Supplement of Geosci. Model Dev., 12, 4099–4113, 2019
https://doi.org/10.5194/gmd-12-4099-2019-supplement

Geoscientific Model Development
Open Access
EGU

Supplement of

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

Xavier Delaunay et al.

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

Introduction

This supplement details the commands and datasets necessary to reproduce the results tabulated in the paper.

The Digit Rounding algorithm

The table below provides tabulated values for the approximation of $\log_{10}(m_i)$ in the implementation of the Digit Rounding algorithm. 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 $v \leq \log_{10}(m_i)$.

Table 1: Tabulated values for the approximation of $\log_{10}(m_i)$ in the implementation of the Digit Rounding algorithm

m_i	Approximated value v for $\log_{10}(m_i)$
$0.5 \leq m_i < 0.6$	-0.301029996
$0.6 \leq m_i < 0.7$	-0.221848749
$0.7 \leq m_i < 0.8$	-0.154901959
$0.8 \leq m_i < 0.9$	-0.096910013
$0.9 \leq m_i < 1.0$	-0.045757490

The results reported in Table 2 and Table 3 of the paper can be reproduced compiling the Digit Rounding software in test and debug mode with the following command lines:

```
cd digiround/
make clean
make test DEBUG=1
```

The results reported in Table 4 can be reproduced using the following script:

```
for nsd in $(seq 7); do
```

```
    # Compress applying Digit Rounding, Shuffle and Deflate with dflt_lvl=1
    h5repack -i MERRA300.prod.assim.inst3_3d_asm_Cp.20130601.nc -o foo.h5 \
       --filter=UD=47987,0,1,$nsd --filter=SHUF --filter=GZIP=1
done
```

Synthetic datasets

Synthetic datasets with known statistics have been generated in order to test the compression algorithms under variable conditions. The following datasets have been generated:

- $s1$ a noisy sinusoid of 1 dimension,
- s3D a noisy sinusoid pulse of 3 dimensions.

The 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. 1.

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. 2.

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

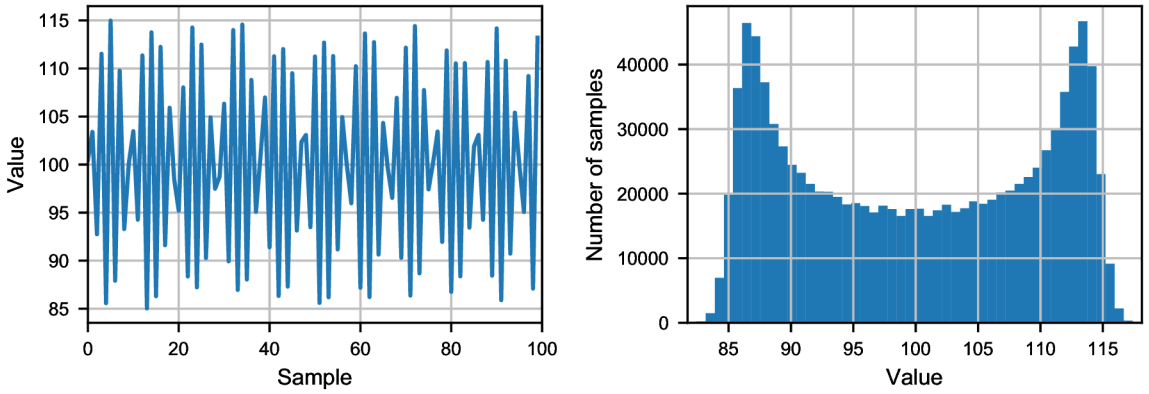

Figure 1: First 100 samples of the dataset *s1* (left) and histogram of the sample values (right).

