



A Scalable Platform for Rapid Data Analytics Through Data Driven Lossy Compression.

Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, November 7th, 2024

Ben O'Neill (RNET)
Rohit Deshmukh (FAU)

Motivation and Vision

Problem:

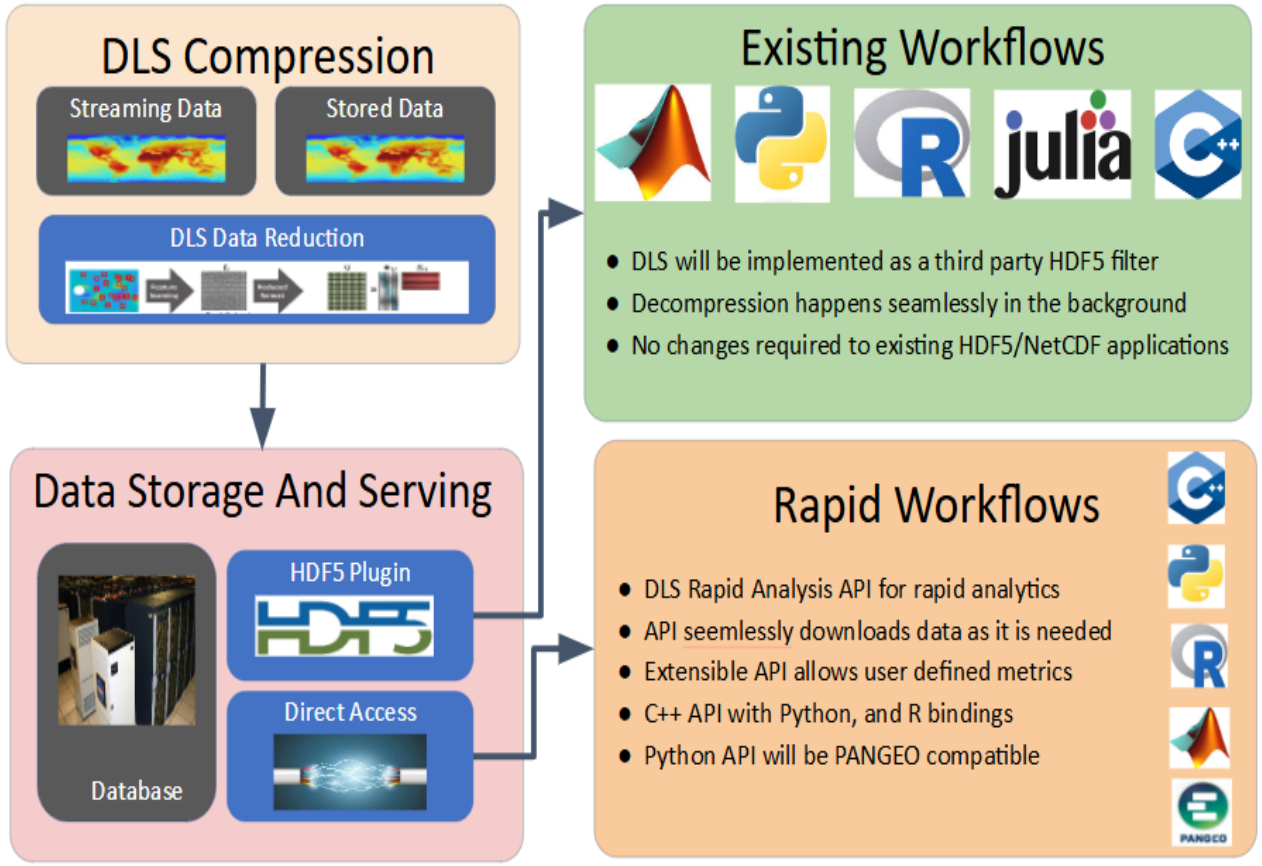
High-Fidelity Scientific Data are expensive to create, store, transfer, and analyze

Solution:

Rapid Analytics through **data-driven**, **geometry-preserving, lossy compression**

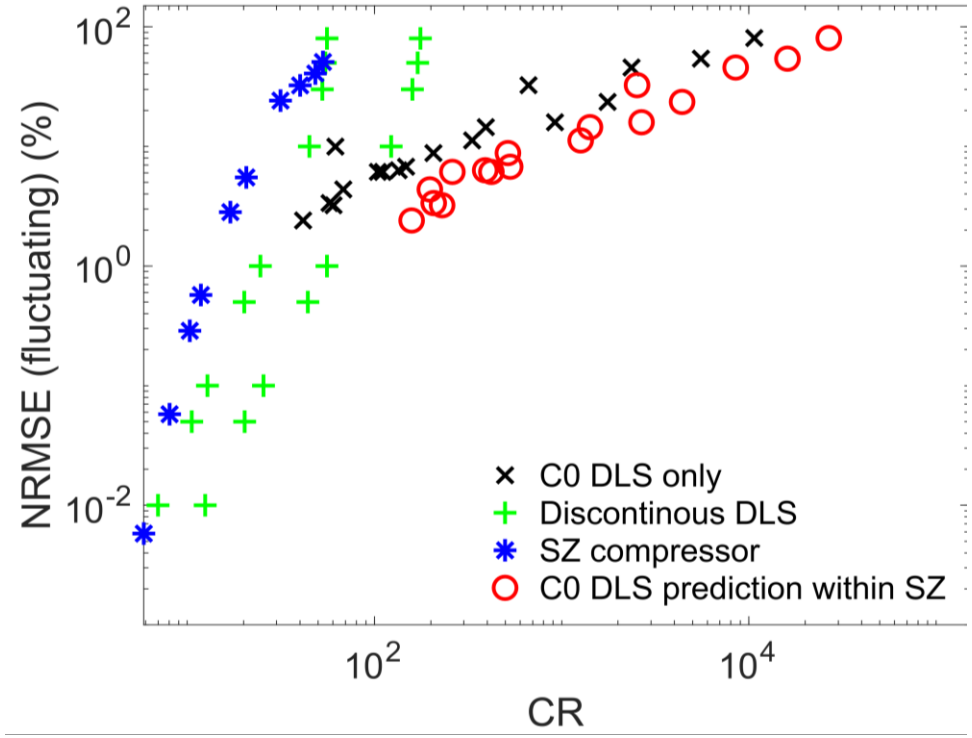
Benefits:

- **Lossy compression** reduces storage and bandwidth without impacting analytic integrity
- **Data-driven compression** algorithms provide state-of-the-art compression-to-error ratios
- **Geometry-preserving compression** enables rapid analytics



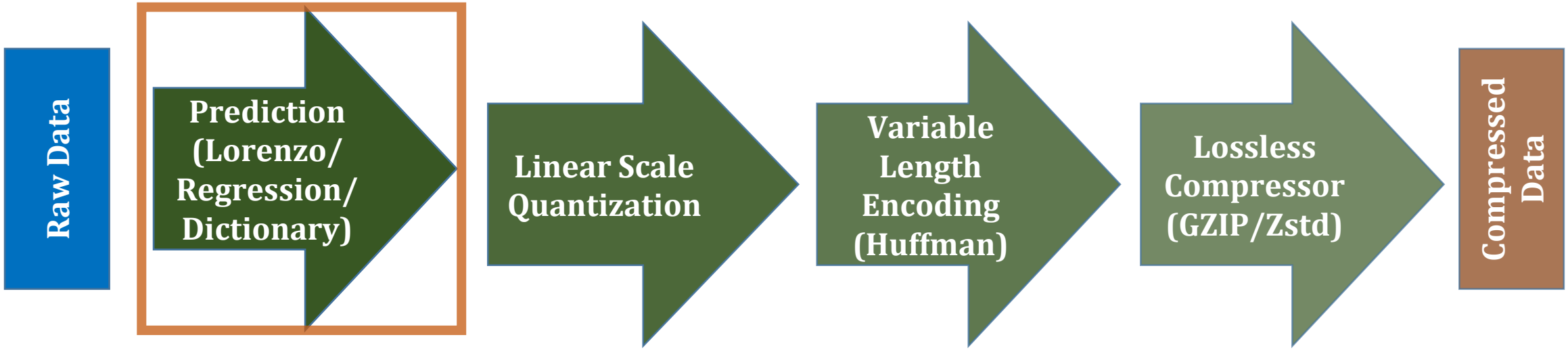
Overview

- Data-Driven Lossy compression
- Data Informed Local Subspaces based Compression (DLS) Algorithms
  - Continuous DLS using GFEM (C0 Continuity)
  - Delayed DLS (fast online/in-situ compression)
  - Zone Based DLS (lower computational cost)
  - Discontinuous DLS (embarrassingly parallelizable)
  - Error bounded DLS (user defined level of error)
- Demonstration
  - High Reynolds Number Turbulent Flow
  - GEOS-5 Nature Run 2m Air Temperature
  - C720 GEOS-FP Forecast
- MLDC Software API and Tools.
- Call to Action



Data-Driven Compression

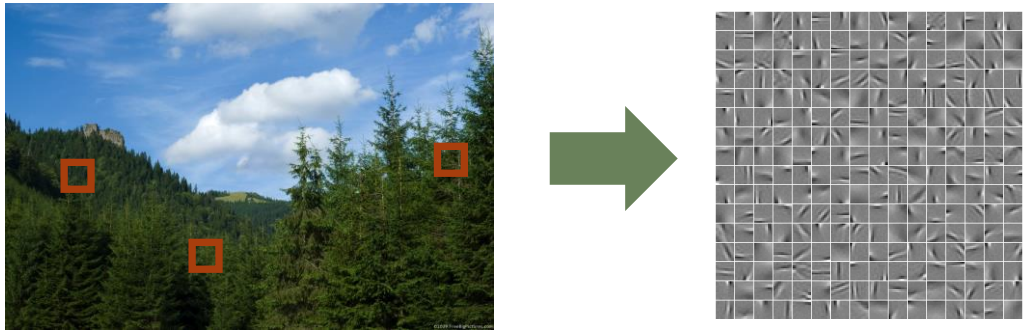
Typical Lossy Compression Workflow



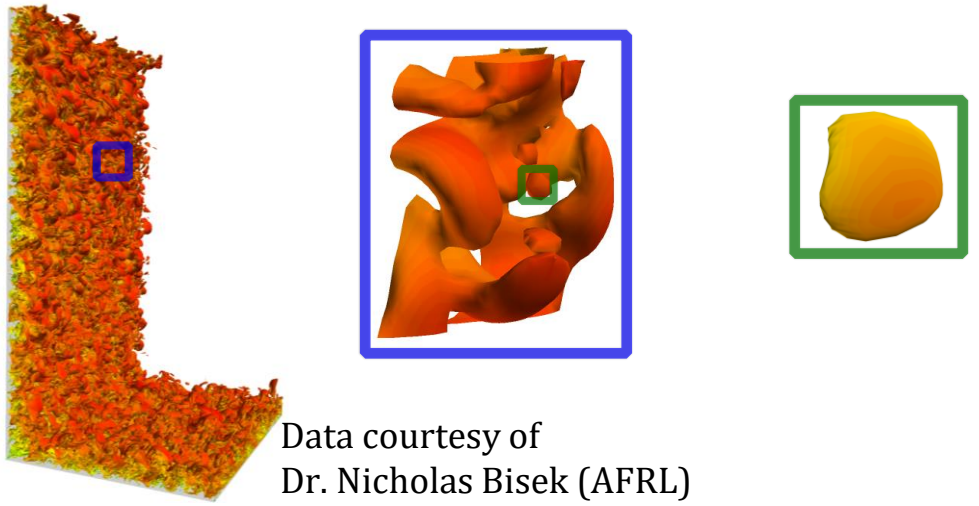
- Bit Grooming typically gives 2x to 4x compression
- **Data-agnostic** prediction steps give 5x to 20x for highly multiscale data
- **Hypothesis: A Data-driven prediction step will increase compression-to-error ratios.**

What is "Data-Driven" Prediction?

