MemSens: Significantly Reducing Memory Overhead in Adjoint Sensitivity Analysis Using Novel Error-Bounded Lossy Compression

Chenxi Li[1] Yihang Feng[1] Fuxing Deng[1] Dingwen Tao[2] Weifeng Liu[2] Zhou Jin[3]

1. SSSLab, Dept. of CST, China University of Petroleum-Beijing, Beijing, China
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
3. College of Integrated Circuits, Zhejiang University, Hangzhou, China

Email: z.jin@zju.edu.cn

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

OUTLINE

1. Background
2. MemSens
3. Experiment
4. Conclusions

OUTLINE

1. Background
2. MemSens
3. Experiment
4. Conclusions

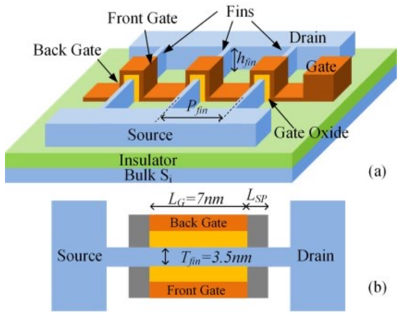
Background — Sensitivity Analysis

Transient sensitivity analysis calculates the sensitivity (or gradient) of any target function of the solution with respect to given parameters. It plays a vital role in various domains, including circuit optimization , performance modeling, and yield estimation.

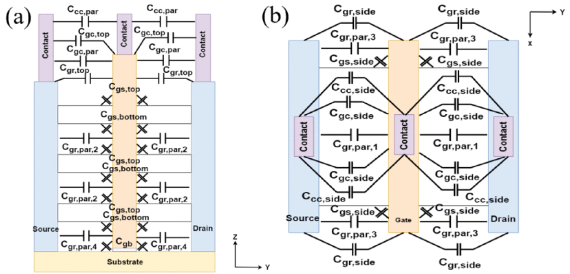
Background — Sensitivity Analysis

Transient sensitivity analysis calculates the sensitivity (or gradient) of any target function of the solution with respect to given parameters. It plays a vital role in various domains, including circuit optimization , performance modeling, and yield estimation.

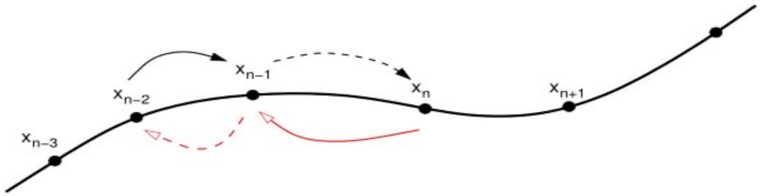
- The performance of a circuit is influenced by numerous critical parameters, such as transistor dimensions, parasitic resistances, capacitances, and more. Sensitivity analysis is a valuable tool for examining the impact of these factors on system output.
- The conventional direct method falls short when dealing with a large number of parameters; as a result, the adjoint method has become the standard in modern circuit simulations.



Transistor Dimension



Parasitic Resistances and Capacitances

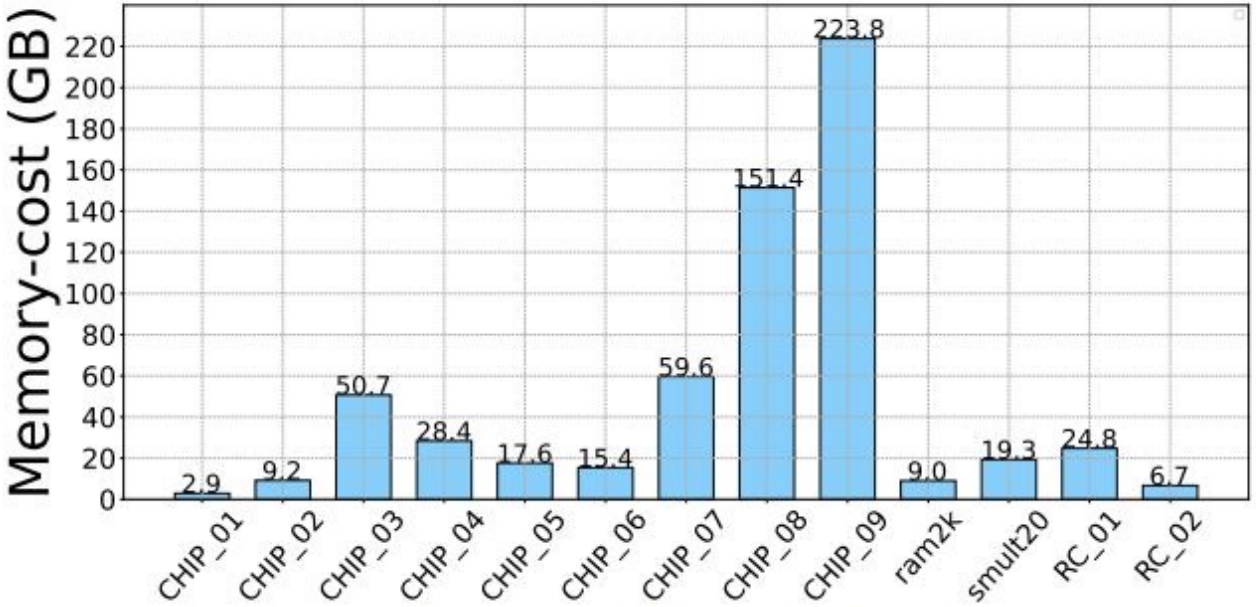


Adjoint System and Original System

Background — Massive Memory Overhead

A major drawback of the adjoint method is its high memory overhead.

- ➢ During the forward solution process, it is necessary to store critical historical information at each time step, such as the state vectors and matrices, to construct differential equations during the adjoint process.
- ✓ In large-scale simulations, this approach incurs substantial spatial and temporal costs.



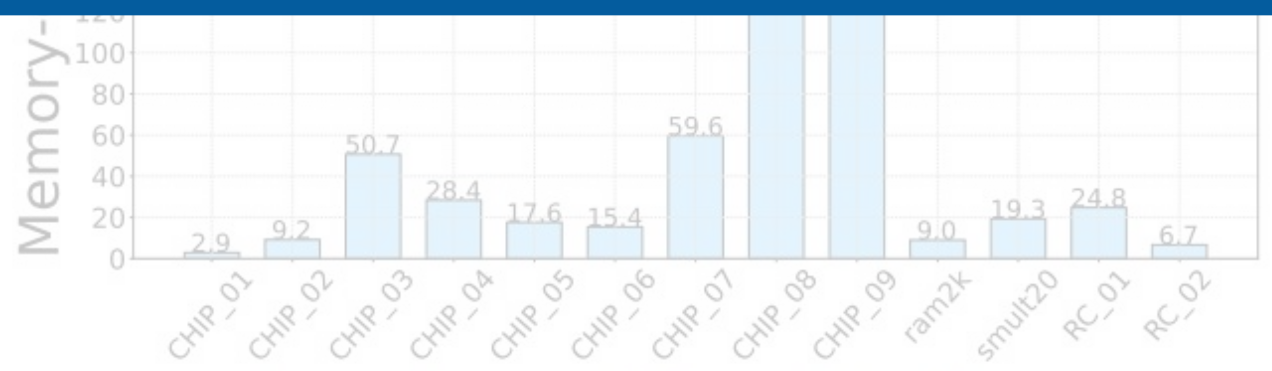
Memory cost of storing the history Jacobian matrices

Background — Massive Memory Overhead

A major drawback of the adjoint method is its high memory overhead.

- During the forward solution process, it is necessary to store critical historical information at each time step, such as the state vectors, to construct differential equations during the adjoint process.

A potential approach to reduce memory overhead is to use advanced data compression techniques.



Memory cost of storing the history Jacobian matrices

Background — Lossy Compression

Error-bounded lossy compression, a current research hotspot in the field of high-performance computing (HPC), can significantly reduce data size while ensuring accuracy. It is widely regarded as the best solution for addressing the challenges posed by large-scale scientific data.

