Supplement of Geosci. Model Dev., 9, 3199–3211, 2016
http://www.geosci-model-dev.net/9/3199/2016/
doi:10.5194/gmd-9-3199-2016-supplement

Geoscientific Model Development
Open Access

Supplement of

Bit Grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4.4.8+)

Charles S. Zender

Correspondence to: Charles S. Zender (zender@uci.edu)

Supplement

This supplement details the commands and datasets necessary to reproduce the results tabulated in the paper. For Tables 1–2, first place the exact value π in a variable named, say, *pi* in a netCDF file named, say, in.nc. (Alternatively, use the file in.nc that comes with NCO). Then apply Bit Grooming and Decimal rounding as follows:

```
# Define pi
ncap2 -s ’pi=3.14159265358979323846264338327950 29’ in.nc in.nc
# Bit Groom to every level from 1 to 9 significant digits
ncks -v pi --ppc pi=1 in.nc nsd1.nc
ncks -v pi --ppc pi=2 in.nc nsd2.nc
ncks -v pi --ppc pi=3 in.nc nsd3.nc
ncks -v pi --ppc pi=4 in.nc nsd4.nc
ncks -v pi --ppc pi=5 in.nc nsd5.nc
ncks -v pi --ppc pi=6 in.nc nsd6.nc
ncks -v pi --ppc pi=7 in.nc nsd7.nc
ncks -v pi --ppc pi=8 in.nc nsd8.nc
ncks -v pi --ppc pi=9 in.nc nsd9.nc
# Decimal rounding to 2 significant decimal places
ncks -v pi --ppc pi=.2 in.nc dsd2.nc
# Print to sixteen decimals
ncks -v pi -s %20.16e -C -H nsd1.nc
```

Many sites like http://www.h-schmidt.net/FloatConverter/IEEE754.html show the IEEE binary format of the resulting decimal numbers.

These instructions produce the statistical evaluation of Bit Grooming vs. Bit Shaving in Table 3.

```
# Convert MERRA assimilation downloaded from NASA from HDF to netCDF
# and extract temperature T
ncks -3 -v T MERRA300.prod.assim.inst3_3d_asm_Cp.20130601.hdf T.nc
# Delete extraneous packing information
ncatted -a scale_factor,,d,, -a add_offset,,d,, T.nc
# Copy MERRA T into SP and DP PPC input files
# Use separate variable name for each Bit Grooming level
# SP (Single Precision):
ncap2 -s ’ppc=T;nsd1=nsd2=nsd3=nsd4=nsd5=nsd6=nsd7=ppc’ T.nc ppc_in.nc
# DP (Double Precision):
```

```
ncap2 -s ’ppc=double(T);nsd1=nsd2=nsd3=nsd4=nsd5=nsd6=nsd7=ppc’ \
      T.nc ppc_in.nc
# Artificial SP dataset
ncap2 -s ’defdim("dmn",1000000);ppc=float(array(1.0,1.e-6,$dmn))’ \
      -s ’nsd1=nsd2=nsd3=nsd4=nsd5=nsd6=nsd7=ppc’ in.nc ppc_in.nc
# Artificial DP dataset
ncap2 -s ’defdim("dmn",1000000);ppc=array(1.0,1.e-6,$dmn);’ \
      -s ’nsd1=nsd2=nsd3=nsd4=nsd5=nsd6=nsd7=ppc’ in.nc ppc_in.nc

# Bit Groom input dataset
ncks --ppc nsd1=1 --ppc nsd2=2 --ppc nsd3=3 --ppc nsd4=4 --ppc nsd5=5 \
     --ppc nsd6=6 --ppc nsd7=7 ppc_in.nc ppc_out.nc
# Decimal Round input dataset
ncks --ppc nsd1=.1 --ppc nsd2=.2 --ppc nsd3=.3 --ppc nsd4=.4 \
     --ppc nsd5=.5 --ppc nsd6=.6 --ppc nsd7=.7 ppc_in.nc ppc_out.nc

# Subtract quantized from exact data
ncbo ppc_out.nc ppc_in.nc ppc_dff.nc
# Ratios of biases to exact data
ncbo -y dvd ppc_dff.nc ppc_in.nc ppc_rat.nc
# Multiply biases by scale factor for easy intercomparison
ncap2 -s ’nsd1*=10;nsd2*=100;nsd3*=1000;nsd4*=10000;nsd5*=100000;’ \
      -s ’nsd6*=1000000;nsd7*=10000000’ ppc_rat.nc ppc_rat_scl.nc
# Compute statistics of biases
ncwa -y avg ppc_rat_scl.nc ppc_avg.nc # Mean bias
ncwa -y max ppc_rat_scl.nc ppc_max.nc # Maximum bias
ncwa -y min ppc_rat_scl.nc ppc_min.nc # Minimum bias
ncwa -y mabs ppc_rat_scl.nc ppc_mabs.nc # Maximum absolute bias
ncwa -y mebs ppc_rat_scl.nc ppc_mebs.nc # Mean absolute bias
ncwa -y mibs ppc_rat_scl.nc ppc_mibs.nc # Minimum absolute bias
```

These instructions produce the compression ratios shown in Tables 4–7. The indicated files (total size ~ 1.2 GB) are available from http://figshare.com after contacting the author (zender at uci dot edu). Run the indicated commands on each input file and compute the compression ratio as the output file-size divided by the initial file-size.

```
# Tables 4-7
```

```
fl=dstmch90_clm.nc
fl=famipc5_ne30_v0.3_00003.cam.h0.1979-01.nc
fl=MERRA300.prod.assim.inst3_3d_asm_Cp.20130601.hdf
fl=OMI-Aura_L2-OMIAuraSO2_2012m1222-o44888_v01-00-2014m0107t114720.h5

# Use ls to obtain filesize for output files
# Compute compression ratio as Row A divided by output filesize
ls -l ${fl} # Row A
bzip2 -1 -f ${fl} # Row B
bzip2 -9 -f ${fl} # Row C
ncks -7 -L 0 ${fl} foo.nc # Row D
ncks -7 -L 1 ${fl} foo.nc # Row E
ncks -7 -L 9 ${fl} foo.nc # Row F
ncpdq -7 -L 0 ${fl} foo.nc # Row G
ncpdq -7 -L 1 ${fl} foo.nc # Row H
ncks -7 -L 1 --ppc default=7 ${fl} foo.nc # Row I
ncks -7 -L 1 --ppc default=6 ${fl} foo.nc # Row J
ncks -7 -L 1 --ppc default=5 ${fl} foo.nc # Row K
ncks -7 -L 1 --ppc default=4 ${fl} foo.nc # Row L
ncks -7 -L 1 --ppc default=3 ${fl} foo.nc # Row M
ncks -7 -L 1 --ppc default=2 ${fl} foo.nc # Row N
ncks -7 -L 1 --ppc default=1 ${fl} foo.nc # Row O
```