- Feature learning from redundant structures in natural images:
- Learn compact set of features that explain large, global, images



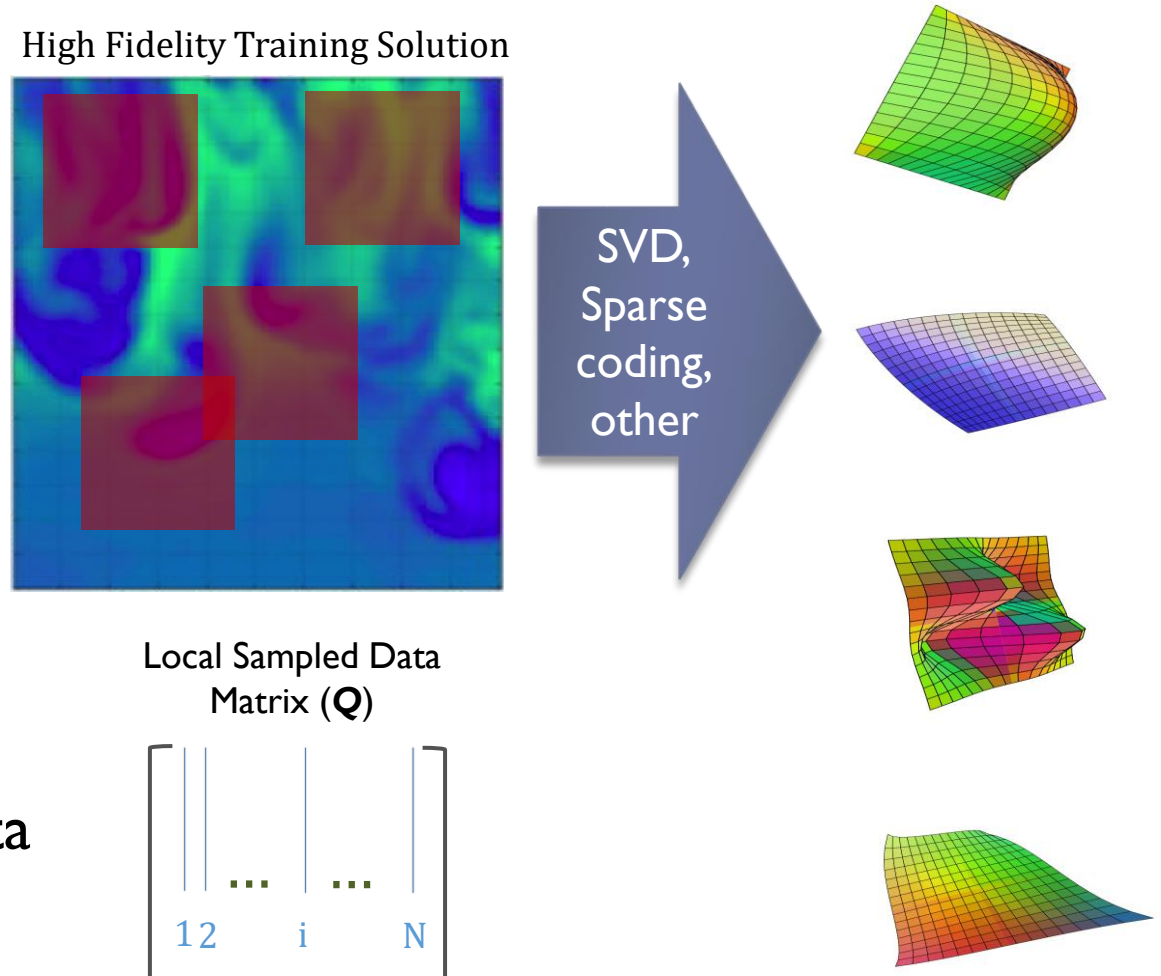
- Similar statistics found in multiscale scientific data:
- Feature learning from large flow fields
  - learn spatially small, simpler structures →
  - more flexibility



Data courtesy of
Dr. Nicholas Bisek (AFRL)

Data Driven Feature Learning

- Step 1: Build the Training Data Set.
  - Randomly select N "patches" from the data set.
  - Assemble patches into a sampled data Matrix (**Q**).
- Step 2: Learn a Local Basis
  - Singular Value Decomposition (SLEPc).
  - Sparse coding
  - Deep learning
- Step 3: Use Learned Features to compress data



Data-Driven Compression using Data-Enriched Finite Elements

Data-informed Local Subspaces (DLS)

Generalized Finite Element Method

[Babuska et al., 1994; Duarte and Oden, 1995]

$$\boldsymbol{u_h}(\boldsymbol{x},t) = \left\{ \sum_{\alpha=1}^{n_{FEM}} \hat{\boldsymbol{u}}_\alpha(t)\varphi_\alpha(\boldsymbol{x}) \right\} \textbf{ FEM}$$

$$+ \left\{ \sum_{\alpha=1}^{n_{FEM}} \varphi_\alpha(\boldsymbol{x}) \sum_{i=1}^{K} \tilde{\boldsymbol{u}}_{\alpha i}(t)\boldsymbol{L_i}(\boldsymbol{x}) \right\} \textbf{ Enrichment of FEM}$$

GFEM enrichment functions are built using the singular vectors of the M largest singular values.



FEM function:

$\varphi(x)$

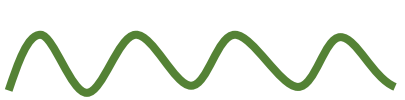


Enrichment functions:

$L_1(x)$



$L_2(x)$



GFEM functions:

$\varphi(x)$



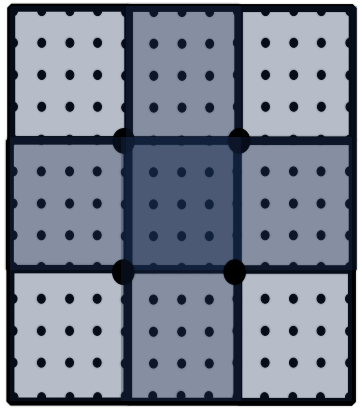
$\varphi(x)L_1(x)$



$\varphi(x)L_2(x)$



C0 Compression over global domain

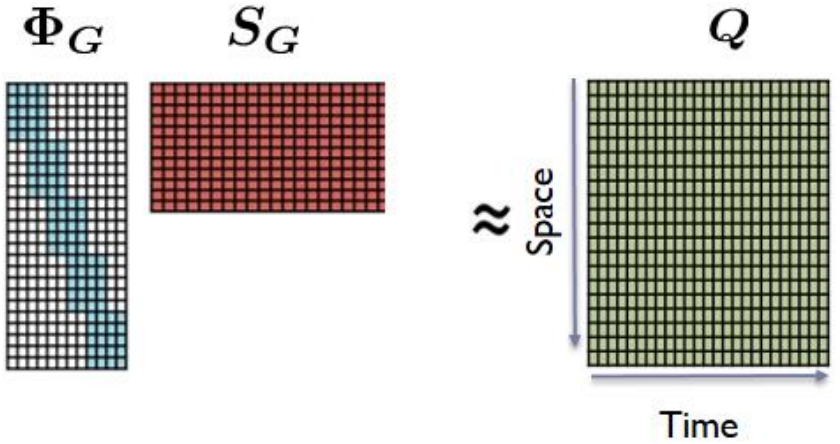


Enrichment functions:

$\boldsymbol{L_1(x)}$ 

$\boldsymbol{L_2(x)}$ 

$$\boldsymbol{u_h}(\boldsymbol{x},t) = \sum_{\alpha=1}^{n_{FEM}} \hat{\boldsymbol{u}}_\alpha(t)\varphi_\alpha(\boldsymbol{x}) + \sum_{\alpha=1}^{n_{FEM}} \varphi_\alpha(\boldsymbol{x}) \sum_{i=1}^{K} \tilde{\boldsymbol{u}}_{\alpha i}(t)\boldsymbol{L_i}(\boldsymbol{x})$$



- Compression is achieved by representing High Fidelity Data inside an enriched finite element space defined on a coarse grid
- Solve equation using Least Squares

$$\Phi_G^T \Phi_G S_G \approx \Phi_G^T Q$$

Global Compression with GFEM

RNET Technologies, Inc.

FAU Florida Atlantic University

One Time Training Step

$$M = \Phi_G^T \Phi_G$$

- Block Banded matrix
- One block per coarse grid element
- Symmetric Positive Semi-definite
- Use Cholesky decomposition to find an upper triangular matrix, **B**, such that:

$$B^T B S_G \approx \Phi_G^T Q$$

Online Compression

For each snapshot q_i

$$B^T B s_i = \Phi_G^T q_i$$

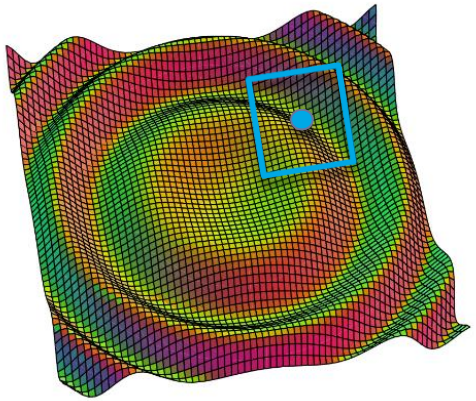
- Use finite element assembly to form the RHS
- Solve using one forward and one backward solve.
- Solution is stored on disk

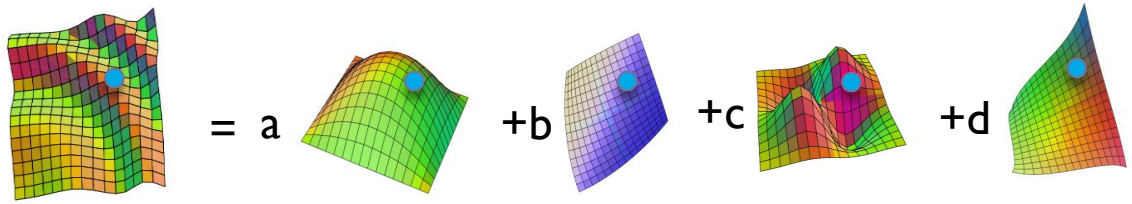
Decompression:

- Geometry Preserving Compressed Data Format allows for Pointwise decompression
- Spatiotemporal separation of variables allows for rapid analytics through batch processing.
- Simple Data layout allows users to only **download what they need**.
- Remote access requires only a **basic** HTML data server with support for the Ranges header (nginx, apache, etc).

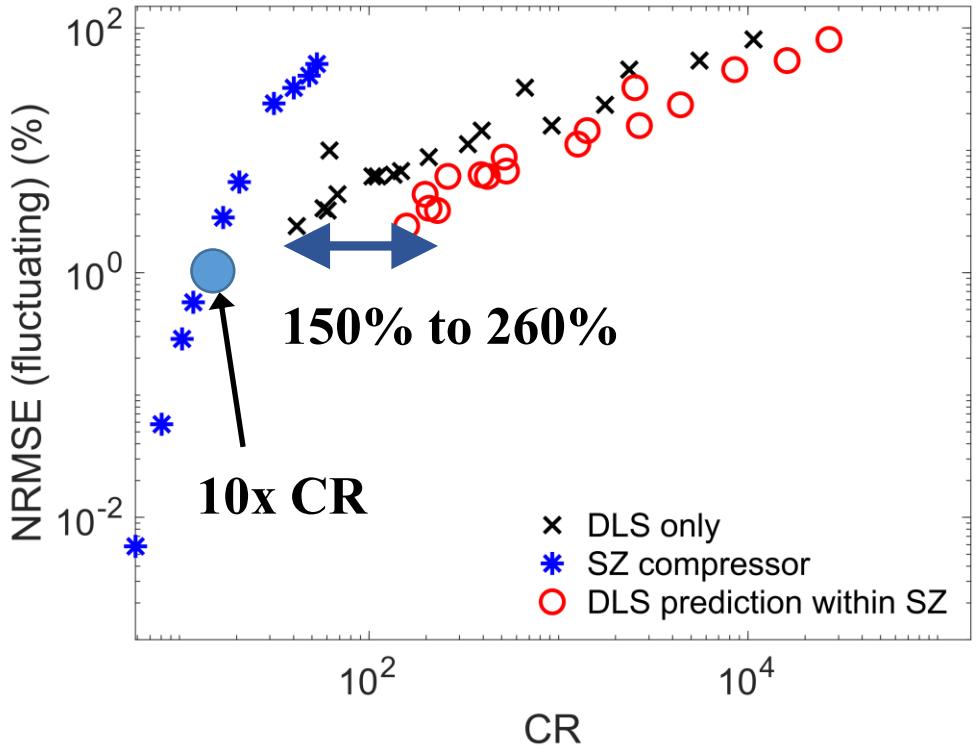
For each snapshot q_i

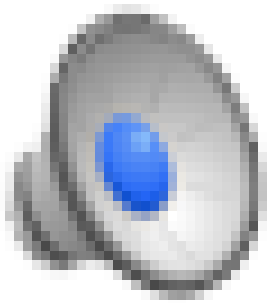
$$q_i \approx \Phi_G^T s_i$$





DLS + Quantization + GZIP





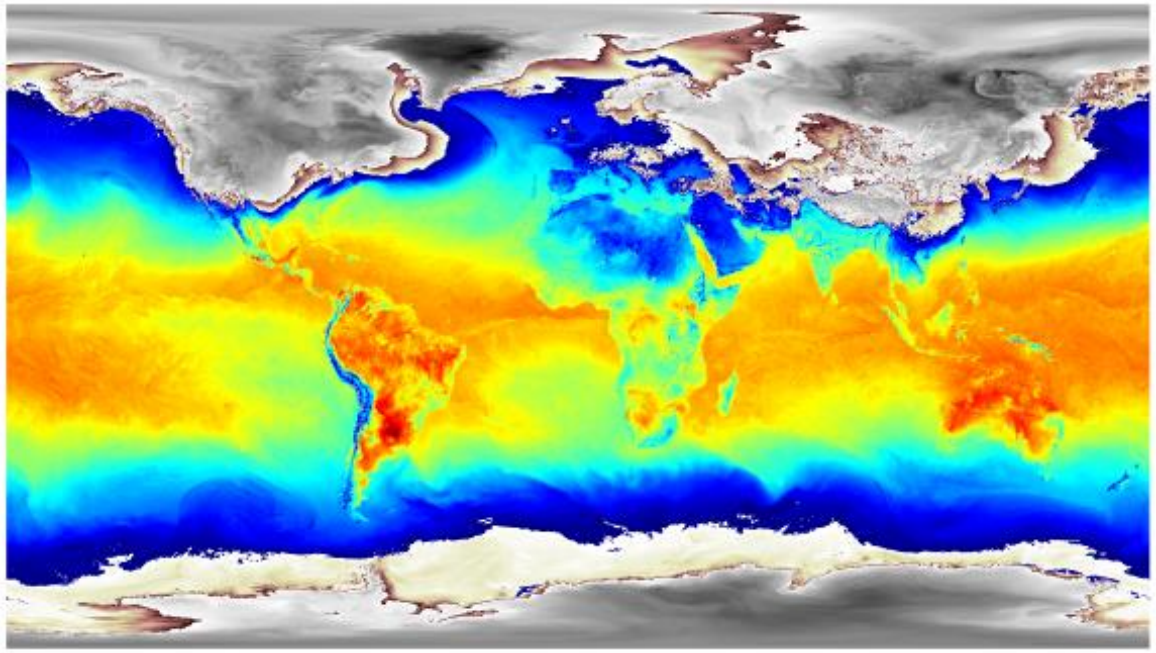
2 TB of unsteady flow data

- Bit-Grooming maxes out at 4x (theoretical, 2-3x in reality) for "reasonable" errors
- The SZ compressor performs well below 10x compression
- **Using DLS prediction within SZ we can push this compression to over 100x!**