- Traditional lossless compression algorithms are less effective for domain-specific data due to low compression ratios.
- Lossy methods offer compression ratios 1 to 2 orders of magnitude higher than lossless ones.

Dataset	Lossless Compression			Lossy Compression under 1e-2 Relative Error			
	GZIP	ZSTD	BZIP2	ZFP [17]	FPZIP [16]	SZ1.4[15]	SZ2.1[15]
add20(vec)	5.23	6.03	4.33	8.52	189.8	1824.81	1783.85
add20(mat)	2.52	9.74	7.19	14.6	52.51	509.418	214.24
mem_plus(vec)	2.57	2.08	3.45	7.34	67.27	1933.368	1515.333

Background — Existing Work

Data-prediction-based compression model

[1]P. Lindstrom and M. Isenburg, “Fast and Efficient Compression of Floating-Point Data,” IEEE Transactions on Visualization and Computer Graphics, 2006.

[2]S. Di and F. Cappello, "Fast Error-Bounded Lossy HPC Data Compression with SZ," IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016.

[3]P. Liakos, K. Papakonstantinopoulou, and Y. Kotidis, “Chimp: Efficient Lossless Floating Point Compression for Time Series Databases,” Proceedings of the VLDB Endowment, 2022.

Domain-transform-based compression model

[1]P. Lindstrom, “Fixed-rate Compressed Floating-point Arrays,” IEEE Transactions on Visualization and Computer Graphics, 2014.

[2]R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, “TTHRESH: Tensor Compression for Multidimensional Visual Data,” IEEE Transactions on Visualization and Computer Graphics, 2019.

[3]N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, “Exploration of Lossy Compression for Application-level Checkpoint/Restart,” IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2015.

Background — Existing Work

Data-prediction-based compression model

[1]P. Lindstrom and M. Isenburg, “Fast and Efficient Compression of Floating-Point Data,” IEEE Transactions on Visualization and Computer Graphics, 2006.

[2]S. Di and F. Cappello, "Fast Error-Bounded Lossy HPC Data Compression with SZ," IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016.

[3] … nt

Co …

Do …

[1] … d

Co …

[2]R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, “TTHRESH: Tensor Compression for Multidimensional Visual Data,” IEEE Transactions on Visualization and Computer Graphics, 2019.

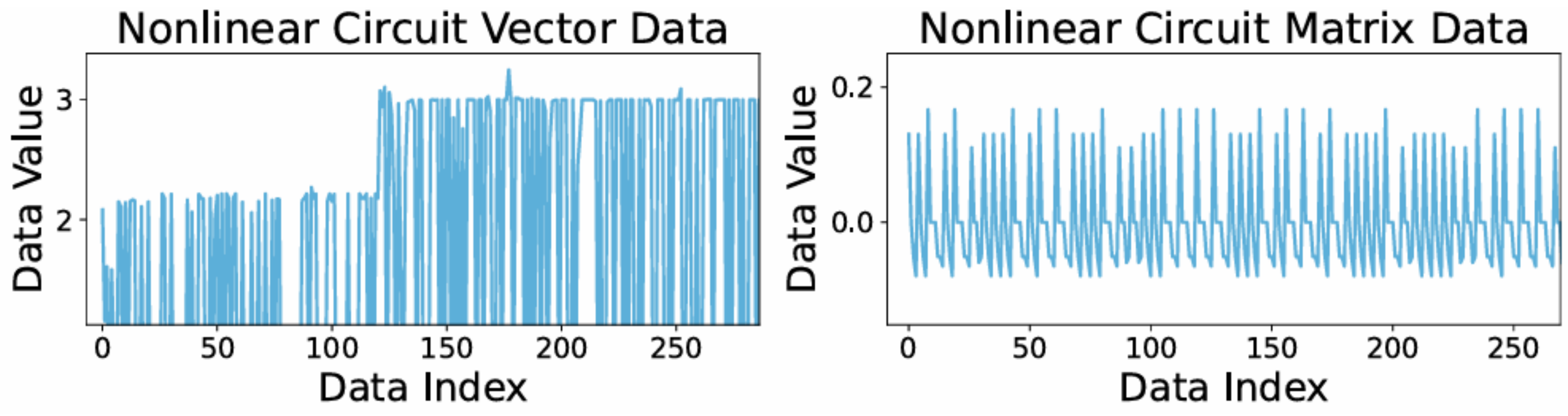
[3]N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, “Exploration of Lossy Compression for Application-level Checkpoint/Restart,” IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2015.

Not fully leveraging the data characteristics in circuit simulation renders them unsuitable.

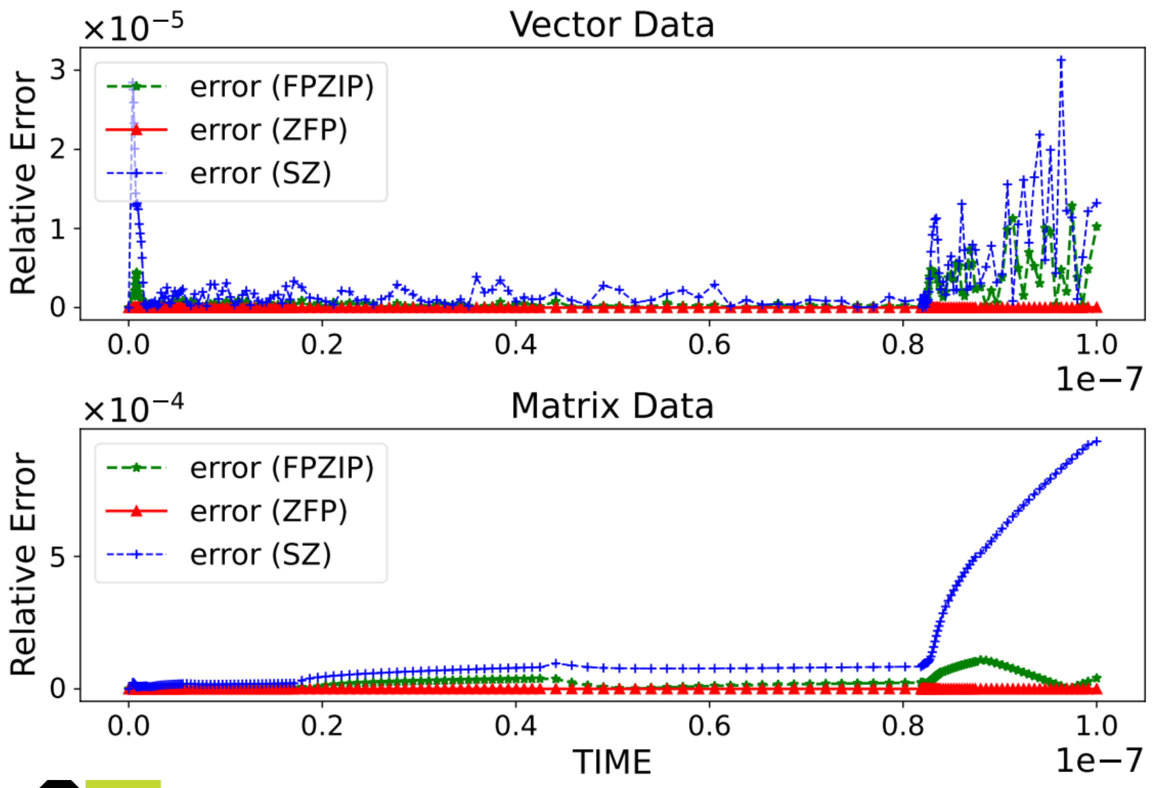
Background — Key Challenges

The matrices and vectors generated by circuit simulation at each time step are spiky data with very low spatial autocorrelation. Existing compressors mostly rely on the smoothness of the data, so their performance will be greatly reduced on circuit simulation datasets.

- ➢ For example, ZFP, its domain transformation will lose effectiveness on simulation data, thus leading to low compression ratio.
- ✓ Therefore, enhancing the smoothness of data in the spatial dimension is essential for simulation data compression.



