Scalable Learning of Dynamical Systems

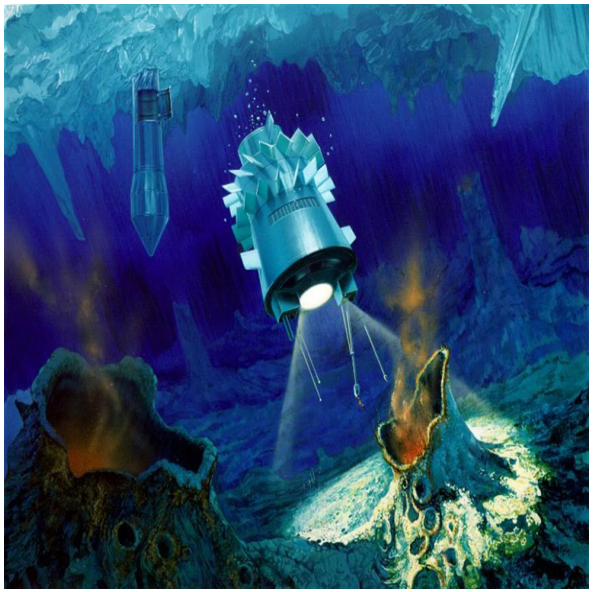
Low-rank approaches for increasing computational efficiency

Physics Informed Machine Learning Workshop,
University of Washington, Seattle, WA, USA
June 6-7, 2019

Alex Gorodetsky, Aerospace Engineering, UM
joint work with Boris Kramer, Cosmin Safta, John Jakeman,
Eurika Kaiser

UNIVERSITY OF MICHIGAN

Motivation: decision making in dynamical systems



- How do we *quickly learn and use stable* models for design, control, and optimization
- Minimize data requirements, reduce dimensionality, fast computation (beyond querying)
- Can we do as much as possible on-board and in real-time

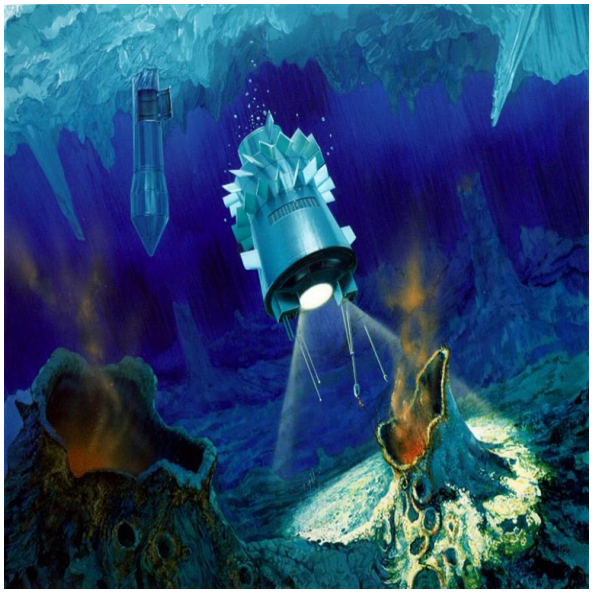
Motivation: decision making in dynamical systems

- How do we ***quickly learn and use stable*** models for design, control, and optimization
- Minimize data requirements, reduce dimensionality, fast computation (beyond querying)
- Can we do as much as possible on-board and in real-time

Motivation: decision making in dynamical systems

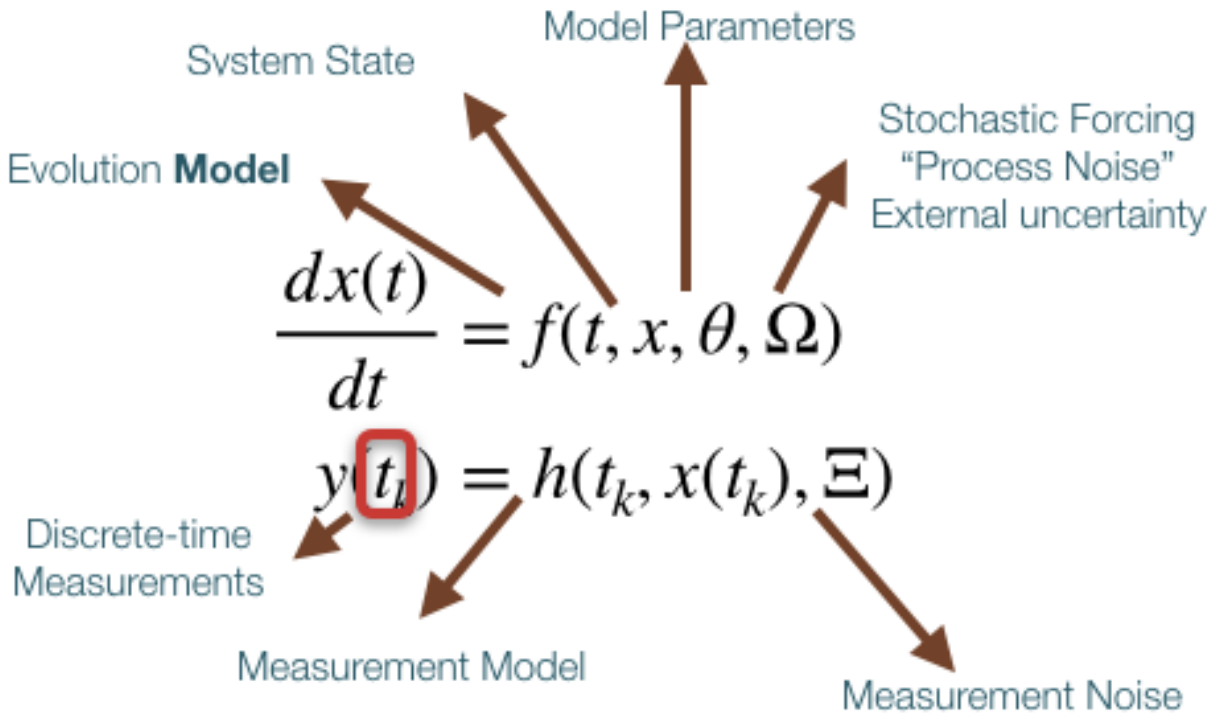
- How do we *quickly learn and use stable* models for design, control, and optimization
- Minimize data requirements, reduce dimensionality, fast computation (beyond querying)
- Can we do as much as possible on-board and in real-time

Motivation: decision making in dynamical systems



- How do we ***quickly learn and use stable*** models for design, control, and optimization
- Minimize data requirements, reduce dimensionality, fast computation (beyond querying)
- Can we do as much as possible on-board and in real-time

