Dear Editor,

Thank you so much for extending our time for the revision of the manuscript. We also like to thank both reviewers for their thorough review of the manuscript. Below, find the main modifications made to the manuscript which covers all the remarks.

- As suggested by the reviewers, the manuscript has been reworked to focus more on the application to CFOSAT and SWOT datasets. We have also moved the details concerning the synthetic datasets in the Supplement which is now provided with the paper.
- The term “Bit Grooming” was employed both for the absolute and the relative error bounded compression modes. But the algorithms are different. In the revised version of the paper we employ the term “Decimal Rounding” in the absolute error bounded mode and the term “Bit Grooming” in the relative absolute error bounded mode to remove any ambiguities.
- Thanks to a remark from C. Zender, we found that the Digit Rounding algorithm was sometimes too conservative. We thus slightly modified the implementation. The compression results reported in the new version of the paper have been obtained with the new version of the Digit Rounding algorithm.
- In the previous version of the paper, we cascaded a call to ncks with a call to h5repack to perform Bit Grooming followed by Zstandard compression. For fairer comparisons on the compression speed, we modified our approach and now only employ ncks tool to run Decimal Rounding and Bit Grooming. However, we could not call Zstandard compression via the ncks tool, but only Deflate compression. Consequently, we now provide results for Decimal Rounding, Bit Grooming, Sz and Digit Rounding followed by Deflate compression.
- The Digit Rounding software source code is now available from CNES GitHub at https://github.com/CNES/Digit_Rounding.
- We now provide a Supplement which details the datasets and provides the command lines used for running the compression tools.
- The grammar and English of the paper has been dramatically improved by reviews from native English speakers.

Reply to Anonymous Referee #1

We are grateful to the referee for her/his constructive and thorough criticism and suggestions to our manuscript. Please, find below a detailed point-by-point reply. *Referee's comments are in blue italic*; our answers are in black and our changes to the manuscript in green.

- This manuscript needs a lot of improvement in terms of the grammar and writing. There are many awkward phrases and incorrect word choices that need to be improved (a subset are listed below). The paragraph structures are also in need of modification (many paragraphs contain only 1 or 2 sentences).

We will improve grammar and writing of the manuscript by contacting a native English speaker/writer. Thank you for pointing out the subset of incorrect word choices. We will also modify the paragraph structures to avoid too small paragraphs.

The grammar and writing have been improved throughout the manuscript. Short paragraphs have also been modified.

-Section 2: I'd be helpful to include more detail for the preprocessing algorithms: shuffle and bitshuffle. Also this section in general needs improvement. It's a bit "choppy" to read (needs smoother and better transitions between topics) and feels like more details would be helpful on the methods (especially the ones that the digit rounding algorithm builds on).

More details will be added on the shuffle and bitshuffle algorithms. More details will also be added on the bit-grooming and decimal rounding algorithm, algorithm on which the digit rounding algorithm is built. We will do what is needed to improve this section in general.

Section 2 has been restructured so as to make the reading smoother. Details have been added on the shuffle and bitshuffle algorithms as well as on the bit-grooming and decimal rounding algorithm.

"The Shuffle algorithm groups all the data samples' first bytes together, all the second bytes together, etc. In smooth datasets, or datasets with highly correlated consecutive sample values, this rearrangement creates long runs of similar bytes, improving the dataset's compression. Bitshuffle extends the concept of Shuffle to the bit level by grouping together all the data samples' first bits, second bits, etc.

…

The Decimal Rounding algorithm achieves a uniform scalar quantization of the data. The quantization step is a power of 2 pre-computed so as to preserve a specific number of decimal digits. The Bit Grooming algorithm creates a bitmask to degrade the least significant bits of the mantissa of IEEE 754 floating-point data. Given a specified total number of significant digits, nsd, the Bit Grooming algorithm tabulates the number of mantissa bits that has to be preserved to guarantee the specified precision of nsd digits: to guarantee 1-6 digits of precision, Bit Grooming must retain 5, 8, 11, 15, 18, and 21 mantissa bits respectively. The advantage is that the number of mantissa bits that must be preserved is computed very quickly. The disadvantage is that this computation is not optimal. In many cases, more mantissa bits are preserved than strictly necessary."

-Section 4: Why does using the synthetic data in 4.1 to assess performance make sense - it seems unrelated to the application area of interest. I'd argue that the metrics used in 4.1 are really minimal requirements as well. Also take care when referring to "performance" as it is overloaded term...do you mean speed or effectiveness (it's used both ways)

The objective of using synthetic data was to control the data parameters, such as the SNR, to be able to assess the impact of these parameters on the compression ratios. The results are not reported in this paper which rather focuses on providing a comparison of the compression ratio and speed of different algorithms. It has also been chosen to present only the minimal set of relevant metrics to avoid overloading the paper. We will be more rigorous and replace the term “performance” by “compression ratio” or “compression speed” in the text.

All the occurrences of the term “performance” have been checked and corrected when needed. The Mean Absolute Error metric has been added to the list of metrics:

“mean absolute error $\overline{e_{abs}}$ to evaluate the mean data degradation. It is defined as the mean of the pointwise absolute difference between the original and compressed data:

$\overline{e_{abs}} = \frac{1}{N}\sum_{i=0}^{N-1}|s_i - \tilde{s}_i|$”

*-fpzip is a fast and effective lossless method that would have been nice to compare (I *think* there is an fpzip filter available). Also I believe that any hdf5 filter can be accessed through NetCDF4 (see last sentence in conclusion) - consider contacting the Unidata folks.*

Thank you for the suggestion. Indeed a HDF5 filter is accessible for fpzip. However, many lossless compression algorithms exist. In our paper, we chose to evaluate the most “popular”, i.e. the lossless compression algorithms the most used in applications. Thank you for pointing this evolution of the NetCDF-4 library: from version 4.6.0 - January 24, 2018, NetCDF fully supports HDF5 dynamic filters. The text of the paper will be modified so as to provide the example usage using the new NetCDF-4 features.

An example using the new NetCDF-4 features with *nccopy* tool has been provided in the supplement and the conclusion has been modified to remove the last sentence. However, nccopy tool does not allow yet linking together different filters.

-Comments on doing compression in parallel?

We do not consider running compression algorithm in parallel in this work and will make it clear in the manuscript. It is a possible extension of this study.

The following sentence has been added in section 4:

“Parallel compression has not been considered in this work.”

-When reading the conclusion, it’s hard to see what the main contributions of this paper are. It’s fairly well known already that preprocessing of scientific data (e.g., bit shuffle or shuffle) improves lossy compression. Also the statements in the conclusion aren’t specific to a particular type of data set, but are presented as more general conclusions.

Given that the effectiveness and performance (speed) of lossy and lossless compression are very data, application, and variable dependent, the general statements here are not well justified by the small sample of data in the paper. I’d suggest focusing the paper more heavily on the data in Section 5 (if it’s of interest) and tailoring the discussion in that manner. Or maybe the focus was to be more on speeds than quality, in which case it’s be important to work to get sz and fpzip working, particularly via netcdf-4...

Thank you for the suggestion that will help highlighting the main contributions of our work. As suggested, the paper will be reworked to focus more on the application to the CFOSAT and SWOT datasets. We will also avoid general statements but attach our conclusions to our application case.

The paper has been reworked to focus more on the application to the CFOSAT and SWOT datasets. Sz compression has been run on CFOSAT dataset and on some parts of SWOT datasets. The NetCDF-4 tool nccopy does not allow yet linking together different filters. This restrains its usability and this is why we prefer using h5repack tool.

——————————- Specific items: ——————————-

-p2, line 20: note that fpzip can also be lossless

Thank you for the remark. Fpzip will be presented both as a lossless and lossy compression algorithm.

The text has been modified as follows:

"Third, some lossy/lossless compression algorithms, such as FPZIP (Lindstrom and Isenburg, 2006), …"

-p. 8: discussion of figure 5: is the width of the bars related to the compression levels? (e.g. line 20 statement is unclear)

No. All the bars have the same width. Each vertical bar represents a compression level. For instance, the 9 compression levels of Deflate are represented by 9 vertical bars. This will be clarified in the text p.8.

Clarifications have been added to the text:

"The vertical bars represent the results for different compression levels: from 1 to 9 for the Deflate level dfl_lvl, from 1 to 22 for Zstandard level zstd_lvl, and only one level for LZ4."

-p.8, lines 28-29: Why is this the case? (Add some discussion beyond describing the figure.)

These lower compression/decompression speeds are not well understood and would require further investigation to be fully understood. It might be related to HDF5 chunking. Indeed, HDF5 split the data into chunks of small size that are independently compressed. This allows HDF5 to improve partial I/O for big datasets but can sometimes reduce the compression/decompression speeds. This discussion will be added to the text.

The following sentence has been added to the text:

"Further investigations are required to understand why the compression/decompression speeds are lower, but it might be related to HDF5 chunking."

-p.8, line 20: I feel like the parameters should be better explained for –filter so that the reader can try them more easily. For example, what does the "32017" mean? I think that the following 0 is for sz, but this is not stated either.

We will add the meaning of each parameters. Each HDF5 filter is identified by a unique ID. “32017” is the identifier of Sz filter. The following “0” is the number of filter parameters. In the case of Sz, the filter does not have any parameter to set. That is why there are 0 parameters. Sz compressor is configured via the sz.config file. The same explanations will be added for the other filters used in the paper.

All the command lines have been moved to the Supplement with the explanation above to avoid overloading the manuscript.

-p. 10, line 6: re: "experiments have shown" - whose or which experiments (cite?)

It is based on our own experiments that haven’t been published. The sentence will be reworked as follows: “We have found that Shuffle or Bitshuffle preprocessing do not increase the compression ratio when applied after Sz. We have also found that and Bitshuffle provide lower compression ratio than Shuffle when applied after Bit Grooming. That is why only Shuffle is applied after Bit Grooming.”

This sentence has been removed since the results have not been published.

-Table 5 : Why is the speed faster for 1d?

As previously, the lower compression/decompression speeds obtained with the dataset s3D are not well understood and might be related to HDF5 chunking. This discussion will be added to the text.

The following sentence has been added to the text:

“The lower compression/decompression speeds obtained with Sz on the dataset s3D are not well understood and might be related to HDF5 chunking as previously mentioned.”

-Section 4.4.1, line 21-24: Any idea why you get these results?

Sz performs better on smooth signals since it makes use of a prediction step. The signal s1 being highly noisy, Sz prediction might often fail. This can explain the lower compression ratio on the signal s1. On the contrary, Bit-grooming does not makes any prediction. This can explain why it achieves better compression than Sz on the signal s1. This hypothesis will be added to the text.

The following sentences have been added to the text:

“Sz may perform better on dataset $s3D$ because it is smoother than dataset $s1$. Indeed, Sz integrates a prediction step. This prediction might often fail because dataset $s1$ is very noisy. This may explain the lower compression ratio for this dataset. Decimal Rounding, however, does not make any predictions, which may explain why it achieves a better compression than Sz for dataset $s1$.”

-Section 4.4.1, last sentences: It’s unclear to me what the value of these synthetic data sets is - especially given the statement on p. 11, line 3, about the dependence on the dataset

As suggested previously, the paper will be reworked to focus more on the application to CFOSAT and SWOT datasets without drawing general conclusions based on the results obtained on the synthetic datasets.

We modified the last sentence of this section as follows:

"Both Sz and Bit Grooming algorithms seem valuable for compression in absolute error-bounded compression mode."

-page 11, line 9-10: I'd include characteristics of the data (e.g., maximum abs. value) earlier in the text when the two datasets are introduced.

Your suggestion will be taken into account: the characteristics of the data will be introduced in section 4.2.

We have added the characteristics of the data in section 4:

"Datasets $s1$ and $s3D$ were generated, $s1$ being a noisy sinusoid of 1 dimension with a maximum absolute value of 118. The data volume of the $s1$ dataset is 4MB. Dataset $s3D$ is a noisy sinusoid pulse of 3 dimensions with a maximum absolute value of 145. The data volume of the $s3D$ dataset is 512MB."

-page 11, line 24: I don't see relative error mentioned in Table 6 - it seems to just be absolute error

The text will be modified to make it clearer: "…all three algorithms respect the maximum absolute error of 0.5 which, for the signal s1, corresponds to a relative error of 0.00424."

The text has been modified as follows:

"… all three algorithms respect the maximum absolute error of 0.5, which corresponds for dataset s1 to a relative error of 0.00424."

-p.11-12: Need more of a discussion of the results in Figure 7. For 3D, it looks like bit grooming and digit rounding are similar - I don't see a clear advantage.

More discussion on the results will be added to the text. For the s3D you are right, there is no clear advantage. It is written "the Digit Rounding algorithm provides compression performance very closed to the one of the Bit Grooming algorithm".

The text has been modified as follows:

"All three algorithms provide similar SNR versus compression ratio results, with a slight advantage for the Bit Grooming algorithm."

-p.12, lines 16-17: SZ compression can be controlled with an absolute error bound, so why is the relative error bound adjusted to get the desired abs. error?

The objective was to see if Sz compression configured with a relative error bound respect the error bound specified. As the digit rounding and bit-grooming algorithm can only be configured on a number of significant digits, they can only “produce” absolute error in 0.5, 0.05, 0.005, etc. In order to be able to compare Sz configured with a relative error bound with those algorithms, we have configured the relative error bound to obtain a maximum absolute error of 0.5. These explanations will be added to the text.

The following sentence has been added in the text:

In order to be able to compare Sz configured with a relative error bound with those algorithms, we configured the relative error bound to obtain a maximum absolute error of 0.5: the pw_relBoundRatio parameter in Sz was set to 0.00424.

-Section 5.1: It is disappointing not to have SZ results on the real data of interest. Were the SZ authors contacted? I would think that they could have helped resolve this issue.

Yes, we had some exchanges. The issue is still under investigation.

A more recent version of Sz has been used and results on CFOSAT and SWOT datasets are now provided.

-p. 15, line 18: "which only a few attributes may be missing" - It’s unclear what this means. It’s super helpful to really detail the data being compressed so that one can make sense of the results.

Details on the datasets will be added to the text.

This part of the sentence has been removed has it is not relevant in the frame of this study. Details on the CFOSAT and SWOT datasets have been added in the Supplement, particularly the precision required for the compression of each variable.

-p. 14, line 30: Please share more specific information about the precision required by the scientists for the data. Again, more information is useful for interpreting results.

The configuration and the precision of each variable will be made available.

These details have been added in the Supplement.

-Section 5 seems like it should be the highlight of the paper as here we are seeing the results on the real data. But it feels like more detail is needed on the data and more discussion of the implications of the results.

Section 5 will be developed to add more details on the data and more discussion on the results obtained.

Section 5 has been reworked. It now provides results on particular variables of the CFOSAT and SWOT datasets. Details on the data have been added in the Supplement.

—————————— Typos, etc.: ——————————

-abstract, line 7: incorrect use of "imposes"

-p.1, line 22: "quite spread"=> "quite prevalent" or "quite popular" , "widely spread" => "widely used"

-p.1., line 26: "reduce significantly" => "significantly reduce"

-p.1. line 27: This sentence (that continues to page 2) is too long.

-p.2, line 3: "can afford for" is awkward

-p2, lines 10-24: this region is 5 paragraphs

-p.3, lines 3-4: awkwardly worded

-p. 3, line 9: one sentence paragraph

-p.3, line 12: missing "," after "Deflate"

-p.3 line 14: not sure what is meant by "new concurrent"

-p.3, line 13: "widely spread" => "widely used"

-p.3, line 16: awkward sentence: "This allows Deflate achieving rather high compression ratios"

-P.3, section 3: again, there are too many tiny paragraphs

-p.4, line 7: "are of same interest" is awkward

-p.4, line 21: Table number is not given

-p.5, line 15: awkwardly worded

-p.5, line 24: One sentence paragraph

-p.7, section 4.2: define f_s, f_ech

-p.7, line 22: "use embarks" is awkward

-p.8, line 7: "declined" doesn't make sense

-p.8, line 14: "embark" - incorrect usage

-p.10, line 10: "This correspond corresponding" needs to be fixed

-p. 10, line 24: Note sure I'd use "performances" here as earlier it was used to indicate speed.

-p. 15, line 21: "ration" => "ratio"

-p.15, line 14: another one sentence paragraph

-p.16, line 9: "Extends to this work" - awkwardly worded

Response: we thank you for highlighting typos that will help us to improve the manuscript.

All these points have been corrected.

Reply to Zender (Referee)

We are grateful to the referee for his constructive and thorough criticism and suggestions to our manuscript. Please find below a detailed point-by-point reply (*referee's comment in italic*).

General Comments

This manuscript presents a new lossy compression algorithm called "Digit Rounding" (DR), and evaluates its performance against and with other lossy and lossless compression algorithms on idealized and remote sensing datasets. The manuscript addresses the growing need to archive meaningful data rather than noise, and to do so reliably and quickly. The study presents an original advance in lossy compression whose implementation unfortunately hampers its utility. The study is understandable yet poorly written. This potentially useful study of lossy compression techniques needs a thorough overhaul before publication.

We will improve the writing of the manuscript by contacting a native English speaker/writer. As suggested by the Anonymous Referee #1, the paper will be reworked to highlight the main contributions of our work and focus more on the application to CFOSAT and SWOT datasets.

The grammar and writing have been improved throughout the manuscript and the paper has been reworked to highlight our main contribution and to focus more on the application to CFOSAT and SWOT datasets.

Specific Comments

Originality: DR is an improvement on "Bit Grooming" (BG) which I invented as an improvement on "Bit Shaving". In that sense I am qualified to comment on its originality. The heart of DR is essentially a continuous version of BG: Whereas BG fixes the number of bits masked for each specified precision, and masks these bits for every value, DR recomputes the number of bits masked for each quantized value to achieve the same precision. BG did not implement the continuous method because I thought that computing the logarithm of each value would be expensive, inelegant, and yield only marginally more compression. However, DR cleverly uses the exponent field instead of computing logarithms, and so deciphers the correct number of bits to mask while avoiding expensive floating point math. This results in significantly more compressibility that (apparently) incurs no significant speed penalty (possibly because it compresses better and thus the lossless step is faster?). Hence DR appears to be a significant algorithmic advance and I congratulate the authors for their insight.

Thank you for your congratulations. They are much appreciated. Indeed, the speed penalty of DR is compensated by the fact that the lossless step is faster.

In the previous version, we cascaded a call to ncks with a call to h5repack to perform Bit Grooming followed by Zstandard compression. For fairer comparisons on the compression speed, we modified our approach and now only employ ncks tool to run Decimal Rounding and Bit Grooming. However, we could not call Zstandard compression via the ncks tool, but only Deflate compression. Consequently, we now provide results for Decimal Rounding, Bit Grooming, Sz and Digit Rounding followed by Deflate compression.

The manuscript stumbles in places due to low quality English, and cries out for more fluent editing. Not only is the word choice often awkward, but the manuscript is like a continuously choppy sea of standalone sentences with few well developed paragraphs that swell with meaning then yield gently to the next idea. GMD readers deserve and expect better.

We will improve the grammar and writing (see first answer). We will also modify the paragraph structures to avoid too small paragraphs and better take care of the transitions.

The grammar and writing have been improved throughout the manuscript. Short paragraphs have also been modified and sentence transitions improved.

Does DR guarantee that it will never create a relative error greater than half the value of the least significant digit? BG chooses the number of digits to mask conservatively, so it can and does guarantee that it always preserves the specified precision. Equations (1)-(7) imply that DR can make the same claim, but this claim is never explicitly tested or made. The absence of this guarantee is puzzling because it would strengthen the confidence of users in the algorithm. However, the guarantee must be explicitly tested, because it undergirds the premise that the comparison between DR and BG is fair. In any case, clearly state whether DR ever violates the desired precision, even if that happens only rarely.

Equations (1)-(7) imply that DR guarantees that it always preserves the specified precision. We will explicitly add that claim in the text and show that DR always provides the desired precision on the number Pi with nsd varying from 1 to 8. We will also provide the maximum absolute error on artificial data of 1 000 000 values spanning [1.0, 2.0) in equal-increment steps of 1e-6.

We have added the following sentence below Eq. 4.

"This condition guarantees that the Digit Rounding algorithm to always preserves a relative error lower than or equal to half the value of the least significant digit."

We have also added results of DR algorithm on the number Pi in Table 2:

"Table 2 provides the result of the Digit Rounding algorithm on the value of π with specified precisions nsd varying from 1 to 8 digits. It can be compared to the Bit Grooming results provided in Table 2 in (Zender, 2016a)."

We also provide the maximum absolute error on artificial data of 1 000 000 values spanning [1.0, 2.0) in equal-increment steps of 1e-6 in Table 3

"Table 3 provides the maximum absolute error obtained with varying nsd values on an artificial dataset composed of 1,000,000 values evenly spaced over the interval [1.0, 2.0). This is the same artificial dataset used in Table 3 in (Zender, 2016a). It shows that Digit Rounding always preserves a relative error lower than or equal to half the value of the least significant digit, i.e. $|s_i - \tilde{s}_i| \leq 0.5 \times 10^{d_i - \mathrm{nsd}}$. "

p. 16 L13: "Code and data availability: The Digit Rounding software source code and the data are currently only available upon request to Xavier Delaunay (xavier.delaunay@thalesgroup.com) or to Flavien Gouillon (Flavien.Gouillon@cnes.fr)." The GMD policy on code and data is here: https://www.geoscientific-model-development.net/about/code_and_data_policy.html. This manuscript provides no code access nor explanation, and no dataset access, and thus appears to violate GMD policy in these areas.

The code and the datasets will be made publicly available on the CNES gitlab.

The code is now publicly available on CNES GitHub at https://github.com/CNES/Digit_Rounding and the dataset are available on demand.

“The Digit Rounding software source code is available from CNES GitHub at https://github.com/CNES/Digit_Rounding. The datasets are available upon request to Xavier Delaunay (xavier.delaunay@thalesgroup.com) or to Flavien Gouillon (Flavien.Gouillon@cnes.fr). The Supplement details the datasets and provides the command lines used for running the compression tools.”

Common comparisons would help build confidence in your results. It would have been more synergistic to evaluate the algorithms on at least one of the same datasets as Zender (2016), which are all publicly available. I am glad the authors used the publicly available NCO executables. Why not release the DR software in the same spirit so that the geoscience community can use (and possibly improve) it?

Comparisons with BG will be provided on the same MERRA dataset used in Zender (2016). The DR software will be released under MIT-style open source license.

We have added results of DR on the same MERRA dataset used in Zender (2016).