Background — Key Challenges



Circuit simulation, particularly during sensitivity analysis, requires high precision.

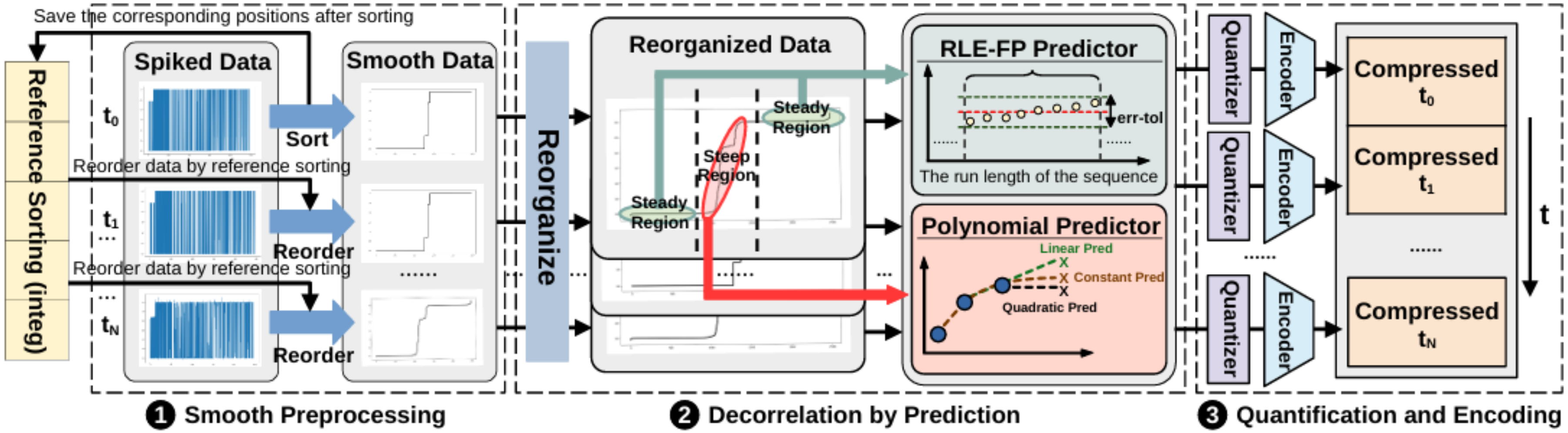
- ➢ While lossy compression algorithms like SZ improve compression ratios, they introduce significant errors that accumulate over time, compromising simulation accuracy. Moreover, the varying error levels across different lossy compression methods underscore the challenge of maintaining data integrity.
- ✓ Therefore, there is an urgent need to develop lossy compression algorithms that reduce data size while preserving the precision necessary for reliable simulations.

OUTLINE

1. Background
2. MemSens
3. Experiment
4. Conclusions

MemSens — Workflow

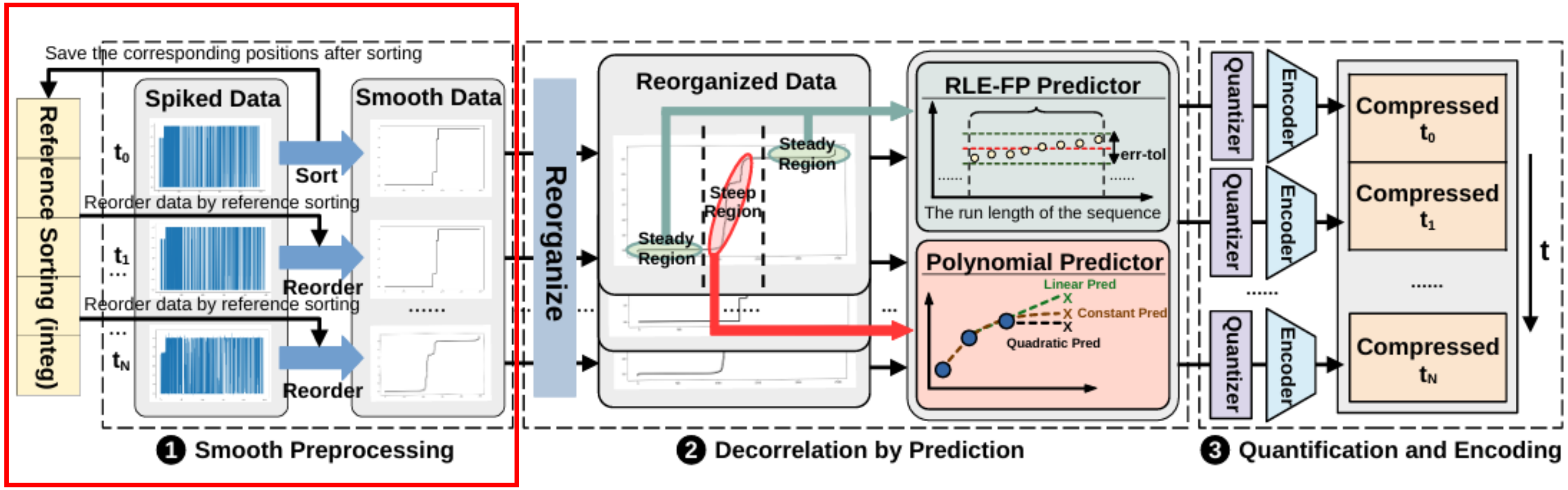
The workflow of the proposed compression algorithm is mainly composed of the following three steps:



MemSens — Workflow

The workflow of the proposed compression algorithm is mainly composed of the following three steps:

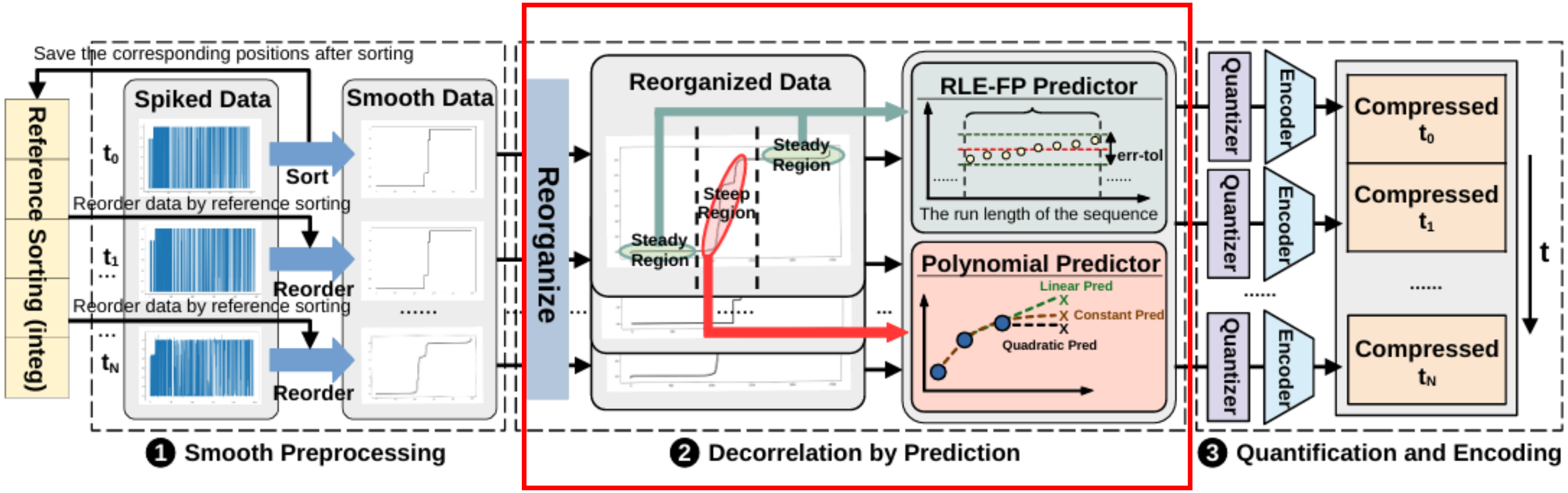
- ➢ efficiently smooth the spiky simulation data using reference sorting;