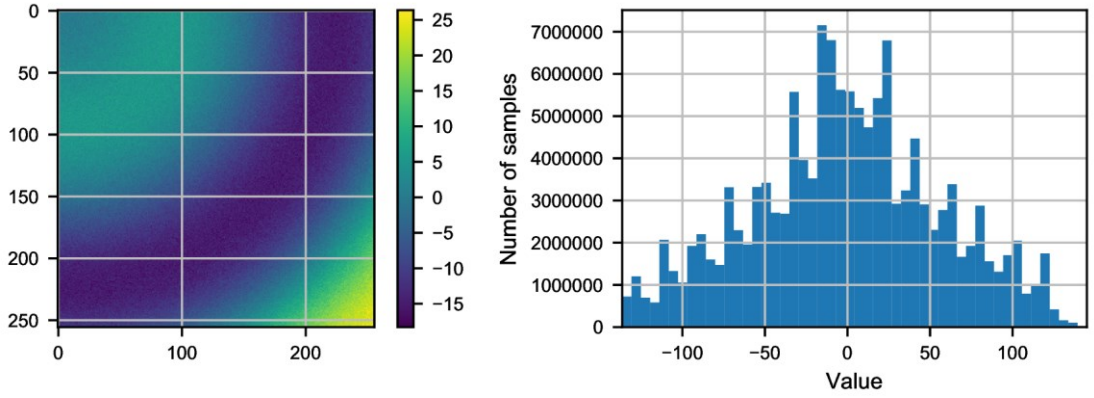

Figure 2: Representation of the first slices $s3D(i_1, i_2, 0)$ (left), and histogram of the sample values (right).

5 Performance assessment of lossless compression methods

We ran lossless compression algorithm using 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 performed calling h5repack tool with a command line formatted as follows:

10 `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. Each HDF5 filter is identified by a unique ID: 32015 for Zstandard and 32008 for Bitshuffle. The following table provides the list of filter options used. They shall replace the filter option between brackets on previous command line.

Table 2: 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	-

The decompression has been 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 were run on a single core.

Performance assessment of lossy compression methods

Performance comparison in the absolute error bounded compression mode

The command lines provided in this section allow reproducing the results provided in Table 5 of the paper.

Sz compression is performed calling h5repack tool with the following command lines:

```
h5repack -i s1.nc -o s1_sz.h5 --filter=signal:UD=32017,0
h5repack -i s3D.nc -o s3D_sz.h5 --filter=signal:UD=32017,0
```

Sz compression filter is identified by its ID (32017) provided on the command line. 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 is configured via the *sz.config* file located in the directory from where h5repack is called. *sz.config* file content is reproduced below:

```
[ENV]
dataEndianType = LITTLE_ENDIAN_DATA
sol_name = SZ
[PARAMETER]
quantization_intervals = 256
szMode = SZ_BEST_SPEED
gzipMode = Gzip_BEST_SPEED
errorBoundMode = ABS
absErrBound = 0.5
```

Decimal Rounding is performed calling the *ncks* tool from NCO toolkit. The algorithm is run with the following command lines (note the period before the *dsd* parameter):

```
ncks -O -4 -L 1 --ppc signal=.0 s1.nc s1_bg.nc
ncks -O -4 -L 1 --ppc signal=.0 s3D.nc s3D_bg.nc
```

Performance comparison in the relative error bounded compression mode

Dataset s1

The command lines provided in this section allow reproducing the results provided in Table 6 of the paper.

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

```
h5repack -i s1.nc -o s1_sz.h5 --filter=signal:UD=32017,0
```

Sz compression is configured via the *sz.config* file located in the directory from where h5repack is called. *sz.config* file content is reproduced below:

```
[ENV]
dataEndianType = LITTLE_ENDIAN_DATA
sol_name = SZ
[PARAMETER]
quantization_intervals = 256
szMode = SZ_DEFAULT_COMPRESSION
gzipMode = Gzip_BEST_SPEED
errorBoundMode = PW_REL
pw_relBoundRatio = 0.00424
pwr_type = MAX
```

Bit Grooming is performed calling the *ncks* with the following command lines:

```
ncks -O -4 -L 1 --ppc signal=3 s1.nc s1_bg_3.nc
ncks -O -4 -L 1 --ppc signal=2 s1.nc s1_bg_2.nc
```

Digit Rounding is performed calling h5repack tool with the following command line:

```
h5repack -i s1.nc -o s1_dr_3.h5 --filter=signal:UD=47987,1,1,3
       --filter=signal:SHUF --filter=signal:GZIP=1
```

Dataset s3D

The command lines provided in this section allow reproducing the results provided in Table 7 of the paper.

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

```
h5repack -i s3D.nc -o s3D_sz.h5 --filter=signal:UD=32017,0
```

Sz compression is configured via the *sz.config* file located in the directory from where h5repack is called. *sz.config* file content is reproduced below:

```
[ENV]
dataEndianType = LITTLE_ENDIAN_DATA
sol_name = SZ
[PARAMETER]
quantization_intervals = 256
szMode = SZ_DEFAULT_COMPRESSION
gzipMode = Gzip_BEST_SPEED
```

```
errorBoundMode = PW_REL
pw_relBoundRatio = 0.00345
pwr_type = MAX
```

Bit Grooming is performed calling the *ncks* with the following command lines:

```
ncks -O -4 -L 1 --ppc signal=3 s3D.nc s3D_bg_3.nc
ncks -O -4 -L 1 --ppc signal=2 s3D.nc s3D_bg_2.nc
```

Digit Rounding is performed calling h5repack tool with the following command line:

```
h5repack -i s3D.nc -o s3D_dr_3.h5 --filter=signal:UD=47987,1,1,3 \
       --filter=signal:SHUF --filter=signal:GZIP=1
```

CFOSAT dataset

We used two version of the same dataset:

- TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc is the 'raw' dataset.
- CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc is the same dataset after clipping and Shuffle + Deflate (3) compression.