“We compare the compression ratio obtained with the Digit Rounding algorithm to that obtained with the Bit Grooming algorithm for the same meteorological data from MERRA re-analysis studied in (Zender, 2016a). Table 4 reports the Bit Grooming results extracted from Table 6 in (Zender, 2016a) and provides the results of the Digit Rounding algorithm. The same lossless compression is employed: Shuffle and Deflate with level 1 compression. From $\mathrm{nsd} = 7$ to $\mathrm{nsd} = 5$, Digit Rounding and Bit Grooming provide similar compression ratios with a slight advantage for the Bit Grooming algorithm. However, from $\mathrm{nsd} = 4$ to $\mathrm{nsd} = 1$, the compression ratios obtained with Digit Rounding are clearly better.”

The DR software is released under LGPL-v3 open source license.

The lossless and lossy compression algorithms analyzed seem like a fairly balanced collection of those most relevant to GMD readers. Most methods that were omitted are, to my knowledge, either non-competitive (e.g., Packing) or not user-friendly, e.g., research grade but not widely available (e.g., Layer Packing) and too hard to independently implement.

Table 6 on p. 19 shows the maximum absolute error (MAE) of BG is quite similar to DR, as I would expect. However, Table 7 on p. 20 shows the maximum absolute error (MAE) of BG is nearly 10x less than DR. Why are the MAEs similar for dataset s1 and significantly different for dataset s3D? I expect DR has a greater mean error (and lower SNR) than BG due to the algorithms, yet the difference in MAEs surprises me. Zender (2016) Table 3 shows that BG is tuned to have an MAE just shy of violating the precision guarantee. An MAE that is nearly 10x larger seems like it might violate the precision guarantee.

These results show that BG can sometimes be too conservative. As shown in Table 1 on the value Pi, BG sometimes preserves more bits in the mantissa than what is strictly necessary to achieve the required precision. This is what happens on the dataset s3D. On the contrary, DR adapts the quantization step to each value of the input dataset. Doing so, it can achieve the required precision while preserving less mantissa bits than DR does. This results both in a higher mean absolute error and in a higher MAE than BG. This explanation will be added to the text.

Thanks to you remark on the MAE on s1 dataset, it has been observed that DR algorithm was also too conservative on some values. It has been enhance in order to provide a MAE closer to what was expected. For this, the value $\log_{10}(\mathrm{m_i})$ is now tabulated with a few values.

"The $\log_{10}(m_i)$ value is tabulated. Only 5 tabulated values are used in our implementation, enough to provide a good precision. The tabulated v values for $\log_{10}(m_i)$ are such that $v \leq \log_{10}(m_i)$. They are provided in the Supplement. This computation slightly underestimates the values for d_i but provides a more conservative quantization, thus guaranteeing the specified number of significant digits."

The following sentence has also been added in the text:

"Bit Grooming is too conservative. It preserves more mantissa bits than strictly necessary to achieve the required precision. This behavior is illustrated in Table 1 with the value of π. In contrast, Digit Rounding adapts the quantization step to each value of the input dataset. Doing so, it can achieve the required precision while preserving less mantissa bits than Bit Grooming does. This results both in a higher maximal absolute error and in a higher mean absolute error than Bit Grooming, but also in a higher compression ratio."

The preceding comment is a request to more carefully analyze the underlying cause of the behaviors reported in the data. The next two comments are to report more results to deepen the analyses and explain the behavior of DR more robustly.

Please include the maximum absolute error or maximum absolute relative error (which normalizes the error by the original value) to Tables 5–10.

MeanAE is an important statistic that is complementary to MaxAE. MeanAE is the average absolute (no compensation between positive and negative) bias in the dataset, and is more familiar and relevant than SNR to at least some geophysicists. Please consider including MeanAE in Tables 5–10.

As suggested, the maximum absolute error and the mean absolute error (MeanAE) will be added to the tables allowing deeper analysis of DR behavior.

The maximum absolute error and the mean absolute error have been added to tables 5, 6 and 7. Tables 9, 12 and 13 provide compression results on CFOSAT and SWOT which are composed of several different datasets. The maximum absolute error and the mean absolute error could only be computed variable per variable. We thus now provide the results obtained on the ground_range_5 variable of the CFOSAT dataset in Table 8, the results obtained on the *height* variable of the SWOT dataset in Table 10, and the results obtained on the *pixel_area* variable of the other SWOT dataset in Table 11.

Zender (2016) and Silver and Zender (2017) consider four primary criteria to evaluate compression algorithms: Compression Ratio, Accuracy, Speed, and User-friendliness. This manuscript neglects explicit consideration of the last, though usability seems (in addition to performance) seems to be an implicit reason why they recommend BG not DR for the "real world" use cases in Sections 5.1 and 5.2. The manuscript would benefit from a more explicit consideration of usability throughout. Examples include software availability, flexibility, and complexity of invocation, as well as transparency (will users have all the necessary software required to read the compressed data?), and instructions to mitigate these issues for DR.

As for BG, there is no "decompression" associated to DR. DR does not require any software to read the rounded data. This argument will be added into the text. The reason why BG is recommended rather than DR for the compression of CFOSAT dataset in section 5.1 is that this dataset is compressed in absolute error bounded compression mode. DR only works for relative error bounded compression mode. Nevertheless, some results using DR on this dataset will be provided for completeness. In section 5.2, BG (in the absolute error bounded compression mode) is recommended rather than DR for the compression SWOT L2 pixel cloud product. This recommendation is based on the compression ratio obtained. We will add the maximum absolute error and the mean absolute error (MeanAE) to Tables 8 to 10 for fairer comparisons. Moreover, we will provide a supplement to the article with the commands and datasets necessary to reproduce the results.

We have added the following sentences in the text:

“We have developed an HDF5 dynamically loaded filter plugin so as to apply the Digit Rounding algorithm to NetCDF-4 or HDF5 datasets. It should be noted that data values rounded by the Digit Rounding algorithm can be read directly: there is no reverse operation to Digit Rounding, and users do not need any software to read the rounded data.”

Moreover, we have added some results using DR on the CFOSAT dataset for completeness.

The maximum absolute error and the mean absolute error have not been added to Tables 9, 12 and 13, because, as explained in the previous answer, CFOSAT and SWOT dataset are composed of several different variable.

We also now provide a supplement to the article with the commands and datasets necessary to reproduce the results.

Tables 1 and 3 follow Tables 1 and 2 of Zender (2016). This should be noted in the text and/or caption of the tables.

The reference to Zender (2016) will be added in the caption of Tables 1 and 3.

The captions have been modified as follows:

“Table 1: Representation of the value of π in IEEE-754 single-precision binary representation (first row) and results preserving 4 significant digits with the Bit Grooming algorithm (second row) or preserving 12 mantissa bits (third row). This table builds on Table 1 in (Zender, 2016a).”

“Table 2: Representation of the value of π in IEEE-754 single-precision binary representation (first row) and results preserving a varying number of significant digits (nsd) with the Digit Rounding algorithm. This table can be compared to Table 2 in (Zender, 2016a) providing the Bit Grooming results for π.”

It seems like Table 2, the algorithm description, should be a figure rather than a table.

This will be corrected as suggested.

The algorithm description is now provided in Figure 2.

The manuscript is awkward in that it introduces a demonstrably superior lossy compression algorithm but recommends a different algorithm (BG) for “real world” cases (Section 5), partly because DR is unavailable in software that potential users have easy access to, and its implementation appears to be too inflexible to use on generic datasets. The recommendation of BG not DR does attest to the objectivity of the study, yet it seems to be an unsatisfying conclusion to what was clearly a time-consuming study. In this sense the manuscript seems premature, since if DR were “ready for primetime” then the authors could have recommended it rather than BG in Section 5. Perhaps the authors should re-evaluate whether the manuscript is premature, i.e., whether it should both introduce *a new lossy algorithm before it is ready to use in optimized workflows for generic geoscientific data compression.*

As previously answered, the manuscript will be reworked to highlight the main contributions of our work and focus on the applications to the CFOSAT and the SWOT datasets. The maximum absolute error and the mean absolute error (MeanAE) will be added to Tables 5 to 10 for fairer comparisons that will allow mitigating the previous conclusions that were based on the compression ratio only. Moreover, some results using DR on CFOSAT dataset will be provided for completeness of the manuscript.

We have added some results using DR on the CFOSAT dataset for completeness, but also maximum and mean absolute error in the tables (see previous answers).

The conclusion has been reworked to make it clearer that we recommend Decimal Rounding for absolute error bounded compression of CFOSAT data but Digit Rounding for relative error bounded compression of SWOT data.

Minor Suggestions

p. 1 L22: "well spread"

p. 2 L22: DEFLATE

p. 4 L1: maxi is redundant. Just use max.

p. 4 L21: Table 1

p. 9 L7: "declined"?

p. 9 L14: "By default, Sz algorithm embark Deflate." is awkward.

p. 14 L27–28: These lines are identical

p. 18 L8: "the number di of significant digit number of digits"???

p. 18 L8: "following Eq." not "following in Eq."

p. 23 Figure 4: Clarify the meaning of the distinct vertical bars.

Response: we thank you for these suggestions that will help us to improve the manuscript.

All these points have been corrected.

Dear Editor,

Thank you so much for extending our time for the revision of the manuscript. We also like to thank both reviewers for their thorough review of the manuscript. Below, find the main modifications made to the manuscript which covers all the remarks.

- As suggested by the reviewers, the manuscript has been reworked to focus more on the application to CFOSAT and SWOT datasets. We have also moved the details concerning the synthetic datasets in the Supplement which is now provided with the paper.
- The term “Bit Grooming” was employed both for the absolute and the relative error bounded compression modes. But the algorithms are different. In the revised version of the paper we employ the term “Decimal Rounding” in the absolute error bounded mode and the term “Bit Grooming” in the relative absolute error bounded mode to remove any ambiguities.
- Thanks to a remark from C. Zender, we found that the Digit Rounding algorithm was sometimes too conservative. We thus slightly modified the implementation. The compression results reported in the new version of the paper have been obtained with the new version of the Digit Rounding algorithm.
- In the previous version of the paper, we cascaded a call to ncks with a call to h5repack to perform Bit Grooming followed by Zstandard compression. For fairer comparisons on the compression speed, we modified our approach and now only employ ncks tool to run Decimal Rounding and Bit Grooming. However, we could not call Zstandard compression via the ncks tool, but only Deflate compression. Consequently, we now provide results for Decimal Rounding, Bit Grooming, Sz and Digit Rounding followed by Deflate compression.
- The Digit Rounding software source code is now available from CNES GitHub at https://github.com/CNES/Digit_Rounding.
- We now provide a Supplement which details the datasets and provides the command lines used for running the compression tools.
- The grammar and English of the paper has been dramatically improved by reviews from native English speakers.

Reply to Anonymous Referee #1

We are grateful to the referee for her/his constructive and thorough criticism and suggestions to our manuscript. Please, find below a detailed point-by-point reply. *Referee's comments are in blue italic*; our answers are in black and our changes to the manuscript in green.

- This manuscript needs a lot of improvement in terms of the grammar and writing. There are many awkward phrases and incorrect word choices that need to be improved (a subset are listed below). The paragraph structures are also in need of modification (many paragraphs contain only 1 or 2 sentences).

We will improve grammar and writing of the manuscript by contacting a native English speaker/writer. Thank you for pointing out the subset of incorrect word choices. We will also modify the paragraph structures to avoid too small paragraphs.

The grammar and writing have been improved throughout the manuscript. Short paragraphs have also been modified.

-Section 2: I'd be helpful to include more detail for the preprocessing algorithms: shuffle and bitshuffle. Also this section in general needs improvement. It's a bit "choppy" to read (needs smoother and better transitions between topics) and feels like more details would be helpful on the methods (especially the ones that the digit rounding algorithm builds on).

More details will be added on the shuffle and bitshuffle algorithms. More details will also be added on the bit-grooming and decimal rounding algorithm, algorithm on which the digit rounding algorithm is built. We will do what is needed to improve this section in general.

Section 2 has been restructured so as to make the reading smoother. Details have been added on the shuffle and bitshuffle algorithms as well as on the bit-grooming and decimal rounding algorithm.

"The Shuffle algorithm groups all the data samples' first bytes together, all the second bytes together, etc. In smooth datasets, or datasets with highly correlated consecutive sample values, this rearrangement creates long runs of similar bytes, improving the dataset's compression. Bitshuffle extends the concept of Shuffle to the bit level by grouping together all the data samples' first bits, second bits, etc.

…

The Decimal Rounding algorithm achieves a uniform scalar quantization of the data. The quantization step is a power of 2 pre-computed so as to preserve a specific number of decimal digits. The Bit Grooming algorithm creates a bitmask to degrade the least significant bits of the mantissa of IEEE 754 floating-point data. Given a specified total number of significant digits, nsd, the Bit Grooming algorithm tabulates the number of mantissa bits that has to be preserved to guarantee the specified precision of nsd digits: to guarantee 1-6 digits of precision, Bit Grooming must retain 5, 8, 11, 15, 18, and 21 mantissa bits respectively. The advantage is that the number of mantissa bits that must be preserved is computed very quickly. The disadvantage is that this computation is not optimal. In many cases, more mantissa bits are preserved than strictly necessary."

-Section 4: Why does using the synthetic data in 4.1 to assess performance make sense - it seems unrelated to the application area of interest. I'd argue that the metrics used in 4.1 are really minimal requirements as well. Also take care when referring to "performance" as it is overloaded term...do you mean speed or effectiveness (it's used both ways)

The objective of using synthetic data was to control the data parameters, such as the SNR, to be able to assess the impact of these parameters on the compression ratios. The results are not reported in this paper which rather focuses on providing a comparison of the compression ratio and speed of different algorithms. It has also been chosen to present only the minimal set of relevant metrics to avoid overloading the paper. We will be more rigorous and replace the term “performance” by “compression ratio” or “compression speed” in the text.

All the occurrences of the term “performance” have been checked and corrected when needed. The Mean Absolute Error metric has been added to the list of metrics:

“mean absolute error $\overline{e_{abs}}$ to evaluate the mean data degradation. It is defined as the mean of the pointwise absolute difference between the original and compressed data:

$\overline{e_{abs}} = \frac{1}{N}\sum_{i=0}^{N-1}|s_i - \tilde{s}_i|$”

*-fpzip is a fast and effective lossless method that would have been nice to compare (I *think* there is an fpzip filter available). Also I believe that any hdf5 filter can be accessed through NetCDF4 (see last sentence in conclusion) - consider contacting the Unidata folks.*

Thank you for the suggestion. Indeed a HDF5 filter is accessible for fpzip. However, many lossless compression algorithms exist. In our paper, we chose to evaluate the most “popular”, i.e. the lossless compression algorithms the most used in applications. Thank you for pointing this evolution of the NetCDF-4 library: from version 4.6.0 - January 24, 2018, NetCDF fully supports HDF5 dynamic filters. The text of the paper will be modified so as to provide the example usage using the new NetCDF-4 features.

An example using the new NetCDF-4 features with *nccopy* tool has been provided in the supplement and the conclusion has been modified to remove the last sentence. However, nccopy tool does not allow yet linking together different filters.

-Comments on doing compression in parallel?

We do not consider running compression algorithm in parallel in this work and will make it clear in the manuscript. It is a possible extension of this study.

The following sentence has been added in section 4:

“Parallel compression has not been considered in this work.”

-When reading the conclusion, it’s hard to see what the main contributions of this paper are. It’s fairly well known already that preprocessing of scientific data (e.g., bit shuffle or shuffle) improves lossy compression. Also the statements in the conclusion aren’t specific to a particular type of data set, but are presented as more general conclusions.

Given that the effectiveness and performance (speed) of lossy and lossless compression are very data, application, and variable dependent, the general statements here are not well justified by the small sample of data in the paper. I’d suggest focusing the paper more heavily on the data in Section 5 (if it’s of interest) and tailoring the discussion in that manner. Or maybe the focus was to be more on speeds than quality, in which case it’s be important to work to get sz and fpzip working, particularly via netcdf-4...

Thank you for the suggestion that will help highlighting the main contributions of our work. As suggested, the paper will be reworked to focus more on the application to the CFOSAT and SWOT datasets. We will also avoid general statements but attach our conclusions to our application case.

The paper has been reworked to focus more on the application to the CFOSAT and SWOT datasets. Sz compression has been run on CFOSAT dataset and on some parts of SWOT datasets. The NetCDF-4 tool nccopy does not allow yet linking together different filters. This restrains its usability and this is why we prefer using h5repack tool.

——————————- Specific items: ——————————-

-p2, line 20: note that fpzip can also be lossless

Thank you for the remark. Fpzip will be presented both as a lossless and lossy compression algorithm.

The text has been modified as follows:

"Third, some lossy/lossless compression algorithms, such as FPZIP (Lindstrom and Isenburg, 2006), …"

-p. 8: discussion of figure 5: is the width of the bars related to the compression levels? (e.g. line 20 statement is unclear)

No. All the bars have the same width. Each vertical bar represents a compression level. For instance, the 9 compression levels of Deflate are represented by 9 vertical bars. This will be clarified in the text p.8.

Clarifications have been added to the text:

"The vertical bars represent the results for different compression levels: from 1 to 9 for the Deflate level dfl_lvl, from 1 to 22 for Zstandard level zstd_lvl, and only one level for LZ4."

-p.8, lines 28-29: Why is this the case? (Add some discussion beyond describing the figure.)

These lower compression/decompression speeds are not well understood and would require further investigation to be fully understood. It might be related to HDF5 chunking. Indeed, HDF5 split the data into chunks of small size that are independently compressed. This allows HDF5 to improve partial I/O for big datasets but can sometimes reduce the compression/decompression speeds. This discussion will be added to the text.

The following sentence has been added to the text:

"Further investigations are required to understand why the compression/decompression speeds are lower, but it might be related to HDF5 chunking."

-p.8, line 20: I feel like the parameters should be better explained for –filter so that the reader can try them more easily. For example, what does the "32017" mean? I think that the following 0 is for sz, but this is not stated either.

We will add the meaning of each parameters. Each HDF5 filter is identified by a unique ID. “32017” is the identifier of Sz filter. The following “0” is the number of filter parameters. In the case of Sz, the filter does not have any parameter to set. That is why there are 0 parameters. Sz compressor is configured via the sz.config file. The same explanations will be added for the other filters used in the paper.

All the command lines have been moved to the Supplement with the explanation above to avoid overloading the manuscript.

-p. 10, line 6: re: "experiments have shown" - whose or which experiments (cite?)

It is based on our own experiments that haven’t been published. The sentence will be reworked as follows: “We have found that Shuffle or Bitshuffle preprocessing do not increase the compression ratio when applied after Sz. We have also found that and Bitshuffle provide lower compression ratio than Shuffle when applied after Bit Grooming. That is why only Shuffle is applied after Bit Grooming.”

This sentence has been removed since the results have not been published.

-Table 5 : Why is the speed faster for 1d?

As previously, the lower compression/decompression speeds obtained with the dataset s3D are not well understood and might be related to HDF5 chunking. This discussion will be added to the text.

The following sentence has been added to the text:

“The lower compression/decompression speeds obtained with Sz on the dataset s3D are not well understood and might be related to HDF5 chunking as previously mentioned.”

-Section 4.4.1, line 21-24: Any idea why you get these results?

Sz performs better on smooth signals since it makes use of a prediction step. The signal s1 being highly noisy, Sz prediction might often fail. This can explain the lower compression ratio on the signal s1. On the contrary, Bit-grooming does not makes any prediction. This can explain why it achieves better compression than Sz on the signal s1. This hypothesis will be added to the text.

The following sentences have been added to the text:

“Sz may perform better on dataset $s3D$ because it is smoother than dataset $s1$. Indeed, Sz integrates a prediction step. This prediction might often fail because dataset $s1$ is very noisy. This may explain the lower compression ratio for this dataset. Decimal Rounding, however, does not make any predictions, which may explain why it achieves a better compression than Sz for dataset $s1$.”

-Section 4.4.1, last sentences: It’s unclear to me what the value of these synthetic data sets is - especially given the statement on p. 11, line 3, about the dependence on the dataset

As suggested previously, the paper will be reworked to focus more on the application to CFOSAT and SWOT datasets without drawing general conclusions based on the results obtained on the synthetic datasets.

We modified the last sentence of this section as follows:

“Both Sz and Bit Grooming algorithms seem valuable for compression in absolute error-bounded compression mode.”

-page 11, line 9-10: I’d include characteristics of the data (e.g., maximum abs. value) earlier in the text when the two datasets are introduced.

Your suggestion will be taken into account: the characteristics of the data will be introduced in section 4.2.

We have added the characteristics of the data in section 4:

“Datasets $s1$ and $s3D$ were generated, $s1$ being a noisy sinusoid of 1 dimension with a maximum absolute value of 118. The data volume of the $s1$ dataset is 4MB. Dataset $s3D$ is a noisy sinusoid pulse of 3 dimensions with a maximum absolute value of 145. The data volume of the $s3D$ dataset is 512MB.”

-page 11, line 24: I don’t see relative error mentioned in Table 6 - it seems to just be absolute error

The text will be modified to make it clearer: “…all three algorithms respect the maximum absolute error of 0.5 which, for the signal s1, corresponds to a relative error of 0.00424.”

The text has been modified as follows:

“… all three algorithms respect the maximum absolute error of 0.5, which corresponds for dataset s1 to a relative error of 0.00424.”

-p.11-12: Need more of a discussion of the results in Figure 7. For 3D, it looks like bit grooming and digit rounding are similar - I don’t see a clear advantage.

More discussion on the results will be added to the text. For the s3D you are right, there is no clear advantage. It is written “the Digit Rounding algorithm provides compression performance very closed to the one of the Bit Grooming algorithm”.

The text has been modified as follows:

“All three algorithms provide similar SNR versus compression ratio results, with a slight advantage for the Bit Grooming algorithm.”

-p.12, lines 16-17: SZ compression can be controlled with an absolute error bound, so why is the relative error bound adjusted to get the desired abs. error?

The objective was to see if Sz compression configured with a relative error bound respect the error bound specified. As the digit rounding and bit-grooming algorithm can only be configured on a number of significant digits, they can only “produce” absolute error in 0.5, 0.05, 0.005, etc. In order to be able to compare Sz configured with a relative error bound with those algorithms, we have configured the relative error bound to obtain a maximum absolute error of 0.5. These explanations will be added to the text.

The following sentence has been added in the text:

In order to be able to compare Sz configured with a relative error bound with those algorithms, we configured the relative error bound to obtain a maximum absolute error of 0.5: the pw_relBoundRatio parameter in Sz was set to 0.00424.

-Section 5.1: It is disappointing not to have SZ results on the real data of interest. Were the SZ authors contacted? I would think that they could have helped resolve this issue.

Yes, we had some exchanges. The issue is still under investigation.

A more recent version of Sz has been used and results on CFOSAT and SWOT datasets are now provided.