Idealized Learning Setup



In this talk:

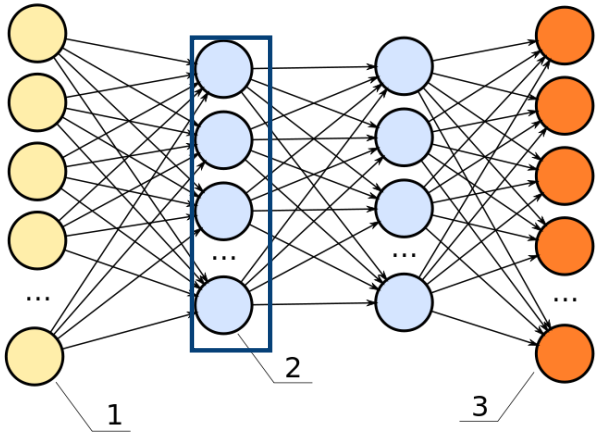
- How do we learn quickly if we have lots of good input-output data
- How do we learn quickly if f is nonlinear, with no stochasticity

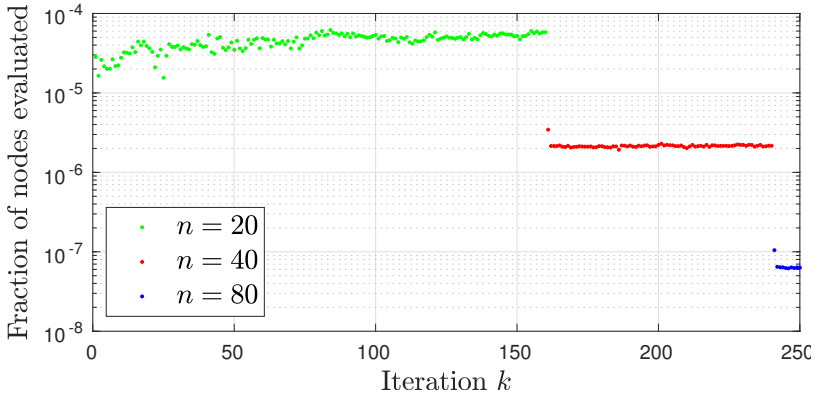
A class of tools: low-rank (tensor) decompositions
Wide-varying applications in ML, Science, and Eng.

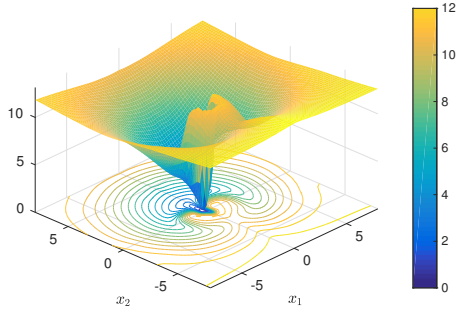
UNIVERSITY OF MICHIGAN

Low-rank tensor formats are widely applied to enable computational feasability

- (Insitu) Compress Neural Networks [Novikov et. al, 2015] [Kossaifi et. al, 2017]
- Compressing large scale scientific data [Austin et. al, 2016], (TB to GB)
- Large scale dynamic programming for Markov decision processes [Gorodetsky et. al., 2018] (TB to MB)







Two Case Studies

1. Linear Reduced Order Models arising in controlled systems
   - Scalability issues arise large numbers of outputs, inputs, and long times
   - Key idea: replace SVD with another low-rank format
2. Approximations of nonlinear systems:
   - Scalability issues arise from:
     - Same things as the linear models
     - Function parameterization: linear subspaces, neural networks, etc.
     - Data requirements for learning nonlinear parameterizations
   - Key idea: enforce and discover low-multilinear-rank structure

Two Case Studies

1. Linear Reduced Order Models arising in controlled systems
   - Scalability issues arise large numbers of outputs, inputs, and long times
   - Key idea: replace SVD with another low-rank format
2. Approximations of nonlinear systems:
   - Scalability issues arise from:
     - Same things as the linear models
     - Function parameterization: linear subspaces, neural networks, etc.
     - Data requirements for learning nonlinear parameterizations
   - Key idea: enforce and discover low-multilinear-rank structure

Takeway: **Assume and exploit** (don't discover) reduced-order structure during learning to improve scalability. Low-rank structure takes many forms and is found widely in physical systems.

Learning LTI via input-output relations

- Discrete time linear time-invariant (LTI) system

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

- $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{p\times n}, D \in \mathbb{R}^{p\times m}$ and $x_k := x(t_k)$
- Δt : sampling time, $t_k := k\Delta t,\ k \in \mathbb{N}_0^+$
- **Markov parameters** and IO relations

$$h_k := \begin{Bmatrix} D, & k = 0 \\ CA^{k-1}B, & k = 1, 2, \ldots \end{Bmatrix} \in \mathbb{R}^{p\times m}, \qquad y_k = \sum_{i=0}^{k} h_i u_{k-i}$$

Data-driven model: $\{h_i\}_{i=1,\ldots,2s} \Rightarrow (A, B, C, D)$?? (Partial realization problem)

Partial realization and Hankel Matrix

The **Hankel matrix** is defined as:

$$H := \begin{bmatrix} h_1 & h_2 & \dots & h_s \\ h_2 & h_3 & \dots & h_{s+1} \\ \vdots & \vdots & \ddots & \vdots \\ h_s & h_{s+1} & \dots & h_{2s-1} \end{bmatrix} = \begin{bmatrix} CB & CAB & \dots & CA^{s-1}B \\ CAB & CA^2B & \dots & CA^sB \\ \vdots & \vdots & \ddots & \vdots \\ CA^{s-1}B & CA^sB & \dots & CA^{2s-2}B \end{bmatrix}$$

The Hankel matrix can be factored as (from systems theory)

$$H = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix}}_{\text{Observability}} \underbrace{\begin{bmatrix} B & AB & \dots & A^{s-1}B \end{bmatrix}}_{\text{Controllability}} = \Theta\, \Gamma$$

Low-rank factorization of Hankel matrix

- Approximate Hankel matrix by

$$H_r = \Theta\,\Gamma \approx H \in \mathbb{R}^{n_r \times n_c},$$

where $\Theta \in \mathbb{R}^{n_r \times r}$ and $\Gamma \in \mathbb{R}^{r \times n_c}$ satisfy

$$\Theta^T\Theta = \Gamma\Gamma^T = S = \text{diag}(s_1, \ldots, s_r) \in \mathbb{R}^{r \times r}, \qquad s_i \in \mathbb{R}^+$$