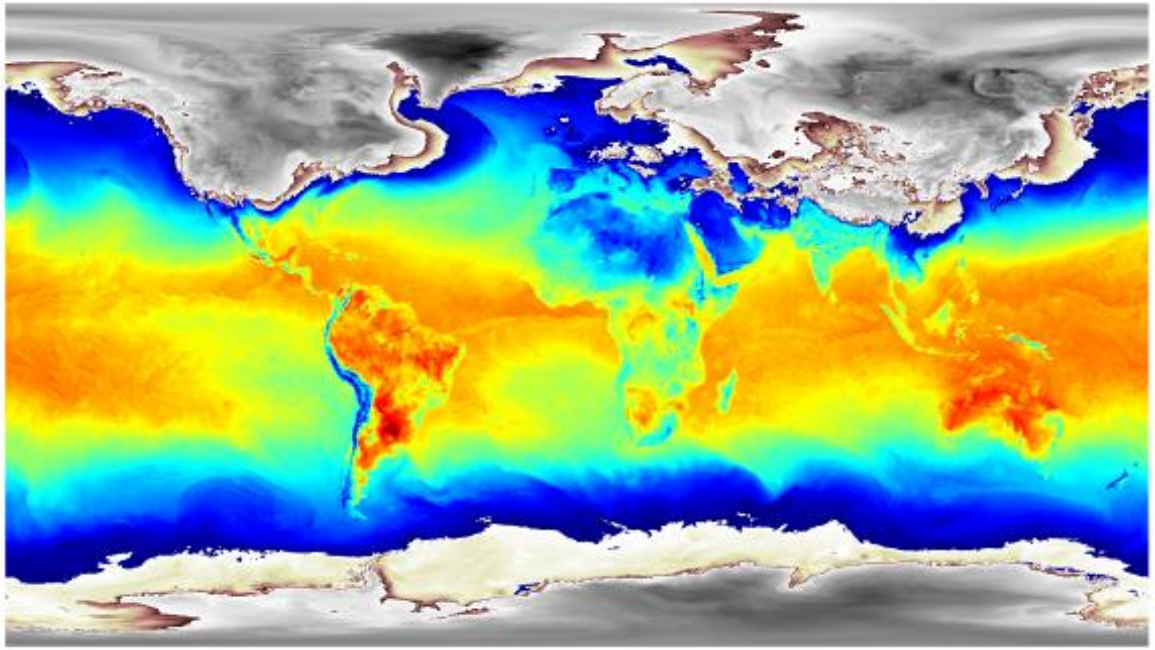
Compression: GEOS-5 NR (Lossy):

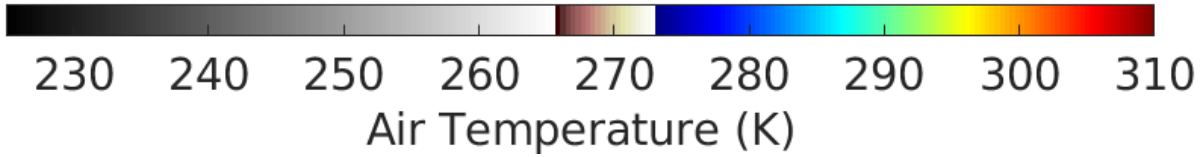
- GEOS-5 Nature Run. Instantaneous 2m Air Temperature **83.97x CR**

Original Solution



Reduced Solution



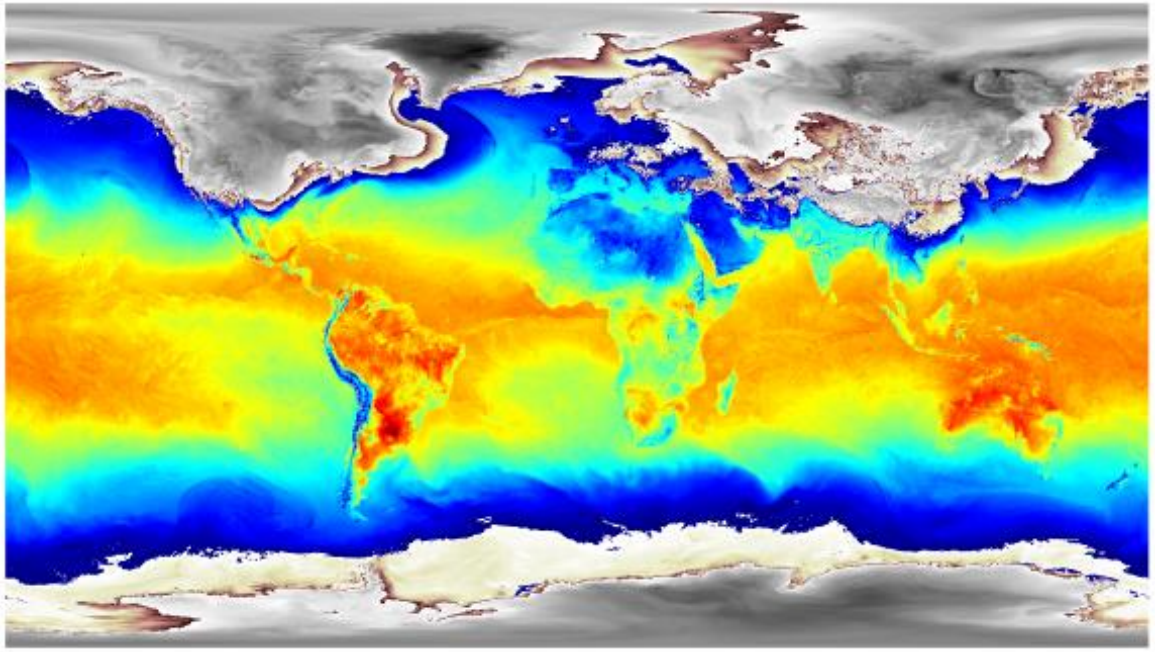


- CR: 83.97x over original size
- NRMSE: 0.2234 %
- Spatial mean temperature diff : -1.56 e-5 K

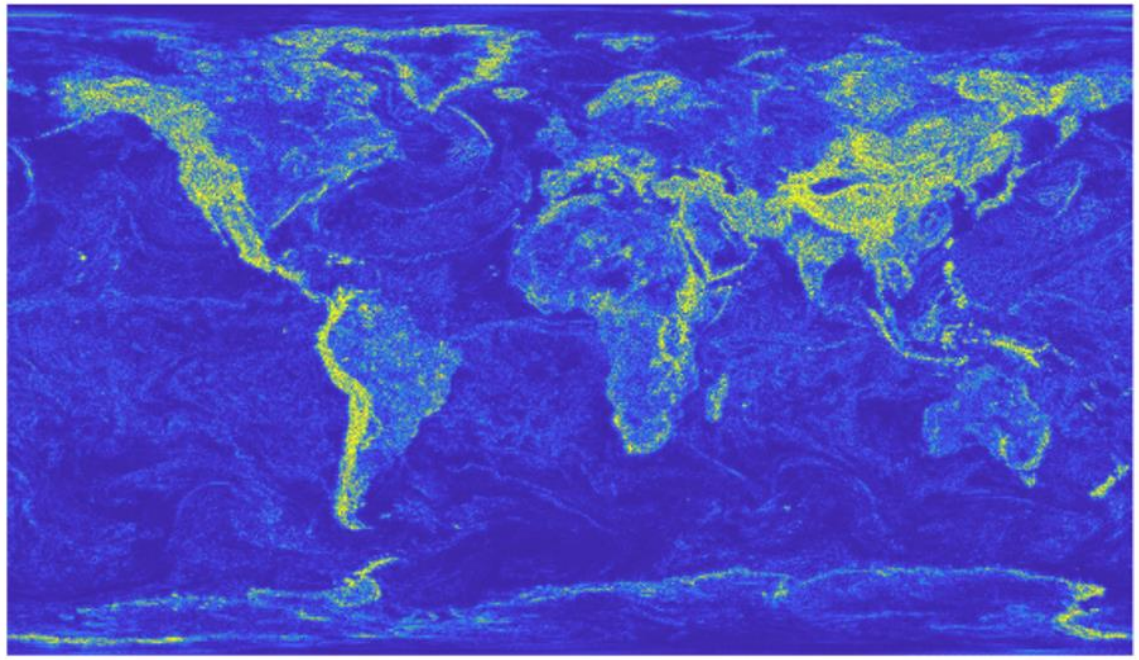
Compression: GEOS-5 NR (Lossy):

- GEOS-5 Nature Run. Instantaneous 2m Air Temperature **83.97x CR**

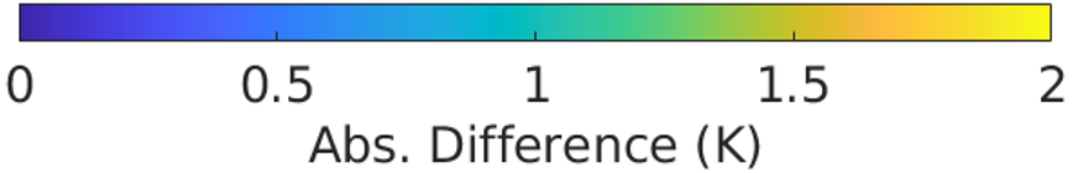
Original Solution



Pointwise Error



- CR: 83.97x over original size
- NRMSE: 0.2234 %
- Spatial mean temperature diff : -1.56 e-5 K



Delayed DLS Compression

Delayed DLS Compression.

- **Problem: In online situations, the cost of the Cholesky decomposition, and/or the forward and backward solves, become an issue.**
- Solution: Complete the Cholesky decomposition and forward and backward solves in post processing.
  - At runtime, we form and store the RHS for each snapshot.

$$d_i = \Phi_G^T q_i$$

  - In postprocessing, we solve the system

$$B^T B s_i = d_i$$

- Decompression is the same as for continuous DLS

Preliminary Results

- 10x speed up in online compression time for single snapshot.
- Compression ratios are identical
- Benefits decrease as the number of snapshots increases

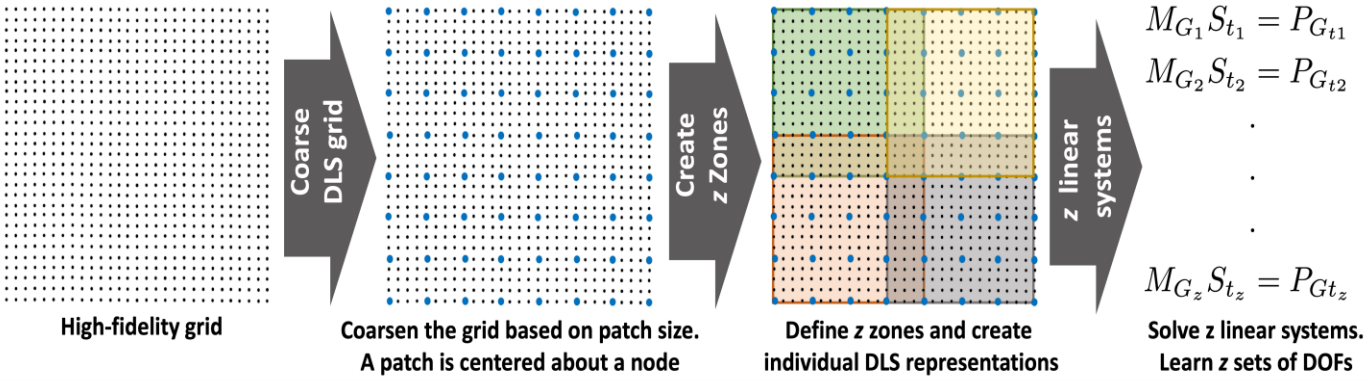
Zone-Based DLS Compression

Zone-Based DLS Compression

- **Problem:** The cost of the DLS algorithm grows with the size of the coarse grid.
- **Solution**: Solve the problem using a set of fixed size compression zones.
  - Features are learned on the entire domain
  - The Correlation Matrix for a fixed size subdomain is formed and decomposed using Cholesky Decomposition.
  - Compression is achieved by solving the DLS equation on Z independent but overlapping zones.

$$\forall z_i \in Z, \text{find } s_{z_i} \text{ s.t}$$

$$Q_{z_i} \approx B^T B S_{z_i}$$



Num Zones	CR	NRMSE (%)	Runtime (ms)
1	74.06x	0.737731	189341
4	73.68x	0.73781	26379
9	73.30	0.73788	18587
16	**72.9**	0.737941	**17229**
25	66.9	0.737465	17581

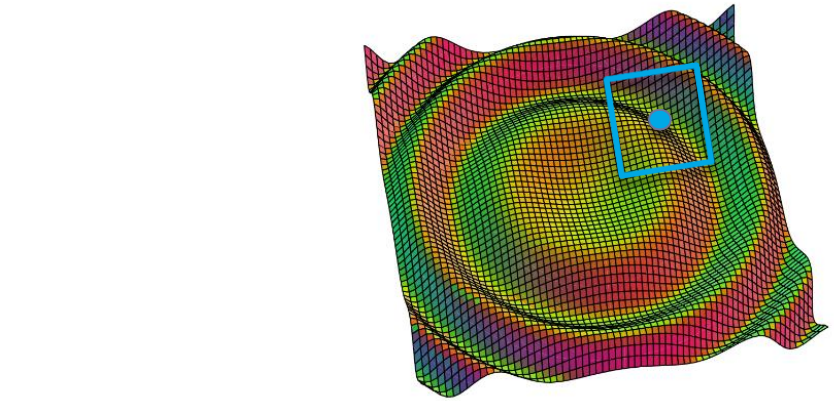
Result: Compression ratios, error and total runtime (training + compression) when compressing air temperature in 2D 5760x5760 lat-lon grid.