-p. 15, line 18: "which only a few attributes may be missing" - It’s unclear what this means. It’s super helpful to really detail the data being compressed so that one can make sense of the results.

Details on the datasets will be added to the text.

This part of the sentence has been removed has it is not relevant in the frame of this study. Details on the CFOSAT and SWOT datasets have been added in the Supplement, particularly the precision required for the compression of each variable.

-p. 14, line 30: Please share more specific information about the precision required by the scientists for the data. Again, more information is useful for interpreting results.

The configuration and the precision of each variable will be made available.

These details have been added in the Supplement.

-Section 5 seems like it should be the highlight of the paper as here we are seeing the results on the real data. But it feels like more detail is needed on the data and more discussion of the implications of the results.

Section 5 will be developed to add more details on the data and more discussion on the results obtained.

Section 5 has been reworked. It now provides results on particular variables of the CFOSAT and SWOT datasets. Details on the data have been added in the Supplement.

——————— Typos, etc.: ———————

-abstract, line 7: incorrect use of "imposes"

-p.1, line 22: "quite spread"=> "quite prevalent" or "quite popular" , "widely spread" => "widely used"

-p.1., line 26: "reduce significantly" => "significantly reduce"

-p.1. line 27: This sentence (that continues to page 2) is too long.

-p.2, line 3: "can afford for" is awkward

-p2, lines 10-24: this region is 5 paragraphs

-p.3, lines 3-4: awkwardly worded

-p. 3, line 9: one sentence paragraph

-p.3, line 12: missing "," after "Deflate"

-p.3 line 14: not sure what is meant by "new concurrent"

-p.3, line 13: "widely spread" => "widely used"

-p.3, line 16: awkward sentence: "This allows Deflate achieving rather high compression ratios"

-P.3, section 3: again, there are too many tiny paragraphs

-p.4, line 7: "are of same interest" is awkward

-p.4, line 21: Table number is not given

-p.5, line 15: awkwardly worded

-p.5, line 24: One sentence paragraph

-p.7, section 4.2: define f_s, f_ech

-p.7, line 22: "use embarks" is awkward

-p.8, line 7: "declined" doesn't make sense

-p.8, line 14: "embark" - incorrect usage

-p.10, line 10: "This correspond corresponding" needs to be fixed

-p. 10, line 24: Note sure I'd use "performances" here as earlier it was used to indicate speed.

-p. 15, line 21: "ration" => "ratio"

-p.15, line 14: another one sentence paragraph

-p.16, line 9: "Extends to this work" - awkwardly worded

Response: we thank you for highlighting typos that will help us to improve the manuscript.

All these points have been corrected.

Reply to Zender (Referee)

We are grateful to the referee for his constructive and thorough criticism and suggestions to our manuscript. Please find below a detailed point-by-point reply (*referee's comment in italic*).

General Comments

This manuscript presents a new lossy compression algorithm called "Digit Rounding" (DR), and evaluates its performance against and with other lossy and lossless compression algorithms on idealized and remote sensing datasets. The manuscript addresses the growing need to archive meaningful data rather than noise, and to do so reliably and quickly. The study presents an original advance in lossy compression whose implementation unfortunately hampers its utility. The study is understandable yet poorly written. This potentially useful study of lossy compression techniques needs a thorough overhaul before publication.

We will improve the writing of the manuscript by contacting a native English speaker/writer. As suggested by the Anonymous Referee #1, the paper will be reworked to highlight the main contributions of our work and focus more on the application to CFOSAT and SWOT datasets.

The grammar and writing have been improved throughout the manuscript and the paper has been reworked to highlight our main contribution and to focus more on the application to CFOSAT and SWOT datasets.

Specific Comments

Originality: DR is an improvement on "Bit Grooming" (BG) which I invented as an improvement on "Bit Shaving". In that sense I am qualified to comment on its originality. The heart of DR is essentially a continuous version of BG: Whereas BG fixes the number of bits masked for each specified precision, and masks these bits for every value, DR recomputes the number of bits masked for each quantized value to achieve the same precision. BG did not implement the continuous method because I thought that computing the logarithm of each value would be expensive, inelegant, and yield only marginally more compression. However, DR cleverly uses the exponent field instead of computing logarithms, and so deciphers the correct number of bits to mask while avoiding expensive floating point math. This results in significantly more compressibility that (apparently) incurs no significant speed penalty (possibly because it compresses better and thus the lossless step is faster?). Hence DR appears to be a significant algorithmic advance and I congratulate the authors for their insight.

Thank you for your congratulations. They are much appreciated. Indeed, the speed penalty of DR is compensated by the fact that the lossless step is faster.

In the previous version, we cascaded a call to ncks with a call to h5repack to perform Bit Grooming followed by Zstandard compression. For fairer comparisons on the compression speed, we modified our approach and now only employ ncks tool to run Decimal Rounding and Bit Grooming. However, we could not call Zstandard compression via the ncks tool, but only Deflate compression. Consequently, we now provide results for Decimal Rounding, Bit Grooming, Sz and Digit Rounding followed by Deflate compression.

The manuscript stumbles in places due to low quality English, and cries out for more fluent editing. Not only is the word choice often awkward, but the manuscript is like a continuously choppy sea of standalone sentences with few well developed paragraphs that swell with meaning then yield gently to the next idea. GMD readers deserve and expect better.

We will improve the grammar and writing (see first answer). We will also modify the paragraph structures to avoid too small paragraphs and better take care of the transitions.

The grammar and writing have been improved throughout the manuscript. Short paragraphs have also been modified and sentence transitions improved.

Does DR guarantee that it will never create a relative error greater than half the value of the least significant digit? BG chooses the number of digits to mask conservatively, so it can and does guarantee that it always preserves the specified precision. Equations (1)-(7) imply that DR can make the same claim, but this claim is never explicitly tested or made. The absence of this guarantee is puzzling because it would strengthen the confidence of users in the algorithm. However, the guarantee must be explicitly tested, because it undergirds the premise that the comparison between DR and BG is fair. In any case, clearly state whether DR ever violates the desired precision, even if that happens only rarely.

Equations (1)-(7) imply that DR guarantees that it always preserves the specified precision. We will explicitly add that claim in the text and show that DR always provides the desired precision on the number Pi with nsd varying from 1 to 8. We will also provide the maximum absolute error on artificial data of 1 000 000 values spanning [1.0, 2.0) in equal-increment steps of 1e-6.

We have added the following sentence below Eq. 4.

"This condition guarantees that the Digit Rounding algorithm to always preserves a relative error lower than or equal to half the value of the least significant digit."

We have also added results of DR algorithm on the number Pi in Table 2:

"Table 2 provides the result of the Digit Rounding algorithm on the value of π with specified precisions nsd varying from 1 to 8 digits. It can be compared to the Bit Grooming results provided in Table 2 in (Zender, 2016a)."

We also provide the maximum absolute error on artificial data of 1 000 000 values spanning [1.0, 2.0) in equal-increment steps of 1e-6 in Table 3

"Table 3 provides the maximum absolute error obtained with varying nsd values on an artificial dataset composed of 1,000,000 values evenly spaced over the interval [1.0, 2.0). This is the same artificial dataset used in Table 3 in (Zender, 2016a). It shows that Digit Rounding always preserves a relative error lower than or equal to half the value of the least significant digit, i.e. $|s_i - \tilde{s}_i| \leq 0.5 \times 10^{d_i - \mathrm{nsd}}$. "

p. 16 L13: "Code and data availability: The Digit Rounding software source code and the data are currently only available upon request to Xavier Delaunay (xavier.delaunay@thalesgroup.com) or to Flavien Gouillon (Flavien.Gouillon@cnes.fr)." The GMD policy on code and data is here: https://www.geoscientific-model-development.net/about/code_and_data_policy.html. This manuscript provides no code access nor explanation, and no dataset access, and thus appears to violate GMD policy in these areas.

The code and the datasets will be made publicly available on the CNES gitlab.

The code is now publicly available on CNES GitHub at https://github.com/CNES/Digit_Rounding and the dataset are available on demand.

“The Digit Rounding software source code is available from CNES GitHub at https://github.com/CNES/Digit_Rounding. The datasets are available upon request to Xavier Delaunay (xavier.delaunay@thalesgroup.com) or to Flavien Gouillon (Flavien.Gouillon@cnes.fr). The Supplement details the datasets and provides the command lines used for running the compression tools.”

Common comparisons would help build confidence in your results. It would have been more synergistic to evaluate the algorithms on at least one of the same datasets as Zender (2016), which are all publicly available. I am glad the authors used the publicly available NCO executables. Why not release the DR software in the same spirit so that the geoscience community can use (and possibly improve) it?

Comparisons with BG will be provided on the same MERRA dataset used in Zender (2016). The DR software will be released under MIT-style open source license.

We have added results of DR on the same MERRA dataset used in Zender (2016).

“We compare the compression ratio obtained with the Digit Rounding algorithm to that obtained with the Bit Grooming algorithm for the same meteorological data from MERRA re-analysis studied in (Zender, 2016a). Table 4 reports the Bit Grooming results extracted from Table 6 in (Zender, 2016a) and provides the results of the Digit Rounding algorithm. The same lossless compression is employed: Shuffle and Deflate with level 1 compression. From $\mathrm{nsd} = 7$ to $\mathrm{nsd} = 5$, Digit Rounding and Bit Grooming provide similar compression ratios with a slight advantage for the Bit Grooming algorithm. However, from $\mathrm{nsd} = 4$ to $\mathrm{nsd} = 1$, the compression ratios obtained with Digit Rounding are clearly better.”

The DR software is released under LGPL-v3 open source license.

The lossless and lossy compression algorithms analyzed seem like a fairly balanced collection of those most relevant to GMD readers. Most methods that were omitted are, to my knowledge, either non-competitive (e.g., Packing) or not user-friendly, e.g., research grade but not widely available (e.g., Layer Packing) and too hard to independently implement.

Table 6 on p. 19 shows the maximum absolute error (MAE) of BG is quite similar to DR, as I would expect. However, Table 7 on p. 20 shows the maximum absolute error (MAE) of BG is nearly 10x less than DR. Why are the MAEs similar for dataset s1 and significantly different for dataset s3D? I expect DR has a greater mean error (and lower SNR) than BG due to the algorithms, yet the difference in MAEs surprises me. Zender (2016) Table 3 shows that BG is tuned to have an MAE just shy of violating the precision guarantee. An MAE that is nearly 10x larger seems like it might violate the precision guarantee.

These results show that BG can sometimes be too conservative. As shown in Table 1 on the value Pi, BG sometimes preserves more bits in the mantissa than what is strictly necessary to achieve the required precision. This is what happens on the dataset s3D. On the contrary, DR adapts the quantization step to each value of the input dataset. Doing so, it can achieve the required precision while preserving less mantissa bits than DR does. This results both in a higher mean absolute error and in a higher MAE than BG. This explanation will be added to the text.

Thanks to you remark on the MAE on s1 dataset, it has been observed that DR algorithm was also too conservative on some values. It has been enhance in order to provide a MAE closer to what was expected. For this, the value $\log_{10}(\mathrm{m_i})$ is now tabulated with a few values.

“The $\log_{10}(m_i)$ value is tabulated. Only 5 tabulated values are used in our implementation, enough to provide a good precision. The tabulated v values for $\log_{10}(m_i)$ are such that $v \leq \log_{10}(m_i)$. They are provided in the Supplement. This computation slightly underestimates the values for d_i but provides a more conservative quantization, thus guaranteeing the specified number of significant digits.”

The following sentence has also been added in the text:

“Bit Grooming is too conservative. It preserves more mantissa bits than strictly necessary to achieve the required precision. This behavior is illustrated in Table 1 with the value of π. In contrast, Digit Rounding adapts the quantization step to each value of the input dataset. Doing so, it can achieve the required precision while preserving less mantissa bits than Bit Grooming does. This results both in a higher maximal absolute error and in a higher mean absolute error than Bit Grooming, but also in a higher compression ratio.”

The preceding comment is a request to more carefully analyze the underlying cause of the behaviors reported in the data. The next two comments are to report more results to deepen the analyses and explain the behavior of DR more robustly.

Please include the maximum absolute error or maximum absolute relative error (which normalizes the error by the original value) to Tables 5–10.

MeanAE is an important statistic that is complementary to MaxAE. MeanAE is the average absolute (no compensation between positive and negative) bias in the dataset, and is more familiar and relevant than SNR to at least some geophysicists. Please consider including MeanAE in Tables 5–10.

As suggested, the maximum absolute error and the mean absolute error (MeanAE) will be added to the tables allowing deeper analysis of DR behavior.

The maximum absolute error and the mean absolute error have been added to tables 5, 6 and 7. Tables 9, 12 and 13 provide compression results on CFOSAT and SWOT which are composed of several different datasets. The maximum absolute error and the mean absolute error could only be computed variable per variable. We thus now provide the results obtained on the ground_range_5 variable of the CFOSAT dataset in Table 8, the results obtained on the *height* variable of the SWOT dataset in Table 10, and the results obtained on the *pixel_area* variable of the other SWOT dataset in Table 11.

Zender (2016) and Silver and Zender (2017) consider four primary criteria to evaluate compression algorithms: Compression Ratio, Accuracy, Speed, and User-friendliness. This manuscript neglects explicit consideration of the last, though usability seems (in addition to performance) seems to be an implicit reason why they recommend BG not DR for the “real world” use cases in Sections 5.1 and 5.2. The manuscript would benefit from a more explicit consideration of usability throughout. Examples include software availability, flexibility, and complexity of invocation, as well as transparency (will users have all the necessary software required to read the compressed data?), and instructions to mitigate these issues for DR.

As for BG, there is no “decompression” associated to DR. DR does not require any software to read the rounded data. This argument will be added into the text. The reason why BG is recommended rather than DR for the compression of CFOSAT dataset in section 5.1 is that this dataset is compressed in absolute error bounded compression mode. DR only works for relative error bounded compression mode. Nevertheless, some results using DR on this dataset will be provided for completeness. In section 5.2, BG (in the absolute error bounded compression mode) is recommended rather than DR for the compression SWOT L2 pixel cloud product. This recommendation is based on the compression ratio obtained. We will add the maximum absolute error and the mean absolute error (MeanAE) to Tables 8 to 10 for fairer comparisons. Moreover, we will provide a supplement to the article with the commands and datasets necessary to reproduce the results.

We have added the following sentences in the text:

"We have developed an HDF5 dynamically loaded filter plugin so as to apply the Digit Rounding algorithm to NetCDF-4 or HDF5 datasets. It should be noted that data values rounded by the Digit Rounding algorithm can be read directly: there is no reverse operation to Digit Rounding, and users do not need any software to read the rounded data."

Moreover, we have added some results using DR on the CFOSAT dataset for completeness.

The maximum absolute error and the mean absolute error have not been added to Tables 9, 12 and 13, because, as explained in the previous answer, CFOSAT and SWOT dataset are composed of several different variable.

We also now provide a supplement to the article with the commands and datasets necessary to reproduce the results.

Tables 1 and 3 follow Tables 1 and 2 of Zender (2016). This should be noted in the text and/or caption of the tables.

The reference to Zender (2016) will be added in the caption of Tables 1 and 3.

The captions have been modified as follows:

"Table 1: Representation of the value of π in IEEE-754 single-precision binary representation (first row) and results preserving 4 significant digits with the Bit Grooming algorithm (second row) or preserving 12 mantissa bits (third row). This table builds on Table 1 in (Zender, 2016a)."

"Table 2: Representation of the value of π in IEEE-754 single-precision binary representation (first row) and results preserving a varying number of significant digits (nsd) with the Digit Rounding algorithm. This table can be compared to Table 2 in (Zender, 2016a) providing the Bit Grooming results for π."

It seems like Table 2, the algorithm description, should be a figure rather than a table.

This will be corrected as suggested.

The algorithm description is now provided in Figure 2.

The manuscript is awkward in that it introduces a demonstrably superior lossy compression algorithm but recommends a different algorithm (BG) for "real world" cases (Section 5), partly because DR is unavailable in software that potential users have easy access to, and its implementation appears to be too inflexible to use on generic datasets. The recommendation of BG not DR does attest to the objectivity of the study, yet it seems to be an unsatisfying conclusion to what was clearly a time-consuming study. In this sense the manuscript seems premature, since if DR were "ready for primetime" then the authors could have recommended it rather than BG in Section 5. Perhaps the authors should re-evaluate whether the manuscript is premature, i.e., whether it should both introduce *a new lossy algorithm before it is ready to use in optimized workflows for generic geoscientific data compression.*

As previously answered, the manuscript will be reworked to highlight the main contributions of our work and focus on the applications to the CFOSAT and the SWOT datasets. The maximum absolute error and the mean absolute error (MeanAE) will be added to Tables 5 to 10 for fairer comparisons that will allow mitigating the previous conclusions that were based on the compression ratio only. Moreover, some results using DR on CFOSAT dataset will be provided for completeness of the manuscript.

We have added some results using DR on the CFOSAT dataset for completeness, but also maximum and mean absolute error in the tables (see previous answers).

The conclusion has been reworked to make it clearer that we recommend Decimal Rounding for absolute error bounded compression of CFOSAT data but Digit Rounding for relative error bounded compression of SWOT data.

Minor Suggestions

p. 1 L22: "well spread"

p. 2 L22: DEFLATE

p. 4 L1: maxi is redundant. Just use max.

p. 4 L21: Table 1

p. 9 L7: "declined"?

p. 9 L14: "By default, Sz algorithm embark Deflate." is awkward.

p. 14 L27–28: These lines are identical

p. 18 L8: "the number di of significant digit number of digits"???

p. 18 L8: "following Eq." not "following in Eq."

p. 23 Figure 4: Clarify the meaning of the distinct vertical bars.

Response: we thank you for these suggestions that will help us to improve the manuscript.

All these points have been corrected.

Evaluation of lossless and lossy algorithms for the compression of scientific datasets in NetCDF-4 or HDF5 ~~formatted~~ files

Xavier Delaunay[1], Aurélie Courtois[1], Flavien Gouillon[2]

[1]Thales ~~Services~~, 290 allée du Lac, 31670 Labège, France
[2]CNES, Centre S~~s~~patial de Toulouse, 18 avenue Edouard Belin, 31401 Toulouse, France

Correspondence to: Xavier Delaunay (xavier.delaunay@thalesgroup.com)

Abstract. The increasing volume of scientific datasets ~~imposes~~ ~~enforces~~ requires the use of compression to reduce ~~the~~ data storage and~~or~~ transmission costs, ~~specifically~~ especially for the oceanographic~~yoceanography~~ or meteorological datasets generated by Earth observation mission ground segments. These data are mostly produced in NetCDF ~~formatted~~ files. Indeed, the NetCDF-4/HDF5 file formats are widely ~~spread~~ used throughout~~in~~ the global scientific community because of the ~~nice~~ useful features they offer. ~~Particularly, the~~ HDF5 in particular offers ~~the~~ a dynamically loaded filter plugin ~~functionality allowing~~ so that users can~~to~~ write ~~filters, such as~~ compression/decompression filters, for example, and~~to~~ process the data before reading or writing ~~it~~ them ~~on~~ to ~~the~~ disk. ~~In t~~This ~~work~~study~~this work, we~~ evaluates~~evaluate~~ ~~the performance of~~ lossy and lossless compression/decompression methods through NetCDF-4 and HDF5 tools on analytical and real scientific floating-point datasets. We also introduce the Digit Rounding algorithm, a new relative erro~~r b~~r-bounded data reduction method inspired by the Bit Grooming algorithm. The Digit Rounding algorithm ~~allows~~ offers a high compression ratio while ~~preserving~~ keeping a given number of significant digits in the dataset. It achieves a higher compression ratio than the Bit Grooming algorithm ~~while keeping similar~~with slightly lower compression speed.

1 Introduction

Ground segments ~~that~~ processing~~process~~ scientific mission data are facing challenges due to the ever-increasing resolution of on-board instruments and the volume of data ~~volume~~ to be~~:~~ processed, stored~~: process, store~~ and transmitted~~transmit~~. This is the case for oceanographic and meteorological missions, for instance. Earth observation mission ground segments produce very large files mostly in NetCDF format, which~~: it~~ is ~~a~~ standard in the oceanography field and ~~quite spread~~widely used ~~in~~ by the meteorological community. This file format is widely ~~spread~~ used throughout~~in~~ the global scientific community because of ~~the~~ its useful~~nice~~ features ~~it offers~~. The fourth version of ~~tsame~~ the~~he~~ NetCDF library, denoted NetCDF-4/HDF5 (as it is based on the HDF5 layer), offers 'Deflate' and 'Shuffle' algorithms as~~some~~ native compression features~~, namely 'Deflate' and 'Shuffle' algorithms~~. However, the compression ~~performance~~ ratio achieved does not fully ~~fulfil~~ meet ~~the~~ ground processing requirements, which are to ~~reduce~~ significantly reduce the storage and dissemination cost as well as the I/O~~IO~~ times between two modules ~~of~~ in the processing chain.

~~Facing ~~In response to the ever-increasing volume of data, scientists are ~~more disposed~~keen to compress data. However, they have certain~~ but with some~~ requirements: ~~science data are generally floating point data; the ~~both compression and decompression have to be fast~~ and either, lossless, or lossy under depending on thesome conditions.~~ ~~:~~Lossy compression is acceptable only if the compression ratios are higher than those of lossless algorithms and if the precision, or data loss, ~~shall~~ can be controlled. There~~, the compression ratio higher than the ones of lossless algorithms. In the lossy case, there~~ is a trade-off between the data volume and the accuracy of the compressed data.

Nevertheless, scientists can ~~afford for~~accept small losses if they remain below ~~under ~~the data's noise level~~ in the data~~. Noise is ~~indeed ~~difficult to~~hardly~~ compress~~ible~~ and of ~~poor ~~little interest ~~for ~~to~~the~~ scientists, ~~thus ~~so they do not consider ~~as loss,~~ data degradation~~alterations~~ that ~~are ~~remains under the noise level as a loss (Baker et al., 2016).

~~Hence, i~~In~~in~~ order to increase the compression ~~performance ~~ratio within the processing chain, 'clipping' methods may be used to~~a~~ degrade~~ationa degradation of~~ the data ~~is considered via the use of so-called "clipping" methods ~~before ~~the~~ compression. ~~Clipping ~~These methods ~~allows ~~increase~~ingincreasing~~ the compression~~ performance~~ ratio by removing the least significant digits ~~or bits ~~in the data. Indeed, at some level, these least significant digits ~~or bits ~~may not be scientifically meaningful in datasets corrupted by noise~~, and this is particularly true for floating- point data~~.