- Think of Θ, Γ as the rank r reduced-order observability and controllability matrices
- Reduced-order observation and control matrices:

$$C_r = [I_p\ 0]\,\Theta, \qquad B_r = \Gamma \begin{bmatrix} I_m \\ 0 \end{bmatrix}$$

- A_r can be obtained from a least squares problem

Kung's SVD-ERA and our CUR-ERA approach

UNIVERSITY OF MICHIGAN

Kung's 1978 Eigensystem Realization Alg. (SVD-ERA) (SVD 1970)

- $H_r = \Theta\,\Gamma$ via SVD, $\Theta = U\Sigma^{1/2}$, and $\Gamma = \Sigma^{1/2}V^T$
- **Computational Bottleneck: SVD** of large Hankel matrix

Our approach: CUR-based Eigensystem Realization Alg. (CUR-ERA)

[Kramer/Gorodetsky, SIAM J. Sci. Comp., '18]

- Develop much faster algorithm due to low-dimensional **CUR** decomposition
- No need to load all data in fast memory
- Show that computed ROMs are stable
- Show that errors scale with first neglected ***singular value***

[Kung, 1978] [Juang and Pappas, 1985] [Ma et al., 2011] [Mahoney and Drineas, 2009] [Oseledets and Tyrtyshnikov, 2010]

Efficiently computing Hankel factorization: $\rightarrow H_r = CUR$

UNIVERSITY OF MICHIGAN

- The **CUR factorization of H is defined as**

$$H_r := H(:, \mathcal{J})H(\mathcal{I}, \mathcal{J})^{\dagger}H(\mathcal{I}, :) \quad = \text{“Columns } \times \; U \; \times \text{ Rows”}$$

- $\mathcal{I} = \{i_1, i_2, \ldots, i_r\}, \; i_k \in \{1, 2, \ldots, n_r\}$: row indices
- $\mathcal{J} = \{j_1, j_2, \ldots, j_r\}, \; j_k \in \{1, 2, \ldots, n_c\}$: column indices
- If H has rank $r < \min(n_r, n_c)$, s.t. $\|H_r^{opt} - H\|_F = 0$, then we can find $\mathcal{I}$ and $\mathcal{J}$ such that the resulting CUR factorization H_r is exact
- CUR factorization can be computed in $\mathcal{O}(r^3 + nr^2)$ (SVD: $\mathcal{O}(n^3)$)

[Goreinov and Tyrtyshnikov, 2001] [Mahoney and Drineas, 2008] [Drineas et al., 2008]

Approximation properties of CUR

- For **approximately low-rank** matrices, find $\mathcal{I}, \mathcal{J}$ such that

$$[\mathcal{I}, \mathcal{J}] = \arg\min_{\hat{\mathcal{I}}, \hat{\mathcal{J}}} \|H - H(:, \hat{\mathcal{J}}) H(\hat{\mathcal{I}}, \hat{\mathcal{J}})^{\dagger} H(\hat{\mathcal{I}}, :)\|$$

- Good strategy: maximum volume submatrix [Goreinov/Tyrtyshnikov, 2001]

$$[\mathcal{I}, \mathcal{J}] = \arg\max_{\hat{\mathcal{I}}, \hat{\mathcal{J}}} |\det(H(\hat{\mathcal{I}}, \hat{\mathcal{J}}))|$$

- With this choice, and σ_{r+1} the $(r+1)$st singular value of H, have:

$$\|H - H_r\|_C = \max_{i,j} |H(i,j) - H_r(i,j)| \leq (r+1)\sigma_{r+1}$$

- Combinatorial optimization problem infeasible to solve
- Cross-approximation of [Oseledets/Tyrtyshnikov, 2010] sequentially increases volume by alternating between rows and columns

[Goreinov and Tyrtyshnikov, 2001] [Goreinov and Tyrtyshnikov, 2011]

Algorithm: CUR-ERA

[Kramer/Gorodetsky, SIAM J. Sci. Comp., '18]

UNIVERSITY OF MICHIGAN

1. Compute $H_r := H(:, \mathcal{J}) H(\mathcal{I}, \mathcal{J})^\dagger H(\mathcal{I}, :)$ via cross-approximation
2. QR decomposition gives basis for the column/row space:

$$Q_{col} R = H(:, \mathcal{J}), \qquad L Q_{row} = H(\mathcal{I}, :)$$

3. Compute SVD of cross-matrix:

$$U_1 \Sigma_1 V_1^T = H(\mathcal{I}, \mathcal{J})$$

4. Form the matrix

$$R \left[H(\mathcal{I}, \mathcal{J}) \right]^\dagger L = R V_1 \Sigma_1^{-1} U_1^T L$$

5. Taking the SVD of this matrix we obtain $(r \times r)$

$$U \Sigma V^T = R V_1 \Sigma_1^{-1} U_1^T L$$

6. The approximation for the Hankel matrix then becomes

$$H \approx H_r = \underbrace{Q_{col} U \Sigma^{1/2}}_{=:\Theta} \underbrace{\Sigma^{1/2} V^T Q_{row}}_{=:\Gamma}$$

Test problem: Mass-Spring-Damper system
Two orders of magnitude reduction in CPU time

UNIVERSITY OF MICHIGAN

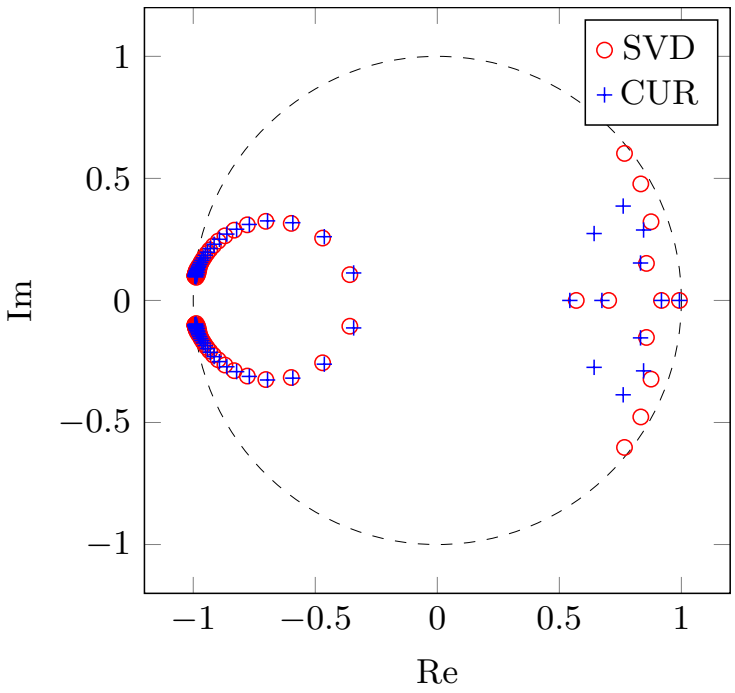
Construct and compare two ROMs:
One with CUR-ERA, another with SVD-ERA

