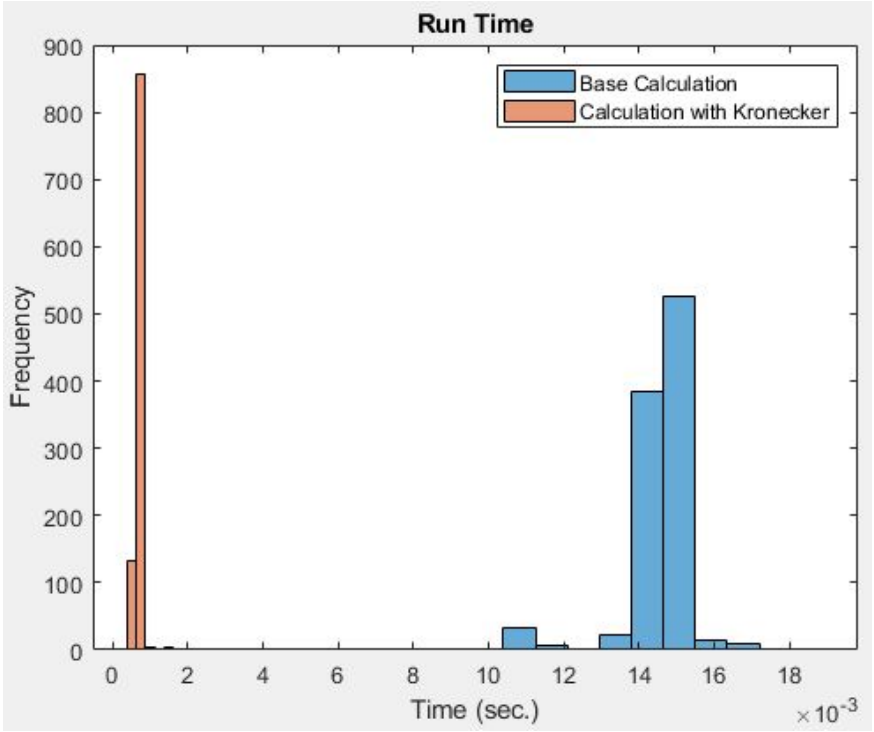
Kronecker Tensor-Vector Multiplication

Experiment

1. B,C = random tensors
2. x,y = random vectors
3. A = B kron C
4. z = x kron y
5. Calculate A times z in *p*-th mode
   a. from B and C (kronecker calculation)
   b. from A (base calculation)
6. record the time and error in the calculations

Results

- The run time distribution is shown to the right for 1000 iterations
- The error over 1000 iterations is on the order of 1e-13 (next slide)
- B,C are 20 x 20 x 20



Kronecker Tensor-Vector Multiplication (Cont.)

Experiment

1. B,C = random tensors
2. x,y = random vectors
3. A = B kron C
4. z = x kron y
5. Calculate A times z in *p*-th mode
   a. from B and C (kronecker calculation)
   b. from A (base calculation)
6. record the time and error in the calculations

Results

- The run time distribution is shown to the right for 1000 iterations
- The largest error over 1000 iterations is on the order of 1e-16
- B,C are 2x2x2

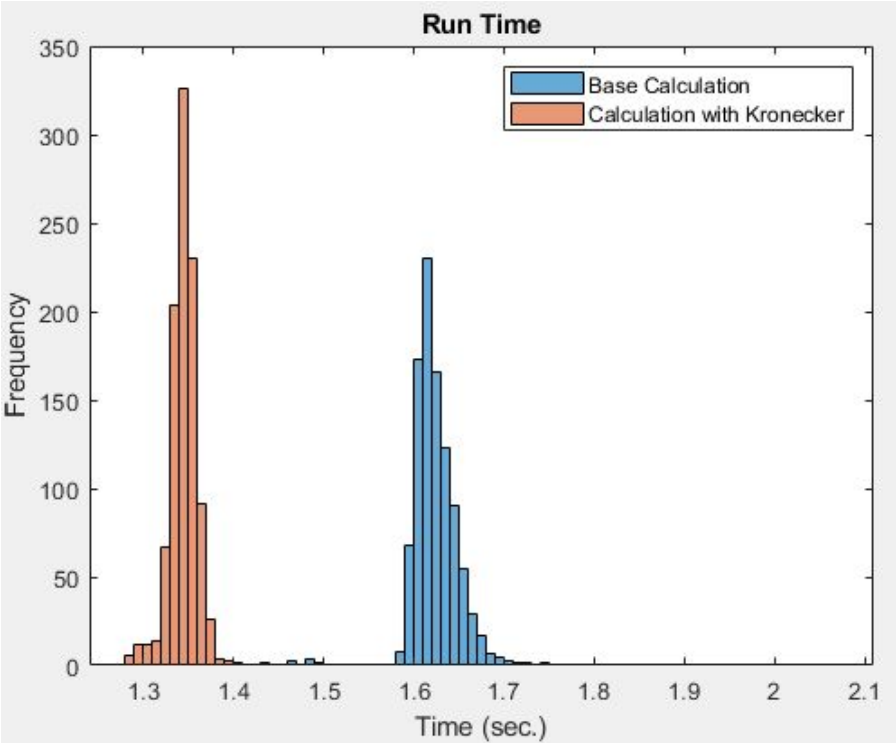
Kronecker Tensor H-Eigenvalue Calculations

Experiment

1. B,C = random tensors
2. A = B kron C
3. Calculate largest eigenvalue of A
   a. from B and C (kronecker calculation)
   b. from A (base calculation)
4. record the time and error in the calculations

Results

- The run time distribution is shown to the right for 1000 iterations
- The largest error over 1000 iterations is on the order of 1e-16
- B,C are 2x2x2



Kronecker SVD: KSVD

The Complete KP-SVD

Given:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} \qquad A_{ij} \in \mathbb{R}^{p \times q}$$

Form $\tilde{A}$ (MN-by-pq) and apply LAPACK SVD:

$$\tilde{A} = \sum_{i=1}^{r_A} \sigma_i u_i v_i^T$$

Then:

$$A = \sum_{i=1}^{r_A} \sigma_i \cdot \texttt{reshape}(u_i, M, N) \otimes \texttt{reshape}(v_i, p, q)$$

Charles Van Loan, *The Kronecker Product SVD* (2009)

Problem: Computing Kronecker factorization of tensors is expensive and requires CPD/HOSVD decomposition which is infeasible for large tensors

Goal: Develop a fast, approximate Kronecker factorization that can be applied to tensors with a limited compute resources

Strategy: Rather than computing the SVD matrix decompositions, we bin the data to generate one factor, similar to viewing 3C at different resolutions, and solve a convex optimization to generate the second factor

Proposed Algorithm

Problem: Given A, quickly find B and C as a reasonable approximation: $A \approx B \otimes C$