This paper studies compression and clipping ~~old and new ~~methods that can be applied to scientific datasets in order to maximize the compression ~~performance ~~ratio while preserving ~~the ~~scientific data content and ~~the ~~numerical accuracy. It focuses on methods that can be applied to scientific datasets, i.e. vectors or matrices of floating~~-~~point numbers.

First, lossless compression algorithms can be applied to any kind of data. The standard is the 'Deflate'~~Deflate~~ algorithm (Deutsch, 1996)~~,~~ native in NetCDF-4/HDF5 libraries. It is widely ~~spread ~~used~~and implemented~~ in compression tools such as zip, gzip, and zlib libraries~~y~~, and has become~~.library. It is~~ a ~~reference ~~benchmark for lossless data compression. Recently, alternative~~s~~ lossless compression algorithms have emerged. These include~~ such as~~ Google Snappy, LZ4 (Collet, 2013) or Zstandard (Collet and Turner, 2016). ~~None of tThese algorithms do not make use of Huffman coding t~~To achieve faster compression than the Deflate algorithm, none of these algorithms use Huffman coding.~~.~~

Second, pre-processing methods such as ~~the ~~Shuffle, available in HDF5, or Bitshuffle (Masui et al., 2015) ~~allow ~~are used to optimize~~ingoptimizing~~ ~~the ~~lossless compression by rearranging~~orderingreordering~~ the data bytes or bits into~~in~~ a "more compressible" order.

Third, some lossy/lossless ~~lossy ~~compression algorithms, such as FPZIP (Lindstrom and Isenburg, 2006), ZFP (Lindstrom, 2014) or Sz (Tao et al, 2017a), are specifically designed for ~~the compression of ~~scientific data~~,~~ and, in particular floating-point data~~,~~ and ~~allow ~~can control~~ling~~ ~~the ~~data loss.

Fourth, data reduction methods such as Linear Packing (Caron, 2014a), Layer Packing (Silver and Zender, 2017), Bit Shaving (Caron, 2014b), and Bit Grooming (Zender, 2016a) ~~introduce some loss~~lose some~~ in the~~ data content without necessarily reducing ~~the data~~its volume. Pre-processing methods and lossless compression can then be applied to obtain a higher compression ratio.

This paper focuses on compression methods implemented for NetCDF-4 or HDF5 files. ~~Indeed, t~~These~~these~~ scientific file formats are wide~~ly~~ spread ~~across~~ among the oceanographic~~oceanography~~ and meteorological~~ical~~ communities~~y~~.~~meteorological community.~~ HDF5 offers ~~the~~ a dynamically loaded filter plugin ~~functionality~~that~~. It~~ allows users to write~~ingwriting filters, such as~~ compression/decompression filters (among others), and~~,~~ to process ~~the~~ data before reading or writing ~~it~~ them to~~on~~ ~~the~~ disk. Consequently, many compression/decompression filters~~,~~ such as Bitshuffle, Zstandard, LZ4, and Sz~~,~~ have been implemented by members of the HDF5 user~~s'~~ community and are freely ~~accessible~~available. ~~On the other hand~~T~~, t~~he NetCDF Operator toolkit (NCO) (Zender, 2016b) also offers some compression features, such as B~~b~~it ~~s~~Shaving, Decimal Rounding~~bit shaving~~ and Bit Grooming.

The rest of this~~This~~ paper is ~~organized~~ divided into five ~~more~~ sections. Section 2 presents the lossless and lossy compression schemes for scientific floating-~~-~~point datasets.~~ and the absolute and relative error bounded compression modes.~~ Section 3 introduces the Digit Rounding algorithm, which~~. This algorithm~~ ~~alters the data in a relative error bounded manner to make them more compressible. It is an alternative,~~is ~~-~~an ~~inspired~~ improvement of ~~by~~ the Bit Grooming algorithm that optimizes the number of mantissa bits preserved. Section 4 defines the performance metrics used in this paper. Section 5~~4~~ describes the performance assessment of a selection of lossless and lossy compression methods on synthetic datasets. It presents the datasets and ~~, the performance metrics, the~~ compression results before~~, and~~ ~~finally provides~~making some recommendations. Section 6~~5~~ provides some compression results obtained ~~on~~ with real CFOSAT and SWOT datasets. ~~Last~~Finally, section 7~~6~~ provides our conclusions.

2 Compression algorithms

Compression schemes for scientific floating-~~-~~point datasets ~~can be composed of~~usually entail several steps: ~~a~~ data~~Data~~ reduction~~ step~~, ~~a~~ preprocessing~~ step~~, and ~~aand~~ lossless coding~~ step~~. These three steps ~~methods~~ can be chained as illustrated i~~o~~n Fig. 1.

The lossless coding step is reversible. It does not ~~introduce any alteration~~ degrade~~in~~ the data ~~but~~ while ~~allows~~ reducing its~~ing the data~~ volume. It can be implemented by ~~This step can make use of~~ lossless compression algorithms such as Deflate, Snappy, LZ4 or Zstandard. The preprocessing step is also reversible. It rearranges~~orders~~~~reorders~~ the data bytes or bits to enhance ~~the~~ lossless coding ~~step performance~~efficiency. It can be~~make use of lossless compression~~ implemented by algorithms such as Shuffle~~,~~ or Bitshuffle. The data reduction step is not reversible because it entails~~:~~ data losses~~ are introduced in this step~~. The ~~strategy~~ goal is to remove irrelevant data such as noise or other scientifically meaningless data. Data reduction can reduce data volume, depending~~Depending~~ on the algorithm used~~use, this step can reduce the data volume~~. For instance, the Linear Packing and Sz algorithms ~~allow~~ reduce~~ingreducing~~ ~~the~~ data volume, but ~~not~~ B~~b~~it S~~bit~~ ~~s~~having and Bit Grooming algorithms do not.

~~In t~~This~~this~~ paper~~, we chosechoose to~~ evaluates~~evaluate~~ ~~the performance of the~~ lossless compression algorithms Deflate, LZ4, and Zstandard; Deflate because it is the ~~reference~~benchmark algorithm, LZ4 because it is a widely~~- spread~~ used, very-high~~-~~-speed compressor, and Zstandard because it provides~~is the new concurrent of~~ better results than Deflate~~,~~ both in terms of~~on~~ compression ratios and ~~on~~ of compression/decompression speeds. The Deflate algorithm ~~make~~ uses~~use~~ ~~of~~ LZ77 dictionary coding (Ziv and Lempel, 1977) and ~~of~~ Huffman entropy coding~~ercoder~~ (Huffman, 1952). Both methods exploit different types of redundancies to enable~~allowing. This allows~~ Deflate to achieve ~~ingachieving~~ ~~rather~~ high compression ratios. However, the computational cost of the Huffman coder is high and makes Deflate compression rather slow.

LZ4 is a dictionary coding algorithm designed to provide high compression/decompression speeds rather than a high compression ratio. ~~For this,~~ ~~I~~It does ~~not~~ this without~~make use of~~ any~~y~~ entropy coder.

Zstandard is a fast lossless compressor offering~~achieving~~ high compression ratios. It makes use of dictionary coding (repcode model~~l~~ling) and ~~of~~ a finite~~-~~-state entropy coder (tANS) (Duda, 2013). It ~~achieves~~ offers ~~similar~~ a compression ratio similar to that of~~than~~ Deflate coupled with high compression/decompression speeds.

~~We also~~This paper also evaluates Shuffle and Bitshuffle. The Shuffle~~The preprocessing step is also reversible. It reorders the data bytes or bits to enhance the lossless coding step performance. It can make use of lossless compression algorithms such as Shuffle~~ algorithm~~-, or Bitshuffle.~~ groups ~~together~~ all the data samples' first bytes together, all the~~-~~ second bytes together, etc.~~, bytes of the data samples.~~ ~~On~~ In smooth datasets, or datasets with highly correlated consecutive sample~~s~~ values, this rearrangement~~ordering~~ creates long runs of similar bytes, improving the dataset's compression~~of the dataset~~. Bitshuffle extends the concept of Shuffle to the bit level by grouping~~: it groups~~ together all the data samples' first bits, second bits, etc.~~, bits of the data samples.~~

Last, we evaluate the lossy compression algorithms Sz, Decimal Rounding and Bit Grooming. The Sz algorithm predicts~~s~~ data samples using an *n*-layers prediction model and performs an error-control quantization of the data before ~~a~~ variable length encoding. Unpredictable data samples are encoded after a binary representation analysis: the insignificant bits are truncated after ~~a~~ computation of the smallest number of mantissa bits required to achieve the specified error bound. The Decimal Rounding algorithm achieves~~performs~~ a uniform scalar quantization of the data. The quantization step is a power of 2 pre-computed so as to preserve a specific number of decimal digits. ~~The data reduction step is not reversible: data losses are introduced in this step. The strategy is to remove irrelevant data such as noise or other scientifically meaningless data. Depending on the algorithm use, this step can reduce the data volume. For instance, the Linear Packing and Sz algorithms allow reducing the data volume but not bit shaving and Bit Grooming algorithm.~~

~~One feature required for lossy the scientific data compression is the control of the amount of loss or the accuracy of the compressed data. Depending on the data, this accuracy can be expressed by an absolute or a relative error bound.~~

~~The maximum absolute error mae is defined by $mae = \max_i |\hat{s}_i - s_i|$ where the s_i are the samples values of the original dataset and the $\hat{s}_i$ are the samples values of the compressed dataset. An absolute error bound specifies the maximum absolute error e_{abs} allowed between any sample of the original and compressed data: $mae \leq e_{abs}$. The maximum relative error mre is defined by $mre = \max_i \left|\frac{\hat{s}_i - s_i}{s_i}\right|$ A relative error bound specifies the maximum relative error e_{rel} allowed between any sample of the original and compressed data: $mre \leq e_{rel}$.~~

~~The absolute error bound can be useful for data with a unique dynamic range of interest. The relative error bound can be useful for data where both very small value and very high values are of same interest.~~

~~The Decimal Rounding algorithm (also mentioned as DSD algorithm for Decimal Significant Digit) and the Bit Grooming algorithm (also mentioned as a NSD algorithm for Number of Significant Digits) proposed in (Zender, 2016a) address both cases. The Decimal Rounding algorithm respects a maximum error bound by preserving the specified number of decimal significant digits. The Bit Grooming algorithm respects a relative error bound by preserving the specified total number of significant digits. One interesting feature of these algorithms is the fact the accuracy of the compressed data can easily be interpreted: rather than defining the number of significant bits, they define the number of significant digit or the number of significant decimal digits.~~

The Bit Grooming algorithm creates a bitmask to ~~alter~~ degrade the least significant bits of the mantissa of IEEE 754 floating-point data. Given a specified total number of significant digits, nsd, the Bit Grooming algorithm tabulates the number of mantissa bits that has to be preserved to guarantee~~yguaranty~~ the specified precision of nsd digits: to guarantee ~~preserving~~ 1-6 digits of precision, Bit Grooming must retain 5, 8, 11, 15, 18, and 21 mantissa bits~~,~~ respectively. The advantage is that ~~the computation of~~ the number of mantissa bits that ~~has t~~must~~oto~~ be preserved is computed very quickly. The disadvantage is that this computation~~fast.fast. However, it~~ is not optimal. In many cases, ~~the number of~~more mantissa bits are preserved ~~is higher~~ than ~~what would have been~~ strictly necessary.

Table ~~Table~~ 1 provides ~~the~~ an example ~~on~~ using the value of π with a specified precision of ~~–~~$nsd = 4$ digits. This table reproduces some of the results ~~extracted~~ from Table 1 in (Zender, 2016a). The Bit Grooming algorithm preserves 15 mantissa bits. Table 1 shows that ~~where it would have been enough to preserve~~ only 12 bits were actually necessary.~~.~~

Optimizing the number of mantissa bits preserved ~~will have~~has a favorable impact on the compression ratios since it allows ~~for~~ more bits to be zeroed,~~ingzeroing more bits and~~ thus creating longer sequences of zero bits. In~~Thus in~~ the next section, we propose the Digit Rounding algorithm to overcome this limitation of the Bit Grooming algorithm.

3 The Digit Rounding algorithm

The Digit Rounding algorithm is similar to the Decimal Rounding algorithm in the sense that it computes a quantization factor q, which is a power of 2, in order to set bits to zero in the binary representation of the quantized floating~~-~~-point value. But it adapts the quantization factor to each sample value.

The Digit Rounding algorithm ~~makes~~ uses~~use of a~~ uniform scalar quantization with ~~a~~ reconstruction at the bin~~s~~ center:

$$\tilde{s}_i = \mathrm{sign}(s_i) \times \left(\left\lfloor \frac{|s_i|}{q_i} \right\rfloor + 0.5\right) \times q_i \tag{1}$$

where $\tilde{s}_i$ is the quantized value of ~~the~~ sample value s_i. The quantization error is bounded by:

$$|s_i - \tilde{s}_i| \le q_i/2 \tag{2}$$

The number of digits d_i before the decimal separator in ~~the~~ value s_i is:

$$d_i = \lfloor \log_{10}|s_i| + 1 \rfloor \tag{3}$$

We want to preserve nsd significant digits of ~~the~~ sample value s. This is approximately equivalent to having a rounding error of less than half the last tenth digit preserved. The quantization error shall thus be lower than or equal to:

$$|s_i - \tilde{s}_i| \le 0.5 \times 10^{d_i - nsd} \tag{4}$$

This condition guarantees that the Digit Rounding algorithm ~~to~~ always preserves a relative error lower than or equal to half the value of the least significant digit.

Combining Eq. (2) and Eq. (4), we look for the highest quantization factor q_i such that:

$$q_i/2 \le 0.5 \times 10^{d_i - nsd}$$

or ~~:~~

$$\log_{10}(q_i) \le d_i - nsd$$

Moreover, in order to lower the computational cost and increase ~~the~~ compression efficiency, we ~~look for~~seek a quantization factor that is a power of two. This allows bit-masking instead of division, and creates sequences of zero bits ~~0~~:

$$q_i = 2^{p_i} \tag{5}$$

We thus look for the greatest integer p_i such that~~:~~

$$p_i \le (d_i - nsd)\log_2 10.$$

Finally, we take ~~the~~ value p_i such that:

$$p_i = \lfloor (d_i - nsd)\log_2 10 \rfloor \tag{6}$$

The log computation in Eq. (3) is the more computationally in~~exp~~tensive, but~~.demanding. Nevertheless,~~ optimization is possible because~~as~~ only the integer part of the result is useful. The ~~optimized version implemented~~optimization consists in computing the number of digits before ~~the~~ decimal separator d from ~~the~~ binary exponent e_i and mantissa m_i of value s_i, which~~:: the value s_i~~ in binary representation is written:

$$s_i = \mathrm{sign}(s_i) \times 2^{e_i} \times m_i \tag{7}$$

~~where~~ The ~~the~~ mantissa m_i is a number between 0.5 and 1. Hence, using Eq. (3) we ~~have~~obtain:

$$d_i = \lfloor \log_{10}(2^{e_i} \times m_i) \rfloor + 1$$ or

$$d_i = \lfloor e_i \log_{10}(2) + \log_{10}(m_i) \rfloor + 1$$

The ~~value~~ $\log_{10}(m_i)$ value is tabulated. Only 5 tabulated values are used in our implementation, enough to provide a good precision. The tabulated ~~values~~ values v for $\log_{10}(m_i)$ are such that $v \le \log_{10}(m_i)$. They are provided in the Supplement. ~~It This computation, thus guaranteeing. The n~~Number d_i of significant digits before the decimal separator in ~~the~~ sample value s_i is thus approximated with the following equation:~~As $-\log_{10}(2) < \log_{10}(m_i) \leq 0$, we use the following approximation in our implementation:~~

~~$d_i \approx \lfloor (e_i - 1)\log_{10}(2) \rfloor + 1$ (7)~~

~~It provides slightly under estimated values for d_i but also a more conservative quantization allowing preserving the specified number of significant digits.~~

$$d_i \approx \lfloor e_i \log_{10}(2) + v \rfloor + 1 \qquad (8)$$

This computation slightly underestimates the values for d_i but provides a more conservative quantization, guaranteeing the specified number of significant digits. ~~This~~The optimization slightly decreases the achievable compression ratios in exchange for a much higher~~strong benefits on the~~ compression speed.

~~Finally, t~~The~~The~~ Digit Rounding algorithm is summarized in ~~Figure~~~~Table 2~~Fig. 2. We have developed an HDF5 dynamically loaded filter plugin ~~for~~ so as to apply the Digit Rounding algorithm ~~to be able to apply it on~~to NetCDF-4 or HDF5 datasets~~ formatted as NetCDF-4 or HDF5 files~~. It ~~has~~ should~~to~~ be noted that data values ~~that have been~~ rounded by the Digit Rounding algorithm can be read directly~~ read~~: there is no reverse operation to ~~the~~ Digit Rounding, and users do not need any software to read the rounded data.

Table 2~~3~~ provides the results of the Digit Rounding algorithm on the value of π with ~~a~~ specified precisions ~~of~~ nsd~~ = 4~~ varying from 1 to 8 digits. It can be compared to the ~~results of the~~ Bit Grooming results provided in Table 2 in (Zender, 2016a). For a specified precision of $nsd = 4$ digits, the~~The~~ Digit Rounding algorithm preserves 11 bits in the mantissa and sets the 12th bit to 1. Compared to the Bit Grooming algorithm, 3 more bits have been set to 0. Table 3 provides the maximum absolute error obtained with varying nsd values on an artificial dataset composed of 1,~~-~~000,~~-~~000 values evenly spaced over~~n~~ the interval [1.0, 2.0). This is the same artificial dataset ~~as the one~~ used in Table 3 in (Zender, 2016a). It shows that Digit Rounding always preserves a relative error lower than or equal to half the value of the least significant digit, *i.e.* $|s_i - \tilde{s}_i| \leq 0.5 \times 10^{d_i - nsd}$. We compare the compression ratio obtained with the Digit Rounding algorithm to that~~e compression ratio~~ obtained with the Bit Grooming algorithm ~~on~~ for the same meteorological data from MERRA re-analysis studied in (Zender, 2016a). Table 4 reports the Bit Grooming results extracted from Table 6 in (Zender, 2016a) and provides the results of the Digit Rounding algorithm. ~~In both cases, t~~The same lossless compression is employed: Shuffle and Deflate with level 1 compression. From $nsd = 7$ to $nsd = 5$, Digit Rounding and Bit Grooming provide similar compression ratios with a slight advantage for the Bit Grooming algorithm. However, from $nsd = 4$ to $nsd = 1$, the compression ratios obtained with Digit Rounding are clearly better.

The ~~next~~ following section~~s~~ first ~~provides the~~ defines~~ition of~~ the various performance metrics used hereinafter~~in the remaining of this paper~~, then studies the performance of various lossless and lossy compression algorithms~~ — ,~~including ~~the~~ Digit Rounding~~ — ,~~ when applied to~~on~~ both synthetic ~~datasets~~ and ~~on~~ real scientific datasets.

4 Performance metrics

~~We have implemented the Digit Rounding algorithm as a new HDF5 dynamically loaded filter plugin to be able to apply it on datasets formatted as NetCDF-4 or HDF5 files.~~

One of the features required for lossy ~~the~~ scientific data compression is ~~the~~ control ~~of~~ over the amount of loss, or the accuracy, of the compressed data. Depending on the data, this accuracy can be expressed by an absolute or a relative error bound. The maximum absolute error is defined by $e_{abs}^{max} = \max|\tilde{s}_i - s_i|$ where ~~the~~ s_i are the sample~~s~~ values of the original dataset and ~~the~~ $\tilde{s}_i$ are the sample~~s~~ values of the compressed dataset. An absolute error bound specifies the maximum absolute error, e_{abs}, allowed between any sample of the original and compressed data: $e_{abs}^{max} \leq e_{abs}$. The maximum relative error is defined by $e_{rel}^{max} = \max\left|\frac{\tilde{s}_i - s_i}{s_i}\right|$. A relative error bound specifies the maximum relative error, e_{rel}, allowed between any sample of the original and compressed data: $e_{rel}^{max} \leq e_{rel}$. The absolute error bound can be useful for data with a ~~unique~~ single dynamic range of interest. The relative error bound can be useful for data where both very ~~small~~ low~~value~~ and very high values are pertinent.

A near-~~ly~~nearly -exhaustive list of metrics for assessing the performance of lossy compression of scientific datasets is provided in (Tao et al., 2017b). For the sake of conciseness, ~~it has been chosen to present~~ only a few of them are presented in this paper. The following metrics ~~have been~~were chosen for this study:

- ~~the~~ compression ratio $CR(F)$ to evaluate the reduction in size ~~reduction~~ as a result of the compression. It is defined by the ratio of the original file size over the compressed file size:

$$CR(F) = \frac{filesize(F_{orig})}{filesize(F_{comp})}$$

- ~~the~~ compression speed $CS(F)$ and decompression speed $DS(F)$ to evaluate the speed of the compression and ~~of the~~ decompression. They are defined by the ratio of the original file size over the compression or decompression time:

$$CS(F) = \frac{filesize(F_{orig})}{t_{comp}}$$

$$DS(F) = \frac{filesize(F_{orig})}{t_{decomp}}$$

The compression ~~speed~~ and ~~the~~ decompression speeds are expressed in MB/s. Those ~~compression and decompression speed~~ reported in this paper ~~have been~~were obtained on a Dell T1600 with an Intel Xeon E31225 4--core~~s~~ CPU at 3.1GHz, and a 4GB memory under the RedHat 6.5 (64--bit~~s~~) OS with compression and decompression run on a single core. Parallel compression has not been considered in this work.

The following metrics were~~have been~~ chosen to assess the data degradation of the lossy compression algorithms:

- ~~the~~ maximum absolute error e_{abs}^{max} defined previously. It is used to evaluate the maximum error between the original and compressed data;

- ~~the~~ mean error $\bar{e}$ to evaluate if any bias is introduced into the compressed data. It is defined as the mean of the pointwise difference between the original and compressed data:

$$\bar{e} = \frac{1}{N}\sum_{i=0}^{N-1}(s_i - \tilde{s}_i)$$

- ~~the~~ mean absolute error $\overline{e_{abs}}$ to evaluate the mean data degradation. It is defined as the mean of the pointwise absolute difference between the original and compressed data:

$$\overline{e_{abs}} = \frac{1}{N}\sum_{i=0}^{N-1}|s_i - \tilde{s}_i|$$

- SNR to evaluate the signal to compression error ratio. It is defined by the ratio of the signal level over the root mean square compression error and~~. It~~ is expressed in decibels (dB):

$$SNR_{dB} = 20\log_{10}\left(\frac{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}{s_i}^2}}{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(s_i - \tilde{s}_i)^2}}\right)$$

These metrics are used in the ~~next~~ following sections to evaluate various lossless and lossy compression algorithms, including ~~the~~ Digit Rounding.