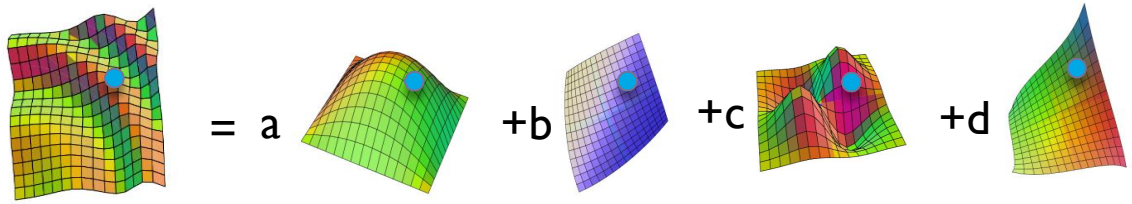
Zone-Based Decompression:

- Decompression at a single point
  1. Determine what zone(s) the point lies in.
  2. Reconstruct the solution in each of those zones.
  3. Final value is the average of all reconstructed values.
- Anecdotal testing suggests we can drop the interpolation step without impacting error.

For each snapshot q_i

$$q_i \approx \Phi_G^T s_i$$





Discontinuous DLS Compression

Error-Bounded Compression

- **Goal:** Maximize the compression ratio subject to a user defined error tolerance.
- Let **P** be the number of nodes in each snapshot.
- Let ε be a user-defined global error tolerance
- Let **f(x)** be a lossless compression function that returns the storage cost for the vector **x**.
- The norm used to measure error can be any user-defined metric.

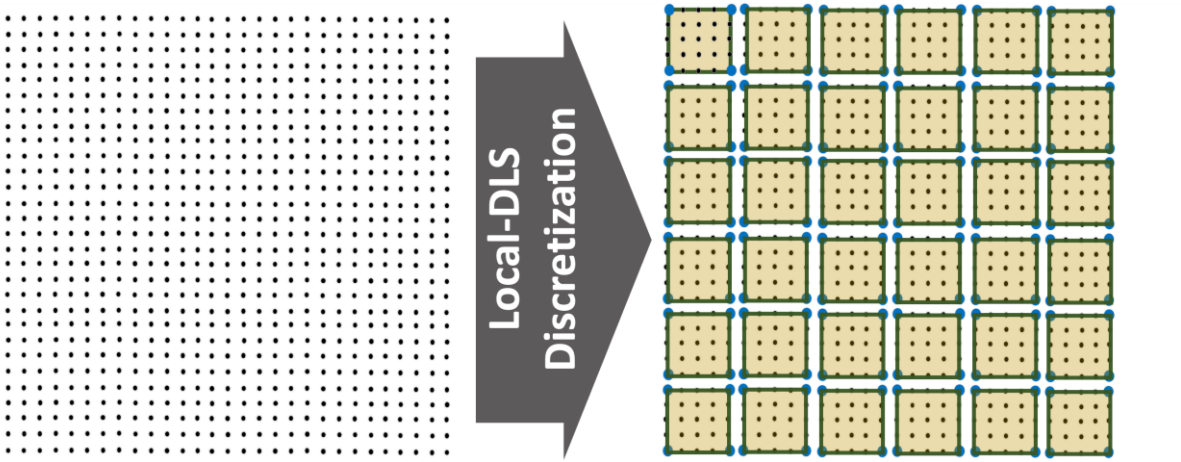
$$\text{For each snapshot } q_i :$$

$$s_i = arg \min_{x \in R^P} f(x)$$

$$||q_i - U s_i|| < \epsilon$$

Error-Bounded Compression with DLS

- Use SVD to learn a complete orthonormal basis for the coarse grid elements → same as C0-DLS
- Use discontinuous DLS to adaptively compress the solution on a set of non-overlapping patches.
- Find a local error bound that, when satisfied on each patch, guarantees the global tolerance will be met.
  - $\varepsilon_l = \varepsilon_G / N$ in the L_1
  - $\varepsilon_l = \varepsilon_G / \text{sqrt}(N)$ in L_2
  - $\varepsilon_l = \varepsilon_G$ in L_∞



$$\text{For each } q_p \in Q, \text{ find}$$

$$d_p = arg \min_{x \in \mathbf{R^P}} (f(x))$$

$$\text{s.t } ||q_p - U d_p|| < \epsilon_l$$

Adaptive Selection

$$\text{For each } q_p \in Q, \text{ find}$$
$$d_p = arg \min_{x \in \mathbf{R^P}} (f(x))$$
$$\text{s.t } ||q_p - U d_p|| < \epsilon_l$$

- Lossless compression functions work best when applied to data with repeating patterns
- Set U to be the learned orthonormal basis ($U^{-1} = U^T$)
- Make the DOFs vector for each patch "easier" to compress by zeroing out least significant DOFs.

The DLS Adaptive Selection Algorithm

1. Calculate the DOFs vector required to represent each patch using our learned basis.

$$s_p = U^T q_p$$

2. Use bisection method to zero elements of s_p (from smallest to largest) until the local error bound can no longer be maintained.

3. Use bisection method to bit-groom non-zero elements in s_p until the local error bound can no longer be maintained

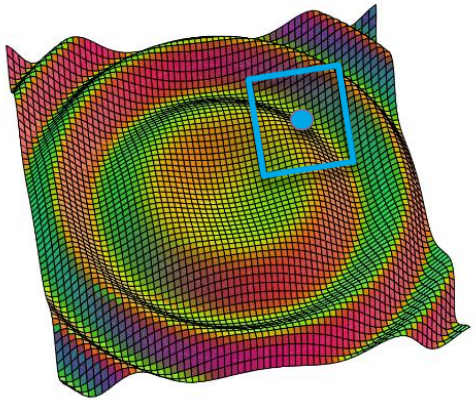
4. Compress reduced solution using GZIP and store the result on disk.

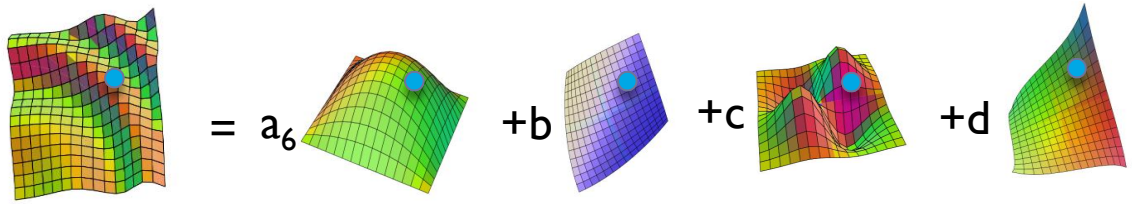
Decompression: Error Bounded DLS

- Discontinuous spatiotemporal separation of variables allows for a reduced set of rapid analytics.
- Users still only **download the data they need**.
  - Location and size of each elements DOFs are defined in the header.
  - Compressed DOFs are downloaded and decompressed on the client.
  - Enrichment functions can be downloaded on an as-needed basis.

For each snapshot patch q_p :

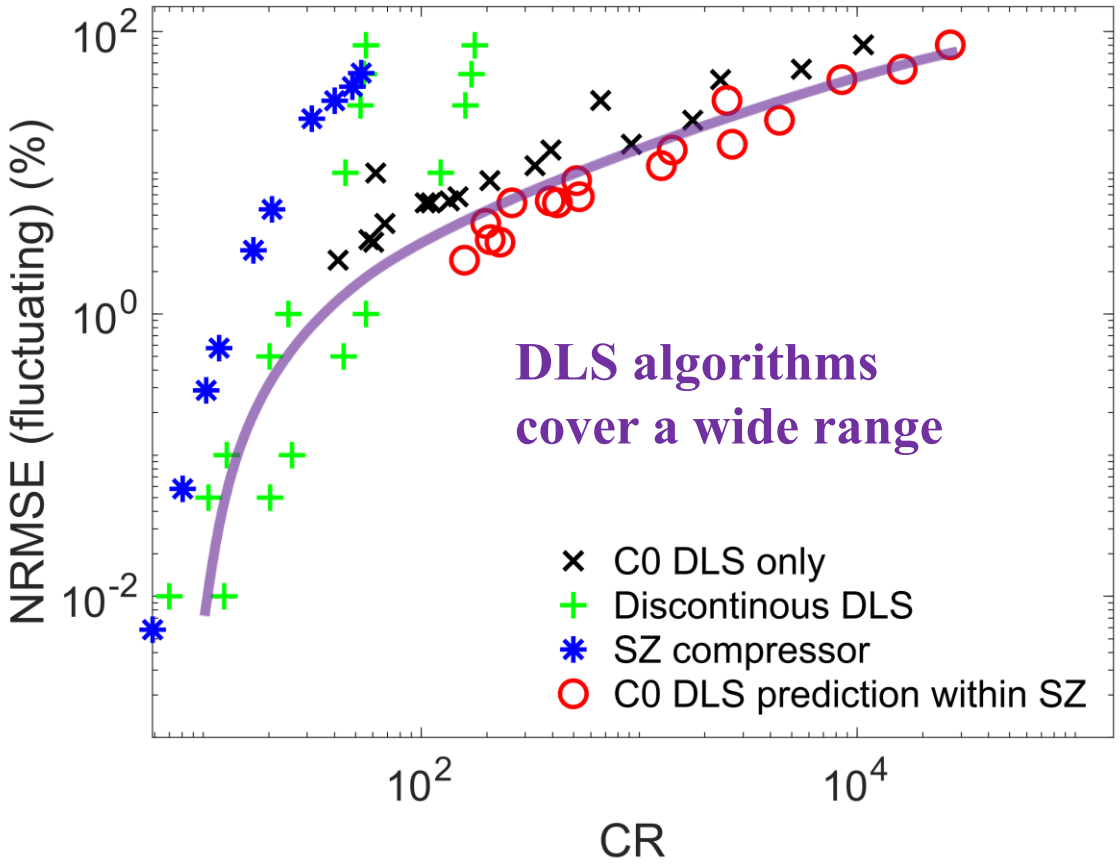
$$q_p \approx U_m s_p$$



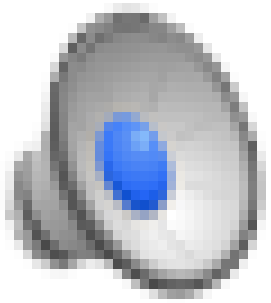


Error Bounded Lossy Compression

Application to 3D flow past a stationary cylinder at Re 100,000



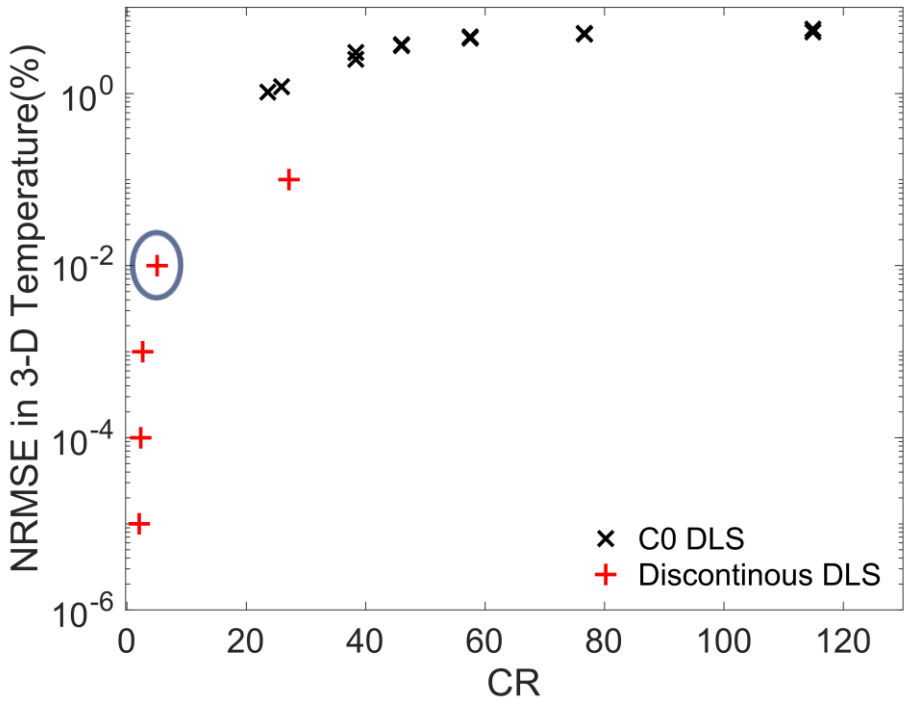
2 TB of unsteady flow data



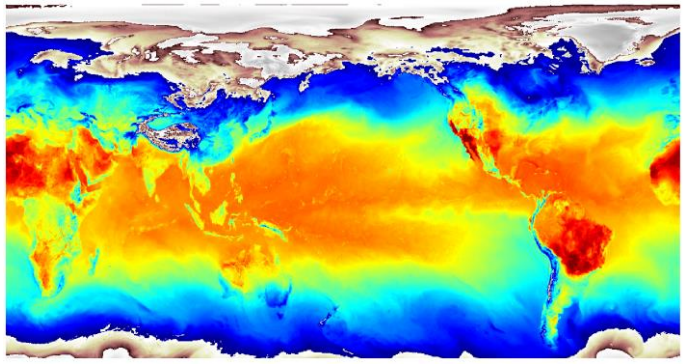
DLS-based compression algorithms provide multiple options to users to optimally compress for low errors and high compression.