- Inputs/outputs: $m = p = 30$
- Data are $2s = 1000$ Markov parameters, $h_k \in \mathbb{R}^{30\times 30}$
- Hankel matrix is of size $15,000 \times 15,000$
- CUR and SVD of Hankel matrix of order $r = 80$
- CUR initialized randomly, results averaged over 20 test runs

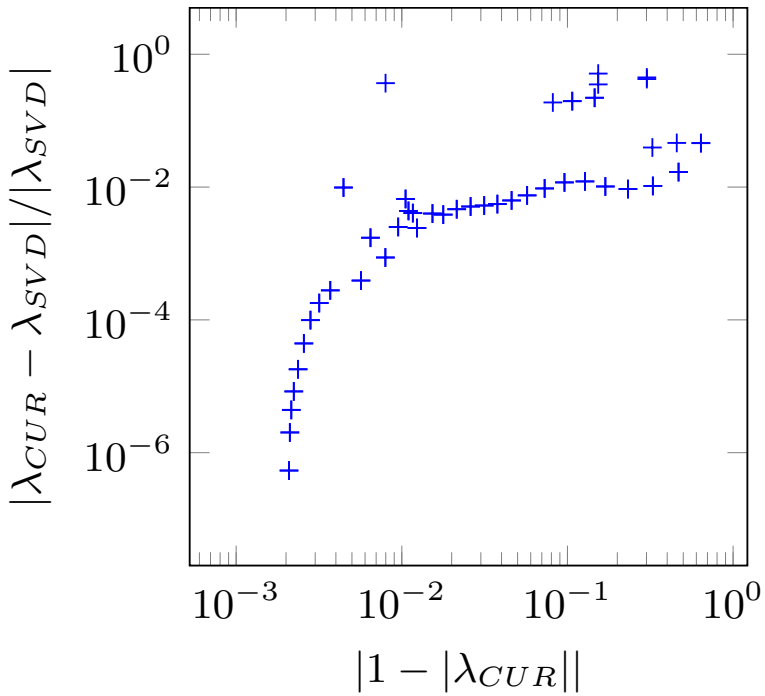
	SVD-ERA	CUR-ERA
CPU time [s]	1.22×10^3	4.77×10^1
$\|H - H_r\|_F / \|H\|_F$	$\mathbf{1.56 \times 10^{-5}}$	$\mathbf{4.35 \times 10^{-5}}$

CPU time on a Intel®Core i7-3770 CPU @ 3.40GHz $\times$ 8, with 32GB of DDR3 RAM and `MATLAB 2015b`.

CUR-ERA approximates eigenvalues of A_r well



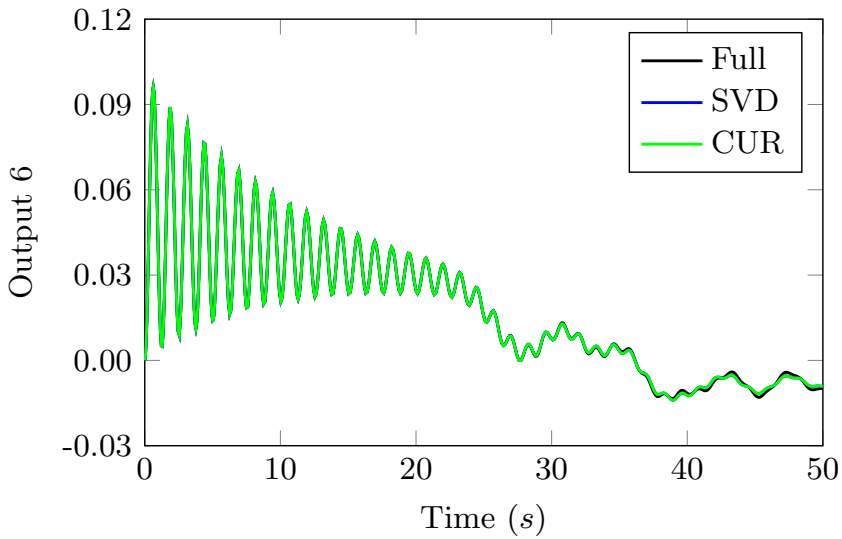
Eigenvalues of the identified system matrix A_r via CUR-ERA and SVD-ERA, respectively.

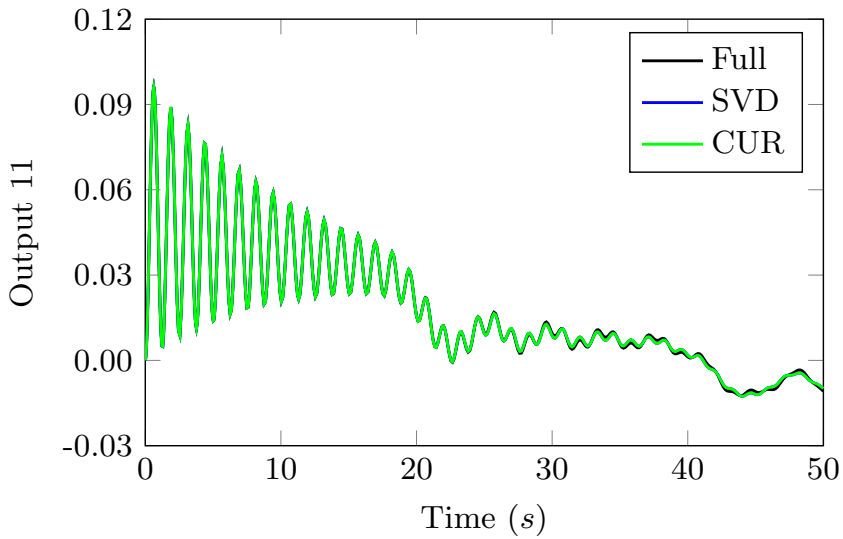


Relative acccuracy of the eigenvalues $|\lambda_{SVD} - \lambda_{CUR}|/|\lambda_{SVD}|$ with respect to distance from the stability boundary $|1 - |\lambda_{CUR}||$.