5 ~~4~~ Performance assessment ~~on~~ with synthetic data

~~4.1 Performance metrics~~

~~A nearly exhaustive list of metrics for assessing the performance of lossy compression of scientific datasets is provided in Z-checker (Tao et al., 2017b). For the sake of conciseness, it has been chosen to present only a few of them in this paper. The following metrics have been chosen:~~

- ~~the compression ratio $CR(F)$ to evaluate the size reduction as a result of the compression. It is defined by the ratio of the original file size over the compressed file size:~~

$$\cancel{CR(F) = \frac{filesize(F_{orig})}{filesize(F_{comp})}}$$

- ~~the compression speed $CS(F)$ and decompression speed $DS(F)$ to evaluate the speed of the compression and of the decompression. They are defined by the ratio of the original file size over the compression or decompression time:~~

$$\cancel{CS(F) = \frac{filesize(F_{orig})}{t_{comp}}}$$

$$\cancel{DS(F) = \frac{filesize(F_{orig})}{t_{decomp}}}$$

~~The compression speed and the decompression speed are expressed in MB/s.~~

~~The following metrics have been chosen to assess the data degradation of the lossy compression algorithms:~~

- ~~the maximum absolute error e_{abs}^{max} to evaluate the maximum error between the original and compressed data. It is defined as the maximum value of the pointwise absolute difference between the original and compressed data:~~

~~$e_{abs}^{max} = \max_i |s_i - \tilde{s}_i|$~~

- ~~the mean error $\bar{e}$ to evaluate if any bias is introduced in the compressed data. It is defined as the mean of the pointwise difference between the original and compressed data:~~

~~$\bar{e} = \frac{1}{N}\sum_{i=0}^{N-1}(s_i - \tilde{s}_i)$~~

- ~~the SNR to evaluate the signal to compression error ratio. It is defined by the ratio of the signal level over the root mean square compression error. It is expressed in decibel (dB):~~

~~$SNR_{dB} = 20\log_{10}\left(\frac{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}s_i^{\,2}}}{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(s_i - \tilde{s}_i)^2}}\right)$~~

5.1~~4.2~~ Analytical datasets

Synthetic datasets $s1$ and $s3D$ with known statistics ~~have been~~were generated in order to test the compression algorithms under variable conditions. ~~The dDatasetsfollowing datasets have beenwere generated,. :~~

~~s1beingis~~ Dataset $s1$ is a noisy sinusoid of 1 dimension with a maximum absolute value of 118. ~~,~~The data volume of ~~the~~ this dataset is 4MB. Dataset ~~$s3D$~~ s

~~s~~$3D$~~3D a~~ is a noisy sinusoid pulse of 3 dimensions with a maximum absolute value of 145. The data volume of this dataset is 512MB. The S~~s~~upplement ~~further~~ describes these datasets in greater detail.

~~5The signal $s1$ is a noisy sinusoid defined by:~~

~~$s1(i) = c + a_1 \times \sin\left(2\pi i \frac{f_{s1}}{f_s}\right) + n(i)$~~

~~Where c is the mean value, a_1 is the amplitude of the sinusoid, f_{s1} is its frequency and $n(i)$ is a zero mean Gaussian noise of variance 1. The signal $s1$ is generated with $c = 100$, a_1 computed so as to obtain a SNR of 20dB, and $\frac{f_{s1}}{f_s} = \frac{17}{19\times2}$. It allows having a bit more than two samples per period with a pattern reproduced every 17 periods. It is generated over $N = 2^{20}$ float sample values, each float value being encoded on 32bits. The volume of the dataset $s1$ is 4MB. The dataset and its histogram are shown in Fig. 2.~~

~~The signal $s3D$ a noisy sinusoid pulse of 3 dimensions defined by:~~

~~$s3D(i_1, i_2, i_3) = a_2 \times \frac{\sqrt{i_1^{\,2} + i_2^{\,2} + i_3^{\,2}}}{\sqrt{L^2 + M^2 + N^2}} \times \sin\left(2\pi\sqrt{i_1^{\,2} + i_2^{\,2} + i_3^{\,2}}\,\frac{f_{s3D}}{f_{ech}}\right) + n(i_1, i_2, i_3)$~~

~~Where L, M, N are the 3 dimensions of the signal $s3D$, a_2 is the amplitude of the sinusoid, f_{s3D} is its frequency and $n(i_1, i_2, i_3)$ is a zero mean Gaussian noise of variance 1~~

~~The signal $s3D$ is generated with $L = 256$, $M = 256$, $N = 2048$, a_2 computed to obtain a SNR of 40dB, and $\frac{f_{s3D}}{f_s} = \frac{17\times8}{19\times N}$ in order to have 4 periods on the main axis. It is generated over $L \times M \times N = 2^{27}$ float sample values, each float value being encoded on 32bits. The volume of the dataset $s3D$ is 512MB. The dataset and its histogram are shown in Fig. 3.~~

~~The datasets $s1$ and $s3D$ datasets have been stored into NetCDF-4 formatted files.~~

~~4.3~~5.2 Performance assessment of lossless compression methods

The lossless compression algorithms evaluated are Deflate and Zstandard with or without the Shuffle or Bitshuffle preprocessing step. ~~Moreover,~~ LZ4 is always evaluated ~~but always~~ with the Bitshuffle preprocessing step because it was imposed in the LZ4 implementation ~~of LZ4~~ we used. ~~embarks Bitshuffle.~~

We ra~~u~~n a~~run~~ lossless compression algorithm using the h5repack tool from the HDF5 library, ~~in~~ version 1.8.19, Deflate implemented in zlib 1.2.11, Zstandard ~~in~~ version 1.3.1 with the corresponding HDF5 filter available on the HDF web portal (http://portal.hdfgroup.org/display/support/Filters), and the implementation of LZ4 and Bitshuffle in ~~the~~ python package Bitshuffle-0.3.4. The compression ~~is~~ was performed by calling the h5repack tool. The S~~s~~upplement provides the command lines and options ~~that have been~~ used.

Figures ~~The compression is performed calling h5repack tool with a command line formatted as follows:~~

~~h5repack -i *in_file.nc* -o *compressed_file.h5* [--filter=*var*:params]~~

~~where *in_file.nc* is the input dataset formatted as a NetCDF-4 file and *compressed_file.h5* is the compressed dataset in HDF5 file format. The input dataset contains the *var* variable processed by one or several HDF5 filter. Table 4 provides the list of filter options used. They shall replace the filter option between brackets on previous command line.~~

~~The decompression is performed calling *h5repack* tool with a command line formatted as follows:~~

~~h5repack -i *compressed_file.h5* -o *out_file.h5* --filter=*var*:NONE~~

~~Compression and decompression has been performed on a Dell T1600 with an Intel Xeon E31225 4 cores CPU at 3.1GHz, and 4GB memory under RedHat 6.5 (64 bits) OS. Compression and decompression are run on a single core.~~

~~In order to obtain meaningful compression and decompression speed results, we employed the following process:~~

- ~~Each compression or decompression is run 10 times.~~
- ~~The elapse time (real clock time) of each run is measured.~~
- ~~The minimum and maximum times measured are removed from the list of measures~~
- ~~The mean of the remaining 8 measures provides the compression or decompression time.~~

~~Figure 3~~4 and ~~Fig 4~~ provide~~s~~ the results obtained for the compression and decompression of ~~the~~ dataset $s1$ and ~~Fig. 5 provides the results obtained for the compression and decompression of the~~ dataset $s3D$ respectively. The vertical bars represent the results for different compression levels: from 1 to 9 for ~~the~~ Deflate level *dfl_lvl*, from 1 to 22 for Zstandard level *zstd_lvl*, and only one level for LZ4.

First, it can be observed that ~~theThe~~ preprocessing steps Shuffle or Bitshuffle have a similarly ~~-~~favorable impact both on the compression ratio and on the compression/decompression speeds ~~in most cases~~.~~. Shuffle and Bitshuffle have similar effects on the compression performances.~~

Second, the ~~The cC~~compression~~compression~~ level~~s~~ parameters *dfl_lvl* and *zstd_lvl* have little influence on the compression ratio. However, the compression/decompression speeds decrease ~~with increasing~~as compression levels increase, particularly with Zstandard compression levels.~~level.~~

Third, ~~T~~the compression ratios~~ratio~~ obtained with Deflate and Zstandard are similar, but the compression speeds of Zstandard at low compression levels are far higher, ~~but~~ and the decompression speeds of Zstandard are always higher~~, and the compression speeds of Zstandard at low compression levels are far higher~~.

Fourth, ~~The compression/decompression speeds~~ obtained~~obtain with~~ Bitshuffle~~ and~~+~~–~~LZ4 provides a slightly lower compression ratio than Shuffle+Deflate or Shuffle+Zstandard, with a compression speeds similar to Shuffle+Deflate or Shuffle+Zstandard at low compression level parameters *dfl_lvl* or *zstd_lvl*~~are not in all cases~~always ~~higher than the compression/decompression speeds obtained with Bitshuffle and Zstandard at low compression level *zstd_lvl*. Nevertheless, the compression~~ ratios~~ratio obtained with Bitshuffle and LZ4 are only slightly lower than the compression ratio obtained with Bitshuffle and Zstandard at low compression level *zstd_lvl*~~.

Finally, the compression/decompression speeds obtained with Zstandard and LZ4 for the compression of ~~the~~ dataset $s3D$ are ~~by far~~much lower than that~~the one~~ achieved for the compression of ~~the~~ dataset $s1$. Further investigations are required to understand why the compression/decompression speeds are lower, but it~~. This~~ might be related to HDF5 chunking.

To summarize, ~~T~~these~~We conclude that for~~ results show that ~~the lossless compression of scientific dataset the~~ preprocessing by Shuffle or ~~of~~ Bitshuffle ~~are~~ is very helpful ~~to~~ in increasing~~increase~~ ~~the~~ compression ~~performance~~efficiency. ~~Then,~~They also show that Zstandard can provide higher compression and decompression speeds than Deflate at low compression levels~~level~~. However, on the $s3D$ dataset, we observed~~observe~~ that Zstandard compression and decompression speeds are lower than those~~the one~~ obtained with Deflate. Therefore, Deflate and Zstandard are ~~thus~~ both options~~option~~ to consider for the lossless compression of scientific datasets as long as they follow~~dataset but always with~~ the Shuffle or Bitshuffle preprocessing step.

5.3~~34.4~~ Performance assessment of lossy compression methods

The lossy compression algorithms evaluated are error-bounded compression algorithms. They can constrain either ~~or both~~ the maximum absolute error or the maximum relative error, or both. The compression algorithms evaluated are Sz, Decimal R~~r~~ounding, Bit Grooming and the Digit Rounding algorithm introduced in this paper.

~~Sz compression algorithms evaluated are Sz, Bit Grooming and the Digit Rounding algorithm introduced in this paper.~~

~~Sz compression algorithm has been designed to work in both error-bounded modes. Bit Grooming is declined in two algorithms: the DSD algorithm (for number of decimal significant digits) and the NSD algorithm (for number of significant digits). The DSD algorithm (also called decimal rounding algorithm)~~ The Sz compression algorithm works in both error-bounded modes. Decimal Rounding allows ~~preserving~~ a specific number of decimal digits to be preserved. In this sense, it bounds the maximum absolute error. ~~The NSD~~ Bit Grooming ~~algorithm~~ allows ~~preserving~~ a specific number of significant digits to be preserved.~~.~~ In this sense, it bounds the maximum relative error. ~~As~~ Like the ~~the NSD~~ Bit Grooming algorithm, ~~the~~ Digit Rounding ~~algorithm allows~~ preserves~~ingpreserving~~ a specific number of significant digits and bounds the maximum relative error.

~~.~~

~~Bit Grooming and Digit Rounding algorithms does not compress the data. They only alter the data to make it more compressible. Thus, lossless compression steps are required afterward. By default, Sz algorithm embark Deflate. Nevertheless, it is possible to configure Sz and deactivate Deflate to use other lossless compression algorithms.~~

We ~~urun~~ ran Sz ~~in~~ version ~~1.4.11.1~~2.1.1 using the h5repack tool and ~~call through its~~ Sz HDF5 filter plugin, ~~and~~ applying the Deflate lossless compression algorithm integrated ~~to~~ in the Sz software. We ~~urun~~ran the Decimal Rounding and Bit Grooming algorithms using NCO ~~in~~ version 4.7.9, applying Shuffle and Deflate compression in the call to the NCO tool~~0~~. Last, we ~~urun~~ ran the Digit Rounding algorithm using the h5repack tool and custom implantation of the algorithm in an HDF5 plugin filter. The S~~s~~upplement provides the command lines and options ~~that have been~~ used.

~~Sz compression is performed calling h5repack tool with a command line formatted as follows:~~

~~h5repack -i *in_file.nc* -o *compressed_file.h5* --filter=*var*:UD=32017,0~~

~~Sz compression is configured via the *sz.config* file located in the directory from where h5repack is called. In this configuration file, quantization_intervals is set to 256 and the *szMode* is set to SZ_BEST_SPEED to achieve high speed compression. The *gzipMode* is set to Gzip_NO_COMPRESSION to deactivate Deflate compression. The *errorBoundMode* is set to ABS, or to PW_REL, to achieve respectively absolute error bounded compression, or relative error bounded compression. In the absolute error bounded compression mode, the *absErrBound* parameter is configured to achieve the desire maximum absolute error. In the relative error bounded compression mode, the parameter *pw_relBoundRatio* is configured to achieve the desire maximum relative error.~~

~~Bit Grooming compression is performed calling the *ncks* tool from NCO toolkit. The DSD algorithm is run with the following command line (note the period before the *dsd* parameter):~~

~~ncks -4 -L *dfl_lvl* --ppc *var=.dsd in_file.nc compressed_file.nc*~~

~~The NSD algorithm is run with the following command line:~~

~~ncks -4 -L *dfl_lvl* --ppc *var=nsd in_file.nc compressed_file.nc*~~

~~In all cases, the decompression is performed calling h5repack tool with a command line formatted as follows:~~

~~h5repack -i *compressed_file.h5* -o *out_file.h5* --filter=*var*:NONE~~

~~4.4~~5.3.1 Performance comparison in ~~the~~ absolute error-~~-~~bounded compression mode

This section compares the performance of the absolute error-bounded compression algorithms: Sz and Decimal Rounding. The results reported were obtained by applying ~~In order to measure the compression ratio and the compression speeds,~~ Sz ~~has been~~was configured with the options SZ_BEST_SPEED and Gzip_BEST_SPEED.~~,~~ Shuffle and Deflate with *dflt_lvl* = 1 ~~are~~ were applied after ~~the~~ Decimal Rounding.~~Zstandard with *zstd_lvl* = 5 has been applied after Sz and Shuffle and Zstandard with *zstd_lvl* = 5 has been applied after Bit Grooming. This compression level provides a good trade-off between compression speed and compression ratio.~~

~~Only Shuffle is only applied after Bit Grooming. Indeed, experiments have shown that Shuffle or Bitshuffle preprocessing do not increase the compression ratio when applied after Sz, and Bitshuffle provide lower compression ratio than Shuffle when applied after Bit Grooming.~~

Table 5 compares the ~~compression performance~~ results obtained in ~~the~~ absolute error-~~-~~bounded compression mode for $e_{abs} = 0.5$. This corresponds ~~corresponding~~ to *dsd* = 0 ~~decimal~~ significant decimal digits preserved, or in other words, a rounding to the nearest integer.

~~Sz compression is performed calling h5repack tool with a command line formatted as follows:~~

~~h5repack -i *in_file.nc* -o *compressed_file.h5* --filter=*var*:UD=32017,0 --filter=*var*:UD=32015,1,5~~

~~With the *absErrBound* parameter set to 0.5 in the *sz.config* file located in the directory from where h5repack is called.~~

~~Bit Grooming compression is performed successively calling *ncks* and h5repack tool with command lines formatted as follows:~~

~~ncks -4 -L 0 --ppc *var=.dsd in_file.nc bitgroomed_file.nc*~~

~~h5repack -i *bitgroomed_file.nc* -o *compressed_file.h5* --filter=*var*:SHUF --filter=*var*:UD=32015,1,5~~

Both Sz and Decimal Rounding ~~Bit Grooming~~ algorithms respect the specified maximum absolute error value. Moreover, none introduces a statistical bias: the mean absolute errors of both algorithms— ~~,~~ not shown in this table— ~~,~~ are very close to zero. The errors introduced by these two algorithms are similar. However, it can be ~~shown~~ seen that Decimal~~Bit Grooming~~ Rounding provided a higher compression ratio than Sz ~~on~~ for~~the~~ dataset $s1$.~~, while the compression speeds are similar.~~ On the ~~contrary~~other hand, Sz provided a~~provide~~ higher compression ratio ~~and than Bit Grooming on~~ for ~~the~~ dataset $s3D$. Sz may perform better on ~~the~~ dataset $s3D$ because it is smoother than ~~the~~ dataset $s1$. Indeed, Sz integrates~~makes use of~~ a prediction step. ~~The~~ This prediction might often fail because dataset $s1$ ~~being~~is ~~highly~~ very noisy~~, Sz prediction might often fail~~. This ~~can~~ may explain the lower compression ratio ~~on~~ for $s1$ -dataset. ~~On the contrary,~~ Decimal Rounding, however, does not make~~s~~ any predictions, which may~~. This can~~ explain why it achieves a better compression than Sz ~~on~~ for~~the~~ dataset $s1$. The lower compression/decompression speeds obtained with Sz on the dataset $s3D$ are not well understood and might be related to HDF5 chunking as previously mentioned.

Figure 5~~6~~ compares ~~the performances of~~ Sz and Bit Grooming algorithms~~algorithms~~ in terms of SNR versus compression ratio. This figure ~~has been~~was obtained with the following parameters:

- For the Sz algorithm, the *absErrBound* parameter ~~is~~ was successively set to 5e-5, 5e-4, 5e-3, 5e-2, 5e-1, 5;
- For the Decimal Rounding ~~Bit Grooming~~ algorithm, the *dsd* parameter ~~is~~ was successively set to 4, 3, 2, 1, 0, -1.

~~As for the results reported in Table 5, Zstandard with *zstd_lvl* = 5 has been applied after Sz and Shuffle and Zstandard with *zstd_lvl* = 5 has been applied after Bit Grooming.~~

~~On the~~For dataset $s1$, ~~the~~ Decimal Rounding ~~provides~~has a ~~Bit Grooming algorithm provides better~~ higher ~~compression performance~~ SNR than Sz for a given compression ratio.~~ except for very high compression ratio (*dsd* ≤ -1 or *absErrBound* ≥ 5).~~ On the contrary, ~~on~~ for~~the~~ dataset $s3D$, Sz~~the Bit Grooming algorithm~~ ~~provides~~ has a higher SNR than Decimal Rounding for a given compression ratio. Both~~ better compression performance than Sz but only for low compression ratio (*dsd* ≥ 2 or *absErrBound* ≤ 5e-3).~~

~~We conclude that both~~ Sz and Bit Grooming algorithms ~~are~~ seem~~s~~ valuable for ~~the compression in the absolute~~ error-bounded compression~~ mode. Bit Grooming tend to provide better performance at low compression ratios while Sz tends to provide better performance at higher compression ratios but the limit depends on the dataset~~.

5.3.23~~4.4.2~~ Performance comparison in ~~the~~ relative error-bounded compression mode

This section compares the performance of the relative error-bounded compression algorithms: Sz, Bit Grooming, and Digit Rounding. The results reported ~~have been~~were obtained by applying Sz configured with the options SZ_DEFAULT_COMPRESSION and Gzip_BEST_SPEED. Shuffle and Deflate with *dflt_lvl*=1 ~~have been~~were applied after the Bit Grooming and Decimal Rounding algorithms.

~~As for the performance comparison in the absolute error bounded compression mode, Zstandard with *zstd_lvl* = 5 has been applied after Sz and Shuffle and Zstandard with *zstd_lvl* = 5 has been applied after Bit Grooming in order to measure the compression ratio and the compression speeds.~~

We first focus on the results obtained ~~on the~~with dataset $s1$.

~~Table 6 compares the compression errors obtained in the relative error- bounded compression mode.~~The number of significant digits—*nsd* parameter—in the Bit Grooming and ~~in the~~ Digit Rounding algorithms ~~is~~ was set to 3. As the maximum absolute value in the *s1* dataset is 118, the maximum absolute error should be lower than 0.5. In order to be able to compare Sz configured with a relative error bound with those algorithms, we configured the relative error bound to obtain a maximum absolute error of 0.5: the *pw_relBoundRatio* parameter in Sz was set to 0.00424. The results are provided in Table 6. It can be observed that all three algorithms respect the maximum absolute error of 0.5, which corresponds for dataset $s1$ to a relative error of 0.00424. On this dataset, Sz provides higher compression ratio and compression speed than the other two algorithms. Bit Grooming is too conservative. It preserves more mantissa bits than strictly necessary to achieve the required precision. This behavior is illustrated in Table 1 with the value of π. In contrast, Digit Rounding adapts the quantization step to each value of the input dataset. Doing so, it can achieve the required precision while preserving less mantissa bits than Bit

Grooming does. This results both in a higher compression ratio but also in higher errors than Bit Grooming. Results obtained for Bit Grooming with *nsd* = 2 are also provided for completeness. With this parameter, Bit Grooming provides slightly higher compression ratio and compression speed than Digit Rounding does.