Error Bounded Lossy Compression

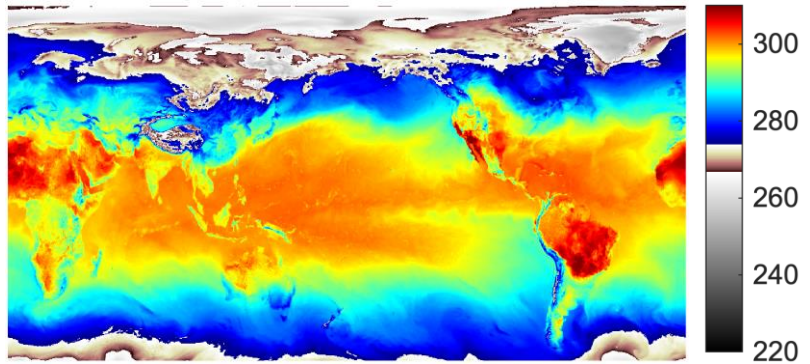
Application to GEOS-Forward Processing forecast (C720 cubed sphere) temperature data



Original surface inst. temperature

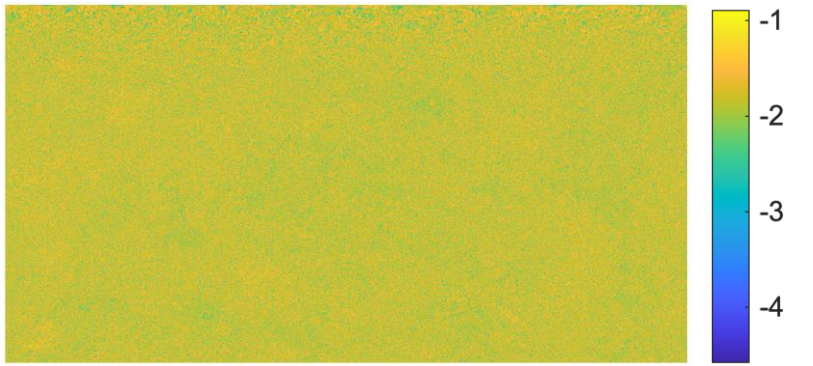


Compressed temperature

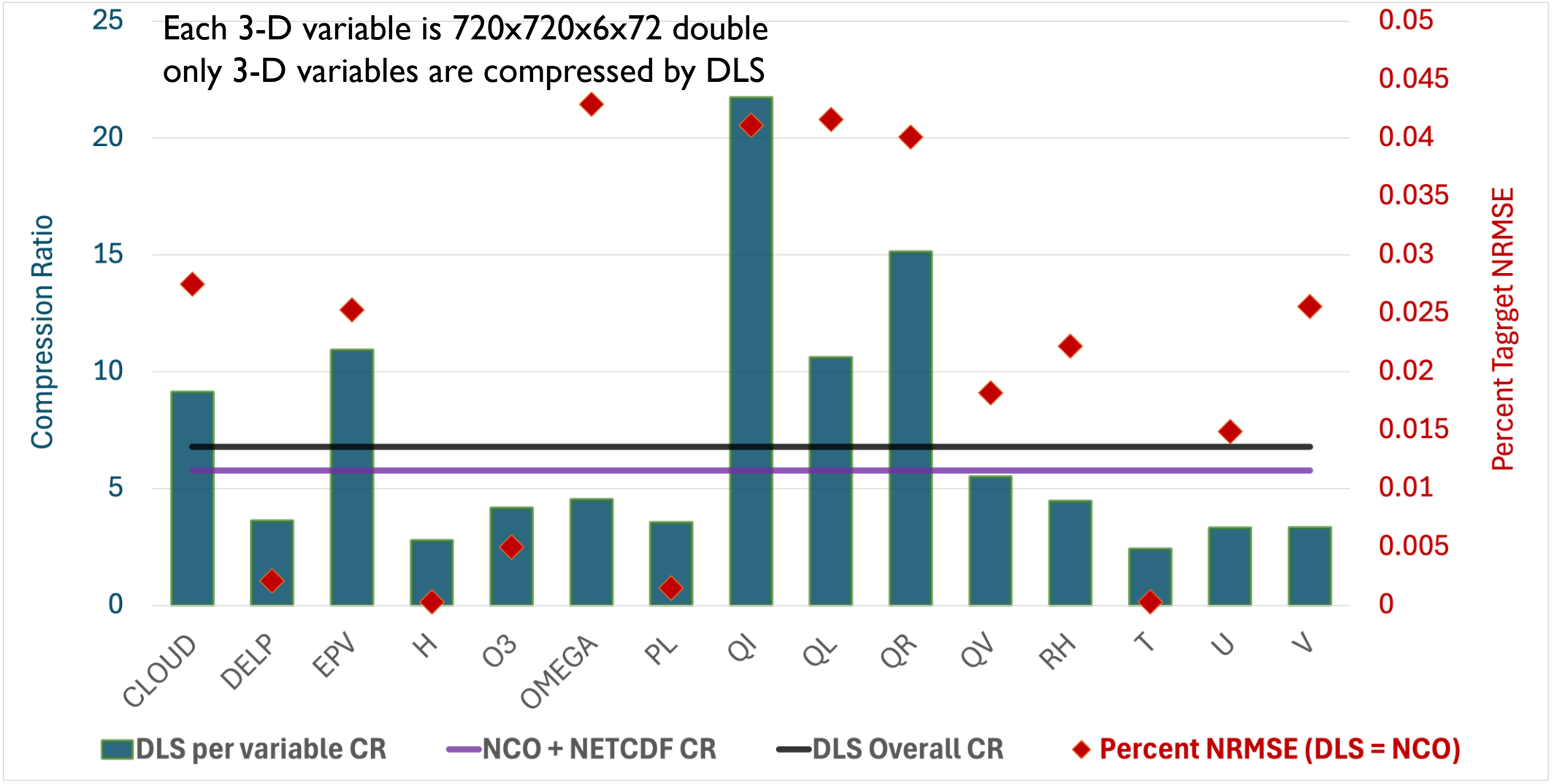


0.01% NRMSE at
Compression Ratio of ~6x

Pointwise error ($\log_{10}$)



Compression of a single GEOS-FP C720 file



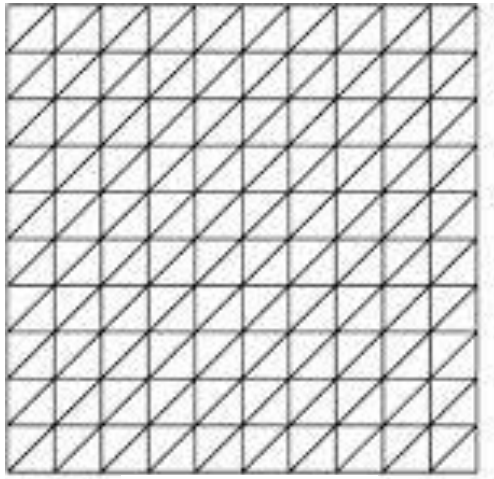
Error-Bounded DLS for Unstructured grids

(Work in Progress)

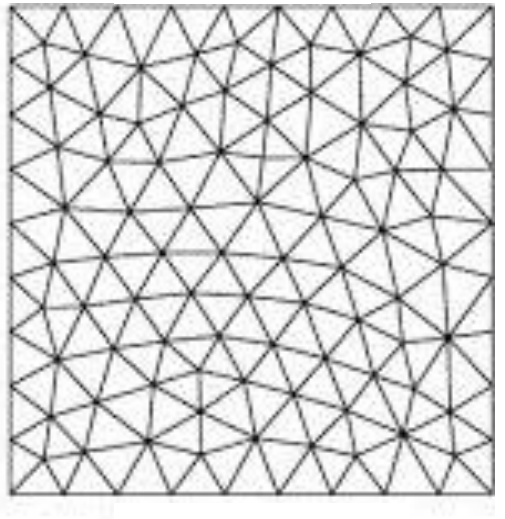
Error-Bounded DLS for Unstructured Grids.

- Coarse grids are not easy to calculate:
  - For structured grids, coarse grids are built by skipping a fixed number of nodes in each direction.
  - For structured grids, sophisticated mesh coarsening algorithms are required

- The distribution of fine grid points is no longer constant.
  - For structured grids, each coarse grid element contains an identical, uniform nested fine grid.
  - For unstructured grids, the number and location of fine grid points in each coarse grid changes.



Structured mesh



Unstructured mesh

Mesh Coarsening

- Generate a Coarse Grid
  - User defined coarse grids.
  - Nested Grids
  - Mesh coarsening algorithms
  - Mesh partitioning with bounding boxes
- User defines an approximate coarsening factor
- Automatic mesh coarsening for simplex grids is next.

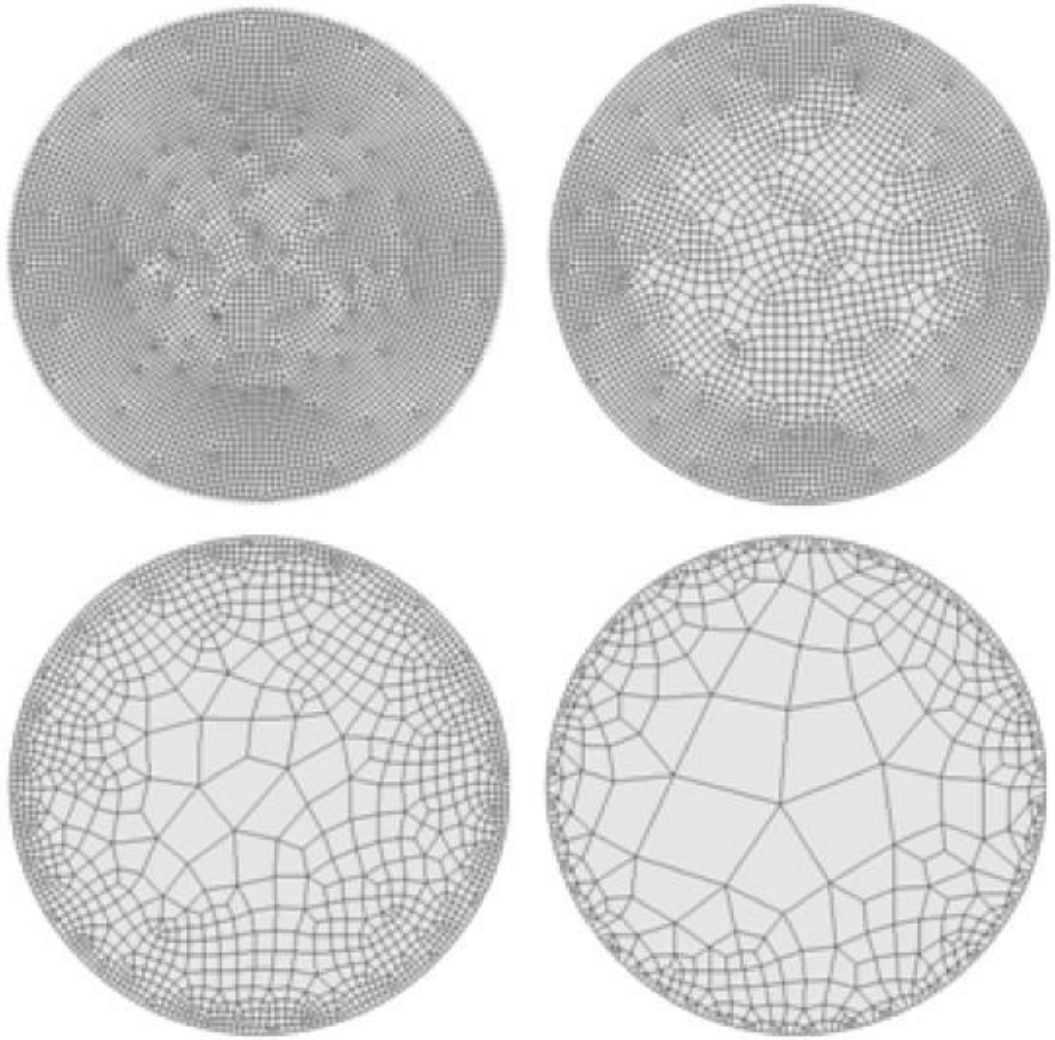
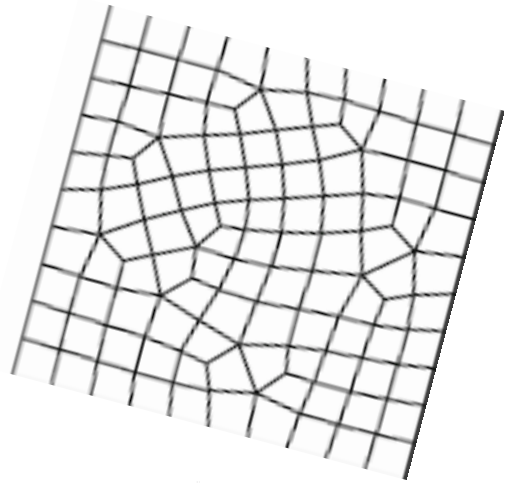