Outputs of identified ROM match FOM

We compare outputs from original data with ROMs from SVD-ERA and CUR-ERA





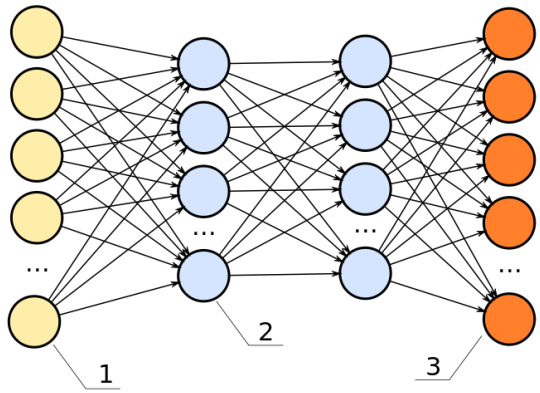
Effectively learning nonlinear representations

$$\frac{dx}{dt} = f(x_1, x_2, \ldots, x_d)$$

Effectively learning nonlinear representations

$$\frac{dx}{dt} = f(x_1, x_2, \ldots, x_d)$$

$$f(x_1, \ldots, x_d) = \sum_{i=1}^{P} a_i \phi_i(x_1, \ldots, x_d)$$



We need a scalable representation that is ...

- Real-valued: scalar or vector
- Expressive: represent complex features using few parameters
- Simple and interpretable: we are making consequential decisions
- **Fast to compute with:** gradients, integrals, marginalization, etc.

Linear parametric representations

- Approximation space: tensor-product basis expansions $\{\psi_i\}_{i=1}^{P}$

$$f(x) = \sum_{i=1}^{\mathsf{P}} \boldsymbol{\theta}_i \psi_i(x) = \sum_{i_1=1}^{P_1} \dots \sum_{i_d=1}^{P_d} \boldsymbol{\theta}_{i_1 \dots i_d} \phi_{1 i_1}(x_1) \dots \phi_{d i_d}(x_d)$$

- Includes wavelets, finite elements, orthonormal polynomials, etc.

Linear parametric representations

- Approximation space: tensor-product basis expansions $\{\psi_i\}_{i=1}^{P}$

$$f(x) = \sum_{i=1}^{\mathsf{P}} \boldsymbol{\theta}_i \psi_i(x) = \sum_{i_1=1}^{P_1} \dots \sum_{i_d=1}^{P_d} \boldsymbol{\theta}_{i_1 \dots i_d} \phi_{1 i_1}(x_1) \dots \phi_{d i_d}(x_d)$$

- Includes wavelets, finite elements, orthonormal polynomials, etc.
- Curse-of-dimensionality: $\mathcal{O}(P^d)$ terms

Linear parametric representations

- Approximation space: tensor-product basis expansions $\{\psi_i\}_{i=1}^{P}$

$$f(x) = \sum_{i=1}^{\mathsf{P}} \boldsymbol{\theta}_i \psi_i(x) = \sum_{i_1=1}^{P_1} \dots \sum_{i_d=1}^{P_d} \boldsymbol{\theta}_{i_1 \dots i_d} \phi_{1 i_1}(x_1) \dots \phi_{d i_d}(x_d)$$

- Includes wavelets, finite elements, orthonormal polynomials, etc.
- Curse-of-dimensionality: $\mathcal{O}(P^d)$ terms
- Expressivity issue: exponentially growing number of unknowns limits order of representation

Optimization/Learning setups

- Ridge regression:
$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_2$$
- $L1$ regression:
$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_1$$
- $L0$ regression:
$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_0$$

Constructing Ψ, storing $\boldsymbol{\theta}$ encounters curse of dimensionality

- Total order expansions: exponential growth of terms
- Low order expansion: limited expressivity

Optimization/Learning setups

- Ridge regression:
$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_2$$
- $L1$ regression:
$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_1$$
- $L0$ regression:
$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_0$$

Constructing Ψ, storing $\boldsymbol{\theta}$ encounters curse of dimensionality

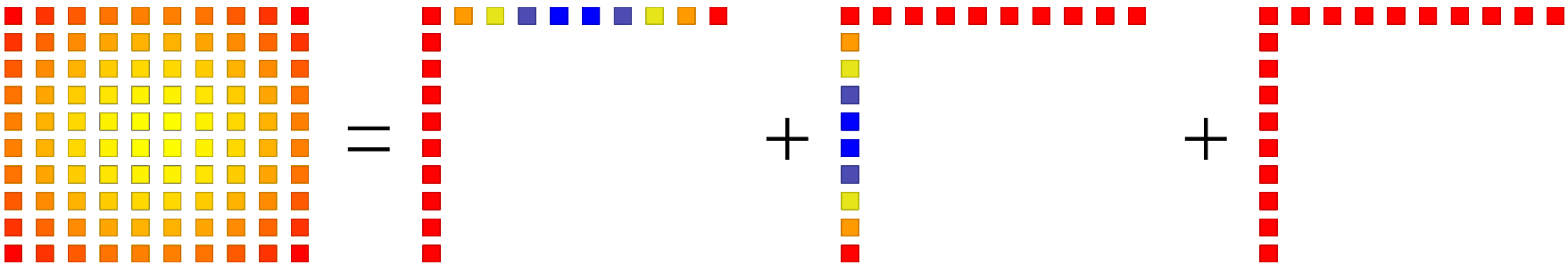
- Total order expansions: exponential growth of terms
- Low order expansion: limited expressivity

Key idea: Assume/Discover reduced-order structure on $\boldsymbol{\theta}$

[Beylkin, 2009] [Doostan, 2013] [Mathelin, 2014] [Rai, 2014]

Simple example of low-rank multilinear structure

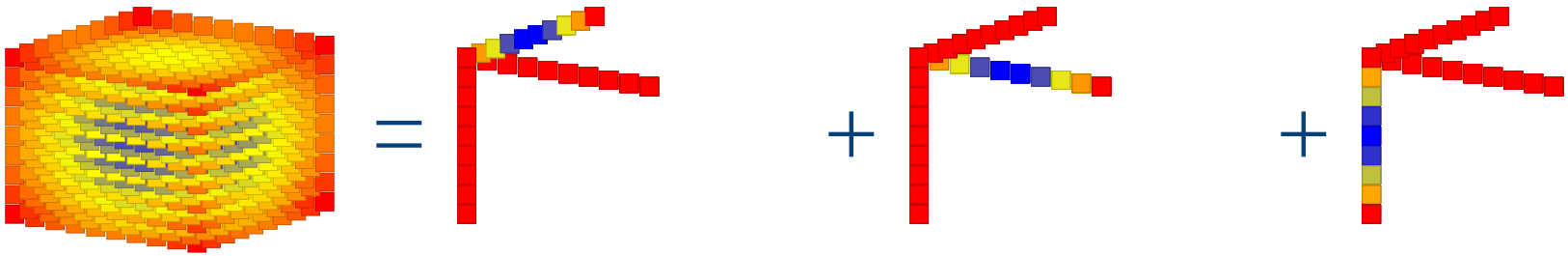
- ***Low-rank structure***, complexity defined by rank r
- Two dimensions: SVD $\mathcal{O}(P^2) \rightarrow \mathcal{O}(2Pr + r)$
- d-dimensions: tensor decompositions $\mathcal{O}(P^d) \rightarrow \mathcal{O}(Pdr)$



$$\mathsf{A} = \boldsymbol{u}_1 \circ \boldsymbol{v}_1 + \boldsymbol{u}_2 \circ \boldsymbol{v}_2 + \boldsymbol{u}_3 \circ \boldsymbol{v}_3$$