~~The algorithms have been configured in order to obtain a maximum absolute error of 0.5. As the maximum absolute value in *s1* dataset is 118, the *pw_relBoundRatio* parameter in Sz is~~ was ~~set to 0.00424It can be observed that all three algorithms respect the maximum absolute error of 0.5, which~~, ~~for and the dataset *s1*,~~ ~~corresponds for dataset *s1* to a relative errornumber of 0.00424. However, as previously mentioned,significant digits *nsd* parameter in the Bit Grooming and in the Digit Rounding algorithm is set to 3 in Table 6. However, as the Bit Grooming algorithm is too conservative.~~ ~~, results with *nsd* = 2 are also provided.~~

~~Sz compression is performed calling h5repack tool with a command line formatted as follows:~~

~~h5repack -i *in_file.nc* -o *compressed_file.h5* --filter=*var*:UD=32017,0 --filter=*var*:UD=32015,1,5~~

~~With the *pw_relBoundRatio* parameter set to 0.00424 in the *sz.config* file located in the directory from where h5repack is called.~~

~~Bit Grooming compression is performed successively calling ncks and h5repack tool with command lines formatted as follows:~~

~~ncks -4 -L 0 --ppc *var=nsd in_file.nc bitgroomed_file.nc*~~

~~h5repack -i *bitgroomed_file.nc* -o *compressed_file.h5* --filter=*var*:SHUF --filter=*var*:UD=32015,1,5~~

~~Digit Rounding is performed calling h5repack tool with a command line formatted as follows:~~

~~h5repack -i *in_file.nc* -o *compressed_file.h5* --filter=*var*:UD=*digitRoundingID*,1,3 --filter=*var*:UD=32015,1,5~~

~~It can be observed in Table 6 that all three algorithms respect the relative error bound specified. However, as previously mentioned the Bit Grooming algorithm is too conservative. The same is observed with the Digit Rounding algorithm for the compression of the dataset *s*1. The quality obtained with the Digit Rounding algorithm is similar to the one obtained with the Bit Grooming. Nevertheless, the compression ratio is higher.~~

Figure 6~~7~~ (left) compares ~~the performances of~~ Sz, Bit Grooming, and Digit Rounding algorithms in terms of SNR versus compression ratio. This figure has been obtained with the following parameters:

- For the Sz algorithm, the *pw_relBoundRatio* parameter ~~is~~ was successively set to ~~4.24e-6,~~ 4.24e-5, 4.24e-4, 4.24e-3~~, 4.24e-2, 4.24e-1~~;
- For the Bit Grooming algorithm, the *nsd* parameter ~~is~~ was successively set to 6, 5, 4, 3, 2, 1;
- For the Digit Rounding algorithm, the *nsd* parameter ~~is~~ was successively set to 6, 5, 4, 3, 2, 1.

All~~As for the results reported in Table 6, Zstandard with *zstd_lvl* = 5 has been applied after Sz and Shuffle and Zstandard with *zstd_lvl* = 5 has been applied after Bit Grooming and Digit Rounding algorithms.~~

~~The Digit Rounding~~All three algorithms provide~~provides~~ similar SNR versus ~~better~~ compression ~~performance~~ ratios results, ~~than Sz or Bit Grooming~~with a slight advantage for the Bit Grooming algorithm.~~. At high compression ratio, Sz provides similar performance as the Digit Rounding algorithm.~~

Figure ~~8~~7 (left) compares the compression ratio obtained as a function of ~~the~~ parameter *nsd*, which is the user-~~-~~specified number of significant digits~~digit~~. Even though~~if the~~ *nsd* is not a parameter of the Sz algorithm, we ~~made the correspondence between~~ related the *pw_relBoundRatio* ~~and~~ to the *nsd* parameters for ~~the~~ dataset $s1$ (i.e. *pw_relBoundRatio* = 4.24e^{-nsd}) and plotted~~plot~~ the compression ratio obtained with the Sz algorithm on the same figure.

It can be seen that, whatever the *nsd* specified by the user, the compression ratios~~ratio~~ obtained with ~~the~~ Digit Rounding are higher than the compression ratio obtained with the Bit Grooming algorithm. It can also be seen that the compressions~~compression~~ obtained with the Sz algorithm are even higher.

We now focus on the results obtained with~~on the~~ dataset $s3D$. The number of significant digits—*nsd* parameter—in the Bit Grooming and ~~in the~~ Digit Rounding algorithms ~~is~~ was set to 3.

As the maximum absolute value in the $s3D$ dataset is 145, the *pw_relBoundRatio* parameter in Sz ~~is~~ was set to 0.00345. Results are provided in Table 7. ~~and the number of significant digits *nsd* parameter in the Bit Grooming and in the Digit Rounding algorithm is set to 3~~ ~~in Table 7.~~

It can be observed ~~in Table 7.~~ ~~in Table 7~~ that all three algorithms ~~respect~~ comply with the relative error bound specified. ~~However, on this dataset, Sz algorithm is twice too conservative. That is why, results obtained with *pw_relBoundRatio* = 0.0069 are also provided in order to obtain a maximum absolute error of 0.5.~~However, as previously mentioned, the Bit Grooming algorithm is too conservative.~~iswith of~~ This~~at~~ is why results obtained with *nsd* = 2 are also provided. On this dataset, Sz provides higher compression ratio than the other two algorithms but lower compression speed than Bit Grooming. At *nsd* = 3, Digit Rounding provides slightly higher compression ratio than Bit Grooming but with lower compression speed. ~~On the contrary~~~~In contrast~~ ~~The compression ratio obtained with the Digit Rounding algorithm is higher than the one obtained with Sz.~~

Figure ~~6~~7 (right) compares ~~the performances of~~ Sz, Bit Grooming, and Digit Rounding algorithms in terms of SNR versus compression ratio. This figure has been obtained with the following parameters:

- For the Sz algorithm, the *pw_relBoundRatio* parameter ~~is~~ was successively set to ~~6.9e-6,~~ 3.45~~6.9~~e-5, ~~6.9~~ 3.45e-4, ~~6.9~~3.45e-3~~;~~, ~~6.9e-2, 6.9e-1~~
- For the Bit Grooming algorithm, the *nsd* parameter ~~is~~ was successively set to 6, 5, 4, 3, 2, 1;
- For the Digit Rounding algorithm, the *nsd* parameter ~~is~~ was successively set to 6, 5, 4, 3, 2, 1.

The~~As for the results reported in Table 7, Zstandard with *zstd_lvl* = 5 has been applied after Sz and Shuffle and Zstandard with *zstd_lvl* = 5 has been applied after Bit Grooming and Digit Rounding algorithms.~~

~~For the dataset *s3D*, the~~ Bit Grooming and Digit Rounding algorithms provide similar compression ratios, but even higher compression ratios are obtained with Sz. ~~algorithm provides better compression performance than Sz. Nevertheless, the Digit Rounding algorithms provides compression performance very closed to the one of the Bit Grooming algorithm.~~

Figure 7~~8~~ (right) compares the compression ratio obtained as a function of the ~~parameter~~ *nsd* parameter, which is the user-specified number of significant digits~~digit~~. As for dataset $s1$, we ~~made~~ related~~the correspondence between the~~

pw_relBoundRatio ~~and~~ to the *nsd* parameters for ~~the~~ dataset *s3D* (i.e. *pw_relBoundRatio* = 3.45e45~~6.9e~~$^{-nsd}$) and plotted~~plot~~ the compression ratio obtained with the Sz algorithm on the same figure.

~~On the dataset *s3D*, and w~~Whatever the *nsd* specified by the user, the compression ratios obtained with the Digit Rounding algorithm are higher than the compression ratio obtained with the Bit Grooming algorithm. The compression ratios obtained with Sz are even higher. ~~or Sz.~~

Those results show that the ~~We conclude that in most cases,~~ Digit Rounding algorithm can be competitive with ~~is superior to~~ the Bit Grooming and Sz algorithms in ~~the~~ relative error~~-~~-bounded compression mode. It is thus applied to real scientific datasets in the next section.

6~~5~~ Application to scientific datasets

~~lossless~~~~Lossless and lossy algorithms are now evaluated for the compression of scientific mission data~~ from~~: CFOSAT and SWOT.~~

6~~5~~.1 Application to a CFOSAT dataset

~~The~~ CFOSAT is a cooperative program ~~is carried out through cooperation~~ between the French and Chinese sSpace a~~Space A~~gencies (CNES and CNSA respectively). CFOSAT is designed to~~aims at~~ characterize~~ing~~~~characterizing~~ the ocean surfaces to better model and predict ~~the~~ ocean states, and improve ~~the~~ knowledge ~~in~~ of ocean/atmosphere exchanges. ~~The~~ CFOSAT products will help ~~for~~ marine and weather forecasting~~forecast~~ and will also be used to monitor the~~for~~ climate ~~monitoring~~. The CFOSAT satellite will carry two scientific payloads— ~~:~~ SCAT, a wind scatterometer; ~~,~~ and SWIM, a wave scatterometer—for the ~~to allow a~~ joint characterization of ocean surface winds and waves. The SWIM (Surface Wave Investigation and Monitoring) instrument delivered by CNES is dedicated to ~~the~~ measuring~~ement~~~~measurement of~~ the directional wave spectrum (density spectrum of wave slopes as a function of direction and wavenumber of the waves). The CFOSAT L1A product contains calibrated and geocoded waveforms~~waveform~~.

Currently, the baseline for ~~the~~ compression of the CFOSAT L1A product involves a ~~“~~clipping~~”~~ method as a data reduction step, ~~the~~ with Shuffle preprocessing and Deflate lossless coding with a compression level *dfl_lvl* of 3. ~~The c~~Compression~~compression~~ with a clipping method~~“clipping”~~ is like~~n to a~~ compression in an absolute error~~-~~-bounded mode. It defines the least significant digit (*lsd*) and ‘~~“~~clips’~~“clips”~~ the data to keep only *lsd* decimal digits. The *lsd* is defined specifically for each dataset variable. The full list is provided in the Supplement with all the command lines and parameters used for running the compression methods described in this section ~~of the dataset~~.~~.~~

We studied~~study~~ the ~~performance of the~~ following ~~alternative br~~ compression methods:

- CFOSAT clipping followed by Shuffle and Deflate (*dflt_lvl* = 3): the baseline for the compression of CFOSAT datasets;

- CFOSAT “clipping” ~~method~~ followed by Shuffle and Zstandard (~~with a compression level~~ *zstd_lvl* = 2) ~~compression level of 1 or 2~~ for higher~~toto achieve favor~~ compression speeds;
- ~~Bit Grooming (in the absolute error bounded compression mode) followed by Shuffle and Deflate or Zstandard.~~Sz ~~br~~ followed by Deflate in the absolute error bounded mode;
- Decimal Rounding followed by Shuffle and Deflate (*dflt_lvl* = 1);
- Bit Grooming (*nsd* = 8) ~~(in the absolute error bounded compression mode)~~ followed by Shuffle and ~~Deflate or Zstandard~~Deflate (*dflt_lvl* = 1);~~.~~
-
- Digit Rounding (*nsd* = 8) ~~br~~ followed by Shuffle and Deflate (*dflt_lvl* = 1).

We first focused on the *ground_range_5* variable of the CFOSAT L1A product. This variable is an array of 18451×3215 values in double precision. The data volume is 452.58-MB (uncompressed). The CFOSAT “clipping” method defines an *lsd* of 3 for this~~at~~ variable. In ~~the~~ absolute error~~-~~bounded mode, ~~Bit Grooming~~Decimal Rounding is~~has been~~ configured to keep the same number of decimal digits as CFOSAT “clipping”: *dsd* = .3; ~~on each variable: *nsd* = *lsd*.~~Sz is configured with *absErrBound* = 5e-4. In ~~the~~ relative error~~-~~bounded mode, Bit Grooming and Digit Rounding are configured with *nsd* = 8 ~~while~~. The c~~C~~ompression results are provided in Table 8.

Compared to the CFOSAT baseline compression, Zstandard compression is more than twice faster while offering a similar compression ratio. On this dataset, the use of Sz instead of the CFOSAT Clipping method increases the compression ratio by a factor of 11. Sz prediction step seems to be very efficient on this dataset. Decimal Rounding increases the compression ratio by a factor of 2.5 “only”, but provides the fastest decompression. In the relative error-bounded mode, Digit Rounding provides a higher compression ratio than Bit Grooming but lower compression/decompression speeds.

~~Bit Grooming has been configured to keep the same number of decimal digits as CFOSAT “clipping” on each variable: *nsd* = *lsd*.~~

~~Unfortunately, Sz crashes on the compression of CFOSAT or SWOT datasets. That is why, no results with Sz are provided in the following tables.~~

The results for the compression of the full~~a~~ CFOSAT L1A product of 7.34GB (uncompressed) are provided in Table 9~~8~~.

The maximum absolute error and the mean absolute error are not provided because this dataset contains several variables compressed with different parameters. ~~Supplementthe br~~ Compared to the CFOSAT baseline compression, Zstandard increases the compression speed ~~of~~ by about 40% while offering a similar compression ratio. It was not possible to apply Sz compression on the full dataset since Sz configuration file has to be modified to adapt the *absErrBound* to the *lsd* defined for each dataset variable. The way around this entails processing each variable one after the other. Sz provides a compression ratio almost 3 times higher than the baseline with faster compression and decompression. Decimal Rounding is configured on a per-variable basis to keep the precision required by the scientists on each variable. ~~The use of Bit Grooming instead of~~

~~the CFOSAT "Clipping" method~~ It increases the compression ratio by a factor of 1.8~~2,~~ with twice faster compression and decompression compared to the baseline~~but decreases the compression speed by 40%~~. The compression ratios achieved with Bit Grooming or Digit Rounding in the relative error-bounded mode are lower. This is not the mode targeted for the compression of CFOSAT datasets. The usability of Sz being reduced by the fact that the error bound cannot be easily configured to achieve the precision required variable per variable, our recommendation is to use the Decimal Rounding algorithm. It achieves faster and more effective compression than CFOSAT Clipping method and bounds the absolute errors. ~~The decompression speeds are similar for all~~ the ~~solutions~~ tested. ~~Our recommendation is thus to use the Bit Grooming algorithm with Zstandard coding rather than the CFOSAT "Clipping" method with Deflate coding to achieve~~ a ~~high compression ratio on~~ for ~~this CFOSAT dataset, at the price of a lower compression speed.~~

6~~5~~.2 Application to SWOT datasets

The Surface Water and Ocean Topography Mission (SWOT) is a partnership between NASA and CNES, and continues~~continue~~ the long history of altimetry missions with an innovative instrument known as~~:~~ KaRin, which is a Ka band synthetic aperture radar. The launch is foreseen for 2021. SWOT addresses both oceanographic~~oceanography~~ and hydrological~~hydrology~~ communities, accurately measuring ~~with a high accuracy~~the water level of ~~the~~ oceans~~ocean~~, rivers, and lakes.

SWOT has two processing modes, ~~of processing and thus~~so two different types of products are generated: ~~the~~ high-resolution products~~,~~ dedicated to hydrology, and low~~-~~resolution products mostly dedicated to oceanography. The Pixel Cloud product (called L2_HR_PIXC) contains data from the KaRi~~I~~n instrument's high-resolution (HR) mode~~ of the KaRIn instrument~~. It contains information on the pixels that are detected as being over water. This product is generated when~~rewhere~~ the HR mask is turned on. The Pixel Cloud product is organized into sub-orbit tiles for each swath and each pass, and this is an intermediate product between the L1 Single Look Complex products and the L2 lake/river ones. The product granularity is a tile ~~of~~ 64 km~~64km~~ long in the along-track direction, and it covers either the left or ~~the~~ right swath (~60 km~~60km~~ wide).

The compression ~~performance is was evaluated on two~~of two different datasets was evaluated:

- A simplified simulated SWOT L2_HR_PIXC pixel cloud product of 460MB (uncompressed);
- A realistic and representative SWOT L2 pixel cloud dataset of 199MB (uncompressed).~~ in which only few attributes may be missing of 199MB (uncompressed).~~

The current baseline for the compression of the simplified simulated SWOT L2 pixel cloud product involves ~~the~~ Shuffle preprocessing and Deflate lossless coding with a compression level *dfl_lvl* of 4. However, the compression method for the official SWOT L2 pixel cloud product has not yet been defined. A required precision is defined by the scientists as a number of significant digits (*nsd*) for each dataset variable. The full list is provided in the Supplement.

We studied the following lossless or relative error bounded compression methods:

- Shuffle and Deflate (*dflt_lvl* = 4): the current baseline for the compression of SWOT datasets;

~~We studiedystudy the performance of the following compression methods:~~

- ~~Shuffle and Deflate;~~
- Shuffle and Zstandard (*zstd_lvl* = 2) lossless alternative;
- Sz with Deflate in the relative error bounded mode;
- Bit Grooming followed by Shuffle and Deflate (*dflt_lvl* = 1);
- Digit Rounding followed by Shuffle and Deflate (*dflt_lvl* = 1).

We first focused on the *height* variable of the SWOT L2_HR_PIXC pixel cloud product. This variable is a list of 1,421,888 values in double precision. The data volume is 10.85MB (uncompressed). A precision of 6 significant digits is required for this variable (*nsd* = 6). Sz is configured in the relative error bounded mode with *pw_relBoundRatio* = 5e-6. Bit Grooming and Digit Rounding are configured with *nsd* = 6. The results are provided in Table 10. Compared to the SWOT baseline compression, Zstandard compression is more than 10 times faster while offering a similar compression ratio. On this dataset, Digit Rounding provides the highest compression ratio with compression/decompression speeds similar to the one obtained with Bit Grooming. The lowest errors are obtained with Bit Grooming but with a compression ratio slightly lower than Digit Rounding. The compression ratio obtained with Sz is even lower.

Next we focused on the *pixel_area* variable of the representative SWOT L2 pixel cloud product. This variable is a list of 1,300,111 values in double precision. The data volume is 9.92MB (uncompressed). A precision of 11 significant digits is required for this variable (*nsd* = 11). Sz is configured in the relative error bounded mode with *pw_relBoundRatio* = 5e-9 only because it cannot achieve higher precision. Bit Grooming and Digit Rounding are configured with *nsd* = 11. The results are provided in Table 11. Compared to the SWOT baseline compression, Zstandard compression is more than 7 times faster while offering a similar compression ratio. Sz provides the highest compression ratio but does not allow achieving the required precision of 11 digits. Moreover, in this configuration Sz compression is very slow. As for the *height* variable, Digit Rounding provides the highest compression ratio with compression/decompression speeds similar to the one obtained with Bit Grooming. The lowest errors are obtained with Bit Grooming but with a compression ratio lower than Digit Rounding.

Table 12 provides the results of the compression of the full simulated SWOT L2_HR_PIXC pixel cloud product. The maximum absolute error and the mean absolute error are not provided because this dataset contains several variables compressed with different parameters.~~Bit Grooming (in the absolute error- bounded compression mode) followed by Shuffle and Deflate or Zstandard;.~~

~~Bit Grooming (in the relative error- bounded compression mode) followed by Shuffle Zstandard;;~~

~~Bit Grooming (in the relative error bounded compression mode) followed by Shuffle Zstandard; ;.~~

~~Digit Rounding followed by Shuffle Zstandard.~~

~~Bit Grooming and Digit Rounding have been configured on a per- variable basis to keep the precision required by the scientists on each variable.~~

~~It was not possible to evaluate the compression time needed to compress the datasets using the Digit Rounding algorithm because *h5repack* only allows defining filters parameters~~ to be defined ~~for a small number of variables. The way around~~ this,

~~in order to compute the compression ratio, has entailsbeen to processingprocess each variable one after the other. Nevertheless, we observed similar speeds for the compression/decompression of the largest variable of in this dataset using the Bit Grooming algorithm in the relative error br-bounded mode or the Digit Rounding algorithm.~~

~~The results for the compression of the simplified simulated SWOT L2 pixel cloud product are provided in Table 9.~~

Compared to the SWOT ~~prototype~~ baseline compression, Zstandard increases ~~more than 5 times~~ the compression speed by over 5 times, while offering a similar compression ratio. Sz compression was not applied because it does not allow achieving the high precision required on some variables. Bit Grooming and Digit Rounding was configured on a per-variable basis to keep the precision required by the scientists on each variable. Compared to the baseline, Bit Grooming and ~~The use of Bit Grooming in the absolute or relative error br-bounded mode, or the use of the~~ Digit Rounding ~~algorithm, increases~~increase the compression respectively by ~~ratio by more~~20% and ~~than~~ 30% with similar compression speeds and faster decompression. ~~, but divides the compression speed by more than 3 the compression speed. The decompression speeds are similar for all the solutions tested. Our recommendation for the compression of this dataset is thus to use of Shuffle and Zstandard in lossless mode to achieve a very high compression speed, or either the Bbit bit grooming Grooming or the Digit Rounding algorithm to achieve a slightly higher compression ratio at the price of a lower compression speed.~~

The results for the compression of the representative SWOT L2 pixel cloud product are provided in Table 13~~0~~.

Compared to ~~Deflate~~the baseline, Zstandard compression is nearly 4 times faster~~increases by more than 2.5 times the compression speed~~ while offering a similar compression ratio. ~~The use of the~~ Bit Grooming increases the compression ratio by 29% with higher compression speed. And Digit Rounding increases the compression ratio by 34% with slightly lower compression speed than Bit Grooming. Bit Grooming and Digit Rounding provides the ~~Bit Grooming algorithm in the absolute error bounded mode increases more than 2 times the compression ratio by over twice but reduces the compression speed. The compression ratios obtained in the relative error bounded mode, either with the Bit Grooming algorithm or the Digit Rounding algorithms, are not as high. The~~fastest decompression ~~decompression speeds are similar for all the solutions tested~~. Our recommendation for the compression of ~~this dataset~~SWOT datasets is thus to use the ~~Bit Grooming~~Digit Rounding algorithm ~~in the absolute error bounded mode~~ to achieve high compression, at the price of a lower compression speed than the lossless solutions, considering that for SWOT the driver is product size ~~is a driver~~, and ~~considering~~ taking into account the ratio~~ration~~ between compression time and processing time.

7~~6~~ Conclusions

~~We have studied~~This study ~~investigated~~evaluated ~~the performance of~~ lossless and lossy compression algorithms both on synthetic datasets and on realistic simulated datasets of future science~~tifiesscientific~~ satellites. The compression methods ~~have been~~were ~~executed~~ applied using NetCDF-4 and HDF5 tools.

~~It has been shown that for the lossless compression of scientific datasets,dataset the compression performance is increased when preprocessing by Shuffle orfof Bitshuffle are used for preprocessingis very helpful to increase the compression~~

~~performance.~~ It has been shown that the impact of ~~T~~the compression level options of Zstandard or Deflate ~~have lower impacts~~ on the compression ratio achieved is not significant compared to the impact of the Shuffle or Bitshuffle preprocessing. However, high compression levels ~~but~~ can significantly reduce the compression speed. ~~Low compression levels are thus a good choices if the goal is to achieve a high compression speed with a satisfactory compression ratio. Zstandard can provide similar as higher a compression speed than as Deflate or LZ4 with similar compression ratios. However, on the three 3-dimensional dataset, we have observed that Zstandard compression and decompression speeds are lower than the one obtained with Deflate. Depending on the dataset,~~ Deflate and Zstandard with low compression levels are ~~thus~~ both reasonable options to consider for the compression of scientific datasets, but must always follow a~~with~~ Shuffle or Bitshuffle preprocessing step. It has been shown that Zstandard can speed-up the compression of CFOSAT and SWOT datasets compared to the baseline solution based on Deflate.