Table 3 provides the least significant digit (*lsd*) of each variable of this dataset.

Table 3: Variables and least significant digit (*lsd*) in CFOSAT dataset.

Variable	*lsd*	Variable	*lsd*	Variable	*lsd*	Variable	*lsd*
altitude	3	elevation_0	5	lat_l1a_0	3	phi_geo	3
cal_ratio_0	3	elevation_1	5	lat_l1a_1	3	pri	1
cal_ratio_1	3	elevation_2	5	lat_l1a_2	3	projected_velocity	3
cal_ratio_2	3	elevation_3	5	lat_l1a_3	3	pseudo_misp	14
cal_ratio_3	3	elevation_4	5	lat_l1a_4	3	radar_range_0	3
cal_ratio_4	3	elevation_5	5	lat_l1a_5	3	radar_range_1	3
cal_ratio_5	3	ground_range_0	3	lon_l1a_0	3	radar_range_2	3
cycle_duration	7	ground_range_1	3	lon_l1a_1	3	radar_range_3	3
earth_radius	3	ground_range_2	3	lon_l1a_2	3	radar_range_4	3
echo_l1_0	12	ground_range_3	3	lon_l1a_3	3	radar_range_5	3
echo_l1_0_nt	12	ground_range_4	3	lon_l1a_4	3		
echo_l1_0_std_nt	12	ground_range_5	3	lon_l1a_5	3		
echo_l1a_0	3	incidence_0	5	ly	2		
echo_l1a_1	3	incidence_1	5	mispointing	14		
echo_l1a_2	3	incidence_2	5	nesig0	3		
echo_l1a_3	3	incidence_3	5	orbital_velocity	3		
echo_l1a_4	3	incidence_4	5	phi	3		
echo_l1a_5	3	incidence_5	5	phi_azimuth	3		

The following command lines allow reproducing the results provided in Table 8 of the paper.

We first extract the *ground_range_5* variable from the 'clipped' dataset and decompress it:

```
ncks -O -4 -C -v ground_range_5 \
      CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc \
      ground_range_5_clipped_tmp.nc
nccopy -d 0 round_range_5_clipped_tmp.nc ground_range_5_clipped.nc
```

Then, CFOSAT Clipping + Shuffle + Deflate (3) compression is performed using the following command line:

```
nccopy -s -d 3 ground_range_5_clipped.nc ground_range_5_clipped_dfl3.nc
```

CFOSAT Clipping + Zstd (2) compression is performed using the following command line:

```
nccopy -F "ground_range_5,32015,2" ground_range_5_clipped.nc ground_range_5_clipped_zstd2.nc
```

For the other compression methods, we first extract the ground_range_5 variable from the 'raw' dataset:

```
ncks -O -4 -C -v ground_range_5 \
      TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc ground_range_5.nc
```

Then, Sz (*absErrBound* = 1e-3, Gzip_BEST_SPEED) compression is performed using the following command line after having correctly configured the sz.config file:

```
h5repack --filter=ground_range_5:UD=32017,0 -i ground_range_5.nc -o ground_range_5_sz.h5
```

Decimal Rounding (*dsd* = .3) + Shuffle + Deflate (1) compression is performed using the following command line:

```
ncks -O -4 -L 1 --ppc ground_range_5=.3 ground_range_5.nc ground_range_5_dr.nc
```

Bit Grooming (*nsd* = 8) + Shuffle + Deflate (1) compression is performed using the following command line:

```
ncks -O -4 -L 1 --ppc ground_range_5=8 ground_range_5.nc ground_range_5_bg.nc
```

Digit Rounding (*nsd* = 8) + Shuffle + Deflate (1) compression is performed using the following command line:

```
h5repack h5 --filter=ground_range_5:UD=47987,1,1,8 --filter=ground_range_5:SHUF \
      --filter=ground_range_5:GZIP=1 -i ground_range_5.nc -o ground_range_5_dr.h5
```

The following command lines allow reproducing the results provided in Table 9 of the paper.