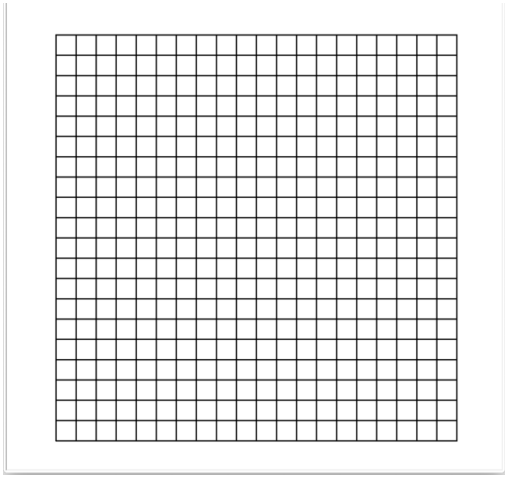
Image Credit: Shepherd et al. https://doi.org/10.1016/j.finel.2009.06.024

Mapping to a structured Reference Element

$$f(u,v) = (1-u)(1-v)\mathbf{a} + (1-u)v\mathbf{b} + (u)(1-v)\mathbf{c} + uv\mathbf{d}$$



- For N randomly selected coarse grid element
  - Map the elements nested fine grid solution onto a structured reference grid.





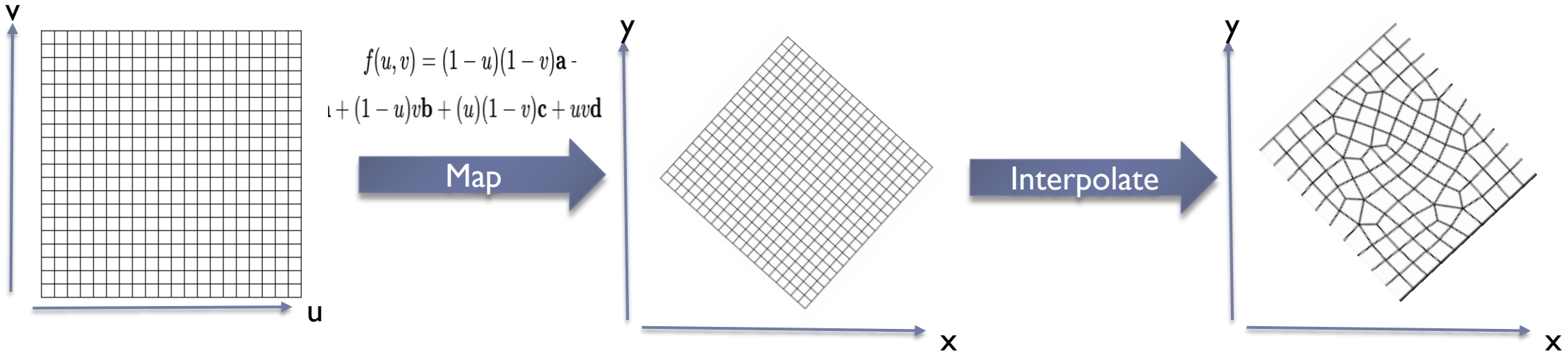
- Mapping an arbitrary point from the reference grid to the element is fast
- Mapping an arbitrary point from the element to the reference grid is slow (nonlinear solve)

D-DLS Compression For Unstructured Grids

1. Coarsen the unstructured grid using a mesh coarsening algorithm
2. Build the training data matrix Q
   - For N random patches **i = 0,1,2,..N**
     - Use Interpolation to map coarse grid element onto the reference grid
     - Encode the reference solution as a vector and store it as column q_i
   - Use SVD to calculate the data-driven orthonormal basis for elements defined on the reference grid
3. Compression: For each coarse grid element:
   - Use Interpolation to map coarse grid element onto the reference grid
   - Use Adaptive D-DLS to compress solution defined on reference grid
4. Storage: Write compressed DOFs, coarse grid and learned basis to disk

D-DLS Decompression For Unstructured Grids

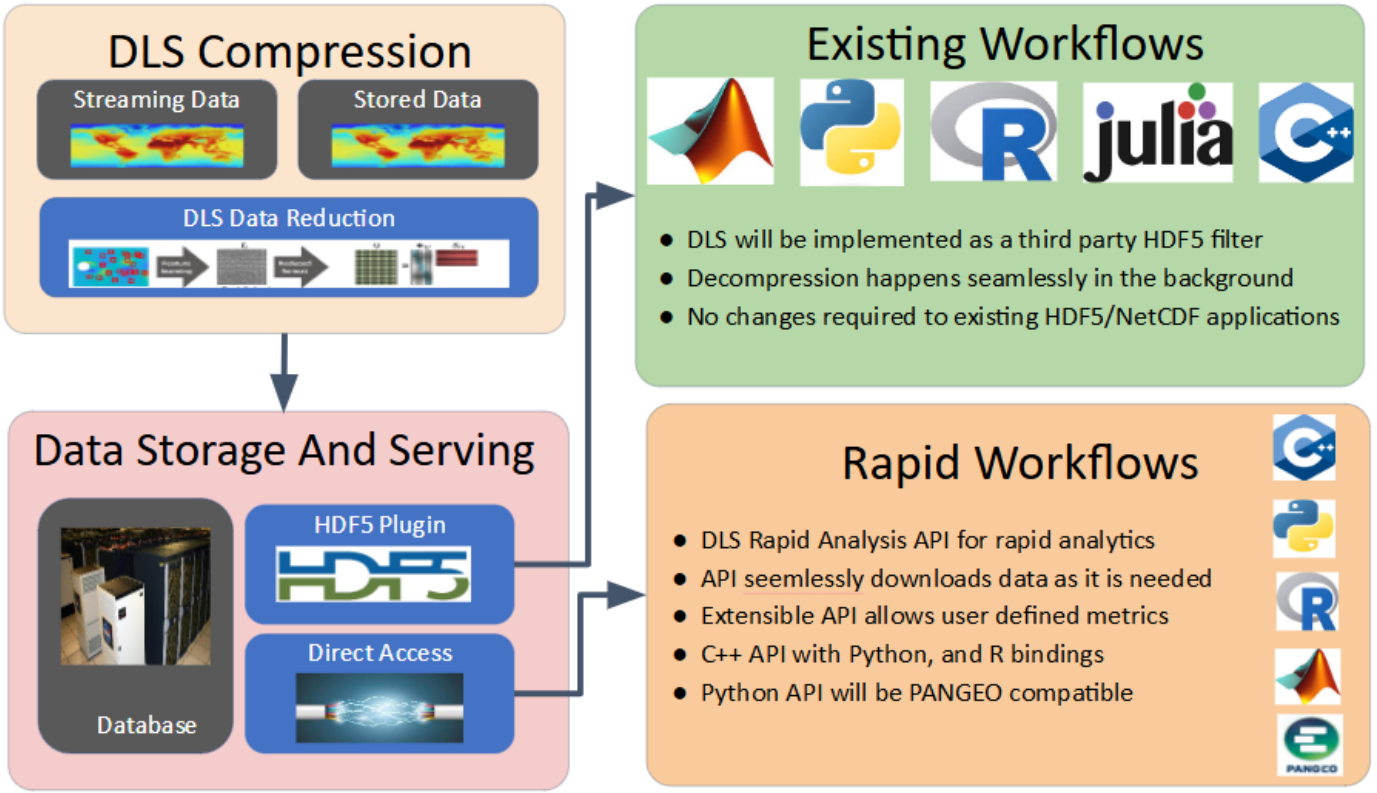
- For each element in the compressed coarse grid
  - Read compressed DOFs from file and decompress
  - Multiply DOFs with basis to reconstruct the solution on the reference element
  - Map the reference grid into XY space
  - Interpolate the XY solution onto the original nested fine grid



Integration and API

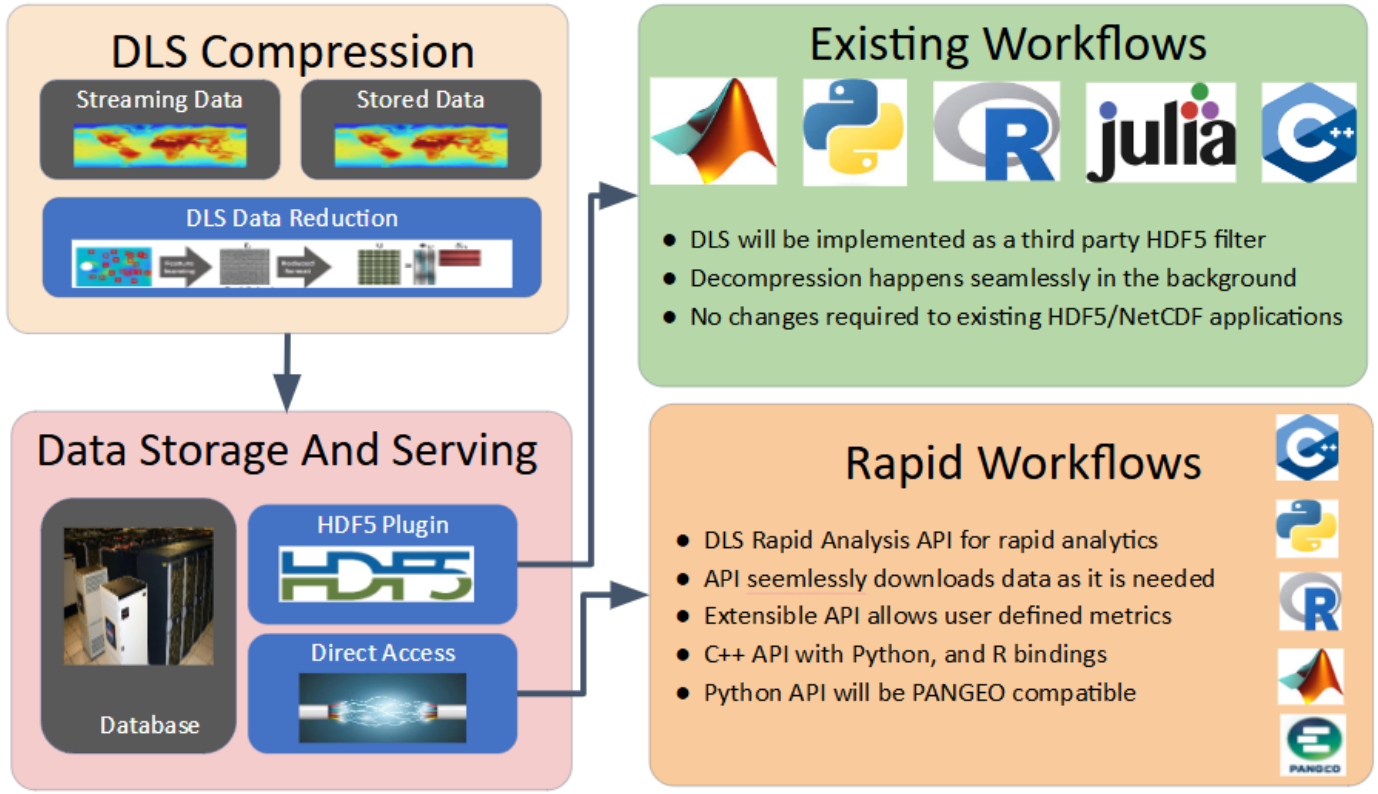
A Platform for Rapid Data Analytics

- DLS Data Compression Library
  - Open-source C++ library + CL interface
  - Uses MPI for spatial domain decomposition
  - Leverage PETSc for linear algebra operations.
  - Offline Data Compression
    - Supports Plot3D, NetCDF, + more
  - Online Data Compression
    - Leverage simulation's domain decomposition