The l~~L~~ossy compression of scientific datasets can be achieved in two different error-~~-~~bounded modes: ~~the~~ absolute and relative error-~~-~~bounded. Four algorithms have been studied: Sz, Decimal Rounding, Bit Grooming and Digit Rounding. One useful feature of the last three is that the accuracy of the compressed data can easily be interpreted: rather than defining an absolute or a relative error bound, they define the number of significant decimal digits or the number of significant digits. ~~mode. Sz and Bit Grooming algorithms can work in both modes.~~ In ~~the~~ absolute error-~~-~~bounded mode, Sz provide higher compression ratios than Decimal Rounding on most datasets. However for the compression of real scientific datasets, its usability is reduced by the fact that only one error bound can be set for all the variables composing the dataset. It cannot be easily configured to achieve the precision required variable per variable.~~both Sz and DecimalBit Grooming algorithms are competitive.~~ This is why we rather recommend the Decimal Rounding algorithm to achieve fast and effective compression of the CFOSAT dataset. ~~Bit Grooming tends to provide higher SNR than Sz at low compression ratios while Sz tends to provide higher SNR than Bit Grooming at higher compression ratios.~~

In ~~the~~ relative error-~~-~~bounded mode, the Digit Rounding algorithm introduced in this work ~~is more efficient inprovides higher~~ provides higher compression ratios ~~efficiency~~ than the Bit Grooming algorithm from which it derives, but with lower compression speed. Sz can provide even higher compression ratios but fails achieving the high precision required for some variables. This is why we rather recommend the Digit Rounding algorithm to achieve relative error bounded compression of SWOT datasets with a compression ratio 30% higher than the baseline solution for SWOT compression.

~~. Moreover, it provides higher SNR than Sz in most cases.~~

~~A follow-upnExtends to this work could be entailto modifyingto modify the implementation of the HDF5 filter for Sz to allow the data loss to be configuredingconfiguring the data loss on a per- variable basis or to adapt the NetCDF-4 library to allow the activation of other filters, not only Shuffle and deflate.~~ usefuldigits,

8~~7~~ Code and data availability

The Digit Rounding software source code is available from CNES GitHub at https://github.com/CNES/Digit_Rounding. ~~and~~ The ~~the~~ datasets are ~~currently only~~ available upon request to Xavier Delaunay (xavier.delaunay@thalesgroup.com) or to ~~to~~ Flavien Gouillon (~~Flavien.Gouillon@cnes.fr~~Flavien.Gouillon@cnes.fr). The Supplement details the datasets and provides the command lines used for running the compression tools.

Author contributions. Xavier Delaunay designed and implemented the Digit Rounding software and wrote most of the manuscript. Aurélie Courtois performed most of the compression experiments and generated the analytical datasets. Flavien Gouillon provided the scientific datasets used in the experiments, supervised the study, and contributed both to its design and to the writing of the manuscript.

Acknowledg~~e~~ments. This work was funded by CNES under contract 170850/00 and ~~realized~~ carried out at Thales Services. We would like to thank Hélène Vadon, Damien Desroches, ~~and~~ Claire Pottier and Delphine Libby-Claybrough for their contributions to the SWOT section and for their help in~~fulthe helpful~~ proofreading. We also thank Charles S. Zender and the anonymous reviewer their comments, which helped improving the quality of this paper.~~.~~

References

Baker, A. H., Hammerling, D. M., Mickelson, S. A., Xu, H., Stolpe, M. B., Naveau, P., Sanderson, B., Ebert-Uphoff, I., Samarasinghe, S., De Simone, F., Carbone, F., Gencarelli, C. N., Dennis, J. M., Kay, J. E., Lindstrom, P., "Evaluating lossy data compression on climate simulation data within a large ensemble", Geosci. Model Dev., 9, 4381–4403, doi:10.5194/gmd-9-4381-2016, 2016.

Caron, J.: Compression by Scaling and Offset, available at: http://www.unidata.ucar.edu/blogs/developer/en/entry/compression_by_scaling_and_offfset (last access: 27 September 2018), 2014a.

Caron, J.: Compression by bit shaving, available at: http://www.unidata.ucar.edu/blogs/developer/entry/compression_by_bit_shaving (last access: 27 September 2018), 2014b.

Collet, Y.: LZ4 lossless compression algorithm, available at: http://lz4.org (last access: 27 September 2018), 2013.

Collet, Y. and Turner, C.: Smaller and faster data compression with Zstandard, available at: https://code.fb.com/core-data/smaller-and-faster-data-compression-with-zstandard/ (last access: 27 September 2018), 2016

Deutsch, L. P.: DEFLATE compressed data format specification version 1.3, Tech. Rep. IETF RFC1951, Internet Engineering Task Force, Menlo Park, CA, USA, doi:10.17487/RFC1951, 1996.

Duda, J.: Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding, arXiv:1311.2540v2 [cs.IT], 2013.

Huffman, D. A.: A method for the construction of minimum redundancy codes, Proceedings of the IRE, 40(9), 1098-1101, doi:10.1109/JRPROC.1952.273898, 1952.

Lindstrom, P.: Fixed-Rate Compressed Floating-Point Arrays, IEEE Transactions on Visualization and Computer Graphics, 20(12), 2674–2683, doi:10.1109/TVCG.2014.2346458, 2014.

Lindstrom, P., and Isenburg, M.: Fast and Efficient Compression of Floating-Point Data, IEEE Transactions on Visualization and Computer Graphics, 12(5), 1245–1250, doi:10.1109/TVCG.2006.143, 2006.

Masui, K., Amiri, M., Connor, L., Deng, M., Fandino, M., Höfer, C., Halpern, M., Hanna, D., Hincks, A. D., Hinshaw, G., Parra, J. M., Newburgh, L. B., Shaw, J. R., and Vanderlinde, K.: A compression scheme for radio data in high performance computing, Elsevier Astronomy and Computing, 12, 181–190, doi:10.1016/j.ascom.2015.07.002, 2015.

Silver, J. D. and Zender, C. S.: The compression–error trade-off for large gridded data sets, Geosci. Model Dev., 10, 413-423, doi:10.5194/gmd-10-413-2017, 2017.

Tao, D., Di, S., Chen, Z., and Cappello, F.: Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization. IEEE International Parallel and Distributed Processing Symposium (IPDPS), 1129–1139, doi:10.1109/IPDPS.2017.115, 2017a.

Tao, D., Di, S., Guo, H., Chen, Z., and Cappello F.: Z-checker: A Framework for Assessing Lossy Compression of Scientific Data. The International Journal of High Performance Computing Application, doi:10.1177/1094342017737147, 2017b

Zender, C. S.: Bit Grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4.4.8+), Geosci. Model Dev., 9, 3199-3211, doi:10.5194/gmd-9-3199-2016, 2016a.

Zender, C. S.: netCDF Operators (NCO), version 4.6.1, Zenodo, doi:10.5281/zenodo.61341, 2016b.

Ziv, J. and Lempel, A.: A universal algorithm for sequential data compression, IEEE T. Inform. Theory, 23, 337–343, doi: 10.1109/TIT.1977.1055714, 1977.

Table 1: Representation of the value of π in IEEE-754 single-precision binary representation (first row) and results preserving 4 significant digits with the Bit Grooming algorithm (second row) or preserving 12 mantissa bits (third row). This table builds on~~follows with~~ Table 1 in (Zender, 2016a).

Sign	Exponent	Mantissa	Decimal	Notes
0	10000000	10010010000111111011011	3.14159265	Exact value of π
0	10000000	10010010000111100000000	3.14154053	Result of ~~the~~ Bit Grooming with *nsd* = 4, 15 mantissa bits preserved
0	10000000	10010010000100000000000	3.14111328	Result preserving only 12 mantissa bits, allows ~~preserving~~ the 4 significant digits of π to be preserved.

~~**Table 2: The Digit Rounding algorithm.**~~

~~**Input:**~~

~~$\{s_i\}_{i=0}^{n}$ input sequence of samples~~

~~**Output:**~~

~~$\{\tilde{s}_i\}_{i=0}^{n}$ output sequence of quantized samples~~

~~**Parameter:**~~

~~*nsd* number of significant digits preserved in each sample~~

~~**Algorithm:**~~

~~For each input sample s_i in $\{s_i\}_{i=0}^{n}$:~~

1. ~~Compute the number d_i of significant digit number of digits before the decimal separator in the sample value s_i following in Eq. (7)~~
2. ~~Compute the quantization factor power p_i following in Eq. (6)~~
3. ~~Compute the quantization factor q_i as in Eq. (5)~~
4. ~~Compute the quantized value $\tilde{s}_i$ as in Eq. (1)~~

Table

~~Table 2~~3: Representation of the value of π in IEEE-754 single-precision binary representation (first row) and results preserving a varying number of ~~4~~ significant digits (*nsd*) with the Digit Rounding algorithm~~. (second row)~~. This table can be compared ~~with~~ to Table 2 in (Zender, 2016a) providing the Bit Grooming results for ~~of~~ π.

Sign	Exponent	Mantissa	Decimal	Notes
0	10000000	10010010000111111011011	3.14159265	Exact value of π

0	10000000	10010010000111111011011	3.14159265	*nsd* = 8
0	10000000	10010010000111111011010	3.14159250	*nsd* = 7
0	10000000	10010010000111111010000	3. 14159012	*nsd* = 6
0	10000000	10010010000111110000000	3. 14157104	*nsd* = 5
0	10000000	10010010000100000000000	3.14111328	~~Result of the Digit Rounding algorithm with~~ *nsd* = 4
0	10000000	10010010100000000000000	3. 14453125	*nsd* = 3
0	10000000	10010100000000000000000	3. 15625000	*nsd* = 2
0	10000000	11000000000000000000000	3. 50000000	*nsd* = 1
0	10000000	00000000000000000000000	4.00000000	*nsd* = 0

Table 3: Maximum absolute errors and mean absolute errors of the Digit Rounding algorithm preserving a varying number of significant digits (*nsd*) on an artificial dataset composed of 1,~~-~~000,~~-~~000 values evenly spaced over~~n~~ the interval [1.0, 2.0).

nsd	Maximum absolute error	Mean absolute error	Mean error
1	0.4999999999	0.1732423125	-0.0796879687
2	0.0312500000	0.0127722254	-0.0003056211
3	0.0039062500	0.0016125222	-0.0000074545
4	0.0004882812	0.0001983929	-0.0000001013
5	0.0000305176	0.0000125886	-0.0000000017
6	0.0000038147	0.0000015736	-0.0000000002
7	0.0000004768	0.0000001937	0.0000000000

Table 4: Comparison ~~of~~ between the compression ratio obtained with the Digit Rounding algorithm ~~to~~ and the compression ratio obtained with the Bit Grooming algorithm reported in (Zender, 2016a) on a MERRA dataset. Shuffle and Deflate with level 1 lossless compression is applied. The reference for the CR computation is Deflate (level 5) compressed data size of 244.3MB.

	Bit Grooming		Digit Rounding	
NSD	Size (MB)	CR (%)	Size (MB)	CR (%)
~7	223.1	91.3	226.1	92.6
6	225.1	92.1	225.8	92.4
5	221.4	90.6	222.0	90.9
4	201.4	82.4	**191.1**	**78.2**
3	185.3	75.9	**165.1**	**67.6**

2	150.0	61.4	**111.1**	**45.5**
1	100.8	41.3	**64.9**	**26.6**

~~Table 4: Command lines and parameters used for the compression with h5repack~~

~~Compression algorithms~~	~~Command line~~	~~Parameters~~
~~Deflate~~	~~--filter=*var*:GZIP=*dfl_lvl*~~	~~*dfl_lvl* from 1 to 9~~
~~Shuffle + Deflate~~	~~--filter=*var*:SHUF --filter=*var*:GZIP=*dfl_lvl*~~	~~*dfl_lvl* from 1 to 9~~
~~Zstandard~~	~~--filter=var:UD=32015,1,*zstd_lvl*~~	~~*zstd_lvl* from 1 to 22~~
~~Shuffle + Zstandard~~	~~--filter=var:SHUF --filter=var:UD=32015,1,*zstd_lvl*~~	~~*zstd_lvl* from 1 to 22~~
~~Bitshuffle + Zstandard~~	~~--filter=var:UD=32008,1,1048576 --filter=var:UD=32015,1,*zstd_lvl*~~	~~*zstd_lvl* from 1 to 22~~
~~Bitshuffle + LZ4~~	~~--filter=var:UD=32008,2,1048576,2~~	-

Table 5: Compression ~~performance~~ results of the absolute error-bounded compression algorithms Sz and ~~Bit Grooming~~Decimal Rounding ~~in the absolute error br-bounded compression mode~~ on ~~the~~ datasets *s1* and *s3D*.

	~~Dataset *s1*~~		~~Dataset *s3D*~~	
~~Algorithm~~	~~Sz (*absErrBound* = 0.5) + Zstd (*zstd_lvl* = 5)~~	~~Bit Grooming (*dsd* = .0) + Shuffle + Zstd (*zstd_lvl* = 5)~~	~~Sz (*absErrBound* = 0.5) + Zstd (*zstd_lvl* = 5)~~	~~Bit Grooming (*dsd* = .0) + Shuffle + Zstd (*zstd_lvl* = 5)~~
~~Maximum absolute error~~	~~0.5~~	~~0.5~~	~~0.5~~	~~0.5~~
~~SNR (dB)~~	~~30.834~~	~~30.830~~	~~45.9687~~	~~45.9689~~
~~Compression ratio~~	~~5.71~~	~~8.98~~	~~8.69~~	~~7.34~~
~~Compression speed (MB/s)~~	~~50~~	~~51~~	~~25~~	~~16~~

Dataset	Compression method	*CR*	*CS* (MB/s)	e_{abs}^{max}	$\overline{e_{abs}}$	*SNR* (dB)
s1	Sz (*absErrBound* = 0.5, Gzip_BEST_SPEED)	5.39	133	0.5	0.2499	30.84

s1	Decimal Rounding (*dsd* = .0, *dflt_lvl*=1)	7.50	100	0.5	0.2501	30.83
s3D	Sz (*absErrBound* = 0.5, Gzip_BEST_SPEED)	12.97	29	0.5	0.2500	45.97
s3D	Decimal Rounding (*dsd* = .0, *dflt_lvl*=1)	5.56	80	0.5	0.2500	45.97

Table 6: Compression ~~performance~~ results of the relative error-bounded compression algorithms Sz, Bit Grooming, and Digit Rounding ~~in the relative error- bounded compression mode~~ on ~~the~~ dataset *s1*.

~~Algorithm~~	~~Sz~~	~~Bit Grooming~~	~~Digit Rounding~~
~~Parameter~~	~~*pw_relBoundRatio* = 0.00424~~	~~*nsd* = 3~~	~~*nsd* = 3~~
~~Maximum absolute error~~	~~0.5~~	~~0.0312~~	~~0.0325~~
~~SNR (dB)~~	~~30.83~~	~~54.93~~	~~54.92~~
~~Compression ratio~~	~~5.68~~	~~3.18~~	~~3.80~~
~~Compression speed (MB/s)~~	~~32~~	~~37~~	~~40~~

Compression method	*CR*	*CS* (MB/s)	e_{abs}^{max}	$\overline{e_{abs}}$	*SNR* (dB)
Sz (*pw_relBoundRatio* = 0.00424, Gzip_BEST_SPEED)	5.08	100	0.484	0.199	32.78
Bit Grooming (*nsd* = 3, *dflt_lvl*=1)	3.09	57	0.0312	0.0156	54.93
Bit Grooming (*nsd* = 2, *dflt_lvl*=1)	4.38	57	0.250	0.125	36.54
Digit Rounding (*nsd* = 3, *dflt_lvl*=1)	4.02	40	0.5	0.195	34.51

Table 7: Compression ~~performance~~ results of Sz, Bit Grooming, and Digit Rounding in ~~the~~ relative error-bounded compression mode on ~~the~~ dataset *s3D*.

Compression method	*CR*	*CS* (MB/s)	e_{abs}^{max}	$\overline{e_{abs}}$	*SNR* (dB)
Sz (*pw_relBoundRatio* = 0.00345, Gzip_BEST_SPEED)	4.32	26	0.487	0.0737	54.56
Bit Grooming (*nsd* = 3, *dflt_lvl*=1)	2.35	46	0.0625	0.0079	73.96

Bit Grooming (*nsd* = 2, *dflt_lvl*=1)	3.04	51	0.5	0.0629	55.89
Digit Rounding (*nsd* = 3, *dflt_lvl*=1)	2.60	18	0.5	0.0239	58.87

~~Algorithm~~	~~Sz~~	~~Bit Grooming~~	~~Digit Rounding~~	
~~Parameter~~	~~*pw_relBoundRatio* = 0.00345~~	~~*nsd* = 3~~	~~*nsd* = 2~~	~~*nsd* = 3~~
~~Maximum absolute error~~	~~0.256~~	~~0.0625~~	~~0.5~~	~~0.5~~
~~*SNR* (dB)~~	~~68.06~~	~~73.96~~	~~55.89~~	~~63.94~~
~~*Compression ratio*~~	~~2.05~~	~~2.45~~	~~3.30~~	~~2.67~~
~~*Compression speed* (MB/s)~~	~~18~~	~~14~~	~~15~~	~~19~~

Table 8: ~~Performance~~ Compression ~~performance~~results for the ~~*compression of the*~~ *ground_range_5* variable in the CFOSAT L1A product.

Compression method	*CR*	*CS* (MB/s)	*DS* (MB/s)	e_{abs}^{max}	$\overline{e_{abs}}$
CFOSAT "Clipping" + Shuffle + Deflate (3)	2.34	38 (*)	123	1.00e-3	5.00e-4
CFOSAT "Clipping" + Zstd (2)	2.20	108 (*)	84	1.00e-3	5.00e-4
Sz (*absErrBound* = 1e-3, Gzip_BEST_SPEED)	26.53	60	42	1.00e-3	4.99e-4
Decimal Rounding (*dsd* = .3) + Shuffle + Deflate (1)	5.85	74	187	4.88e-4	2.36e-4
Bit Grooming (*nsd* = 8) + Shuffle + Deflate (1)	4.78	67	190	2.44e-4	1.22e-4
Digit R~~r~~ounding (*nsd* = 8) + Shuffle + Deflate (1)	5.83	37	38	4.88e-4	2.44e-4

(*) The time taken for the CFOSAT "Clipping" method is not taken into account int~~o~~ the compression speed computation.

Table ~~8~~9: Compression ~~pPerformance~~results for ~~the compression of~~ the CFOSAT L1A product~~products~~.

Compression method	*CR*~~Compression ratio~~	*CS* (MB/s)~~Compression speed (MB/s)~~	*DS* (MB/s)~~Decompression speed (MB/s)~~

~~Baseline CFOSAT compression method:~~ CFOSAT “Clipping” + Shuffle + Deflate (3)	5.21	51 (*)	68
~~CFOSAT “Clipping” + Shuffle + Zstandard (1)~~	~~67~~		
CFOSAT “Clipping” + Shuffle + Zstd~~Zstandard~~ (2)	5.38	72 (*)	78
Sz (*absErrBound*, Gzip_BEST_SPEED)	15.45	88	89
~~Bit Grooming (abs) + Shuffle + Deflate (3)~~	~~74~~		
~~Bit Grooming (abs) + Shuffle +~~ Zstd~~Zstandard (2)~~	~~12.68~~	~~35~~	~~81~~
Decimal Rounding + Shuffle + Deflate (1)	9.53	101	268
Bit Grooming (*nsd* = 8) + Shuffle + Deflate (1)	4.16	75	262
Digit Rounding (*nsd* = 8) + Shuffle + Deflate (1)~~R~~	4.32	37	85

(*) The time taken for the CFOSAT “Clipping” method is not taken into account in~~to~~ the compression speed computation.

Table 10: Compression results for the *height* variable in the simplified simulated SWOT L2_HR_PIXC pixel cloud product.

Compression method	*CR*	*CS* (MB/s)	*DS* (MB/s)	e_{abs}^{max}	$\overline{e_{abs}}$
Shuffle + Deflate (4)	1.12	24	212	0	0
Shuffle + Zstd (2)	1.12	271	181	0	0
Sz (*pw_relBoundRatio* = 5e-6, Gzip_BEST_SPEED)	2.06	35	155	3.16e-5	1.19e-7
Bit Grooming (*nsd* = 6) + Shuffle + Deflate (1)	2.34	33	217	7.58e-6	2.53e-7
Digit Rounding (*nsd* = 6) + Shuffle + Deflate (1)	2.38	35	217	3.05e-5	7.95e-7

Table 11: Compression results for the *pixel_area* variable in the representative SWOT L2 pixel cloud product.

Compression method	*CR*	*CS* (MB/s)	*DS* (MB/s)	e_{abs}^{max}	$\overline{e_{abs}}$
Shuffle + Deflate (4)	1.50	32	248	0	0
Shuffle + Zstd (2)	1.50	237	165	0	0
Sz (*pw_relBoundRatio* = 5e-9, Gzip_BEST_SPEED)	3.24	0.3	165	2.51e-6	4.56e-7
Bit Grooming (*nsd* = 11) + Shuffle + Deflate (1)	2.11	43	245	1.86e-9	3.16e-10

Digit Rounding (*nsd* = 11) + Shuffle + Deflate (1)	2.40	40	240	3.73e-9	1.86e-9

Table 12~~9~~: Compression ~~p~~~~Performance~~results for the ~~compression of the~~ simplified simulated SWOT L2_HR_PIXC pixel cloud product.

Compression method	*CR*~~Compression ratio~~	*CS* (MB/s)~~Compression speed (MB/s)~~	*DS* (MB/s)~~Decompression speed (MB/s)~~
~~Baseline SWOT compression method:~~ Shuffle + Deflate (4)	14.37	107	92
~~Shuffle + Zstandard (1)~~			
Shuffle + ~~Zstandard~~Zstd (2)	14.36	589	97
Bit Grooming + Shuffle + Deflate (1)	17.44	141	336
Digit Rounding + Shuffle + Deflate (1)	18.92	100	393
~~Bit Grooming (abs) + Shuffle + Deflate (4)~~			
~~Bit Grooming (abs) + Shuffle +~~ Zstd~~Zstandard (2)~~	~~20.66~~	~~79~~	~~108~~
~~Bit Grooming (rel) + Shuffle +~~ Zstd~~Zstandard (2)~~	~~18.87~~	~~50~~	~~101~~
R~~Digit Rounding + Shuffle + Zstandard (2)~~	~~21.04~~	~~N/A~~	~~N/A~~

Table 13~~0~~: Compression ~~p~~~~Performance~~results for the ~~compression of the~~ representative SWOT L2 pixel cloud product.