Algorithm 1 New Algorithm

Input: $\mathbf{A}$ is $n \times n$, integers m and l where $n = m \times l$
$\mathbf{F}^{(1)} := \text{ZEROS}(m, m)$
$\mathbf{F}^{(2)} := \text{ZEROS}(l, l)$
for block indices $i, j \in 1, \ldots, m$ **do**
    $\mathbf{F}^{(1)}_{i,j} = \text{MEAN}([\mathbf{A}]_{i,j})$
    $\mathbf{F}^{(2)} = \mathbf{F}^{(2)} + [\mathbf{A}]_{i,j}$
end for
$\mathbf{F}^{(2)} = \mathbf{F}^{(2)}/m^2$
Return: $\mathbf{F}^{(1)}, \mathbf{F}^{(2)}$

Algorithm Intuition:

- Part 1: construct matrix B by binning matrix blocks of A (low resolution i.e. binning the data)
- Part 2: with A given and B fixed, C averages elements across each matrix block (solution to convex optimization problem)
- Parts 1 & 2 can be accomplished within a single loop

Algorithm Complexity

Algorithm 1 New Algorithm

1: **Input:** $\mathbf{A}$ is $n \times n$, integers m and l where $n = m \times l$
2: $\mathbf{F}^{(1)} := \text{ZEROS}(m, m)$
3: $\mathbf{F}^{(2)} := \text{ZEROS}(l, l)$
4: **for** block indices $i, j \in 1, \ldots, m$ **do**
5: $\quad \mathbf{F}^{(1)}_{i,j} = \text{MEAN}([\mathbf{A}]_{i,j})$
6: $\quad \mathbf{F}^{(2)} = \mathbf{F}^{(2)} + [\mathbf{A}]_{i,j}$
7: **end for**
8: $\mathbf{F}^{(2)} = \mathbf{F}^{(2)}/m^2$
9: **Return:** $\mathbf{F}^{(1)}, \mathbf{F}^{(2)}$

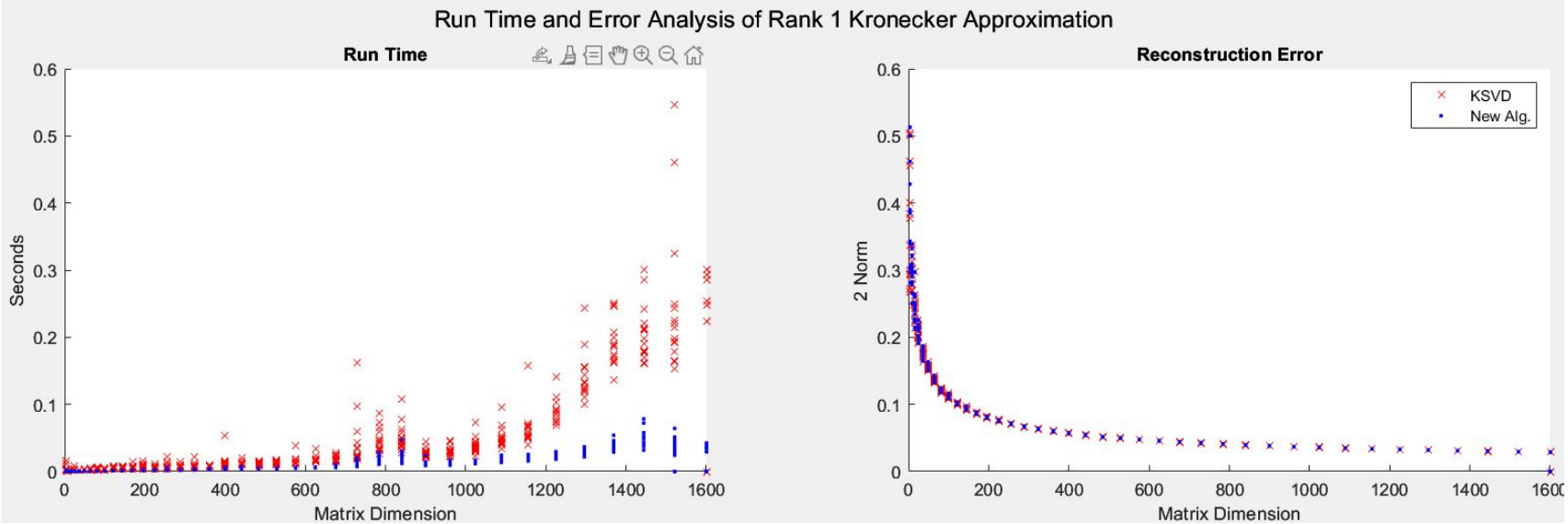
Loop complexity

- Line 4 O(n^2/m^2)
  - Line 5 O(m^2)
  - Line 6 O(m^2)

Total complexity is O(n^2), and for a k-way tensor the total complexity is O(n^k)

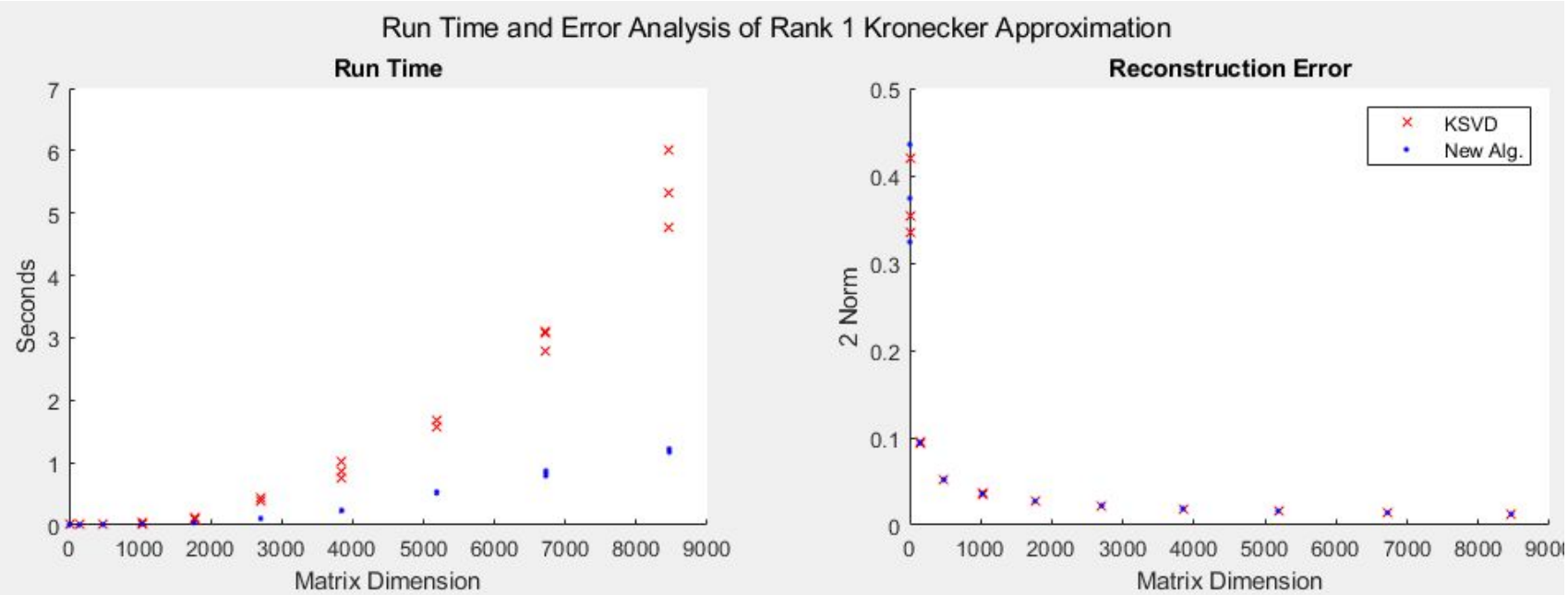
(K)SVD of a symmetric matrix is O(n^3)