We first decompress the 'clipped' dataset:

```
nccopy -d 0 CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc \
      CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500_decompressed.nc
```

Then, CFOSAT Clipping + Shuffle + Deflate (3) compression is performed using the following command line:

```
nccopy -s -d 3 CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500_decompressed.nc \
      CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500_dfl3.nc
```

CFOSAT Clipping + Shuffle + Zstd (2) compression is performed using the following command line:

```
h5repack --filter=SHUF --filter=UD=32015,0,1,2 \
       -i CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500_decompressed.nc \
       -o CFO_TEST_SWI_L1A____F_20160830T150000_20160830T164500_zstd2.h5
```

For the other compression methods, we used the 'raw' dataset.

Decimal Rounding + Shuffle + Deflate (1) compression is performed using the following command line, setting the *dsd* parameter on a per variable basis and corresponding to the *lsd* parameters of CFOSAT clipping (see Table 3):

```
ncks -O -4 -L 1 --ppc default=.3 --ppc cycle_duration=.7 --ppc echo_l1_0.?=.12 \
       --ppc elevation_.?=.5 --ppc incidence_.?=.5  --ppc ly=.2 --ppc mispointing=.14 \
       --ppc pri=.1 --ppc pseudo_misp=.14 \
       TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc \
       TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500_dr.nc)
```

Bit Grooming (nsd = 8) + Shuffle + Deflate (1) compression is performed using the following command line:

```
ncks -O -4 -L 1 --ppc 8 TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc \
       TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500_bg.nc)
```

Digit Rounding (*nsd* = 8) + Shuffle + Deflate (1) compression is performed using the following command line:

```
h5repack h5 --filter=UD=47987,1,1,8 --filter=SHUF --filter=GZIP=1 \
       -i TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc \
       -o TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500_dr.h5
```

Sz compression has been applied variable per variable using the following command lines:

```
# Get the list of variables in the dataset
var_list=$(h5ls TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc | awk '{print $1}')
# Loop over the variables
for var in $var_list;
do
    # Extract the variable from the dataset
    ncks -O -4 -C -v $var \
       TMP_TEST_SWI_L1A____F_20160830T150000_20160830T164500.nc tmp.nc
    # Read the lsd attribute from the input file
    lsd=$(h5ls -v tmp.nc/$var | sed -n '/least_significant_digit.*/,/Data.*/p' \
       | tail -1 | awk '{print $2}')
    # modify sz.config according to the lsd

    sed -i "s/^absErrBound.*=.*/absErrBound = 1e-${lsd}/g" sz.config
    # Compress
    h5repack -i tmp.nc -o ${var}.h5 --filter=${var}:UD=32017,0
done
```

SWOT datasets

We used two different SWOT datasets:

- SWOT_L2_HR_PIXC_000_210_46N-R_main.nc is a simplified simulated SWOT L2_HR_PIXC pixel cloud product;
- pixel_cloud.nc is a realistic and representative SWOT L2 pixel cloud product

Table 4 provides the precision in number of significant digits (*nsd*) required for each variable of the SWOT_L2_HR_PIXC_000_210_46N-R_main.nc dataset.

Table 4: Precision in number of significant digits (*nsd*) required for each variable of the SWOT_L2_HR_PIXC_000_210_46N-R_main.nc dataset.

Variable	*nsd*	Variable	*nsd*	Variable	*nsd*
azimuth_index	4	dlook_dphase_z	6	look_unit_x	8
classification	3	dphase	6	look_unit_y	8
coherent_power	8	dry_tropo_range_correction	4	look_unit_z	8
continuous_classification	8	height	6	num_rare_looks	2
cross_track	15	ice_flag	8	phase_screen	6
dark_water_flag	8	ifgram_imag	15	pixel_area	11
delta_x	8	ifgram_real	15	power_left	8
delta_y	8	illumination_time	15	power_right	8
delta_z	8	instrument_attitude_correction	6	range_index	4
dheight_dbaseline	6	instrument_baseline_correction	6	reference_layover_flag	8
dheight_dphase	8	instrument_phase_correction	6	reference_layover_height_error	8
dheight_drange	6	instrument_range_correction	6	sensor_s	11
dheight_droll	8	ionosphere_range_correction	6	solid_earth_tide_height_correction	4
dlook_dphase_x	6	latitude	15	wet_tropo_range_correction	4
dlook_dphase_y	6	longitude	15	xover_roll_correction	4

Table 5 provides the precision in number of significant digits (*nsd*) required for each variable of the pixel_cloud.nc dataset.

Table 5: Precision in number of significant digits (*nsd*) required for each variable of the pixel_cloud.nc dataset.