Simple example of low-rank multilinear structure

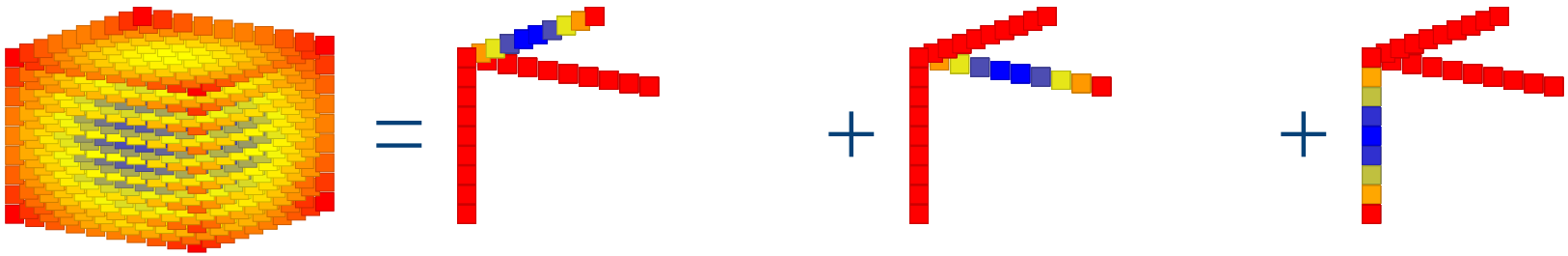
- ***Low-rank structure***, complexity defined by rank r
- Two dimensions: SVD $\mathcal{O}(P^2) \rightarrow \mathcal{O}(2Pr + r)$
- d-dimensions: tensor decompositions $\mathcal{O}(P^d) \rightarrow \mathcal{O}(Pdr)$



$$\mathsf{A} = \boldsymbol{u}_1 \circ \boldsymbol{v}_1 \circ \boldsymbol{w}_1 + \boldsymbol{u}_2 \circ \boldsymbol{v}_2 \circ \boldsymbol{w}_2 + \boldsymbol{u}_3 \circ \boldsymbol{v}_3 \circ \boldsymbol{w}_3$$

Simple example of low-rank multilinear structure

- ***Low-rank structure***, complexity defined by rank r
- Two dimensions: SVD $\mathcal{O}(P^2) \rightarrow \mathcal{O}(2Pr + r)$
- d-dimensions: tensor decompositions $\mathcal{O}(P^d) \rightarrow \mathcal{O}(Pdr)$



$$\mathsf{A} = \boldsymbol{u}_1 \circ \boldsymbol{v}_1 \circ \boldsymbol{w}_1 + \boldsymbol{u}_2 \circ \boldsymbol{v}_2 \circ \boldsymbol{w}_2 + \boldsymbol{u}_3 \circ \boldsymbol{v}_3 \circ \boldsymbol{w}_3$$

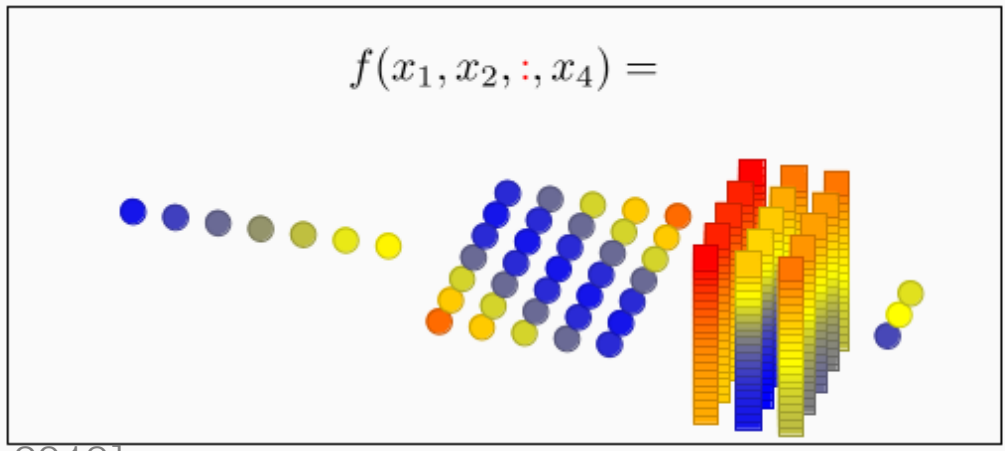
- Many types: CP, TT, HT (tree-networks), tensor-grid, etc.

Functional tensor-train
Tensor-train as a functional ansatz

- Approximate multivariate functions via set of univariate functions
- Strong ties to separability via SVD
- Evaluation through products of matrix-valued functions

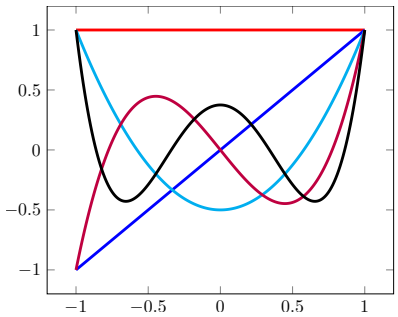
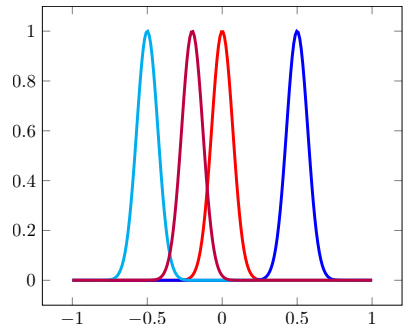
$$f(x_1, x_2, \ldots, x_d) = \sum_{i_0=1}^{r_0} \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} f_1^{(i_0 i_1)}(x_1) f_2^{(i_1 i_2)}(x_2) \ldots f_d^{(i_{d-1} i_d)}(x_d)$$
$$= \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d)$$

$$\underbrace{\begin{bmatrix} f_k^{(11)}(x_k) & \cdot & f_k^{(1 r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1} 1)}(x_k) & \cdot & f_k^{(r_{k-1} r_k)}(x_k) \end{bmatrix}}_{\mathcal{F}_k(x_k)}$$