Run Time & Error Analysis of New Alg. v. KSVD

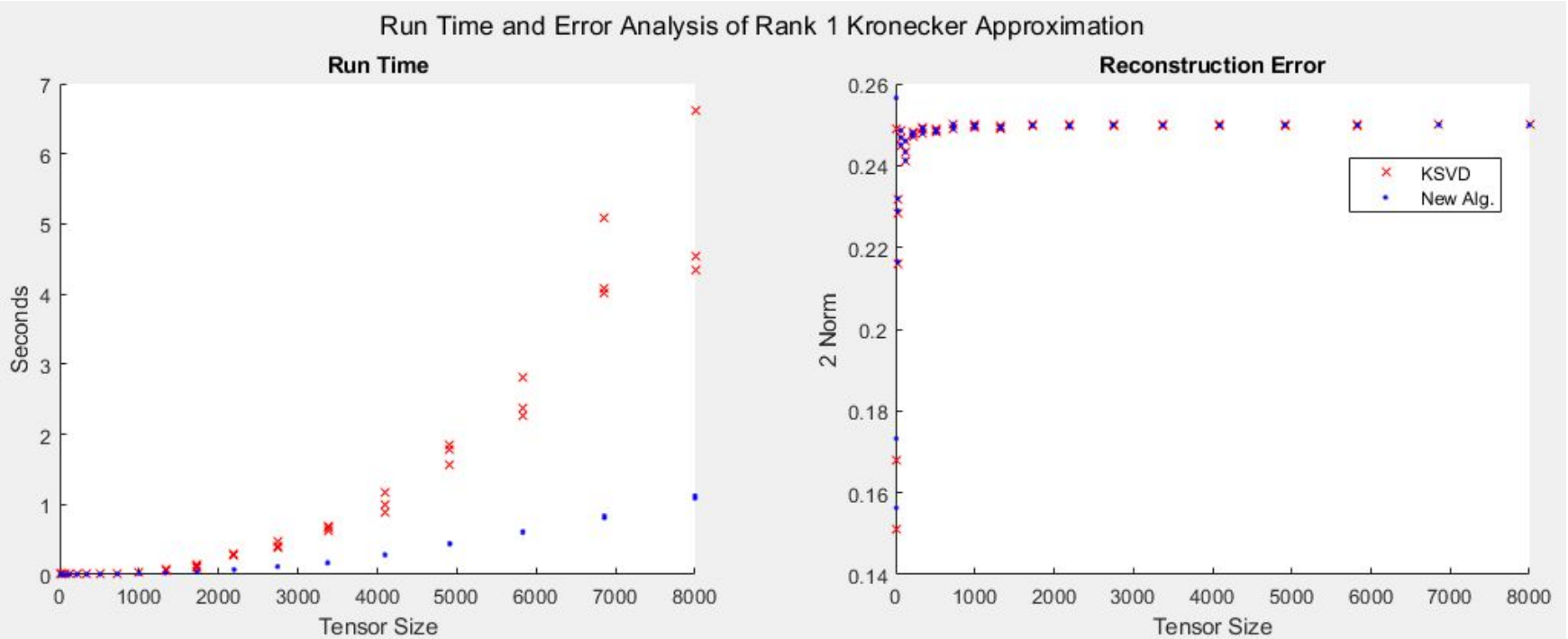


- 2 Norm = ||A - E||/||A|| where A is the data and E is the rank 1 estimate of A
- The above experiment was performed on random matrices of different sizes and 15 iterations at each size
- New algorithm is faster than traditional KSVD
- New algorithm has similar performance on the rank 1 estimation of a matrix

Run Time & Error Analysis (Cont.)

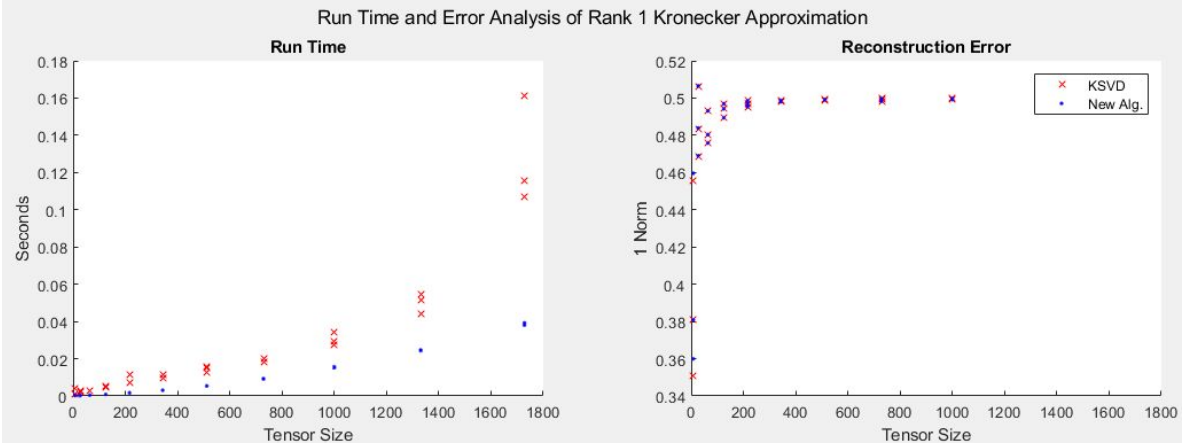


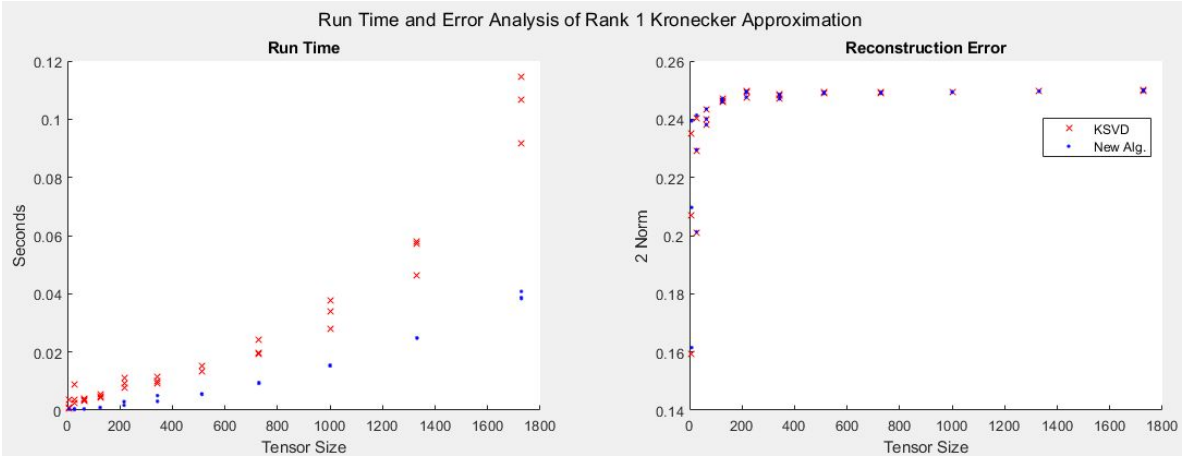
Run Time & Error Analysis Tensor Approximation