Variable	*nsd*	Variable	*nsd*	Variable	*nsd*
azimuth_index	4	ifgram	7	power_left	8
classification	3	illumination_time	15	power_right	8
coherent_power	8	incidence_angle	8	range_index	4
continuous_classification	8	latitude	15	regions	7
cross_track	15	longitude	15	sigma0	7
dheight_dphase	8	num_med_looks	3	x_factor_left	7
dlatitude_dphase	5	num_rare_looks	2	x_factor_right	7
dlongitude_dphase	5	phase_noise_std	7		
height	6	pixel_area	11		

The following command lines allow reproducing the results provided in Table 10 of the paper.

We first extract the *height* variable from SWOT dataset and decompress it:

```
ncks -O -v height SWOT_L2_HR_PIXC_000_210_46N-R_main.nc height_tmp.nc
nccopy -d 0 height_tmp.nc height.nc
```

Then, Shuffle + Deflate (4) compression is performed using the following command line:

```
nccopy -s -d 4 height.nc height_dfl4.nc
```

Shuffle + Zstd (2) compression is performed using the following command line:

```
h5repack --filter=height:SHUF --filter=height:UD=32015,1,1,2 \
       -i height.nc -o height_zstd2.h5
```

Sz (*pw_relBoundRatio* = *5e-6*, Gzip_BEST_SPEED) compression is performed using the following command line after having correctly configured the sz.config file:

```
h5repack --filter=ground_range_5:UD=32017,0 -i height.nc -o height_sz.h5
```

Bit Grooming (*nsd* = 6) + Shuffle + Deflate (1) compression is performed using the following command line:

```
ncks -O -4 -L 1 --ppc height=6 height.nc height_bg.nc
```

Digit Rounding (*nsd* = 6) + Shuffle + Deflate (1) compression is performed using the following command line:

```
h5repack h5 --filter=height:UD=47987,1,1,6 --filter=height:SHUF --filter=height:GZIP=1 \
       -i height.nc -o height_dr.h5
```

The following command lines allow reproducing the results provided in Table 11 of the paper.

We first extract the *pixel_area* variable from SWOT dataset and decompress it:

```
ncks -O -v pixel_area pixel_cloud.nc pixel_area_tmp.nc
nccopy -d 0 pixel_area_tmp.nc pixel_area.nc
```

Then, Shuffle + Deflate (4) compression is performed using the following command line:

```
nccopy -s -d 4 pixel_area.nc pixel_area_dfl4.nc
```

Shuffle + Zstd (2) compression is performed using the following command line:

```
h5repack --filter=pixel_area:SHUF --filter=pixel_area:UD=32015,1,1,2 \
       -i pixel_area.nc -o pixel_area_zstd2.h5
```

Sz (*pw_relBoundRatio* = *5e-9*, Gzip_BEST_SPEED) compression is performed using the following command line after having correctly configured the sz.config file:

```
h5repack --filter=ground_range_5:UD=32017,0 -i pixel_area.nc -o pixel_area_sz.h5
```

Bit Grooming (*nsd* = 11) + Shuffle + Deflate (1) compression is performed using the following command line:

```
ncks -O -4 -L 1 --ppc pixel_area=11 pixel_area.nc pixel_area_bg.nc
```

Digit Rounding (*nsd* = 11) + Shuffle + Deflate (1) compression is performed using the following command line:

```
h5repack h5 --filter=pixel_area:UD=47987,1,1,11 --filter=pixel_area:SHUF \
      --filter=pixel_area:GZIP=1 -i pixel_area.nc -o pixel_area_dr.h5
```

The following command lines allow reproducing the results provided in Table 12 of the paper.

We first decompress SWOT dataset:

```
nccopy -d 0 SWOT_L2_HR_PIXC_000_210_46N-R_main.nc SWOT_L2_HR_PIXC_decomp.nc
```

Then, Shuffle + Deflate (4) compression is performed using the following command line:

```
nccopy -s -d 4 SWOT_L2_HR_PIXC_decomp.nc SWOT_L2_HR_PIXC_dfl4.nc
```

Clipping + Shuffle + Zstd (2) compression is performed using the following command line:

```
h5repack --filter=SHUF --filter=UD=32015,0,1,2 \
      -i SWOT_L2_HR_PIXC_decomp.nc -o SWOT_L2_HR_PIXC_zstd2.h5
```