[Oseledets, 2013] [Gorodetsky/Karaman/Marzouk 2019]

Linear and nonlinear parameterizations

	Linear	Nonlinear
		
Evaluation	$\sum_{\ell=1}^{p_{kij}} \theta_{kij\ell} \phi_{k\ell}^{(ij)}(x_k)$	$\sum_{\ell=1}^{p_{kij}} \exp\left((x_k - \theta_{kij\ell})^2\right)$
Gradient	$\phi_{k\ell}^{(ij)}(x_k)$	$-2\,(x_k - \theta_{kij\ell}) \exp\left((x_k - \theta_{kij\ell})^2\right)$

- Linear: gradient independent of parameter
  $\rightarrow$ faster optimization, lower expressivity
- Nonlinear: gradient dependent on parameters
  $\rightarrow$ slower optimization, higher expressivity

Example: 2D equivalent to CUR

1. Fix "rows" $[x_1\ x_2]$ and "columns" $[y_1\ y_2]$

$$f(x,y) = \begin{bmatrix} f(x,y_1) & f(x,y_2) \end{bmatrix} \mathbf{G} \begin{bmatrix} f(x_1,y) \\ f(x_2,y) \end{bmatrix}$$

2. Approximate ***univariate*** row and column – e.g., in some basis

$$f(x,y) = \begin{bmatrix} \sum_{i=1}^{P} a_i\phi_i(x) & \sum_{i=1}^{P} b_i\phi_i(x) \end{bmatrix} \mathbf{G} \begin{bmatrix} \sum_{i=1}^{P} c_i\psi_i(y) \\ \sum_{i=1}^{P} d_i\psi_i(y) \end{bmatrix}$$

3. Store parameters of ***each*** row and column and $\mathbf{G}$ matrix $\mathcal{O}(Pr + r^2)$

Example: Additive function

- Additive functions are rank 2

$$f(x_1, \ldots, x_d) = f_1(x_1) + \cdots + f_d(x_d)$$

- FT format:

$$f(x_1, x_2, \ldots, x_d) = [f_1(x_1)\ 1] \begin{bmatrix} 1 & 0 \\ f_2(x_2) & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 \\ f_d(x_d) \end{bmatrix}$$

Constrained least squares

$$f^* = \arg\min_{f \in \mathscr{F}_{\boldsymbol{r}}} \quad \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - f\left(x^{(i)} \right) \right)^2$$

$$\mathscr{F}_{\boldsymbol{r}} = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \begin{array}{l} f = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d) \\ \mathcal{F}_1 : \mathcal{X}_1 \to \mathbb{R}^{1 \times r_1} \\ \mathcal{F}_k : \mathcal{X}_k \to \mathbb{R}^{r_{k-1} \times r_k},\ 2 \le k < d \\ \mathcal{F}_d : \mathcal{X}_d \to \mathbb{R}^{r_{d-1} \times 1} \end{array} \right\}$$

- Gradient based procedure for optimization (SGD, BFGS, etc.)
  - Computational complexity is $\mathcal{O}(dr^2)$ [Gorodetsky/Jakeman, JCP., '18]
- Number of unknowns $\mathcal{O}(dr^2P) \to$ amount of data follows

Results

- Mean squared error on validation data for a subset of regression algorithms and data sets from [Kandasamu 2016]
  - 22 algorithms: both parametric and non parametric
  - 10 datasets from UCI database
- Only the algorithms which scored the best in at least one data set and the CV-based LASSO are shown
- **The FT proved second best overall**

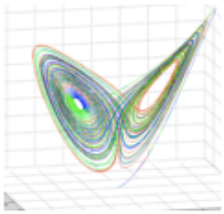
Dataset (d, n)	SALSA	nSVR	RT	GBRT	MARS	FT	CVLASSO
Housing (12,256)	**0.26241**	0.38600	1.06015	0.42951	0.42379	0.32798	0.35218
Galaxy (20,2000)	**0.00014**	0.15798	0.02293	0.01405	0.00163	0.00056	0.00249
Skillcraft (18,1700)	0.54695	0.66311	1.08047	0.57273	0.54595	**0.54434**	0.56879
CCPP (59,2000)	0.06782	0.09449	1.04527	**0.06181**	0.08189	0.06631	0.06466
Speech (21,520)	0.02246	0.06994	0.05430	0.03515	**0.01647**	0.02684	0.03131
Music (90,1000)	0.62512	**0.59399**	1.45983	0.66652	0.88779	0.72094	0.64485
Telemonit (19,1000)	0.03473	0.05246	**0.01375**	0.04371	0.02400	0.04040	0.06487
Propulsion (15,200)	0.00881	0.00910	0.02341	0.00061	0.01290	**0.00009**	0.02660
Airfoil (40,750)	0.51756	0.55118	0.45249	**0.34461**	0.54552	0.46920	0.46444
Forestfires (10,211)	0.35301	0.43142	0.41531	**0.26162**	0.33891	0.39465	0.44175

System ID: Chaotic Systems
Simultaneous learning of all outputs reduces data requirements

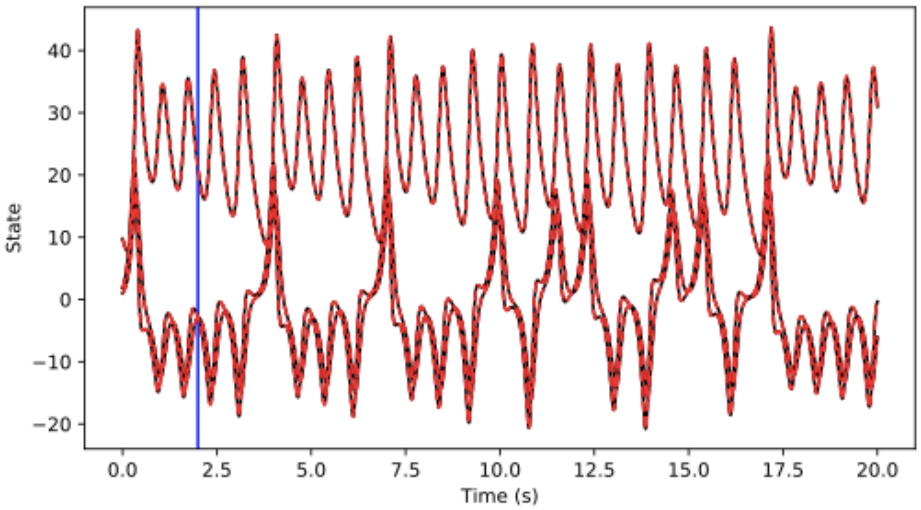
$$\dot{x}_1 = \sigma(x_2 - x_1)$$
$$\dot{x}_2 = x_1(\rho - x_3) - x_2$$
$$\dot{x}_3 = x_1 x_2 - \beta x_3$$