MemSens — Workflow

The workflow of the proposed compression algorithm is mainly composed of the following three steps:

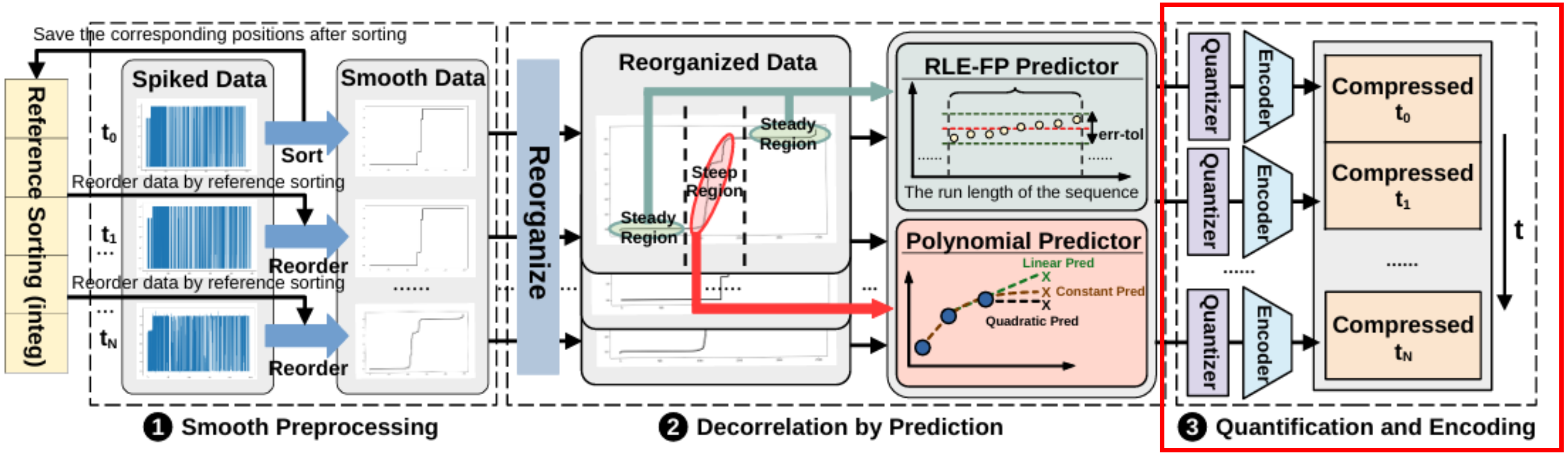
- efficiently smooth the spiky simulation data using reference sorting;
- apply RLE-FP or polynomial interpolation predictors to decorrelate data based on regional characteristics;



MemSens — Workflow

The workflow of the proposed compression algorithm is mainly composed of the following three steps:

- efficiently smooth the spiky simulation data using reference sorting;
- apply RLE-FP or polynomial interpolation predictors to decorrelate data based on regional characteristics;
- quantize floating-point data within a user specified error bound and encode it, achieving a significant data reduction.



MemSens — Data Smoothing

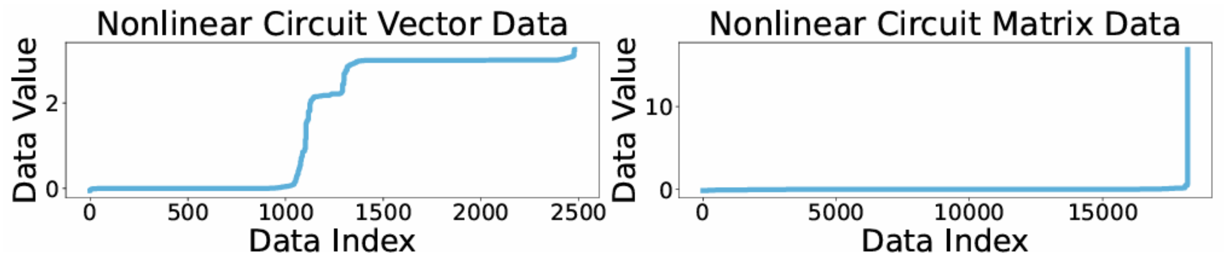
Data smoothing consists of three steps:

- sort the floating-point queue in an ascending order at the initial time step, record the sorting indices, and use this sorting as a reference;
- reorder the data in subsequent time steps based on the reference sorting;
- for longer integration processes, set validity check points to reinsert invalid data points into the sorted queue and update the reference sorting.

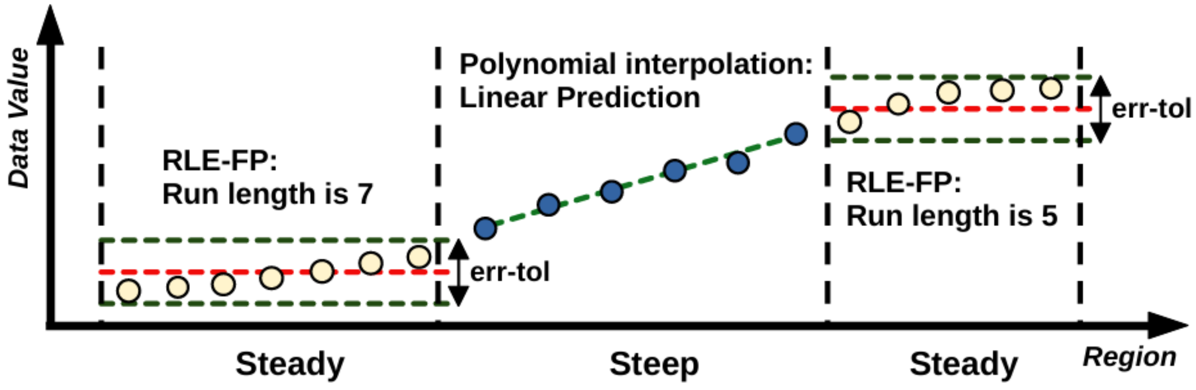
This approach is based on three key observations:

- ✓ in circuits, nodes with large admittance (or small impedance) connections typically have identical or similar potentials. Sorting the state vector groups these corresponding floating-point values together.
- ✓ for matrices, elements with similar parameters contribute stamping values that are also similar, and sorting naturally clusters these corresponding floating-point values.
- ✓ in time series data, values at adjacent time points are often similar or identical, leading to overall similarity in sorting across time steps.

