minimize}} \ \|UX - Y\|_F^2$$

where

- $X \in \mathbf{R}^{mT \times p}$
- U and Y are Toeplitz matrices
- Solution via QR decomposition: $O(NT(mT)^2)$

Result

DIHS applied to OLS problem

- Assume number of "rollouts", $N > 8mT + 16\log(T/\delta)$
- Define $\kappa = mNT^2$

Theorem (Informal)

Fix $\delta \in (0,1)$ and $\rho \in (0, \frac{1}{2})$. If the sketch dimension satisfies

$$s > \frac{c_0 \log^4(\kappa)}{\rho^2} mT,$$

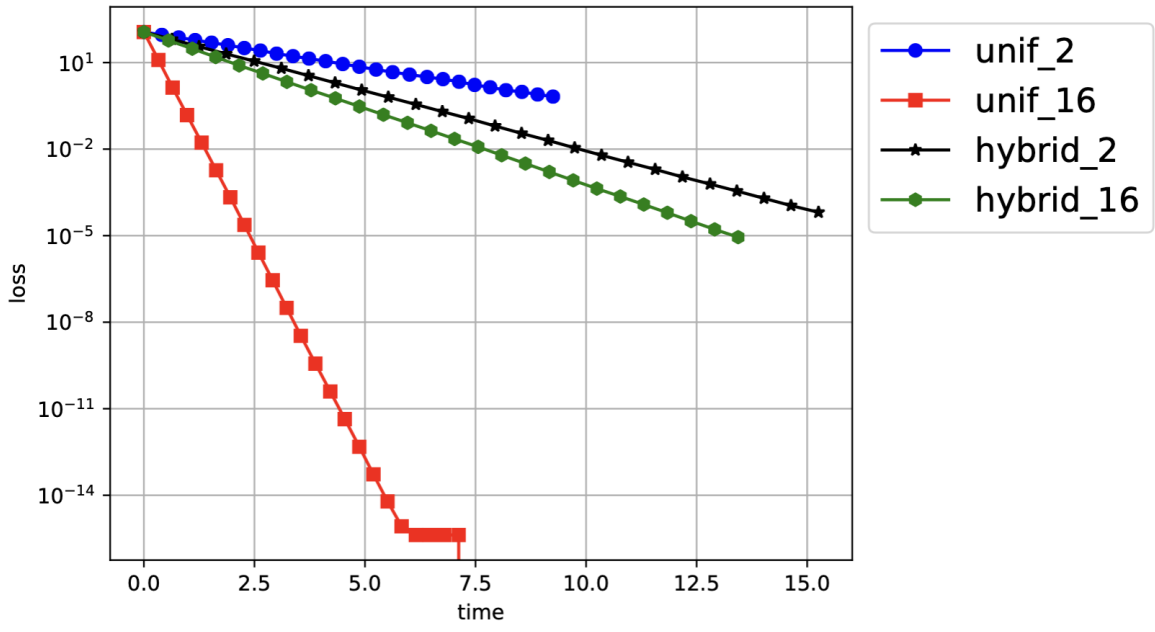
then at iteration k, DIHS satisfies

$$\|X_k - X^{\mathrm{LS}}\|_F \leq 3\rho^k \|X^{\mathrm{LS}}\|_F$$

with high probability.