- Train to blue line
- Test after the blue line

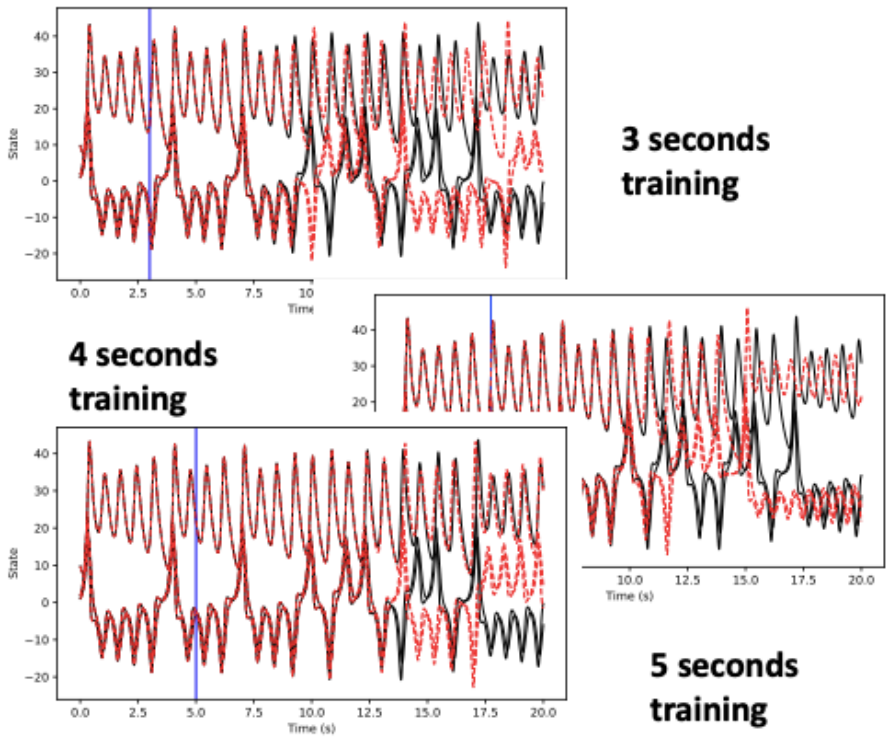
Lorenz 63



Simultaneous Learning (2 seconds)



Independent Learning

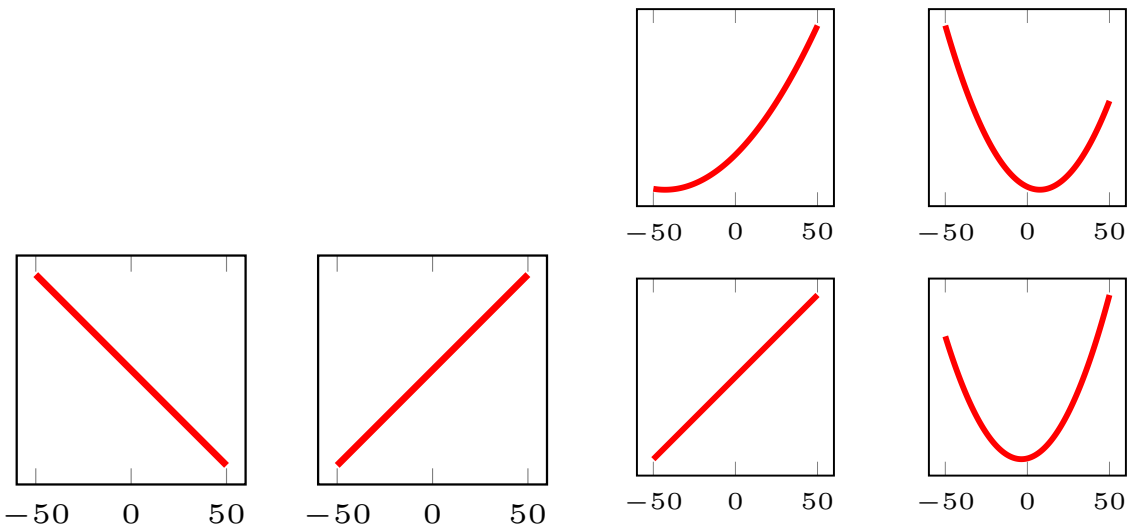


3 seconds training

4 seconds training

5 seconds training

Separation of variables enables interpretability



(a) $\mathcal{F}_1(x_1)$ (b) $\mathcal{F}_2(x_2)$ (c) $\mathcal{F}_3(x_3)$

FT cores for the first equation of the Lorenz 63. In the FT format one can essentially read off the consituent univariate functions and directly determine how they interact with the other dimensions.

Conclusions

- Takeaways:
  - Assuming a structure exists can provide opportunities for scalability
  - Low-rank and low-multilinear-rank structure can be exploited for scalable ROM and function representations
- Code:
  - Available at: `github.com/goroda/Compressed-Continuous-Computation`
  - Includes:
    - General purpose utilities for computing with low-rank functions
    - Optimal sampling and cross approximation for converting functions into FTs
- Papers:
  - B. Kramer and A.A. Gorodetsky ***System identification via CUR-factored Hankel approximation***, SISC 2018
  - A.A. Gorodetsky and J.D. Jakeman. ***Gradient-based Optimization for Regression in the Functional Tensor-Train Format***, JCP 2018
  - A.A. Gorodetsky, S. Karaman, Y.M. Marzouk ***Continuous analogue of the tensor-train decomposition***, CMAME 2019

Acknowledgements

1. DOE Office of Science ASCR (John Von Neumann Fellowship)
2. DARPA PAI
3. AFOSR

System matrix A_r from shift-invariance property

$$H = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix}}_{\text{Observability}} \underbrace{[B\ AB\ \dots\ A^{s-1}B]}_{\text{Controllability}} \approx \Theta\,\Gamma$$

- **Shift-invariance property** of the observability matrix is

$$\underbrace{\begin{bmatrix} C \\ \vdots \\ CA^{s-2} \end{bmatrix}}_{\Theta^f} A = \underbrace{\begin{bmatrix} CA \\ \vdots \\ CA^{s-1} \end{bmatrix}}_{\Theta^l}$$

- Solve for A_r via

$$\Theta^f A_r = \Theta^l$$