MemSens — Hybrid Predictors



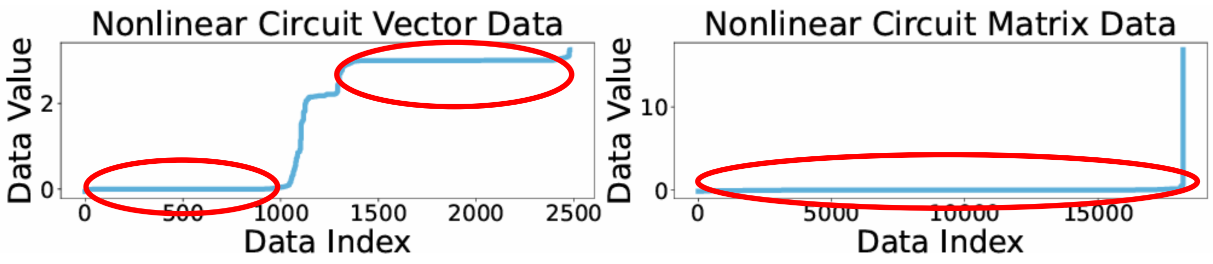
Smoothed simulation data



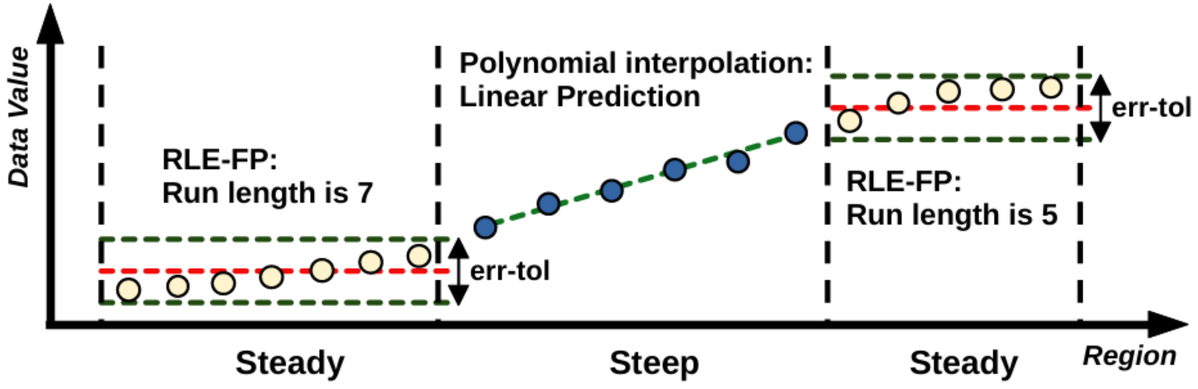
Hybrid Predictor

- After smoothing the simulation data, the compressor applies a hybrid predictor to decorrelate the dataset.

MemSens — Hybrid Predictors



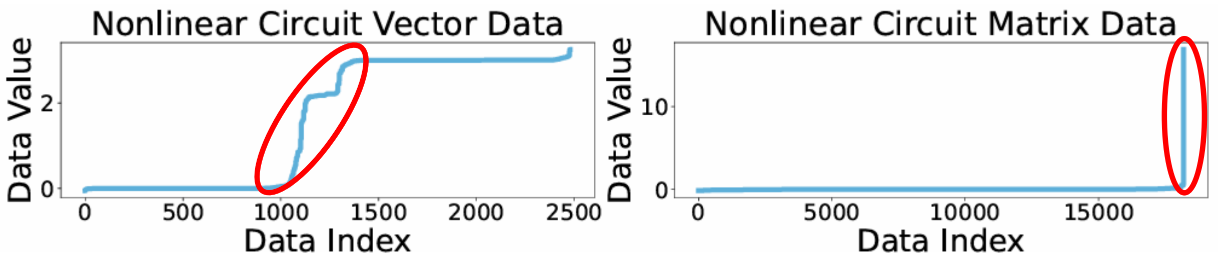
Smoothed simulation data



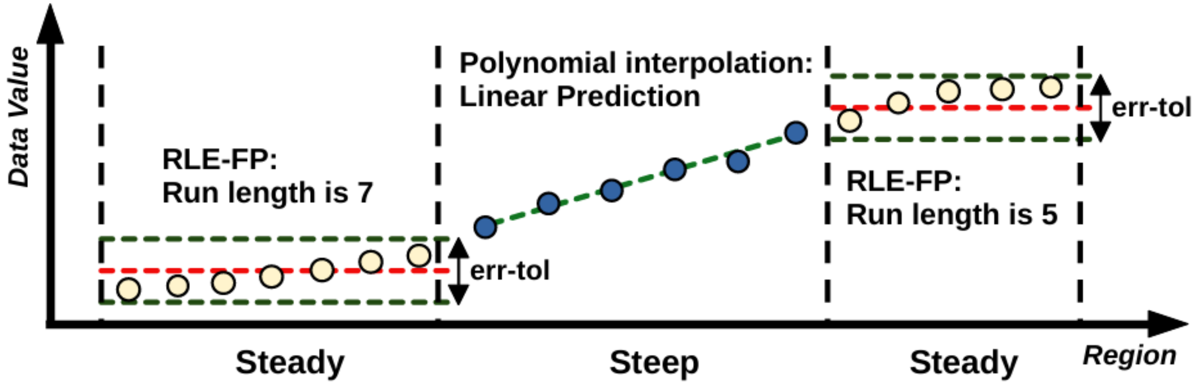
Hybrid Predictor

- After smoothing the simulation data, the compressor applies a hybrid predictor to decorrelate the dataset.
- The smoothed dataset contains both long numerical stable regions and short regions with sharp numerical increases.

MemSens — Hybrid Predictors



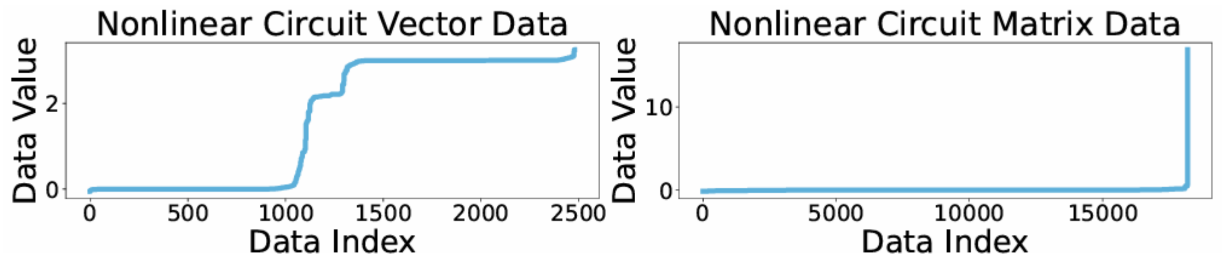
Smoothed simulation data



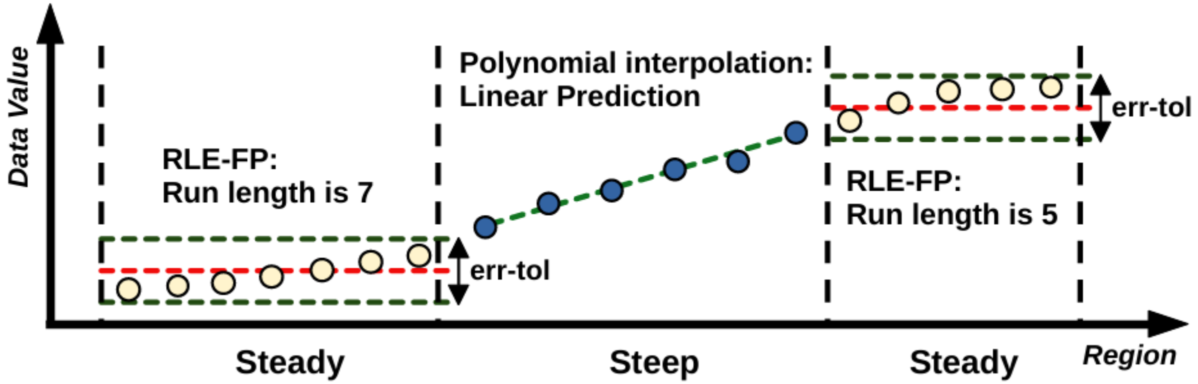
Hybrid Predictor

- After smoothing the simulation data, the compressor applies a hybrid predictor to decorrelate the dataset.
- The smoothed dataset contains both long numerical stable regions and short regions with sharp numerical increases.