Numerical Experiments

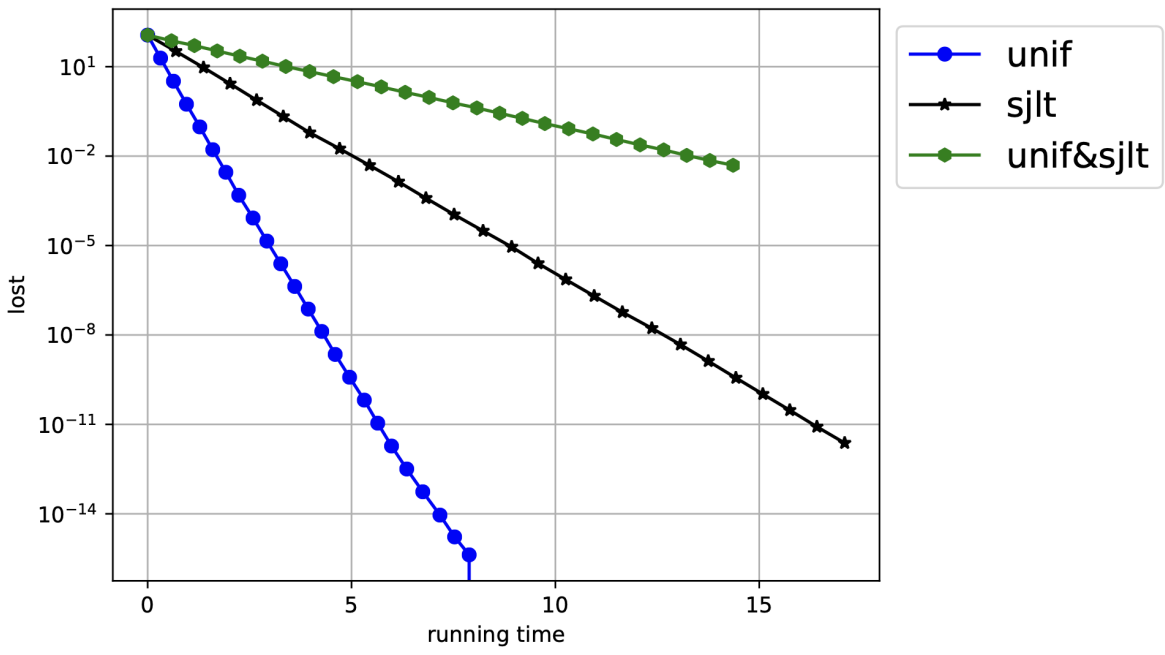
Sketch selection and number of workers



- 40 states, 30 inputs, 20 outputs
- $\sim 45M$ data points

Numerical Experiments

12 workers: same system



Hessian Sketching: Conclusions

- Randomized numerical linear algebra can be applied to full Sys-ID pipeline
- General least squares problems (and beyond)
- Applications to control synthesis?
- Skipped most of the theoretical results - see our papers!

Semidefinite Programming Solvers

iterative algorithms for solving

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{s.t.} & x \in \mathcal{C} \end{array}$$

often involve a projection step

- projected subgradient methods

$$x_{k+1} = \Pi_{\mathcal{C}}(x_k - \alpha_k g_k)$$

- alternating direction method of multipliers

$$\begin{aligned} x_{k+1} &= \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{\rho}{2} \|x - z_k + u_k\|^2 \right) \\ z_{k+1} &= \Pi_{\mathcal{C}}(x_{k+1} + u_k) \\ u_{k+1} &= u_k + x_{k+1} - x_{k+1} \end{aligned}$$

algorithm performance depends on projection tractability

Semidefinite Programming

- conical form SDP

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{s.t.} & x \in \mathcal{C} \end{array}$$

- $\mathcal{C} = \mathbb{S}^n_+$

- optimal projection of a symmetric matrix onto $\mathbb{S}^n_+$:

$$\Pi_{\mathbb{S}^n_+}(X) = U \Lambda_+ U^T$$

where

$$\Lambda_+ = \begin{bmatrix} \max\{\lambda_1, 0\} & & \\ & \ddots & \\ & & \max\{\lambda_n, 0\} \end{bmatrix}$$

- applications in control: performance analysis, synthesis, sum-of-squares

Semidefinite Programming

- conical form SDP

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{s.t.} & x \in \mathcal{C} \end{array}$$

- $\mathcal{C} = \mathbb{S}^n_+$

- optimal projection of a symmetric matrix onto $\mathbb{S}^n_+$:

$$\Pi_{\mathbb{S}^n_+}(X) = U \Lambda_+ U^T$$

where

$$\Lambda_+ = \begin{bmatrix} \max\{\lambda_1, 0\} & & \\ & \ddots & \\ & & \max\{\lambda_n, 0\} \end{bmatrix}$$