Compression method	*CR*~~*Compression ratio*~~	*CS* (MB/s)~~*Compression speed* (MB/s)~~	*DS* (MB/s)~~*Decompression speed* (MB/s)~~
Shuffle + Deflate (4~~2~~)	~~1.9~~1.99~~8~~	35~~52~~	258~~83~~

~~Shuffle + Zstandard (1)~~	~~91~~		
Shuffle + Zstd~~Zstandard~~ (2)	1.99	139	90
Bit Grooming + Shuffle + Deflate (1)	2.55	52	276
Digit Rounding + Shuffle + Deflate (1)	2.65	42	228
~~Bit Grooming (abs) + Shuffle + Deflate (4)~~	~~98~~		
~~Bit Grooming (abs) + Shuffle + ZstdZstandard (2)~~	~~4.4~~	~~65~~	~~104~~
~~Bit Grooming (rel) + Shuffle + Zstandard (2)~~	~~2.56~~	~~43~~	~~93~~
~~RDigit Rounding + Shuffle + Zstandard (2)~~	~~2.85 (*)~~	~~N/A (*)~~	~~N/A (*)~~

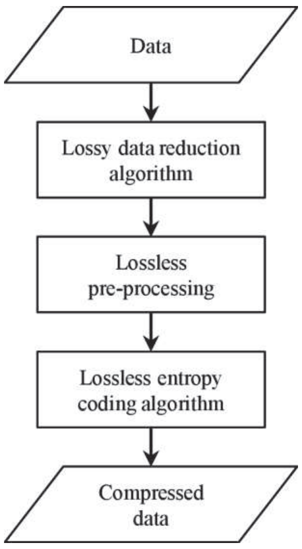

Data
Lossy data reduction algorithm
Lossless pre-processing
Lossless entropy coding algorithm
Compressed data

Figure 1: Compression chain ~~in which appears~~showing the data reduction, pre-processing and lossless coding steps.

Input:

$\{s_i\}_{i=0}^{n}$ input sequence of samples

Output:

$\{\tilde{s}_i\}_{i=0}^{n}$ output sequence of quantized samples

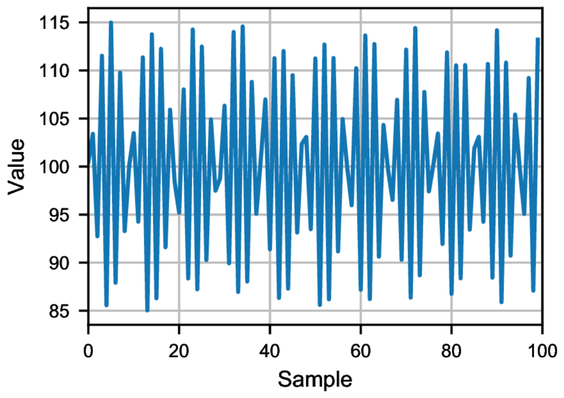
Parameter:

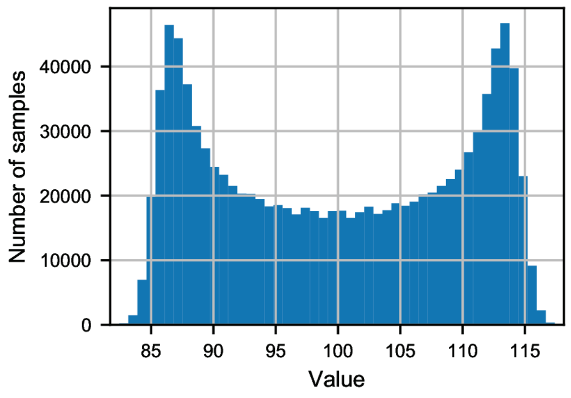
nsd number of significant digits preserved in each sample

Algorithm:

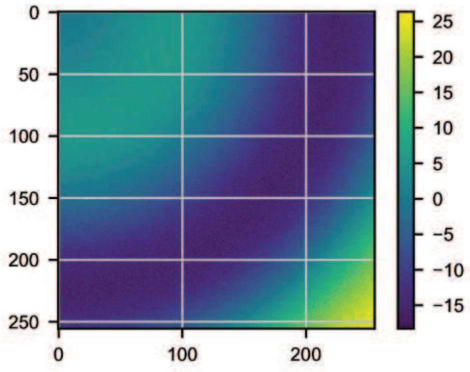
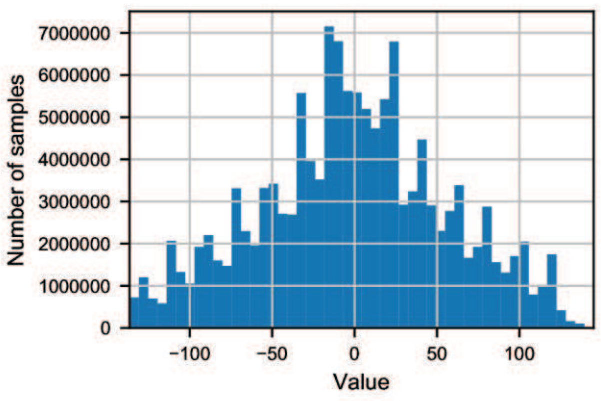
```
For each input sample $s_i$ in $\{s_i\}_{i=0}^{n}$:
    1. Get ~~the~~ binary exponent $e_i$ and mantissa $m_i$ of value $s_i$ according to Eq. (7)
    2. Tabulate ~~the~~ value v for $\log_{10}(m_i)$
    3. Compute the approximated number of digits before the decimal separator in ~~the~~ sample value $s_i$
       following Eq. (8)
    4. Compute ~~the~~ quantization factor power $p_i$ following Eq. (6)
    5. Compute ~~the~~ quantization factor $q_i$ as in Eq. (5)
    6. Compute ~~the~~ quantized value $\tilde{s}_i$ as in Eq. (1)
```

Figure 2: The Digit Rounding algorithm.

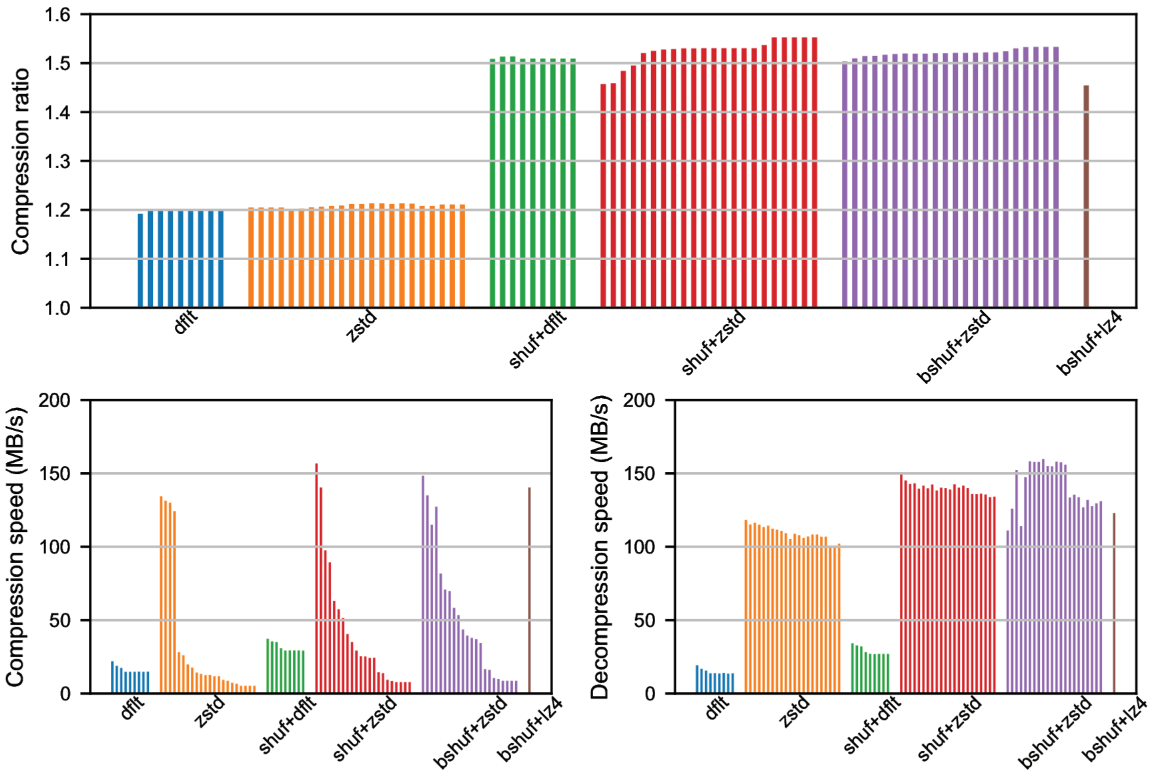




~~Figure 2: First 100 samples of the dataset s_1 (left) and histogram of the sample values (right).~~

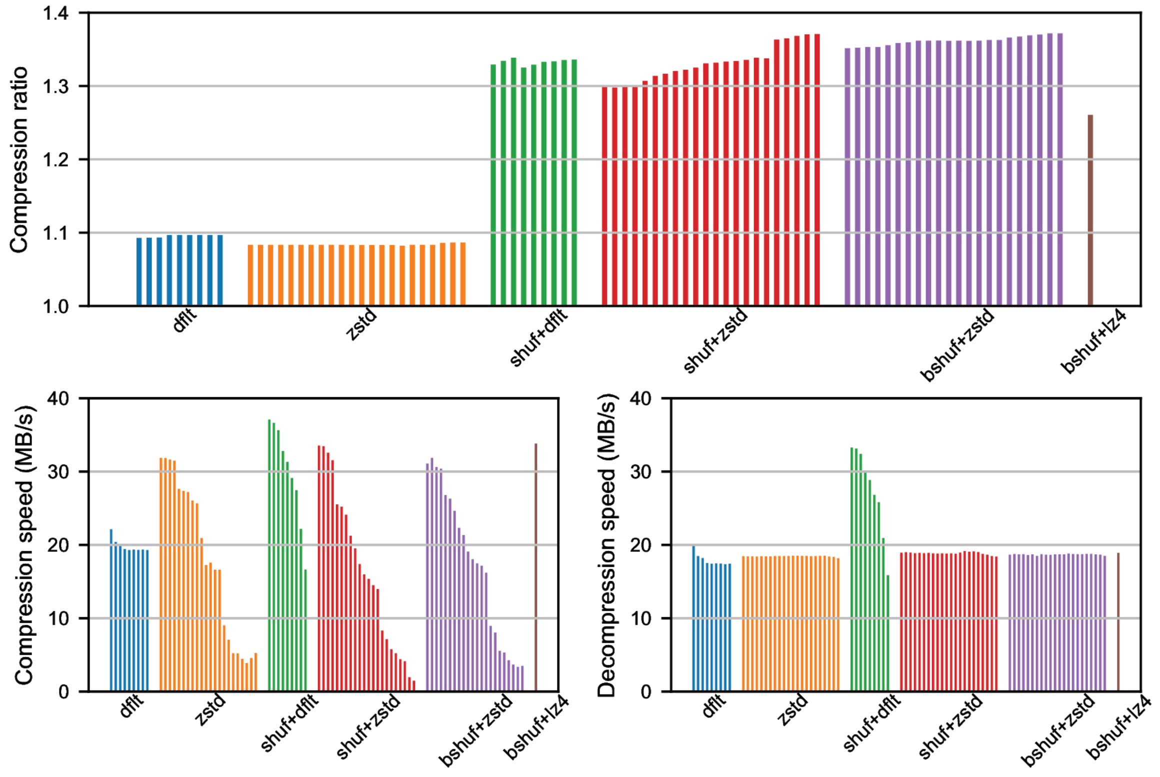



~~Figure 3: Representation of the first slices $s3D(i_1, i_2, 0)$ (left), and histogram of the sample values (right).~~

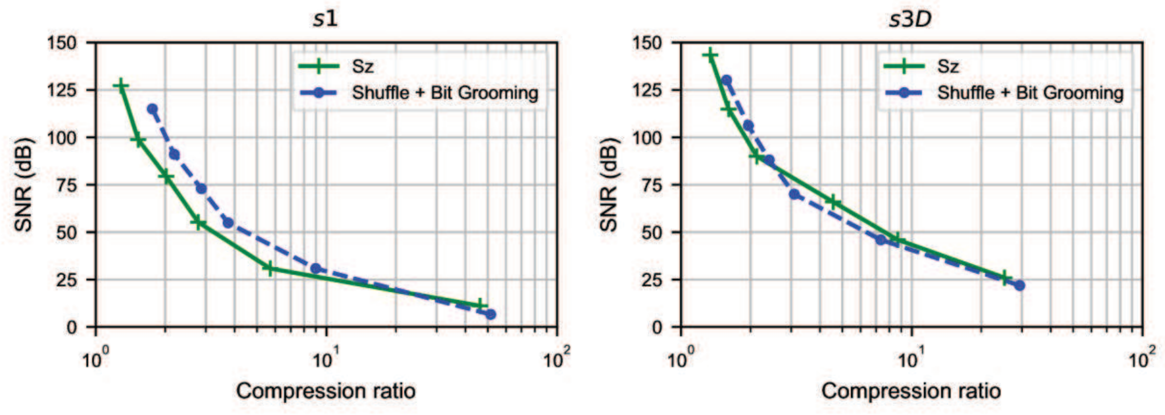


~~Figure 4~~Figure 3: ~~Performance~~ Results obtained for the lossless compression of the *s1* dataset with Deflate (dflt), Zstandard (zstd), Shuffle and Deflate (shuf+dflt), Shuffle and Zstandard (shuf+zstd), Bitshuffle and Zstandard (bshuf+zstd), Bitshuffle and LZ4 (bshuf+lz4). Compression ratios (top), c~~C~~ompression speeds (bottom-left), and decompression speeds (bottom-right). Vertical bars represent the results for different compression levels: from 1 to 9 for Deflate, from 1 to 22 for Zstandard, only one level for LZ4

.



~~Figure 5~~Figure 4: ~~Performance~~ Results obtained for the lossless compression of the *s3D* dataset with Deflate (dflt), Zstandard (zstd), Shuffle and Deflate (shuf+dflt), Shuffle and Zstandard (shuf+zstd), Bitshuffle and Zstandard (bshuf+zstd), Bitshuffle and LZ4 (bshuf+lz4). Compression ratios (top), c~~C~~ompression speeds (bottom- left), and decompression speeds (bottom -right). 5



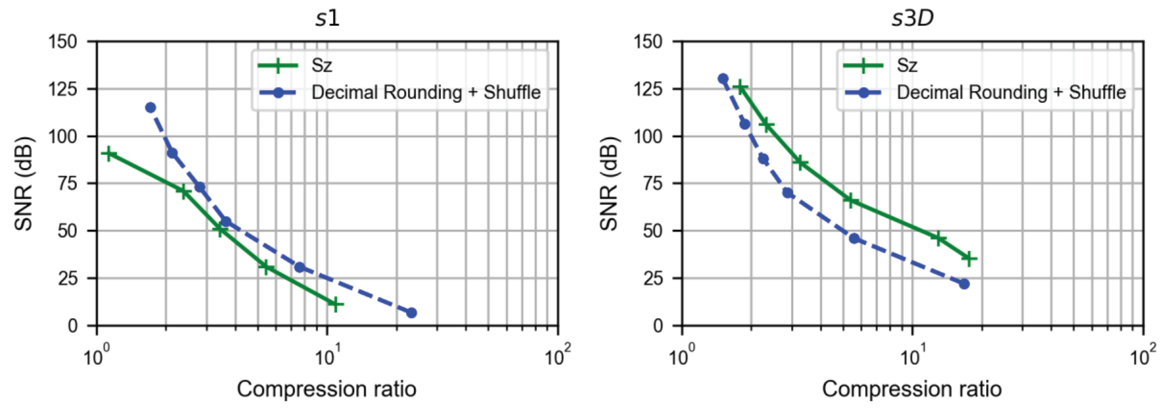
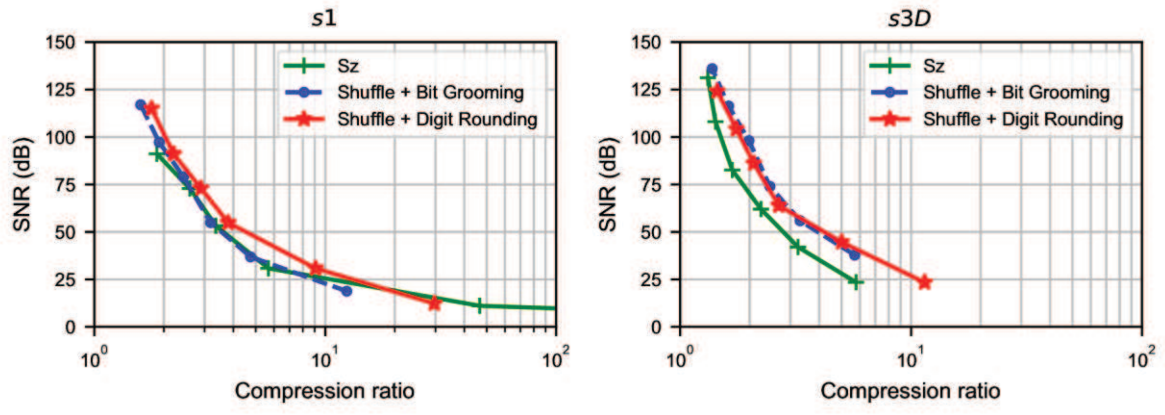


Figure ~~5~~6: Comparison of the compression ~~performance~~ results (SNR vs. compression ratio) of the Sz and ~~Bit Grooming~~Decimal Rounding algorithms in ~~the~~ absolute error-bounded compression mode, on~~.~~ ~~Compression performance obtained on~~ the *s1* dataset (left) and *s3D* dataset (right).



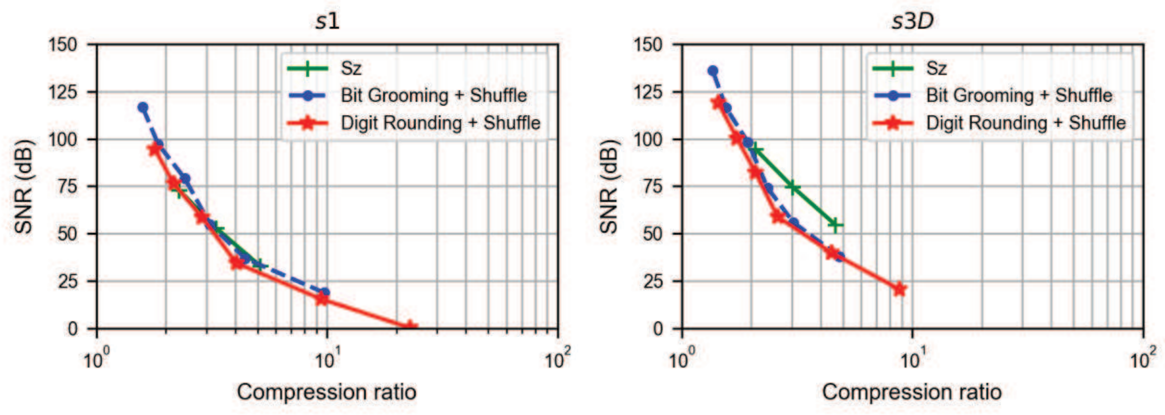


Figure 6~~7~~: Comparison of the compression ~~performance~~ results (SNR vs. compression ratio) of the Sz, Bit Grooming and Digit Rounding algorithms in ~~the~~ relative error-bounded compression mode~~.~~ , on the ~~Compression performance obtained on the~~ *s1* dataset (left) and *s3D* dataset (right).

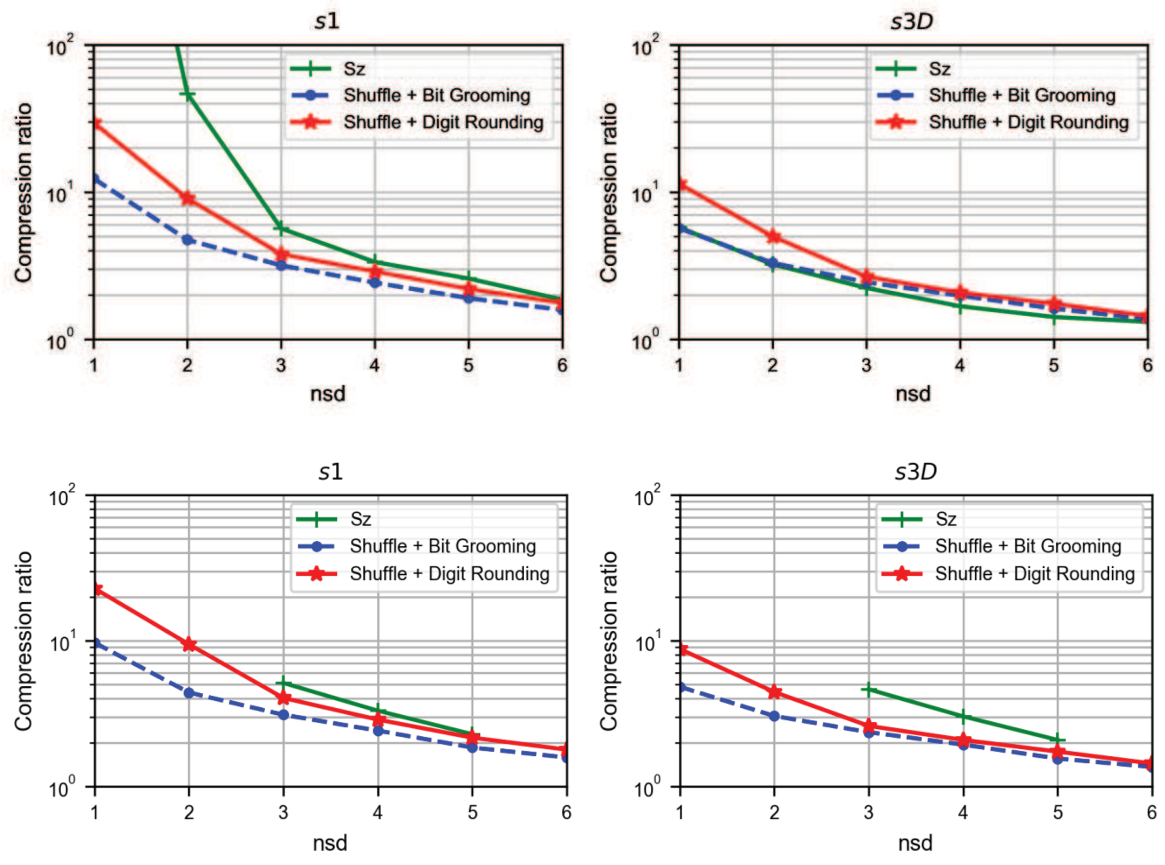


Figure 7~~8~~: Compression ratio as a function of the user-~~-~~specified number of significant digits~~digit~~ (*nsd*) for the Sz, Bit Grooming and Digit Rounding algorithms, on ~~algorithm. Compression performance~~ the ~~obtained on the~~ *s1* dataset (left) and *s3D* dataset (right).