MemSens — Hybrid Predictors



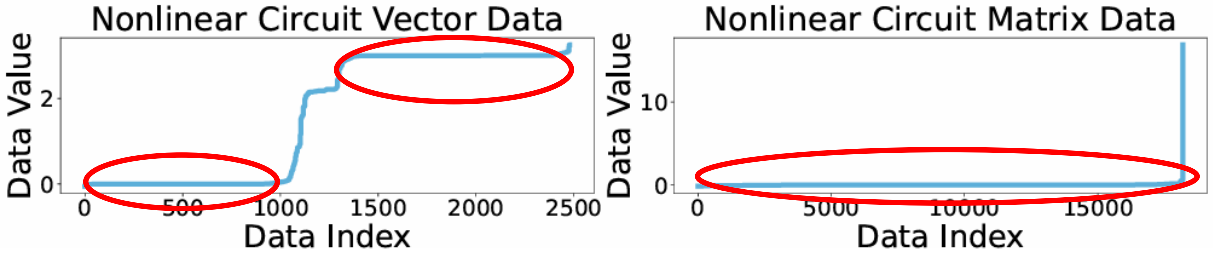
Smoothed simulation data



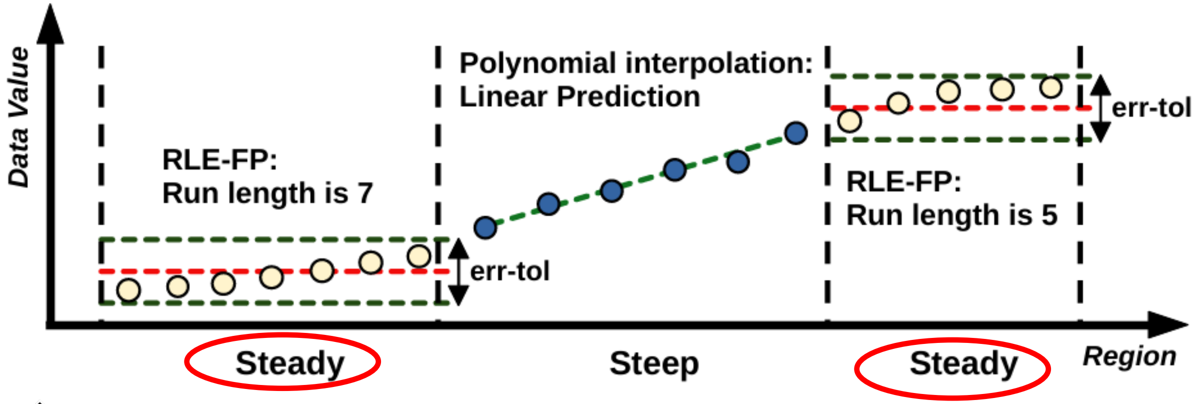
Hybrid Predictor

- The compressor organizes the entire dataset into different regions (or blocks) and applies different data prediction techniques according to the data characteristics within each region.

MemSens — Hybrid Predictors



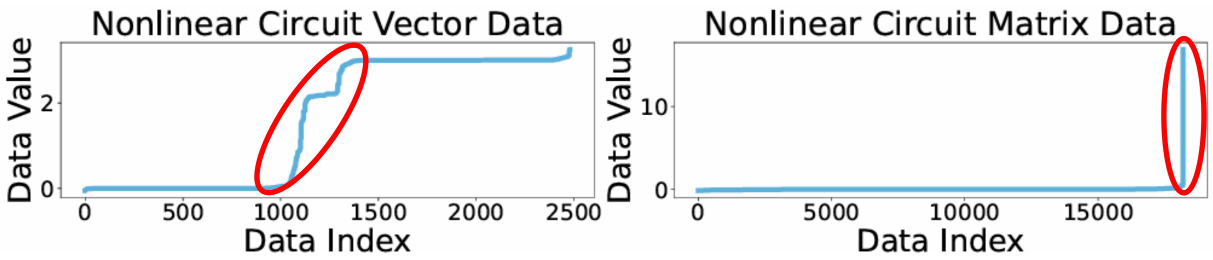
Smoothed simulation data



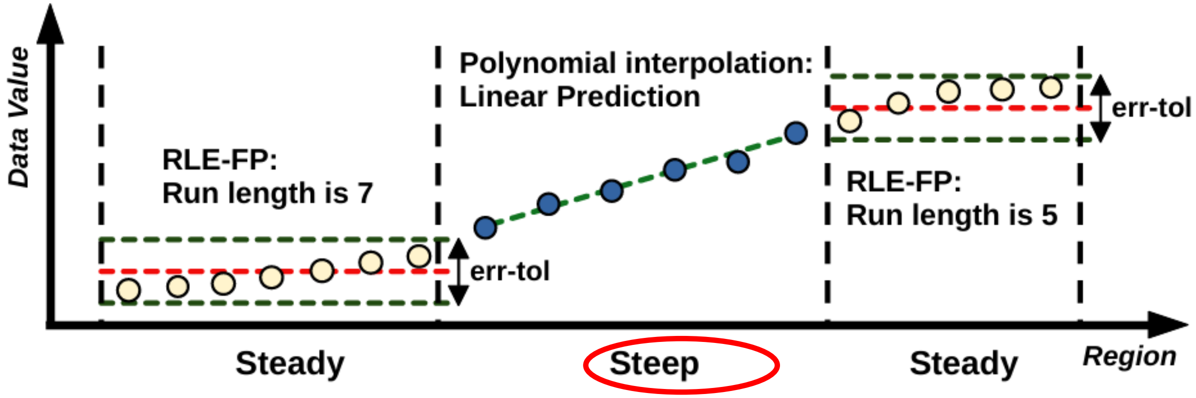
Hybrid Predictor

- The compressor organizes the entire dataset into different regions (or blocks) and applies different data prediction techniques according to the data characteristics within each region.
  - ✓ In the stable regions, this work introduces the RLE-FP technique to effectively decorrelate the data. Its core idea is to record the length of floating-point sequences that remain within the same error range consecutively.

MemSens — Hybrid Predictors



Smoothed simulation data



Hybrid Predictor

- The compressor organizes the entire dataset into different regions (or blocks) and applies different data prediction techniques according to the data characteristics within each region.
  - ✓ In the stable regions, this work introduces the RLE-FP technique to effectively decorrelate the data. Its core idea is to record the length of floating-point sequences that remain within the same error range consecutively.
  - ✓ In the sharply changing regions, this work employs a polynomial interpolation predictor to effectively decorrelate the data. The polynomial interpolation predictor is a prediction technique based on polynomial functions, which predicts future values according to historical points.

MemSens — Error Control Quantization

Error boundary setting:

We provide both absolute error bounds (denoted as ϵ) and relative error bounds (denoted as δ). To ensure simulation accuracy, both of these error bounds must be satisfied simultaneously. Therefore, there is a requirement between the true value V and the predicted value $\hat{V}$ (which is also the decompressed value):

$$\left|V - \hat{V}\right| \le \epsilon \quad and \quad \frac{\left|V - \hat{V}\right|}{|V|} \le \delta$$

So the final error bound e specified by the user is:

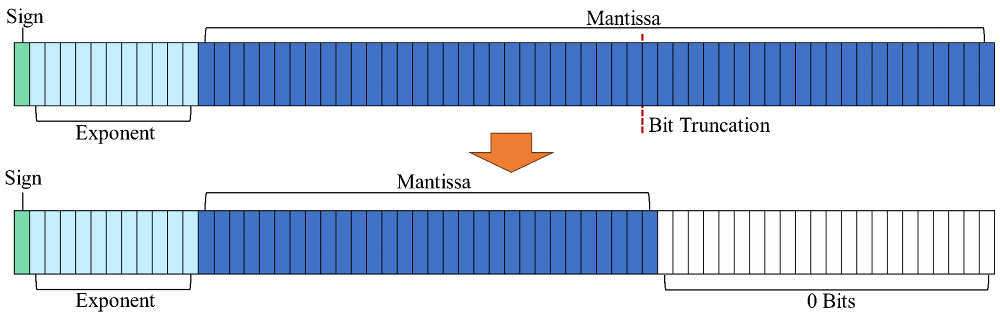
$$e = min(\epsilon, \delta|V|)$$

Clearly, when $|V| < \frac{\epsilon}{\delta}$, we focus solely on relative error; in contrast, we pay more attention to absolute error.

MemSens — Error Control Quantization

A quantizer is responsible for quantizing the error between predicted values and true values, as well as handling unpredictable values. It significantly reduces the entropy of the original data while respecting the user-specified error boundaries.