A Platform for Rapid Data Analytics

- HDF5 Filter Plugin
  - Seamless compression of variables in NetCDF/HDF5 files
  - Compressed data can be used in existing workflows without changes
- Python API for Rapid Analytics
  - XArray data structures
  - Supports rapid analytics
  - Data will be downloaded as needed
- Basic Data Server
  - Only requirement is a server that supports HTTP Range Header



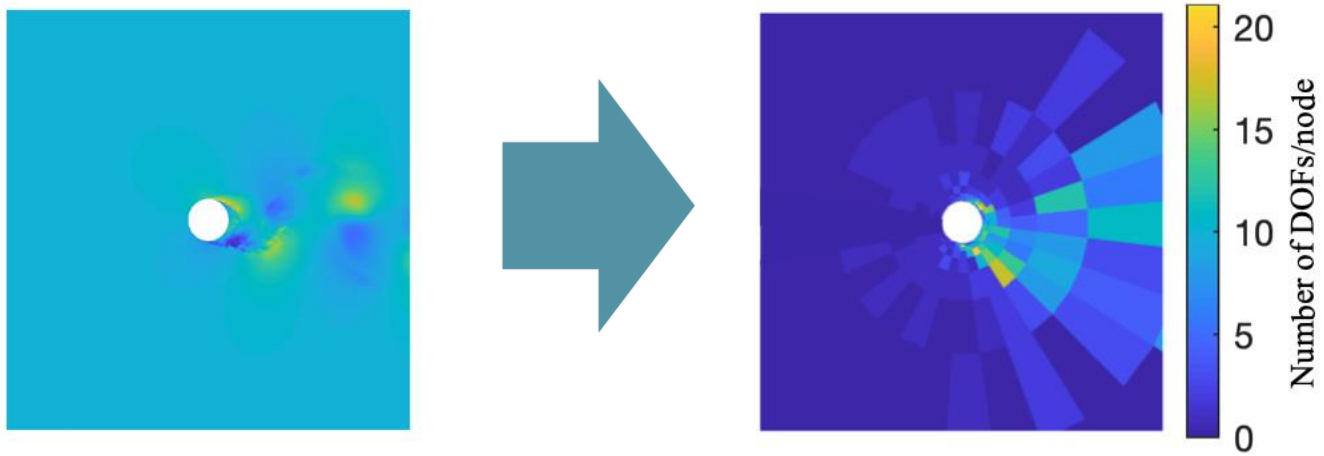
SBIR Project Information

- **Project Title:** Rapid Data Analytics Using Machine Learning
- **SBIR Topic:** NASA 2023, Topic S17 (01), Technologies for Large Scale Numerical Simulation (Exascale Computing)
- **Prime:** RNET Technologies, Inc.
  - PI: Ben O'Neill (BONeill@RNET-Tech.com)
  - Contract: V. "Nagu" Nagarajan (VNagarajan@RNET-Tech.com)
- **Subcontractor: Florida Atlantic University**
  - PI: Rohit Deshmukh (rdeshmukh@fau.edu)
- **NASA Technical Monitor**
  - Sharad Gavali

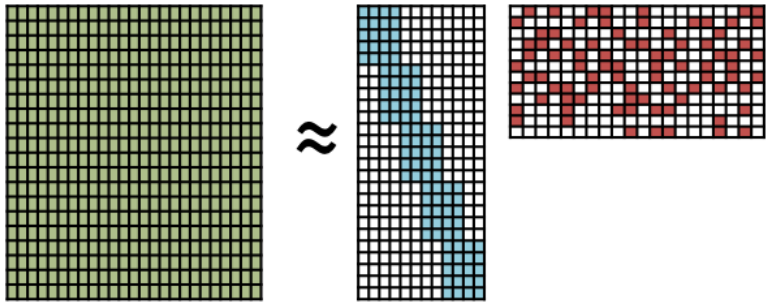
Backup Slides

Sparse GFEM: towards hyperreduction

Further improve the efficiency in decomposition by sparsifying the coefficient matrix



$$Q \approx \Phi S$$



Greedy selection of 'best' features

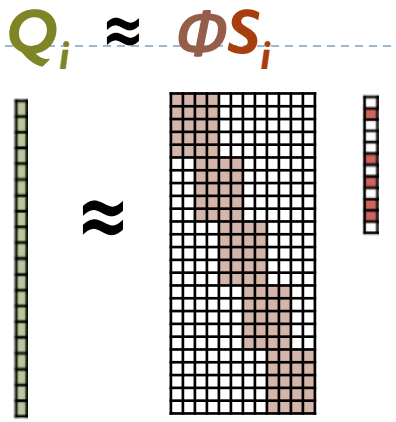
- Select k vectors out of K vectors in Φ

[Rubinstein et al., 2008]

$$\min_{S_i} \quad \|Q_i - \Phi S_i\|_F^2$$

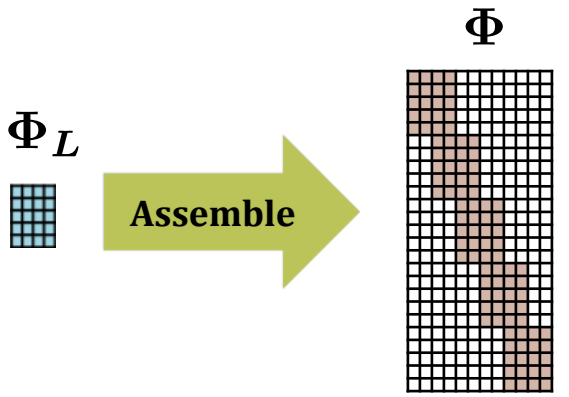
$$\text{such that:} \quad \|S_i\|_0 \leq k$$

$$\text{where,} \quad \|\Phi_j\| = 1 \ \forall \ j$$



The minimization is carried out iteratively, for one row of S_i at a time

1. Begin with a large dictionary of local enrichment features Φ_L
2. Construct the global basis Φ from Φ_L
3. Set a constraint on the desired compression ratio
4. Compute k
5. Solve the minimization problem



Batch-OMP + Finite Elements

Algorithm 2: Batch-Orthogonal Matching Pursuit.[Rubinstein et al., 2008]

Input : Dataset to be compressed in terms of global snapshots matrix $u_e = [q_1 q_2 \cdots q_N]$ of size $M \times N$, Global Dictionary matrix $\Phi_{M \times K}$, target error difference $\triangle\epsilon_0$

Output: Sparse degrees of freedom matrix S

1 Init: $\mathbf{G} := \Phi^T\Phi$

2 **for** *each* t **do**

3 Init: $\alpha^0 := \Phi^T x$, $I := ()$, $\mathbf{L^1} := [1]$, $\delta^0 := 0$, $n := 1$, $\alpha_0 := \alpha$

5 **while** $\triangle\epsilon > \triangle\epsilon_0$ **do**

6 $\hat{k} := Argmax(|\alpha|)$

8 **if** $n > 1$ **then**

9 $g := \mathbf{G}_{I^n,\hat{k}}$

11 $w :=$ Solve for $w \left\{ \mathbf{A^{n-1}} w = g_{I_n} \right\}$

13 $\mathbf{A^n} := \begin{bmatrix} \mathbf{A}^{\tilde{n-1}} & \underline{0} \\ \underline{w}^T & \sqrt{1 - \underline{w}^T \underline{w}} \end{bmatrix}$

14 **end**

15 $I := (I, \hat{k})$

17 $c^n :=$ Solve for $c \{ \mathbf{A^n}(\mathbf{A^n})^T c = \alpha^0_{I^n} \}$

19 $\beta^n := G_{I^n} c^n$

21 $\alpha := \alpha - \beta^n$

23 $\delta^n := (c^n)^T \beta^n_{I^n}$

25 $\triangle\epsilon := \delta^n - \delta^{n-1}$

27 $n := n + 1$

28 **end**

29 $S_t := c^n$

30 **end**

- Explicit computation of Φ is expensive → use FEM matrix assembly instead
- Two types of operations where Φ is used:
- Correlation $\Phi^T\Phi$
- Projection $\Phi^T y$
- Correlation matrix is also sparse → further improvements made by using sparse matrices and operations

Increase efficiency by:

- Embedding ML algorithms in your computational mechanics frameworks
- Introducing sparsity

Reduction of the flow past a cylinder at Re 100,000

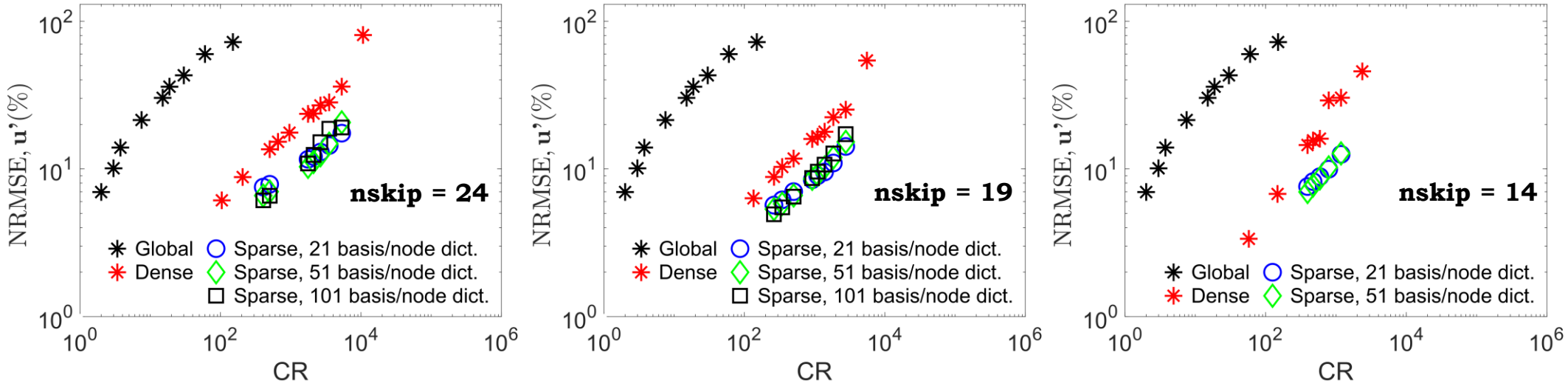
- Begin with a large dictionary of the local enrichment basis with N columns: 21, 51, and 101 basis/node
- Set the compression ratio to the values obtained by dense-GFEM

$$\min_{\boldsymbol{S_i}} \ \|\boldsymbol{Q_i} - \boldsymbol{\Phi S_i}\|_F^2$$

$$\text{such that: } \|\boldsymbol{\Phi}_j\| = 1 \ \forall \ j$$

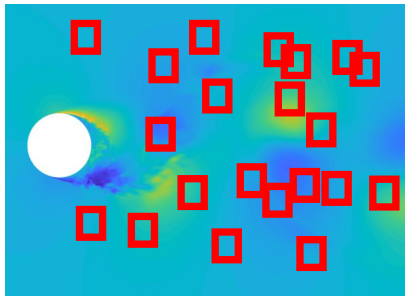
$$\|\boldsymbol{S}_i\|_0 \leq k$$

Approximation of the **fluctuating** velocity test fields

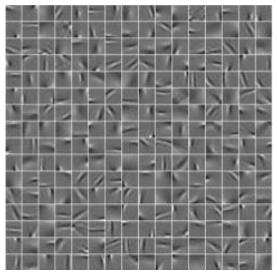


Factor of 2-4 improvement in CR for fixed NRMSE

Methodology: Reduce-then-Analyze



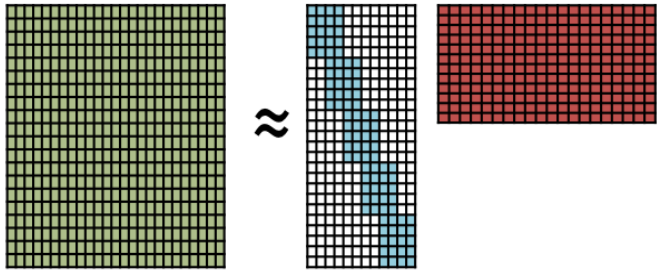
Feature learning



L

Spatially local features

Reduced format

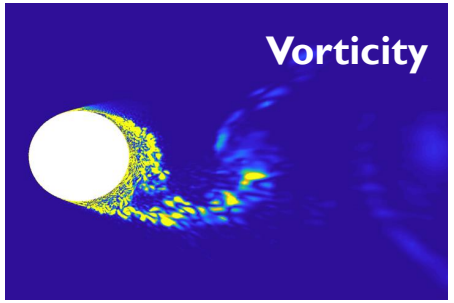


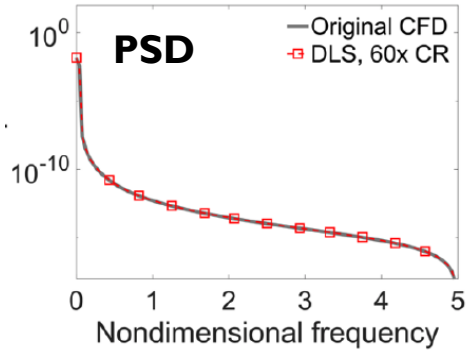
Highly sparse, generalized, space-time decomposition: Data-informed Local Subspaces (DLS)