- applications in control: performance analysis, synthesis, sum-of-squares

Idea

trade optimal projection for a computationally cheap approximation

Low-rank approximation of symmetric matrices

- computing a low-rank approximation and then projecting doesn't work

$$X = \begin{bmatrix} -3 & & \\ & -2 & \\ & & 1 \end{bmatrix} \Longrightarrow \underbrace{\begin{bmatrix} -3 & & \\ & 0 & \\ & & 0 \end{bmatrix}}_{\text{low-rank approx}} \Longrightarrow \underbrace{\begin{bmatrix} 0 & & \\ & 0 & \\ & & 0 \end{bmatrix}}_{\text{project}}$$

- the optimal low-rank projection is

$$\begin{bmatrix} -3 & & \\ & 0 & \\ & & 0 \end{bmatrix}$$

- our approach: map X to $B := \frac{X+\alpha I}{\alpha}$, which provides the relation:

$$\lambda_i(X) < 0 \iff \sigma_i(B) \in [0, 1) \quad \text{and} \quad \lambda_i(X) \geq 0 \iff \sigma_i(B) \in [1, \infty)$$

where α approximates minimum eigenvalue

Polynomial minimization via SoS

Degree= 4, SDP size: $[n, m] = [3026, 715]$	Mosek	`eig`	k=[0.2n] scal=1	k=[0.1n] scal=1	k=[0.05n] scal=1	k=[0.2n] scal=0	k=[0.1n] scal=0	k=[0.05n] scal=0
Computation time (s)	0.176	1.65	2.1	1.8	1.6	1.8	1.5	1.46
$\sqrt{\sum_{i=1}^{m}(\mathrm{Tr}(A_i^\top X) - b_i)^2}$	1.75e-9	4.5e-3	0.1	0.15	0.14	2.08	5.5	2.08
$\lvert\gamma - \gamma^*\rvert$	6.52e-10	1.54e-5	4.5e-5	1.76e-5	5.1e-5	3.1e-5	9.56e-4	1e-3
Degree= 6, SDP size: $[n, m] = [48401, 5005]$								
Computation time (s)	8.82	19.26	22.1	17.8	16.9	16.2	11.9	10.7
$\sqrt{\sum_{i=1}^{m}(\mathrm{Tr}(A_i^\top X) - b_i)^2}$	7e-12	4.5e-2	0.33	0.44	0.43	2.37	2.37	2.33
$\lvert\gamma - \gamma^*\rvert$	1.9e-12	2.7e-4	1.25e-4	2.3e-4	1.43e-4	3.53e-4	2.1e-3	9.9e-3
Degree= 8, SDP size: $[n, m] = [511226, 24310]$								
Computation time (s)	690.6	201	188	156.8	141.7	160.8	134.7	114.6
$\sqrt{\sum_{i=1}^{m}(\mathrm{Tr}(A_i^\top X) - b_i)^2}$	6.83e-11	0.17	3.37	3.1	3.3	3.73	9.78	9.75
$\lvert\gamma - \gamma^*\rvert$	1.1e-11	1.7e-3	1.7e-3	1.6e-3	1.2e-3	1.07e-4	6.4e-3	1.9e-2
Degree= 10, SDP size: $[n, m] = [4008005, 92378]$								
Computation time (s)	∞	2275.7	2389.9	1992.8	1820.6	2179	1909.5	1734.6
$\sqrt{\sum_{i=1}^{m}(\mathrm{Tr}(A_i^\top X) - b_i)^2}$	∞	3.5e-2	0.86	0.99	1.2	1.05	5.27	1.05
$\lvert\gamma - \gamma^*\rvert$	∞	4.2e-5	1.19e-4	7.5e-5	1.19e-4	4.8e-5	3e-4	1.3e-3

Conclusions

- 3 applications of RNLA applied to control
- mostly avoid the theoretical results here, check our papers for performance bounds!
- huge potential to improve on our results
- plenty more applications

Bibliography

- **[HMT]**: Halko, Martinsson, and Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate decompositions*. SIAM Review, 53.2, 2011.

- **[OO]**: Oymak and Ozay, *Non-asymptotic identification of LTI systems from a single trajectory*. Proc. of the American Control Conference, 2019.

- **[PW]**: Pilanci and Wainwright, *Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares*. Journal of Machine Learning Research 17, 2016.

- **[BP]**: Bartan and Pilanci, *Distributed averaging methods for randomized second order optimization*, arXiv:2002.0654, 2020.