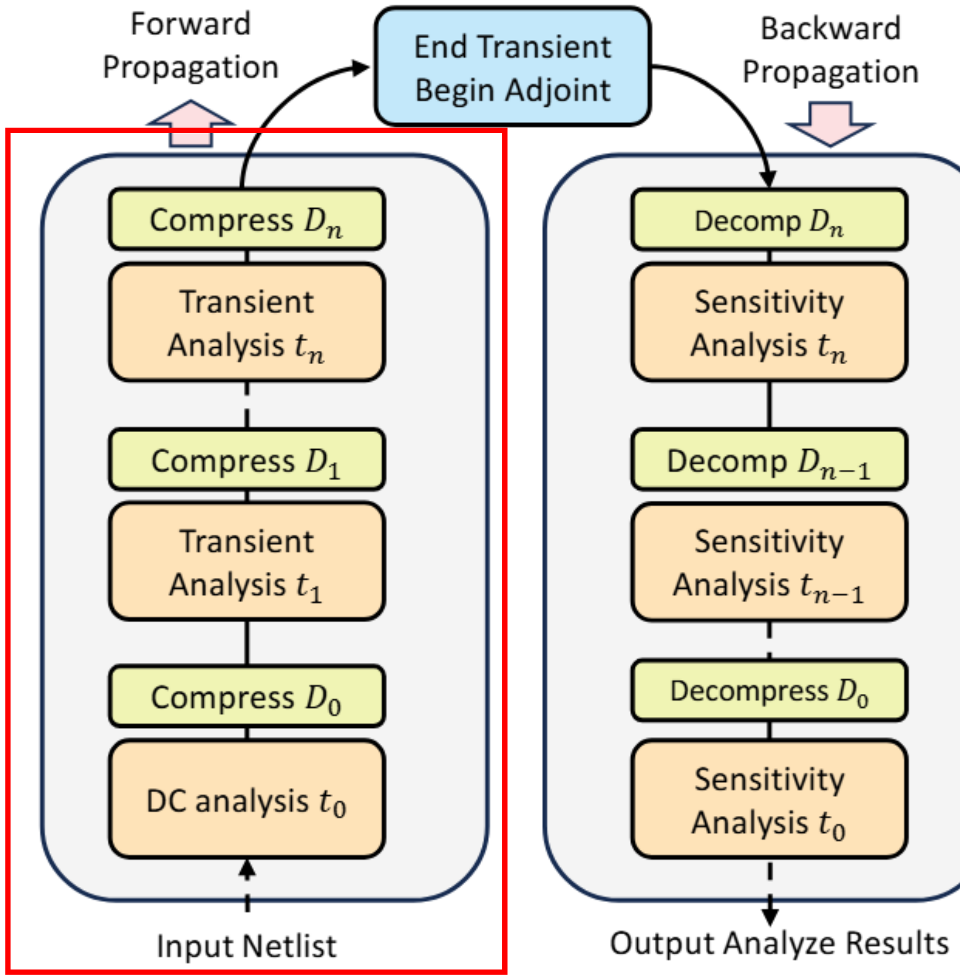
- Calculate the error between the predicted value and the true value.
- If the result falls within the error bound *e*, the quantizer replaces the true value with the predicted value.
- Otherwise, the quantizer treats this true value as an unpredictable value.
  - ✓ Truncate the insignificant bit planes of floating-point numbers based on absolute or relative error bounds.
  - ✓ Further reduce the data size based on XOR lossless floating-point compression.
- Finally, the data size can be further reduced by using the advanced lossless compressor ZSTD.



$$R_{abs} = \begin{cases} 0, p(V) - p(\epsilon) < 0 \\ 52, p(V) - p(\epsilon) > 52 \\ p(V) - p(\epsilon), otherwise \end{cases}$$

$$R_{rel} = -p(\delta)$$

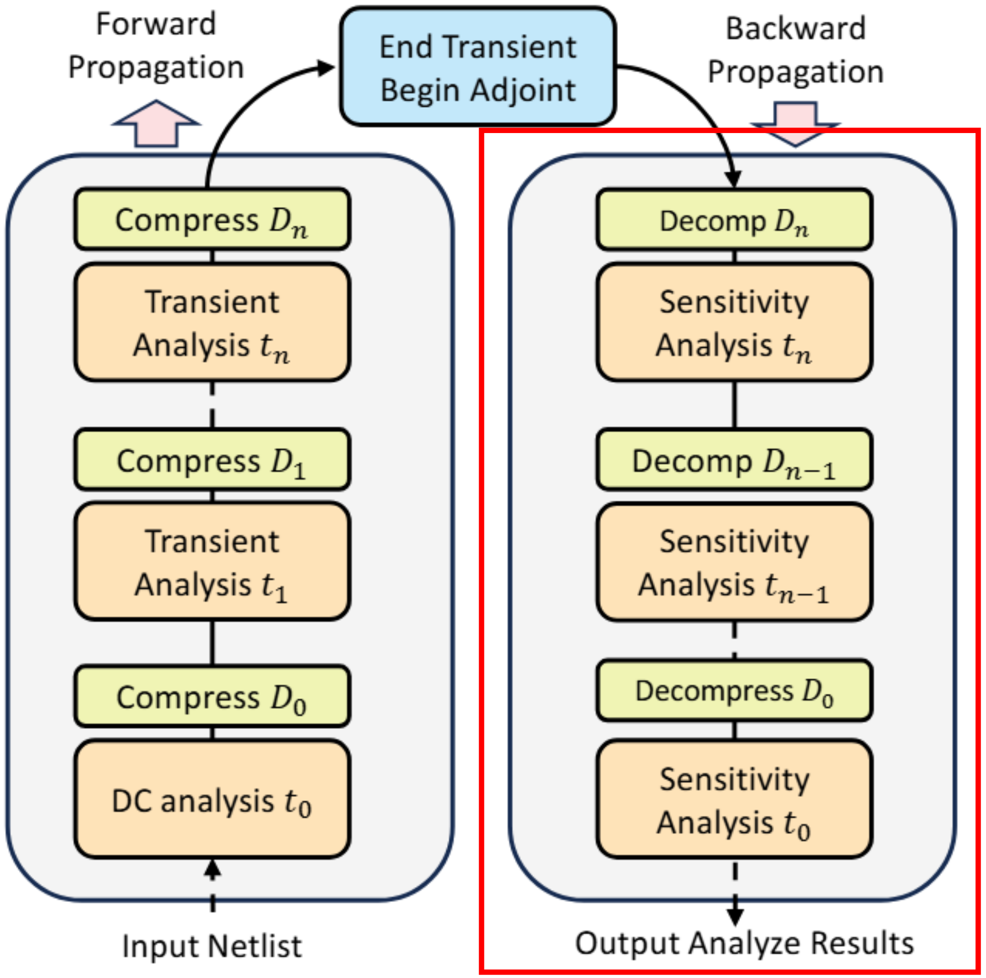
MemSens — Simulation Integration



We integrate the proposed error-bounded lossy compression algorithm into the simulation process, effectively reducing memory usage during both forward and backward propagation.

- In transient analysis, after successfully solving each time step, we store all the necessary state variables (e.g., state vectors and Jacobian matrices), denoted as D, and compress them to reduce storage.

MemSens — Simulation Integration



We integrate the proposed error-bounded lossy compression algorithm into the simulation process, effectively reducing memory usage during both forward and backward propagation.

- In transient analysis, after successfully solving each time step, we store all the necessary state variables (e.g., state vectors and Jacobian matrices), denoted as D, and compress them to reduce storage.
- Subsequently, during the adjoint process, these variables are decompressed at the necessary moments to reconstruct the differential equations.

OUTLINE

1. Background
2. MemSens
3. Experiment
4. Conclusions

Experiment — Configurations and Setup

Platform and software. We implement our proposed compression algorithm in C++ and integrate it into the adjoint sensitivity simulation. The platform used is Xyce, an open-source, SPICE-compatible circuit simulator from Sandia National Laboratories. The experiments are conducted on an AMD Ryzen 7 4800H CPU operating at a clock speed of 2.9 GHz.