Bit Grooming + Shuffle + Deflate (1) compression is performed using the following command line, setting the *nsd* parameter on a per variable basis corresponding to the required precision (see Table 4):

```
ncks -4 -L 1 -O SWOT_L2_HR_PIXC_decomp.nc SWOT_L2_HR_PIXC_bg.nc --ppc default=8 \
      --ppc azimuth_index=4 --ppc classification=3 --ppc cross_track=15 \
      --ppc dheight_.?=6 --ppc dlook_dphase_.?=6 --ppc dphase=6 \
      --ppc dry_tropo_range_correction=4 --ppc height=6 --ppc ifgram_imag=15 \
      --ppc ifgram_real=15 --ppc illumination_time=15 --ppc instrument_.?=6 \
      --ppc ionosphere_range_correction=6 --ppc latitude=15 --ppc longitude=15 \
      --ppc num_rare_looks=2 --ppc phase_screen=6 --ppc pixel_area=11 --ppc range_index=4 \
      --ppc sensor_s=11 --ppc solid_earth_tide_height_correction=4 \
      --ppc wet_tropo_range_correction=4 --ppc xover_roll_correction=4
```

Digit Rounding + Shuffle + Deflate (1) compression is performed using the following command line, setting the *nsd* parameter on a per variable basis corresponding to the required precision (see Table 4):

```
h5repack -i SWOT_L2_HR_PIXC_decomp.nc -o SWOT_L2_HR_PIXC_dr.h5 \
      --filter=azimuth_index:UD=47987,0,1,4 --filter=azimuth_index:SHUF --
      filter=azimuth_index:GZIP=1 --filter=classification:UD=47987,0,1,3 --
      filter=classification:SHUF --filter=classification:GZIP=1 --
      filter=coherent_power:UD=47987,0,1,8 --filter=coherent_power:SHUF --
      filter=coherent_power:GZIP=1 --filter=continuous_classification:UD=47987,0,1,8 --
      filter=continuous_classification:SHUF --filter=continuous_classification:GZIP=1 --
      filter=cross_track:UD=47987,0,1,15 --filter=cross_track:SHUF --
      filter=cross_track:GZIP=1 --filter=dark_water_flag:UD=47987,0,1,8 --
      filter=dark_water_flag:SHUF --filter=dark_water_flag:GZIP=1 --
      filter=delta_x:UD=47987,0,1,8 --filter=delta_x:SHUF --filter=delta_x:GZIP=1 --
      filter=delta_y:UD=47987,0,1,8 --filter=delta_y:SHUF --filter=delta_y:GZIP=1 --
      filter=delta_z:UD=47987,0,1,8 --filter=delta_z:SHUF --filter=delta_z:GZIP=1 --
      filter=dheight_dbaseline:UD=47987,0,1,6 --filter=dheight_dbaseline:SHUF --
      filter=dheight_dbaseline:GZIP=1 --filter=dheight_dphase:UD=47987,0,1,8 --
      filter=dheight_dphase:SHUF --filter=dheight_dphase:GZIP=1 --
      filter=dheight_drange:UD=47987,0,1,6 --filter=dheight_drange:SHUF --
      filter=dheight_drange:GZIP=1 --filter=dheight_droll:UD=47987,0,1,8 --
      filter=dheight_droll:SHUF --filter=dheight_droll:GZIP=1 --
      filter=dlook_dphase_x:UD=47987,0,1,6 --filter=dlook_dphase_x:SHUF --
      filter=dlook_dphase_x:GZIP=1 --filter=dlook_dphase_y:UD=47987,0,1,6 --
```

```
      filter=dlook_dphase_y:SHUF --filter=dlook_dphase_y:GZIP=1 --
      filter=dlook_dphase_z:UD=47987,0,1,6 --filter=dlook_dphase_z:SHUF --
      filter=dlook_dphase_z:GZIP=1 --filter=dphase:UD=47987,0,1,6 --filter=dphase:SHUF --
      filter=dphase:GZIP=1 --filter=dry_tropo_range_correction:UD=47987,0,1,4 --
      filter=dry_tropo_range_correction:SHUF --filter=dry_tropo_range_correction:GZIP=1 --
      filter=height:UD=47987,0,1,6 --filter=height:SHUF --filter=height:GZIP=1 --
      filter=ice_flag:UD=47987,0,1,8 --filter=ice_flag:SHUF --filter=ice_flag:GZIP=1 --
      filter=ifgram_imag:UD=47987,0,1,15 --filter=ifgram_imag:SHUF --
      filter=ifgram_imag:GZIP=1 --filter=ifgram_real:UD=47987,0,1,15 --
      filter=ifgram_real:SHUF --filter=ifgram_real:GZIP=1 --
      filter=illumination_time:UD=47987,0,1,15 --filter=illumination_time:SHUF --
      filter=illumination_time:GZIP=1 --filter=instrument_attitude_correction:UD=47987,0,1,6
      --filter=instrument_attitude_correction:SHUF --
      filter=instrument_attitude_correction:GZIP=1 --
      filter=instrument_baseline_correction:UD=47987,0,1,6 --
      filter=instrument_baseline_correction:SHUF --
      filter=instrument_baseline_correction:GZIP=1 --
      filter=instrument_phase_correction:UD=47987,0,1,6 --
      filter=instrument_phase_correction:SHUF --filter=instrument_phase_correction:GZIP=1 --
      filter=instrument_range_correction:UD=47987,0,1,6 --
      filter=instrument_range_correction:SHUF --filter=instrument_range_correction:GZIP=1 --
      filter=ionosphere_range_correction:UD=47987,0,1,6 --
      filter=ionosphere_range_correction:SHUF --filter=ionosphere_range_correction:GZIP=1 --
      filter=latitude:UD=47987,0,1,15 --filter=latitude:SHUF --filter=latitude:GZIP=1 --
      filter=longitude:UD=47987,0,1,15 --filter=longitude:SHUF --filter=longitude:GZIP=1 --
      filter=look_unit_x:UD=47987,0,1,8 --filter=look_unit_x:SHUF --
      filter=look_unit_x:GZIP=1 --filter=look_unit_y:UD=47987,0,1,8 --
      filter=look_unit_y:SHUF --filter=look_unit_y:GZIP=1 --
      filter=look_unit_z:UD=47987,0,1,8 --filter=look_unit_z:SHUF --
      filter=look_unit_z:GZIP=1 --filter=num_rare_looks:UD=47987,0,1,2 --
      filter=num_rare_looks:SHUF --filter=num_rare_looks:GZIP=1 --
      filter=phase_screen:UD=47987,0,1,6 --filter=phase_screen:SHUF --
      filter=phase_screen:GZIP=1 --filter=pixel_area:UD=47987,0,1,11 --
      filter=pixel_area:SHUF --filter=pixel_area:GZIP=1 --filter=power_left:UD=47987,0,1,8 -
      -filter=power_left:SHUF --filter=power_left:GZIP=1 --filter=power_right:UD=47987,0,1,8
      --filter=power_right:SHUF --filter=power_right:GZIP=1 --
      filter=range_index:UD=47987,0,1,4 --filter=range_index:SHUF --
      filter=range_index:GZIP=1 --filter=reference_layover_flag:UD=47987,0,1,8 --
      filter=reference_layover_flag:SHUF --filter=reference_layover_flag:GZIP=1 --
      filter=reference_layover_height_error:UD=47987,0,1,8 --
      filter=reference_layover_height_error:SHUF --
      filter=reference_layover_height_error:GZIP=1 --filter=sensor_s:UD=47987,0,1,11 --
      filter=sensor_s:SHUF --filter=sensor_s:GZIP=1 --
      filter=solid_earth_tide_height_correction:UD=47987,0,1,4 --
      filter=solid_earth_tide_height_correction:SHUF --
      filter=solid_earth_tide_height_correction:GZIP=1 --
      filter=wet_tropo_range_correction:UD=47987,0,1,4 --
      filter=wet_tropo_range_correction:SHUF --filter=wet_tropo_range_correction:GZIP=1 --
      filter=xover_roll_correction:UD=47987,0,1,4 --filter=xover_roll_correction:SHUF --
      filter=xover_roll_correction:GZIP=1)
```