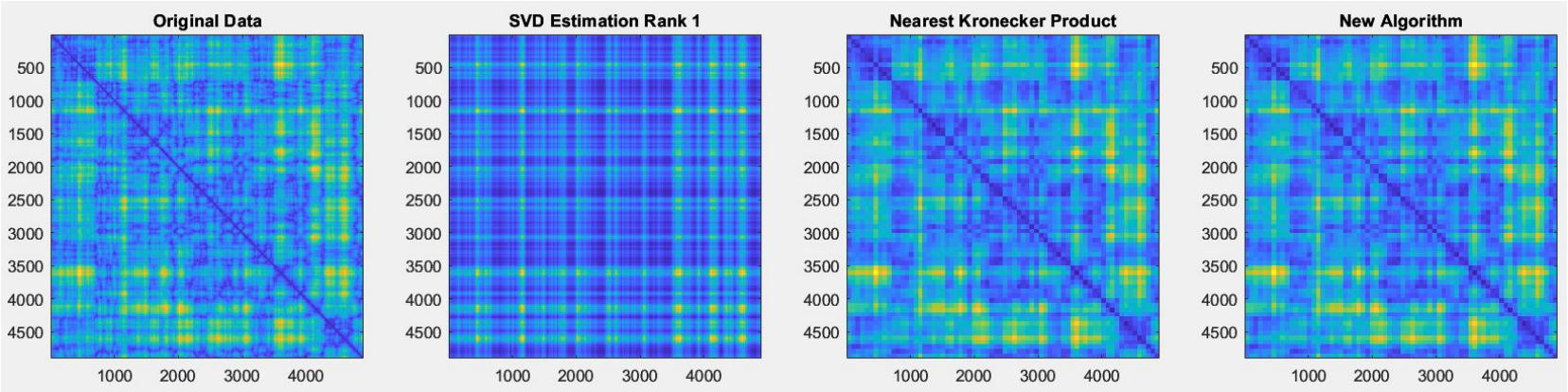
approximated.

Run Time & Error Analysis (Cont.)





Examples (Rank 1)



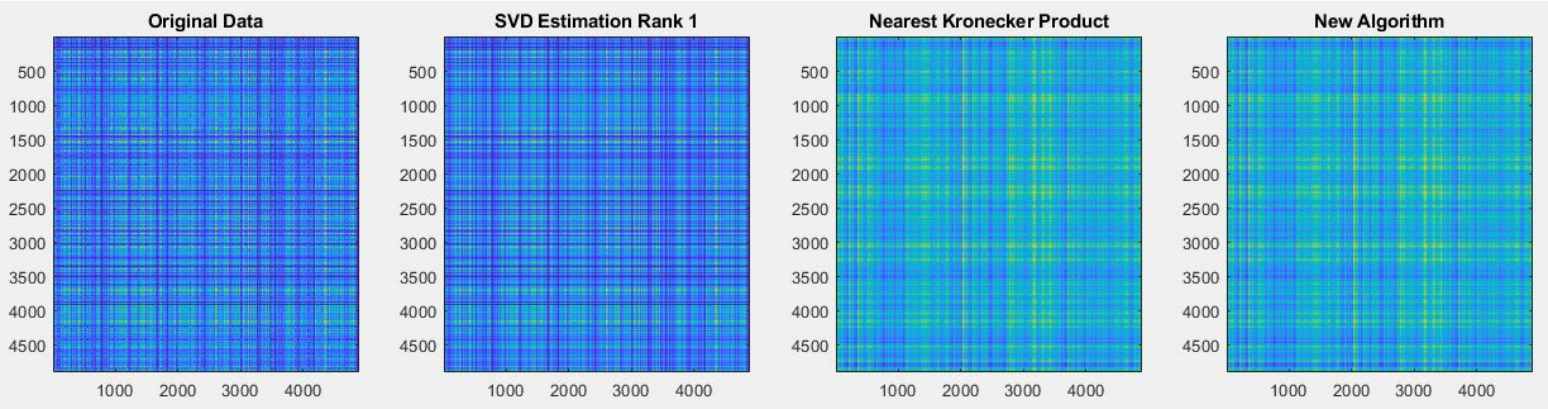
- All 3 estimations have an equal compression ratio or number of parameters used in the estimation
- SVD Rank 1 estimation preserves information in rows and columns (strips)
- Kronecker Rank 1 estimations preserve information in matrix blocks (pixels)
- If we can only use a rank 1 compression, we should use a Kronecker method

* Data has norm 2.113e-4

	|| Data - Estimate ||	Compute Time (sec)
SVD	4.652e-5	0.816
NKP	2.174e-5	1.033
New Alg.	2.1761e-5	0.267

Failure Modes (Rank 1)

This algorithm will work only if there is an ordered structure to the rows/columns of the data



The data is a rank 3 matrix where rank is the classical definition of linear independence between rows and columns (as opposed to Kronecker rank, which follows from the KSVD)

* the times on the slide are not comparable to the previous slide, different computers were used. Data had a norm 2.327e-4

	|| Data - Estimate ||	Compute Time (sec)
SVD	2.903e-5	0.313
NKP	9.234e-5	0.592
New Alg.	9.235e-5	0.201

When to use SVD vs Nearest Kronecker (NKP)?

The optimal SVD and Kronecker SVD approximations are not equivalent, and there errors are not equivalent.

Theorem 2.4.8 (The Eckhart-Young Theorem). If $k < r = \text{rank}(A)$ and

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

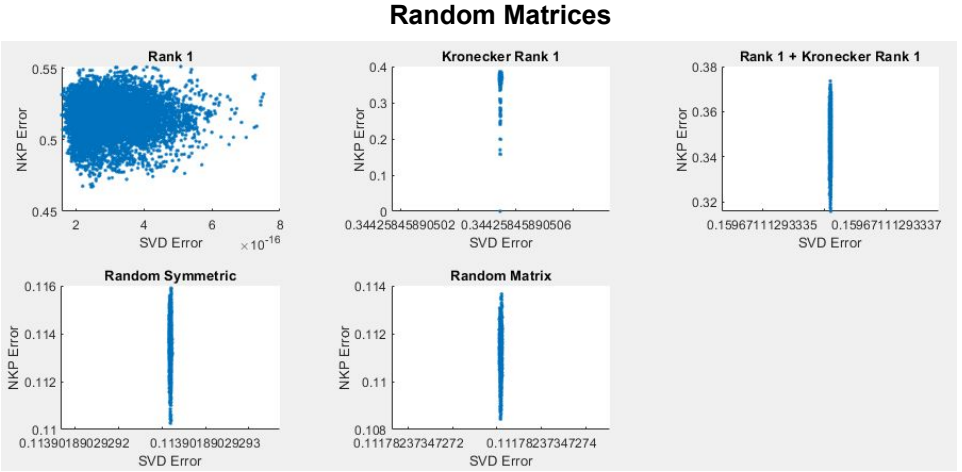
then

$$\min_{\text{rank}(B)=k} ||A - B||_2 = ||A - A_k||_2 = \sigma_{k+1}$$

Theorem 2.1 *Assume that* $A \in \mathbb{R}^{m \times n}$ *with* $m = m_1 m_2$ *and* $n = n_1 n_2$. *If* $B \in \mathbb{R}^{m_1 \times n_1}$ *and* $C \in \mathbb{R}^{m_2 \times n_2}$, *then*

$$\| A - B \otimes C \|_F = \| \mathcal{R}(A) - vec(B) vec(C)^T \|_F.$$

R(A) is a rearranged version of the original A matrix



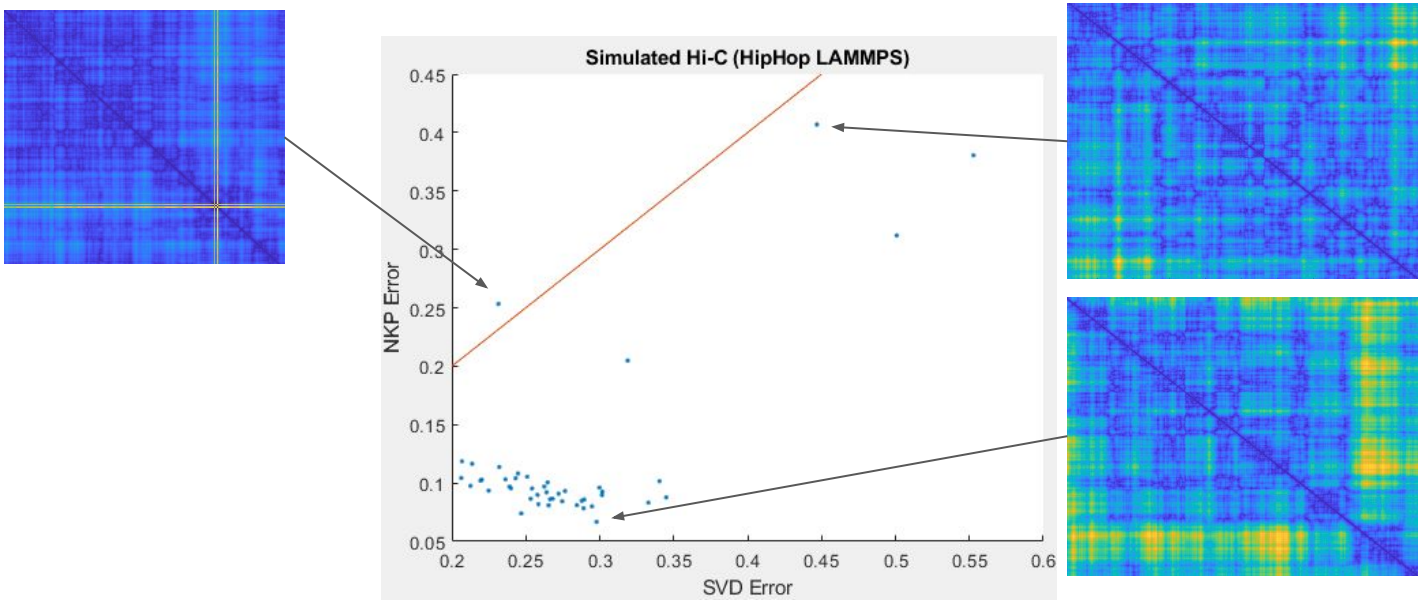
Experiment Procedure:

1. Generate “random” matrix **A**
2. Compute rank 1 KSVD and SVD approximation errors
3. Randomly swap 2 rows or columns of **A**
4. Repeat steps 2 - 3
5. Visualize the errors

Eckhart-Young Theorem from course notes. Theorem 2.1 from Approximation with Kronecker Products by Charles Van Loan and Nikos Pitsianis

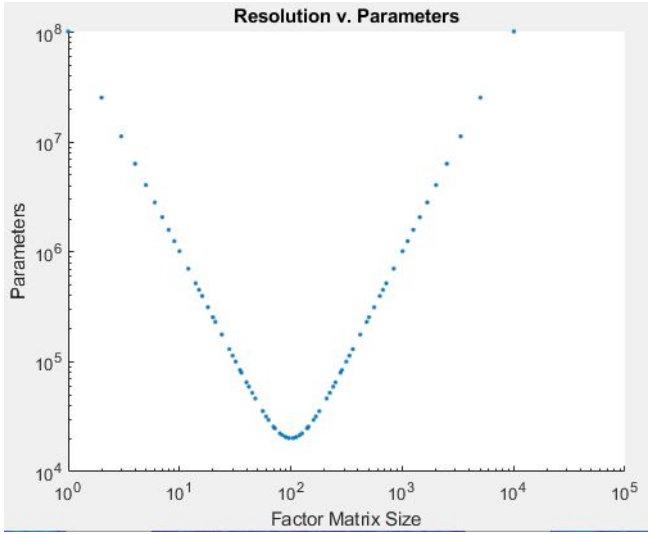
SVD vs NKP (Cont.)

Kronecker Nearest Product outperforms SVD for real data matrices



We compare the error of rank 1 SVD and NKP approximation on several matrices derived from polymer simulations. The *i,j*-th entry of each matrix represents the distance between 2 beads of the polymer in the simulation. These matrices and their simulations have been used in prior work to generate pseudo Hi-C or pseudo Pore-C. For this data, NKP outperforms the SVD for all but one instance of matrix approximation

Data Resolution vs. Num. Parameters (Compression Size)



- Optimal compression is achieved when the factor matrices have dimensions that are the square root of the dimension of the data
- KoPA paper proposes a method of selecting the number of parameters according to an information theoretic criteria, but they also take into account specific amounts or models of noise present in their data

Backing Out the SVD from KSVD

The optimal SVD and Kronecker SVD approximations are not equivalent, and there errors are not equivalent.

Theorem 2.4.8 (The Eckhart-Young Theorem). If $k < r = \text{rank}(A)$ and

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

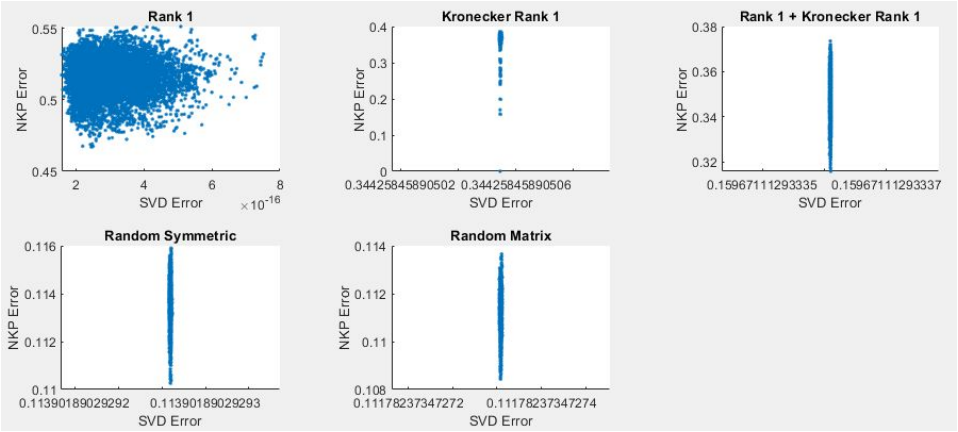
then

$$\min_{\text{rank}(B)=k} ||A - B||_2 = ||A - A_k||_2 = \sigma_{k+1}$$

Theorem 2.1 *Assume that* $A \in \mathbb{R}^{m \times n}$ *with* $m = m_1 m_2$ *and* $n = n_1 n_2$. *If* $B \in \mathbb{R}^{m_1 \times n_1}$ *and* $C \in \mathbb{R}^{m_2 \times n_2}$, *then*

$$\| A - B \otimes C \|_F = \| \mathcal{R}(A) - vec(B) vec(C)^T \|_F.$$

R(A) is a rearranged version of the original A matrix



Experiment Procedure:

1. Generate “random” matrix **A**
2. Compute rank 1 KSVD and SVD approximation errors
3. Randomly swap 2 rows or columns of **A**
4. Repeat steps 2 - 3

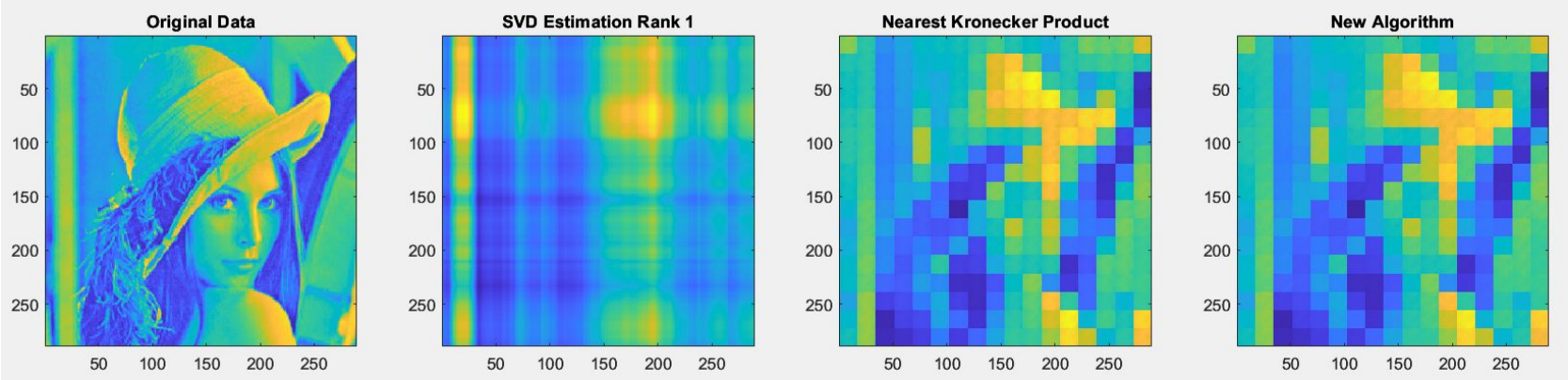
Eckhart-Young Theorem from course notes. Theorem 2.1 from Approximation with Kronecker Products by Charles Van Loan and Nikos Pitsianis

Next Steps

Here are a few items Joshua thinks he should work on to pursue using this algorithm:

- **higher rank approximations** with this algorithm (only rank 1 examples are shown in previous slides. I have)
- **Tensor version** of algorithm (should
- Perform experiment to clock how the algorithm performs on **large matrices** (ex. 10k x 10k)
-

Examples (Rank 1)



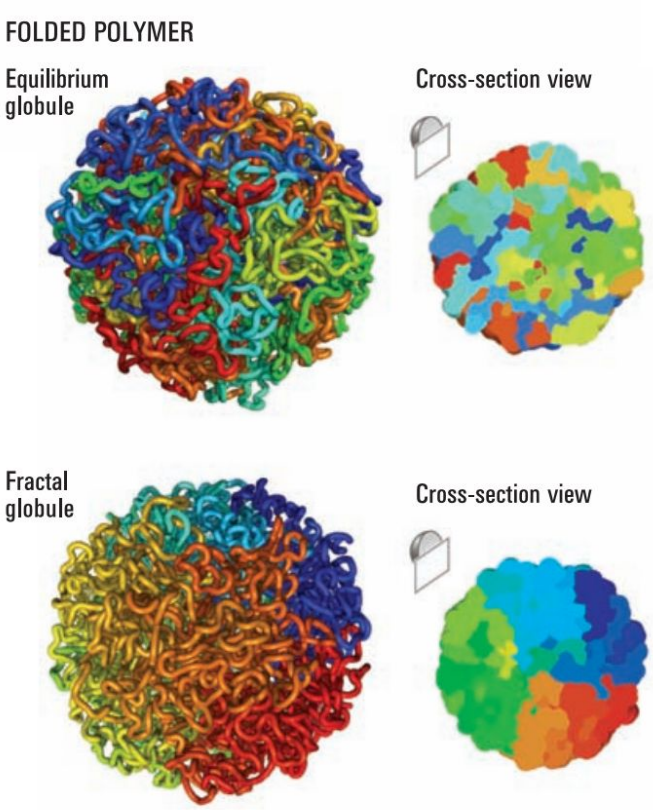
Model: Kronecker Representation of Chromatin

Chromatin Organization

- Chromatin is highly compartmentalized at multiple scales (A/B compartments, TADs, looping)
- Within compartment contact distributions are similar across compartments

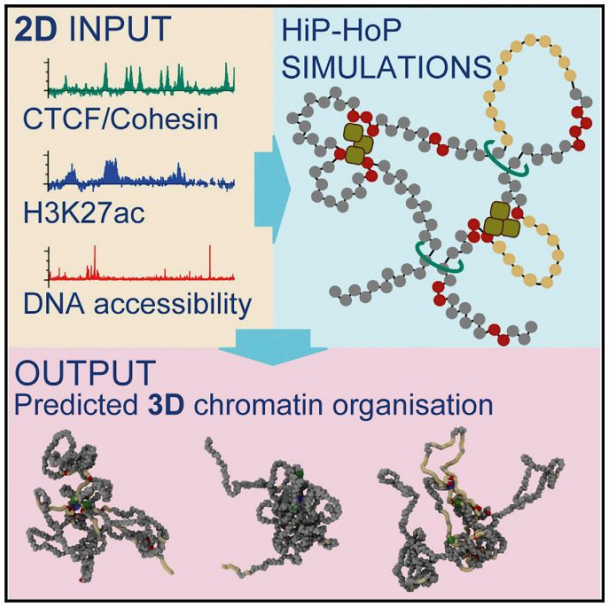
Kronecker Representation

- Chromosomal loci are characterized by their compartment (hypergraph 1) and their location in the compartment (hypergrah 2)
- 2 assumptions to idealize the system
    1. All compartments have identical internal structure
    2. The likelihood of a loci forming a contact in hypergraph 1 (2) is independent of its position in hypergraph 2 (1)



Lieberman-Aiden, Erez, et al. "Comprehensive mapping of long-range interactions reveals folding principles of the human genome." *science* 326.5950 (2009): 289-293.

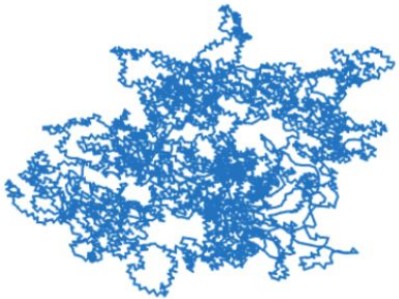
Data: Simulated Chromatin Structures



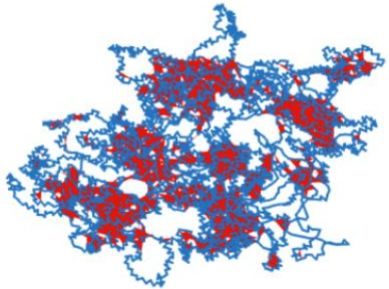
Data

- polymer structures from LAMMPS
- 3 sets of simulation parameters
- Parameters correspond to expression of the Pax6 gene (OFF, ON, HIGH)
- 600 individual simulations

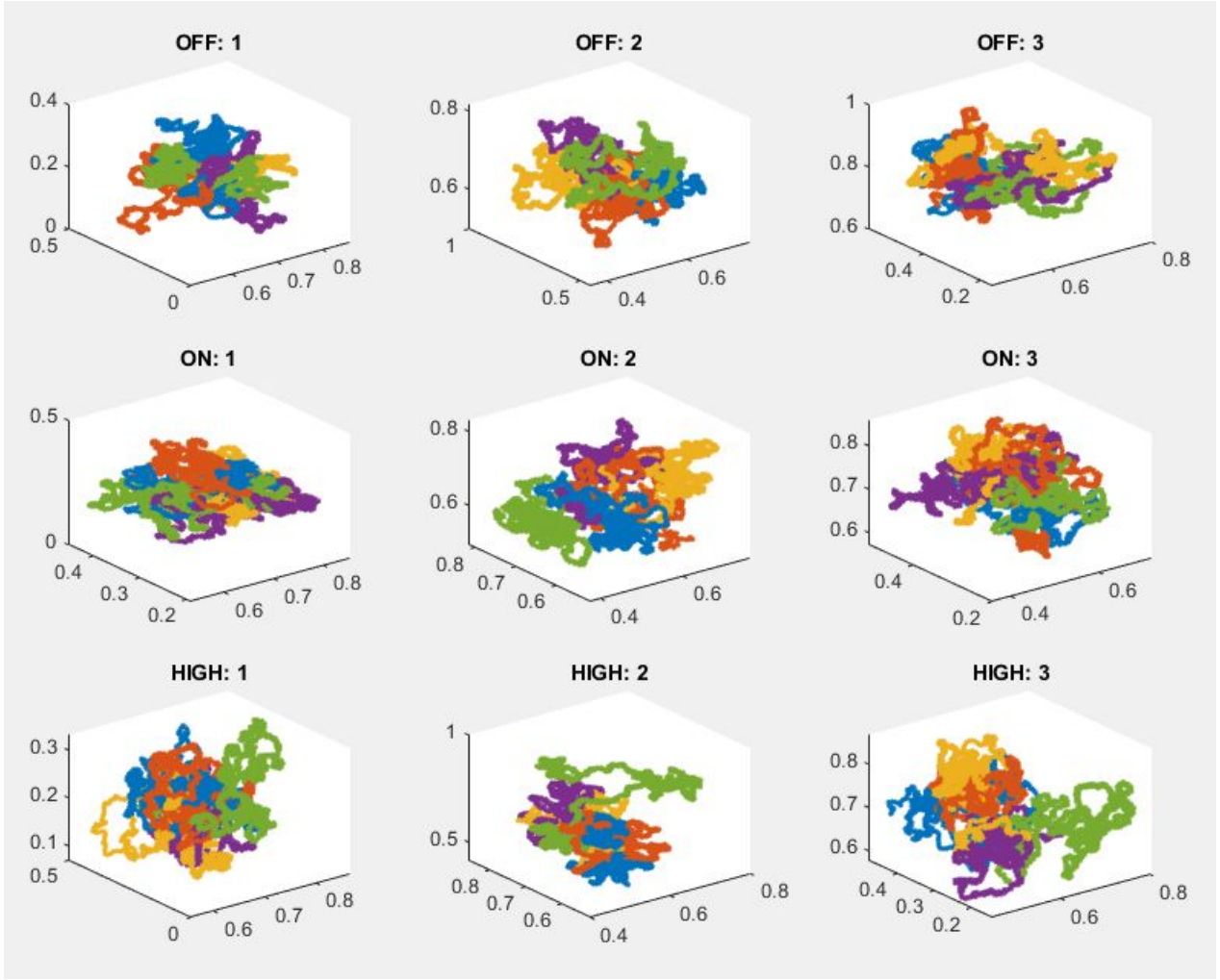
Pax6 Polymer Structure

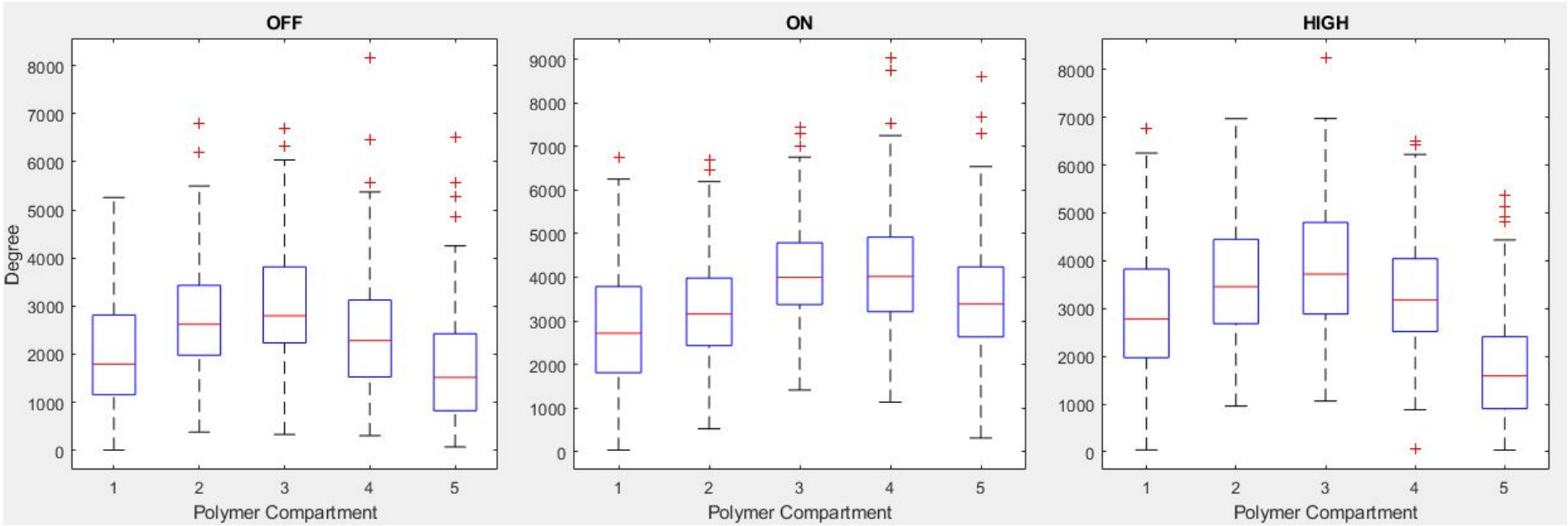


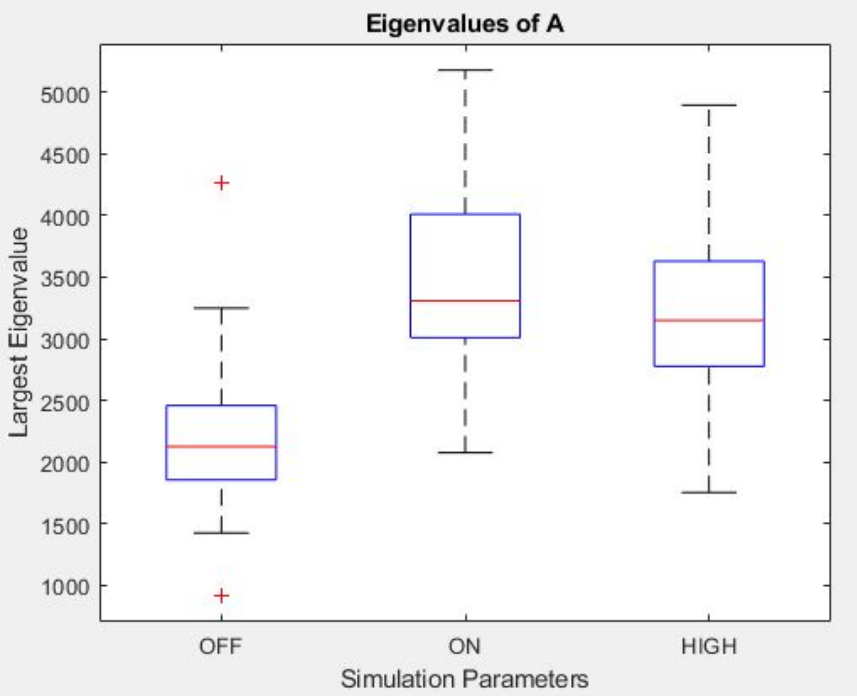
Highlighted Multi-way Contacts

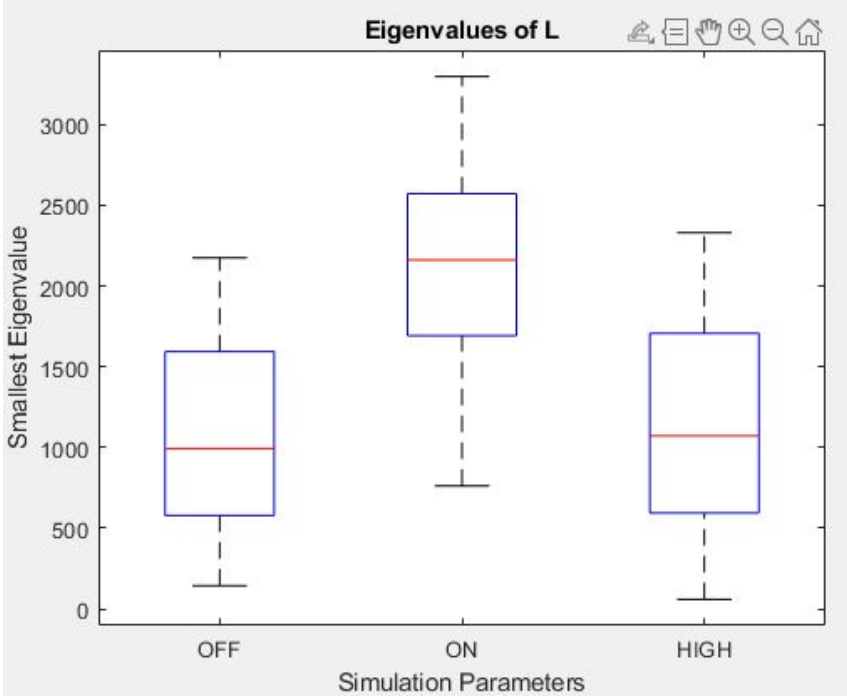


Buckle, Adam, et al. "Polymer simulations of heteromorphic chromatin predict the 3D folding of complex genomic loci." *Molecular cell* 72.4 (2018): 786-797.
Plimpton, Steve. "Fast parallel algorithms for short-range molecular dynamics." *Journal of computational physics* 117.1 (1995): 1-19.


OFF: 1
OFF: 2
OFF: 3
ON: 1
ON: 2
ON: 3
HIGH: 1
HIGH: 2
HIGH: 3


OFF
ON
HIGH
Degree
Polymer Compartment
Polymer Compartment
Polymer Compartment
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
1
2
3
4
5


Eigenvalues of A
Largest Eigenvalue
5000
4500
4000
3500
3000
2500
2000
1500
1000
OFF
ON
HIGH
Simulation Parameters


Eigenvalues of L
Smallest Eigenvalue
3000
2500
2000
1500
1000
500
0
OFF
ON
HIGH
Simulation Parameters

Over all 200 instance how often was

Edge	OFF	ON	HIGH
1,2	102	56	123
1,3	63	47	77
1,4	22	27	24
1,5	5	21	5
2,3	140	81	121
2,4	50	35	59
2,5	17	17	7
3,4	103	118	128
3,5	48	63	22
4,5	50	135	34

Hyperedge	OFF	ON	HIGH
1,2,3	136	93	123
1,2,4	91	46	72
1,2,5	32	29	22
1,3,4	48	53	79
1,3,5	22	40	23
1,4,5	16	39	25
2,3,4	100	92	117
2,3,5	49	54	37
2,4,5	43	45	29
3,4,5	63	109	73

Experimental Validation of Chromosome Simulations