Query/Analysis

- The reduced data format consists of a spatial and temporal separation of variables

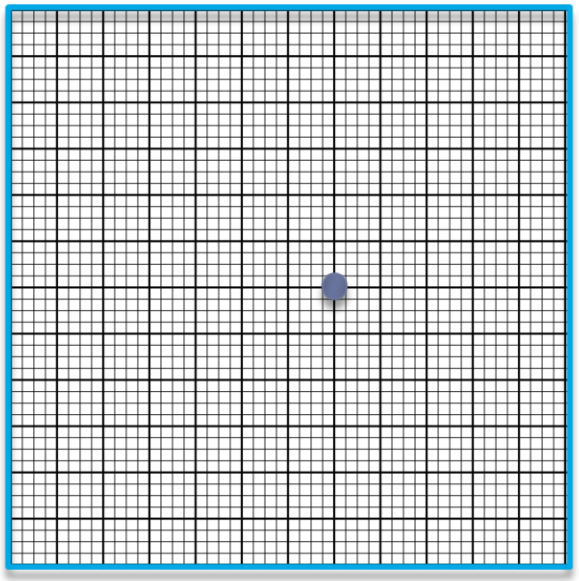
- Efficiency from:
  - Reduced data sizes (20-1000x)
  - Batch processing of basis and coeffs
  - Ease of accessing data subsets





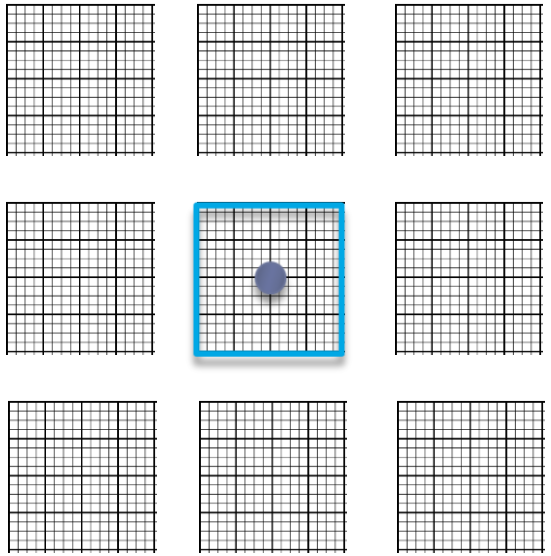
RA 1: Subset Extraction

Basic



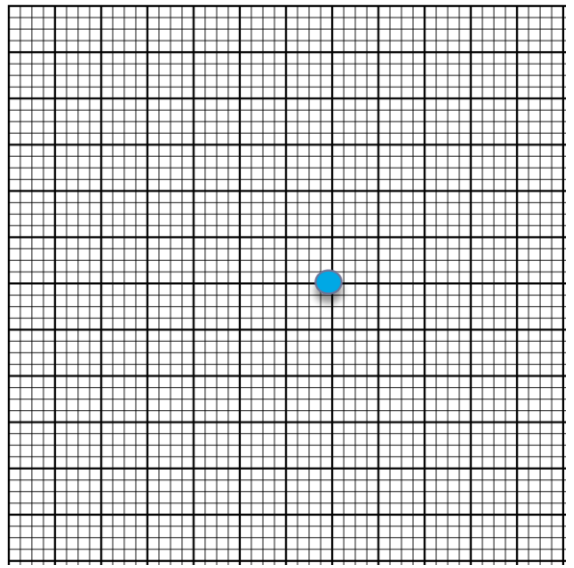
- Decompress the entire solution
- Full dataset must fit in memory
- Not feasible for high fidelity data

HDF5



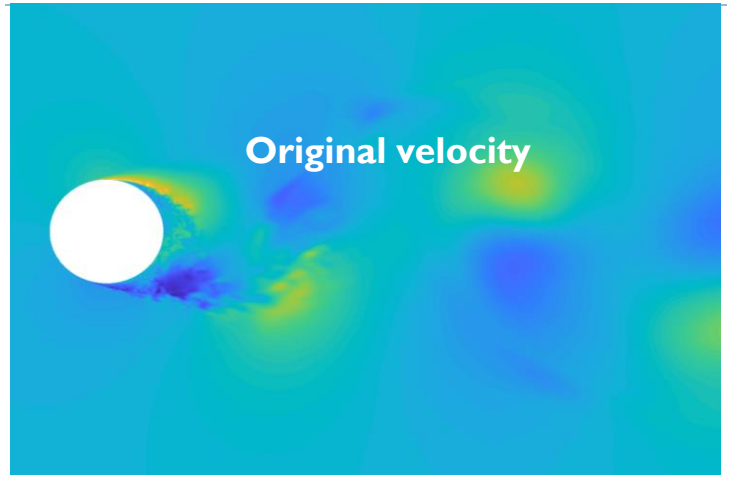
- Decompress the "chunk"
- Chunk size is defined at compression
- Chunk cache maximizes performance

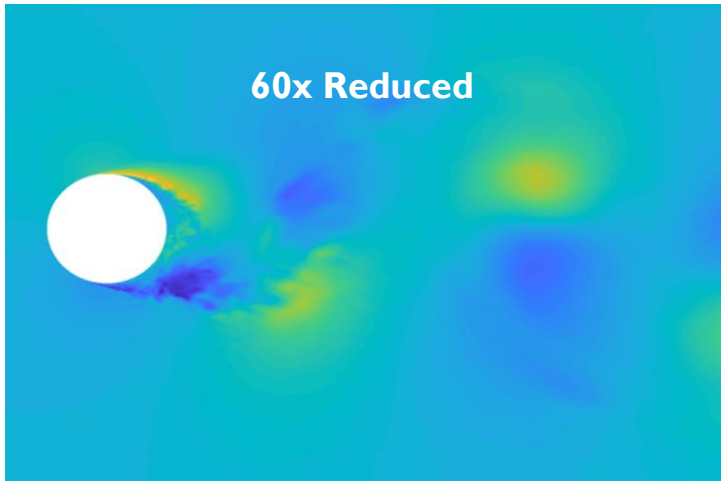
DLS

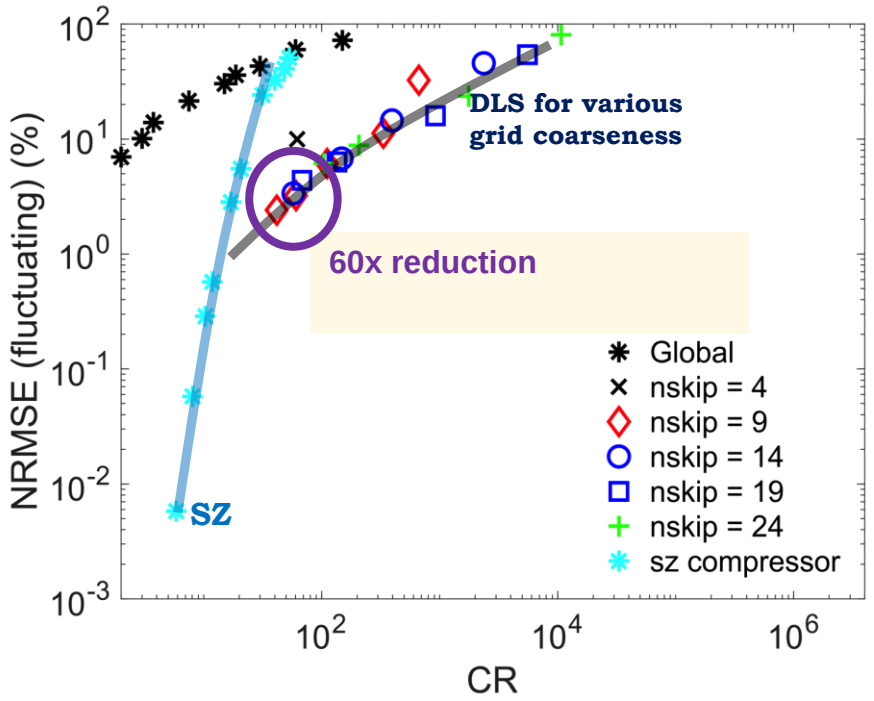


- Decompress the Singular Point
- Low memory footprint
- Can leverage HDF5 for more compression

Compression: Flow past Cylinder



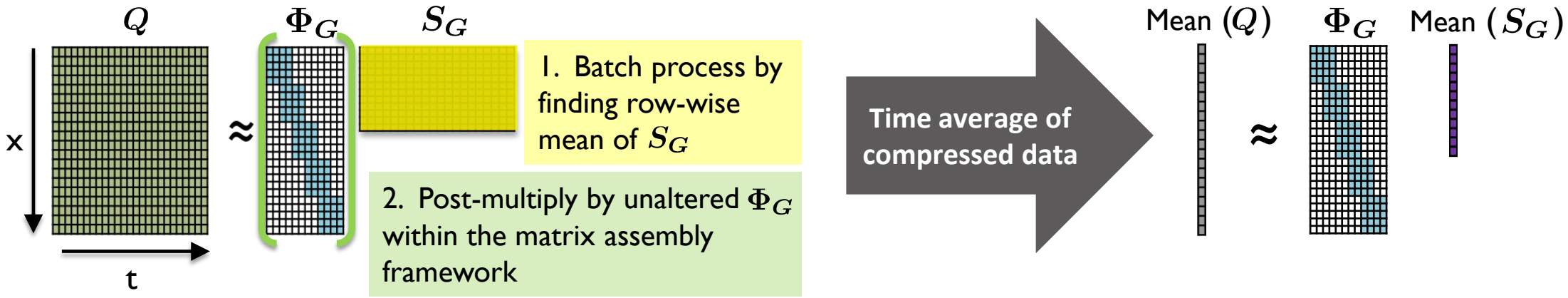


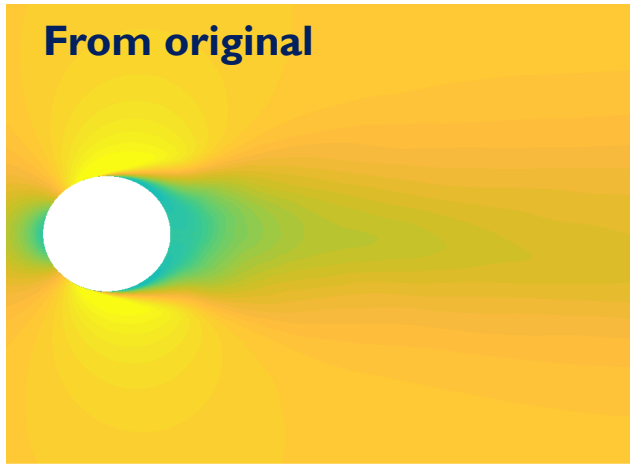


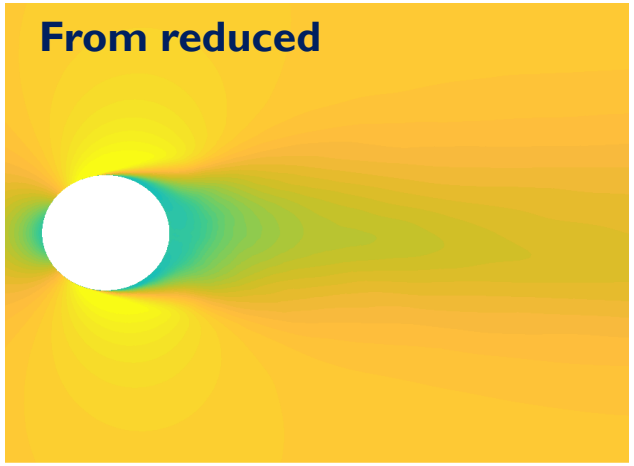
A range of significant data reduction and cost savings can be obtained. The level of savings depends on the user's tolerance for error and analysis type.

RA 1: Leverage Spatial Invariance

- Batch processing on the temporal component. Example: **Time Average**



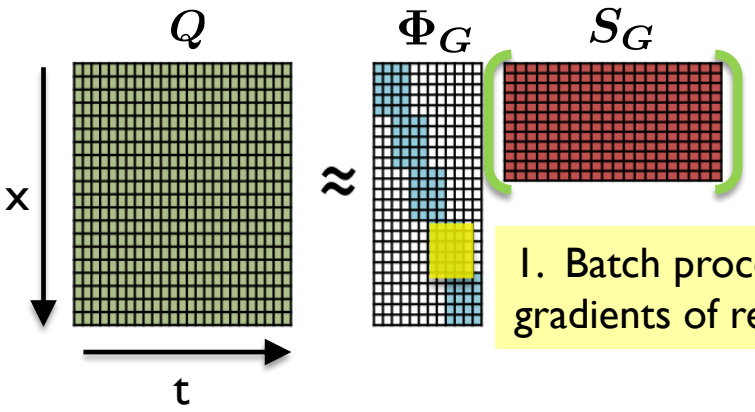




- Analysis on 60x compressed data
- Roughly 25x speedup in time-average calculation over 3000 instances
- Less than 0.1% NRMSE

RA 2: Leverage Temporal Invariance

- Batch processing on the spatial component



2. Post-multiply by unaltered S_G

1. Batch process by finding gradients of repeating Φ_L block

Vorticity of compressed time-history data