The following command lines allow reproducing the results provided in Table 13 of the paper.

Shuffle + Deflate (4) compression is performed using the following command line:

```
nccopy -s -d 4 pixel_cloud.nc PIXEL_CLOUD_dfl4.nc
```

Clipping + Shuffle + Zstd (2) compression is performed using the following command line:

```
h5repack --filter=SHUF --filter=UD=32015,0,1,2 \
       -i pixel_cloud_decomp.nc -o pixel_cloud_zstd2.h5
```

Bit Grooming + Shuffle + Deflate (1) compression is performed using the following command line, setting the *nsd* parameter on a per variable basis corresponding to the required precision (see Table 5):

```
ncks -4 -L 1 -O pixel_cloud_decomp.nc -o pixel_cloud_bg.nc --ppc default=8 \
       --ppc azimuth_index=4 --ppc classification=3 --ppc cross_track=15 \
       --ppc dlatitude_dphase=5 --ppc dlongitude_dphase=5 --ppc height=6 --ppc ifgram=7 \
       --ppc illumination_time=15 --ppc latitude=15 --ppc longitude=15 \
       --ppc num_med_looks=3 --ppc num_rare_looks=2 --ppc phase_noise_std=7 \
       --ppc pixel_area=11 --ppc power_left=7 --ppc range_index=4 --ppc regions=7 \
       --ppc sigma0=7 --ppc x_factor.?=7
```

Digit Rounding + Shuffle + Deflate (1) compression is performed using the following command line, setting the nsd parameter on a per variable basis corresponding to the required precision (see Table 5):

```
h5repack -i pixel_cloud_decomp.nc -o pixel_cloud_dr.h5 \
       --filter=azimuth_index:UD=47987,0,1,4 --filter=azimuth_index:SHUF --
       filter=azimuth_index:GZIP=1 --filter=classification:UD=47987,0,1,3 --
       filter=classification:SHUF --filter=classification:GZIP=1 --
       filter=coherent_power:UD=47987,0,1,8 --filter=coherent_power:SHUF --
       filter=coherent_power:GZIP=1 --filter=continuous_classification:UD=47987,0,1,8 --
       filter=continuous_classification:SHUF --filter=continuous_classification:GZIP=1 --
       filter=cross_track:UD=47987,0,1,15 --filter=cross_track:SHUF --
       filter=cross_track:GZIP=1 --filter=dheight_dphase:UD=47987,0,1,8 --
       filter=dheight_dphase:SHUF --filter=dheight_dphase:GZIP=1 --
       filter=dlatitude_dphase:UD=47987,0,1,5 --filter=dlatitude_dphase:SHUF --
       filter=dlatitude_dphase:GZIP=1 --filter=dlongitude_dphase:UD=47987,0,1,5 --
       filter=dlongitude_dphase:SHUF --filter=dlongitude_dphase:GZIP=1 --
       filter=height:UD=47987,0,1,6 --filter=height:SHUF --filter=height:GZIP=1 --
       filter=ifgram:UD=47987,0,1,7 --filter=ifgram:SHUF --filter=ifgram:GZIP=1 --
       filter=illumination_time:UD=47987,0,1,15 --filter=illumination_time:SHUF --
       filter=illumination_time:GZIP=1 --filter=incidence_angle:UD=47987,0,1,8 --
       filter=incidence_angle:SHUF --filter=incidence_angle:GZIP=1 --
       filter=latitude:UD=47987,0,1,15 --filter=latitude:SHUF --filter=latitude:GZIP=1 --
       filter=longitude:UD=47987,0,1,15 --filter=longitude:SHUF --filter=longitude:GZIP=1 --
       filter=num_med_looks:UD=47987,0,1,3 --filter=num_med_looks:SHUF --
       filter=num_med_looks:GZIP=1 --filter=num_rare_looks:UD=47987,0,1,2 --
       filter=num_rare_looks:SHUF --filter=num_rare_looks:GZIP=1 --
       filter=phase_noise_std:UD=47987,0,1,7 --filter=phase_noise_std:SHUF --
       filter=phase_noise_std:GZIP=1 --filter=pixel_area:UD=47987,0,1,11 --
       filter=pixel_area:SHUF --filter=pixel_area:GZIP=1 --filter=power_left:UD=47987,0,1,8 -
       -filter=power_left:SHUF --filter=power_left:GZIP=1 --filter=power_right:UD=47987,0,1,8
       --filter=power_right:SHUF --filter=power_right:GZIP=1 --
       filter=range_index:UD=47987,0,1,4 --filter=range_index:SHUF --
       filter=range_index:GZIP=1 --filter=regions:UD=47987,0,1,7 --filter=regions:SHUF --
       filter=regions:GZIP=1 --filter=sigma0:UD=47987,0,1,7 --filter=sigma0:SHUF --
       filter=sigma0:GZIP=1 --filter=x_factor_left:UD=47987,0,1,7 --filter=x_factor_left:SHUF
       --filter=x_factor_left:GZIP=1 --filter=x_factor_right:UD=47987,0,1,7 --
       filter=x_factor_right:SHUF --filter=x_factor_right:GZIP=1
```

Example usage with netCDF-4 tools

From version 4.6.0 - January 24, 2018, netCDF supports HDF5 dynamic filters. It is now possible to make use of compression filters through *nccopy* tool. However, currently it supports only one filter: it does not yet allow chaining several filters such as Shuffle, Digit Rounding and Deflate. Nevertheless, below is provided an example of command line calling the Digit Rounding algorithm with *nsd* = 3 on the dataset ground_range_5_clipped.nc.

```
nccopy -F "ground_range_5,47987,3" ground_range_5_clipped.nc ground_range_5_clipped_dr.nc
```