Datasets. We evaluate the proposed error-bounded lossy compression algorithm on six datasets, consisting of vector and matrix data generated from simulations of both linear and nonlinear circuits. As shown in the table, the first column represents the name of the circuit and whether it is vector data or matrix data, the second column is the type of the circuit, the third column indicates the number of elements in the circuit, and the fourth and fifth columns respectively represent the dimensions and size of the data.

Dataset	Circuit Type	#CirElem	Dimension Size	Data Size
ibm1t (vec)	RLC	76934	54265×1252	518.34MB
ibm1t (mat)	RLC	76934	175022×1252	1.6GB
add20 (vec)	MOS	5091	2479×42799	809.47MB
add20 (mat)	MOS	5091	18189×42799	5.8GB
smult20 (vec)	MOS	46075	28759×9048	1.94GB
smult20 (mat)	MOS	46075	213185×9048	14.37GB

Experiment — Configurations and Setup

Configuration. In scientific computing, the absolute and relative error bounds for lossy compression are typically set to 10^{-5} and 10^{-3}, respectively, providing sufficient accuracy while significantly reducing simulation memory overhead. We adopt this configuration for our experiment as well.

Baselines. We compare the proposed algorithm with the state-of-the-art lossless compressor ZSTD, the lossless compression algorithm MASC specifically designed for matrices, and two cutting-edge lossy compressors, ZFP and SZ. All algorithms use the latest versions.

ZSTD: ZHENG L, WU Y, ZHU M, et al. Design and optimization of Zstandard algorithm based on concurrent streaming of multiple hash tables[C] // Proceedings of the International Conference on Laser, Optics and Optoelectronic Technology. Bellingham: SPIE, 2022.

MASC: LI C, ZHANG B, DUAN Y, LI Y, YE Z, LIU W, TAO D, JIN Z. MASC: A Memory-Efficient Adjoint Sensitivity Analysis through Compression Using Novel Spatiotemporal Prediction[C]// Proceedings of the 61th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2024.

ZFP: LINDSTROM P. Fixed-rate compressed floating-point arrays[J]. IEEE Transactions on Visualization and Computer Graphics, 2014, 20(12).

SZ: DI S, CAPPELLO F. Fast error-bounded lossy HPC data compression with SZ[C]// Proceedings of the IEEE International Parallel and Distributed Processing Symposium. Piscataway: IEEE, 2016.

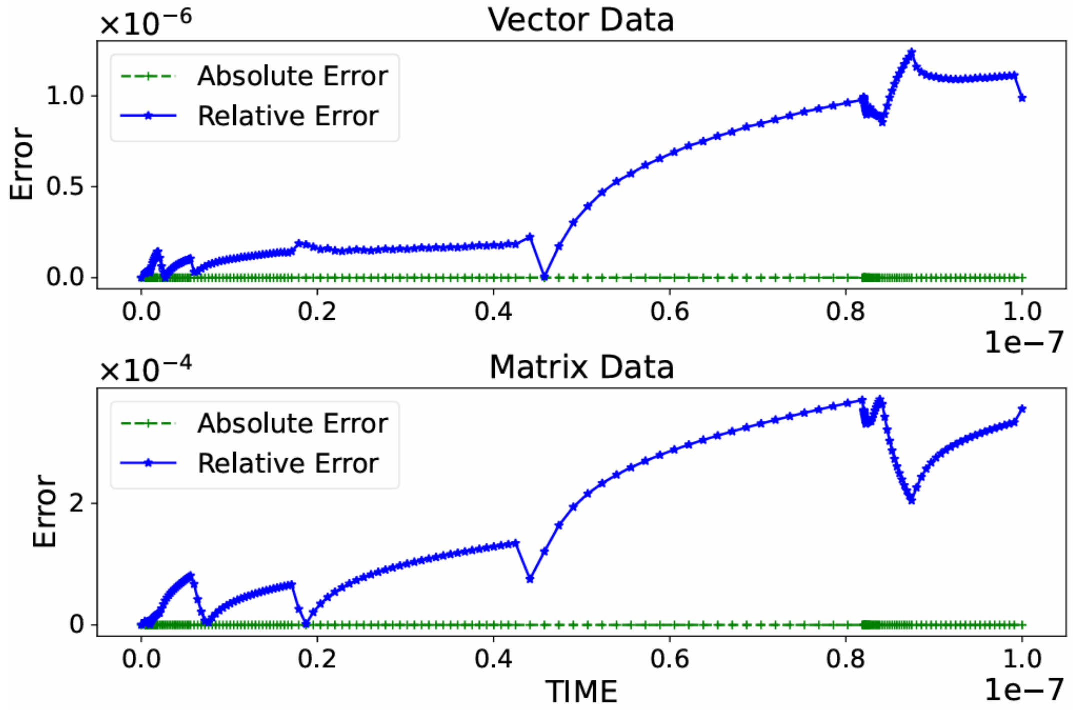
Experiment — Compression Performance

We compare the proposed error-bounded lossy compression algorithm with the aforementioned compression methods.

- In terms of compression ratio, our algorithm achieves average ratios of 23.70x and 39.07x higher than the state-of-the-art lossless compression methods ZSTD and MASC, respectively.
- When compared to advanced lossy compression methods such as ZFP and SZ, the proposed algorithm achieves compression ratios of 36.24x and 2.44x higher, respectively.
  - ✓ This improvement can be attributed to the algorithm's effective exploitation of the characteristics of circuit simulation data.

Dataset	ZSTD [26]			MASC [7]			ZFP [17]			SZ [15]			Our work		
	CR	T_{comp}	T_{decomp}	CR	T_{comp}	T_{decomp}	CR	T_{comp}	T_{decomp}	CR	T_{comp}	T_{decomp}	CR	T_{comp}	T_{decomp}
ibm1t(vec)	1.45	4.49	1.85	1.98	5.23	2.72	4.08	1.3	1.38	11.6	2.02	1.4	12.78	1.97	1.34
ibm1t(mat)	29.59	1.73	1.67	7.17	11.35	8.61	3.54	3.62	0.65	196.23	5.45	2.24	970.11	5.29	1.85
add20(vec)	6.03	3.63	2.09	2.95	6.79	3.97	5.02	1.79	1.77	76.08	2.67	1.33	72.28	2.76	1.45
add20(mat)	9.74	10.68	4.92	11.90	36.72	29.28	4.48	12.6	12.72	80.46	18.6	8.52	125.85	18.01	5.52
smult20(vec)	5.69	4.85	3.51	4.18	10.65	7.84	8.19	4.16	3.25	92.56	7.89	6.56	146.39	4.18	3.27
smult20(mat)	12.18	25.23	15.72	11.07	86.51	75.06	16.97	29.82	18.68	169.25	50.28	25.72	205.63	45.42	18.17
Average	10.78	8.44	4.96	6.54	26.21	21.25	7.05	8.88	6.41	104.36	14.49	8.31	255.51	12.94	7.93

Experiment — Error Impact Evaluation



We integrate the proposed error-bounded lossy compression algorithm into the adjoint sensitivity simulation to analyze the impact of floating-point errors introduced by lossy compression on the accuracy of sensitivity simulation results. The absolute error bound is set to 10^{-5} and the relative error bound to 10^{-3}.

- ➢ It can be observed that the sensitivity analysis results at each time point remained within the predefined error bounds after integrating the lossy compression.
- ✓ This is attributed to the proposed algorithm's strict quantization of errors introduced during the compression process.

OUTLINE

1. Background
2. MemSens
3. Experiment
4. Conclusions

Conclusions

This work introduces the first error-bounded lossy compression algorithm capable of efficiently compressing both vector and matrix data in circuit simulations. This approach significantly reduces memory overhead in adjoint sensitivity simulations.

- Based on the characteristics of data in circuit simulations, we propose an effective smoothing method tailored for simulation data.
- Subsequently, we design different data prediction algorithms for regions with distinct properties.
- Finally, through rigorous error quantization and efficient data encoding, the proposed approach significantly reduces the data size while preserving accuracy.

Lossy compression is valuable for reducing the overhead of circuit simulation.

Thanks for listening

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY