Compression of LIGO data with wavelets

Presented by S.Klimenko

University of Florida

- Outline
  - Compression without losses
    - Random Data Compression (*rdc*)
    - data de-correlation with wavelets
    - results for the engineering run data
  - Compression with losses
    - data dynamic range reduction in wavelet domain
    - results for the engineering run data
  - Conclusion

LIGO data

- LIGO data (LSC data analysis White Paper)
  - Level 0 - full data (~250 TB/year)
  - Level 1 - archived data (~50 TB/year)
  - Level 2 - IFO + data quality channels (~5 TB/year)
  - Level 3 - whitened GW strain data (~0.5 TB/year)
- for Level 1 data lossles compression or compression with "minimal" losses (*quasi-lossless*) is desired.
- Level 2 - Level 3 are "science data sets". Reasonable lossy compression can be used to generate reduced data sets from Level 1 data.
- How can wavelet transforms be used for lossless and lossy compression of LIGO data?

Data Compression Concept

- data de-correlation
  - ➢ transformation that allows *more compact* representation of data
  - ➢ wavelets can be used to de-correlate data and decompose data into components that can be fairly well described as *white Gaussian (WG) noise.*
  - ➢ wavelets allow to combine data de-correlation and data reduction and use the same tool for lossless and lossy compression
- data compression with lossless encoder
  - ➢ many LIGO signals are mainly random Gaussian noise with admixture of non-Gaussian components.
  - ➢ wavelet transform makes data even "more random"
  - ➢ use encoder that is optimized for compression of Gaussian noise.

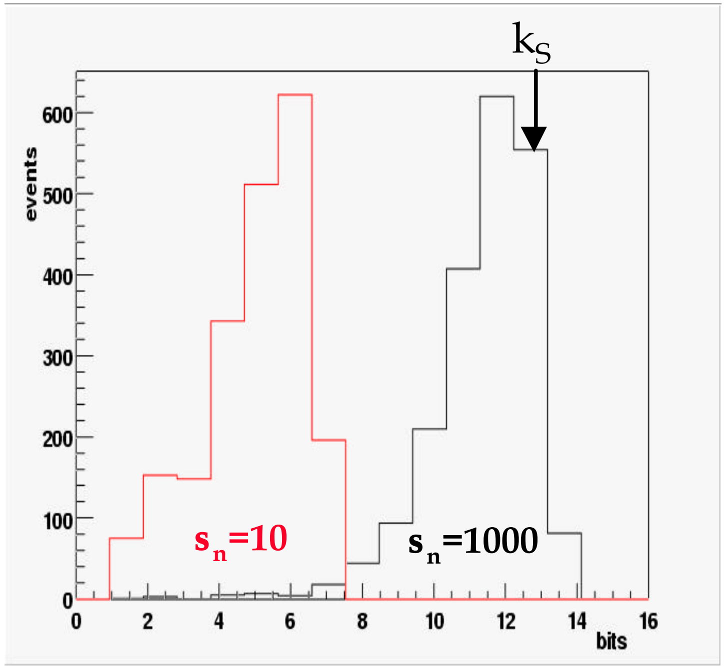
Compression of Random Data

- LIGO data is presented with 16bit words sampled at rate 1Hz - 16 kHz. A raw data set x_i with N samples is $2*N$ bytes long
- In case of a WG noise, $L \sim N \log_2(s_n)/8$ bytes needed to encode x_i ,where s_n is the noise rms
- *rdc* algorithm:
  - modification of 0 suppression method
  - find how many bits k_i needed to encode each number of data set x_i
  - k_L - “long” word (max(k_i))
  - k_S - “short” word (*min(L)*)

    $$L(k_S) = N_S(k_S) + N_L(k_S)\ (k_L + k_o)$$

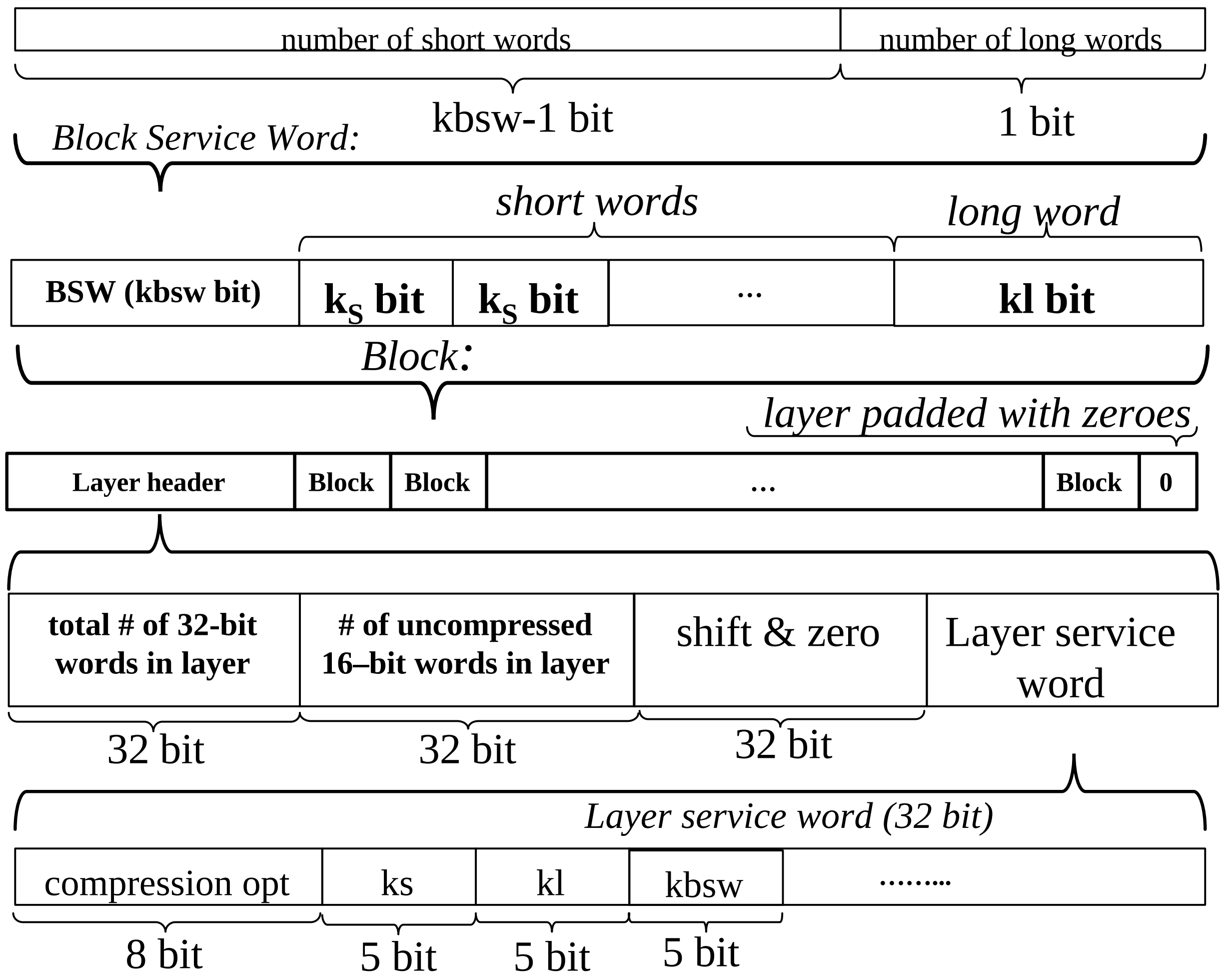
    k_o is overhead constant
  - code {x_i} as a sequence of blocks

BSW	k_S	k_S	k_S	k_S	k_L	k_L



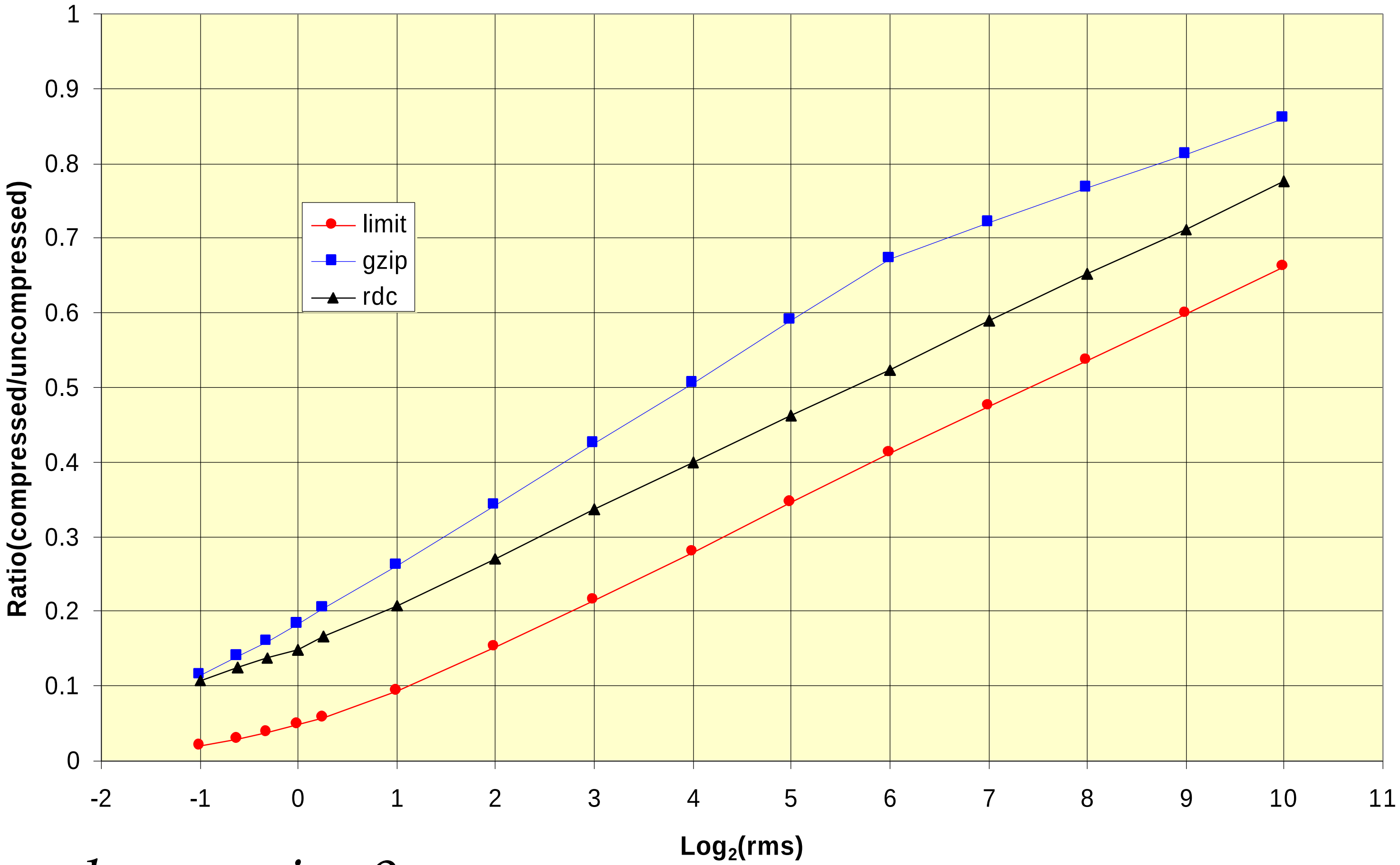
S.Klimenko

RDC encoder



number of short words
number of long words
kbsw-1 bit
1 bit
Block Service Word:
short words
long word
BSW (kbsw bit)
k_s bit
k_s bit
…
kl bit
Block:
layer padded with zeroes
Layer header
Block
Block
…
Block
0
total # of 32-bit words in layer
# of uncompressed 16–bit words in layer
shift & zero
Layer service word
32 bit
32 bit
32 bit
Layer service word (32 bit)
compression opt
ks
kl
kbsw
………
8 bit
5 bit
5 bit
5 bit

Compression of white Gaussian noise

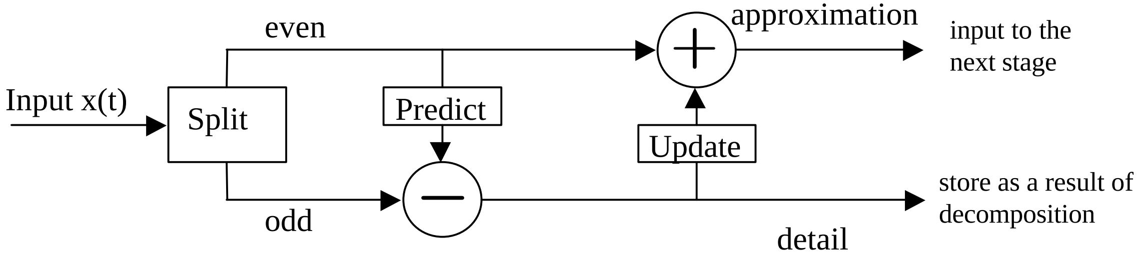


- *rdc* vs *gzip -9*
  - ➢ better compression factor for random Gaussian signals
  - ➢ ~5 times faster then *gzip -1*.

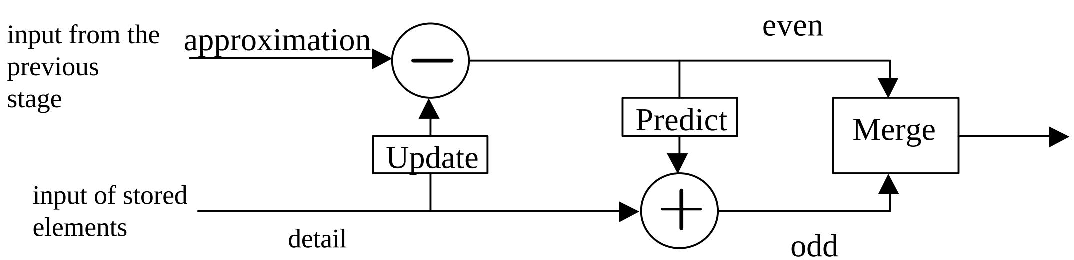
Lifting Wavelet Transform

R.C.Calderbank, I.Daubechies, W,Sweldwns, B.L.Yeo. ACHA, V5, N3, pp. 332-369, 199
Wavelet Transforms that Maps Integers to Integers.

decomposition (a)



reconstruction (b)

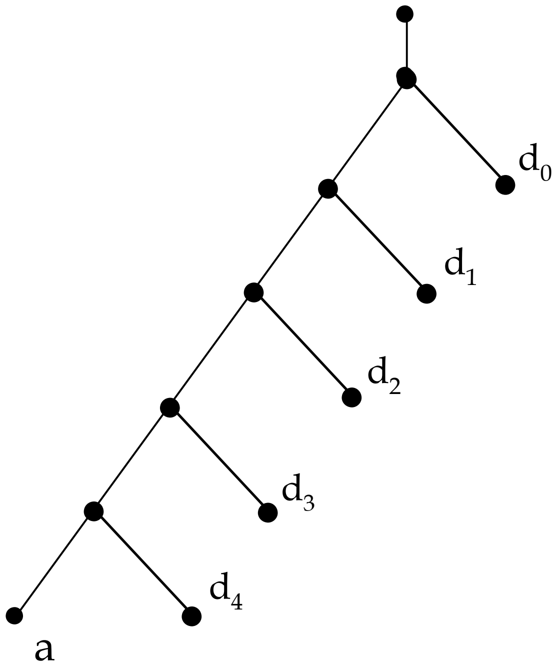


- twice faster than Fast Wavelet Transform.
- allows transforms that map integers to integers:

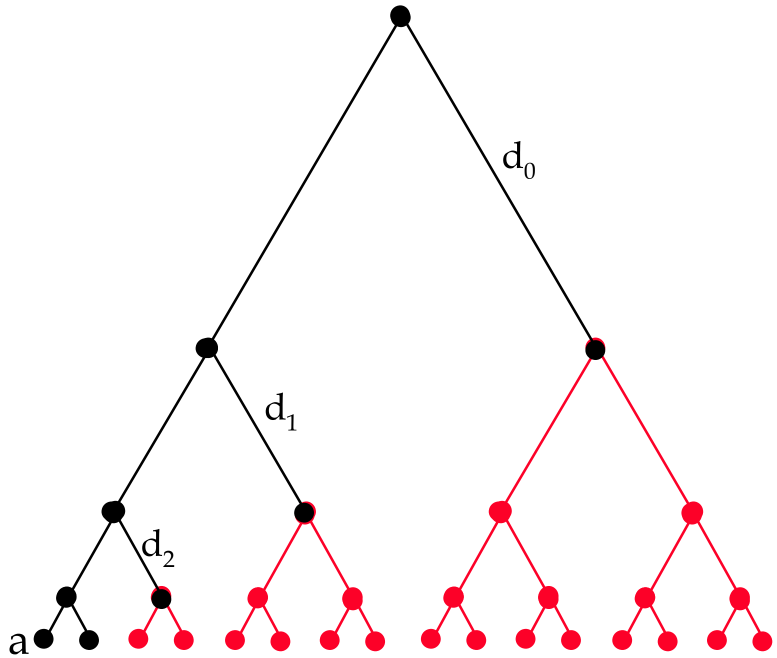
P_I = int(P), U_I = int(U) (for lossless compression)

S.Klimenko

Wavelet Transform Tree



a. wavelet transform tree



b. wavelet transform binary tree

- detail coefficients d_i represent data in different frequency bands
  - a. $df = f/2, f/4, f/8,$ ….. - dyadic basis
  - b. $df = f/n$, n - number of nodes in last layer - linear basis

Compression without losses

Compression ratios for 16kH data channels and different compression methods

Data type =>	Time domain data			TD data + differentiation			Wavelet domein data			WBTree	
Compression method => / Channel name	gzip	ERI	RDC	gzip	ERI	RDC	gzip	ERI	RDC	RDC	RDC+ gzip
H2:IOO-MC_F	1.48	2.30	1.55	1.88	2.06	2.33	1.87	1.98	2.40	2.35	2.36
H2:IOO-MC_I	1.41	1.68	1.75	1.38	1.63	1.65	1.39	1.61	1.75	1.78	1.78
H2:PSL-FSS_FAST_F	3.34	6.30	1.92	5.06	6.35	5.98	4.66	5.58	5.27	4.64	4.69
H2:PSL-FSS_MIXERM_F	1.22	1.35	1.34	1.24	1.35	1.40	1.24	1.35	1.42	1.44	1.44
H2:PSL-FSS_PCDRIVE_F	1.20	1.36	1.33	1.21	1.36	1.35	1.22	1.29	1.40	1.42	1.43
H2:PSL-ISS_ISERR_F	3.05	4.04	3.08	3.17	4.04	3.86	3.10	3.98	3.91	3.72	3.76
H2:LSC-AS_Q_TEMP	2.33	4.35	2.08	3.34	3.57	3.82	3.41	4.25	4.31	4.05	4.08
H2:LSC-AS_I_TEMP	1.13	1.24	1.22	1.16	1.26	1.27	1.17	1.29	1.30	1.43	1.44
H2:LSC-AS_DC_TEMP	3.64	6.01	2.46	4.96	6.23	6.00	4.61	5.68	5.48	4.81	4.86
Average compression ratio	1.72	2.19	1.71	1.89	2.13	2.18	1.88	2.11	2.23	2.24	2.25

Compression ratios for 2kH data channels and different compression methods

Data type =>	Time domain data			TD data + differentiation			Wavelet domein data			WBTree	
Compression method => / Channel name	gzip	ERI	RDC	gzip	ERI	RDC	gzip	ERI	RDC	RDC	RDC+ gzip
H0:PEM-BSC7_ACCX	1.60	1.90	2.00	1.47	1.70	1.83	1.52	1.71	2.05	2.12	2.14
H0:PEM-BSC5_ACCZ	1.84	2.34	2.28	1.71	1.97	2.20	1.77	1.88	2.41	2.46	2.48
H0:PEM-BSC7_ACCZ	1.73	2.21	2.21	1.66	1.87	2.18	1.71	1.86	2.27	2.27	2.29
H2:ASC-ETMX_P	1.12	1.17	1.21	1.08	1.09	1.13	1.11	1.12	1.22	1.23	1.10
H2:ASC-BS_P	1.17	1.27	1.30	1.12	1.17	1.22	1.15	1.20	1.31	1.29	1.30
H0:PEM-HAM7_ACCX	1.64	2.20	2.06	1.48	1.86	1.92	1.56	1.77	2.14	2.24	2.25
H0:PEM-BSC5_ACCY	1.72	2.15	2.20	1.54	1.74	1.98	1.61	1.77	2.17	2.27	2.28
H2:SUS-EMTX_SENSOR_UL	1.87	2.09	2.40	1.67	1.94	2.17	1.78	1.88	2.47	2.50	2.52
Average compression ratio	1.53	1.80	1.84	1.43	1.59	1.72	1.48	1.59	1.88	1.91	1.88

- Documentation: LIGO-T000076-00-D

Compression with losses

- applications
  - generate reduced data sets for data analysis (Level 2, Level 3)
  - compress environmental and control channels, where the very detail information may not be important
  - compress Level 1 data with *quasi-lossless* (very small losses) compression
- main problems
  - possibly loss of useful information (how to control it?)
  - artifacts can be added to compressed signal
  - different channels may require different compression algorithms
- possible solution
  - frequency dependant reduction of the data dynamic range in wavelet domain

Lossy compression with wavelets

- data dynamic range reduction

$$y = int(x/k); \;\; x' = ky$$

x - integer data set, x′ - reconstructed data set, k - reduction factor

y - data with reduced dynamic range that can be compressed by lossless compression program (gzip, rdc,…)

$$x' = x + d, \qquad d^2 << x^2$$

  - ➢ d - white noise generated by random process *int*, $rms = k/sqrt(12)$
  - ➢ no correlation between d and x, no artifacts
  - ➢ loss of information due to additional noise δ

- dynamic range reduction in wavelet domain

$$w_L' = k_L \; int(w_L/k_L), \;\; k_L \text{ - reduction factor for layer } L$$

  - ➢ wavelet allows to select different reduction factors for different frequency bands.

H2:LSC-AS_I_TEMP

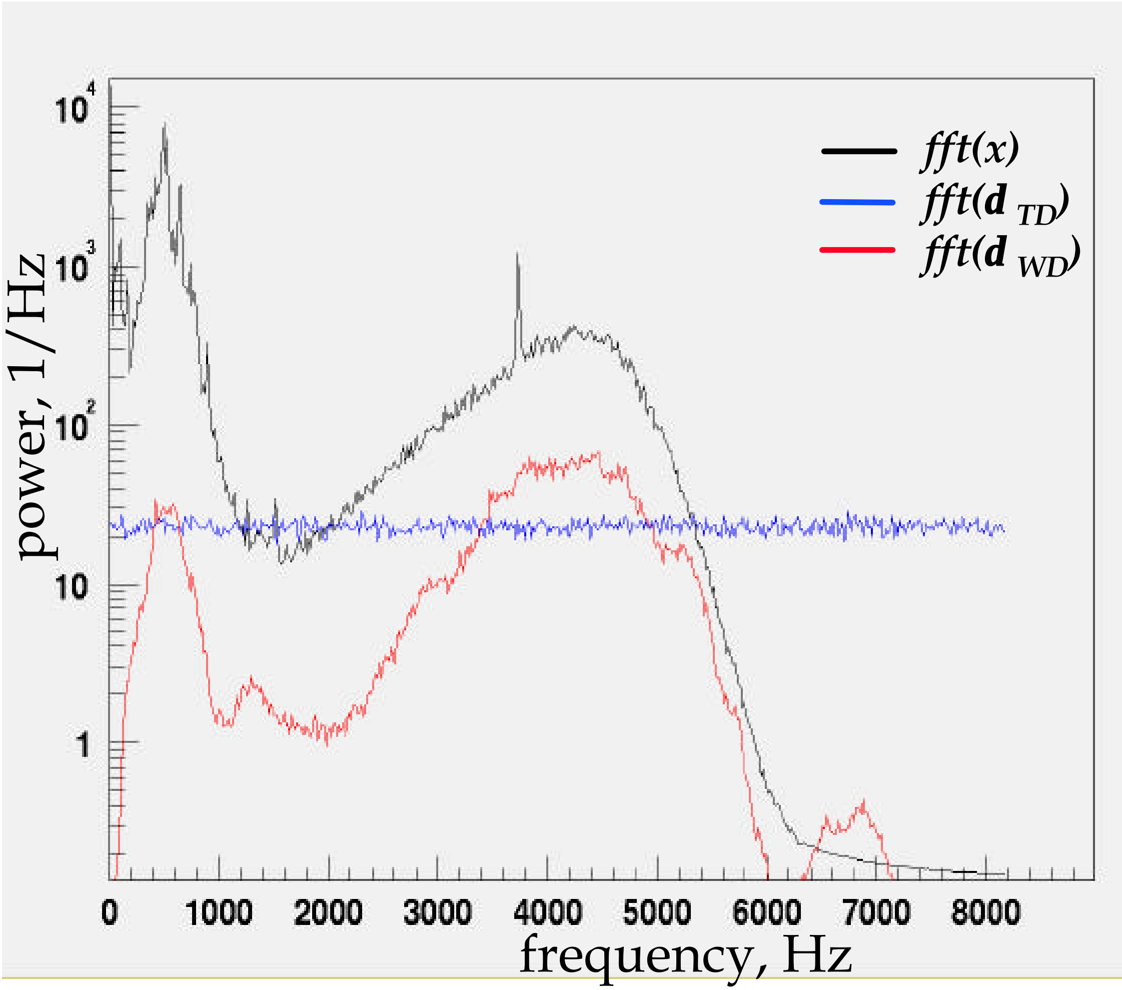
	losses ε,%	comp. factor
RDC	0	1.3
TD	8.5	6.7
WD	4.6	6.7

$$\bar{x}^2 = \overline{(x - \bar{x})^2}$$
$$\delta^2 = \overline{(x - x')^2}$$

x - original signal

x' - uncompressed signal

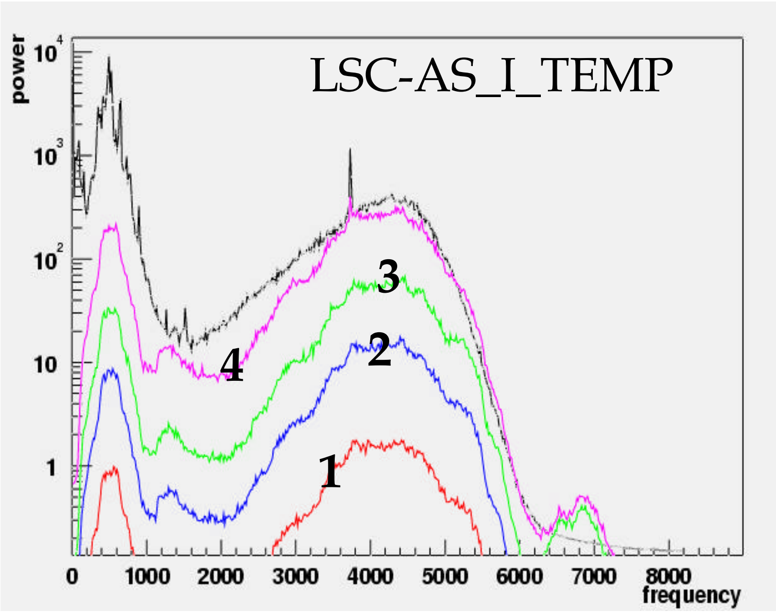
"losses": $\varepsilon = \delta^2/\bar{x}^2$

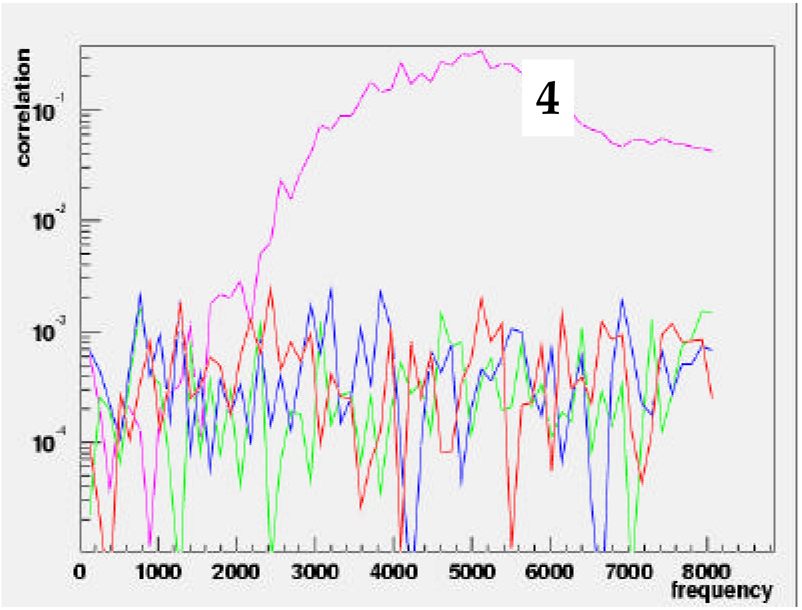


compression & x(d) correlation vs losses

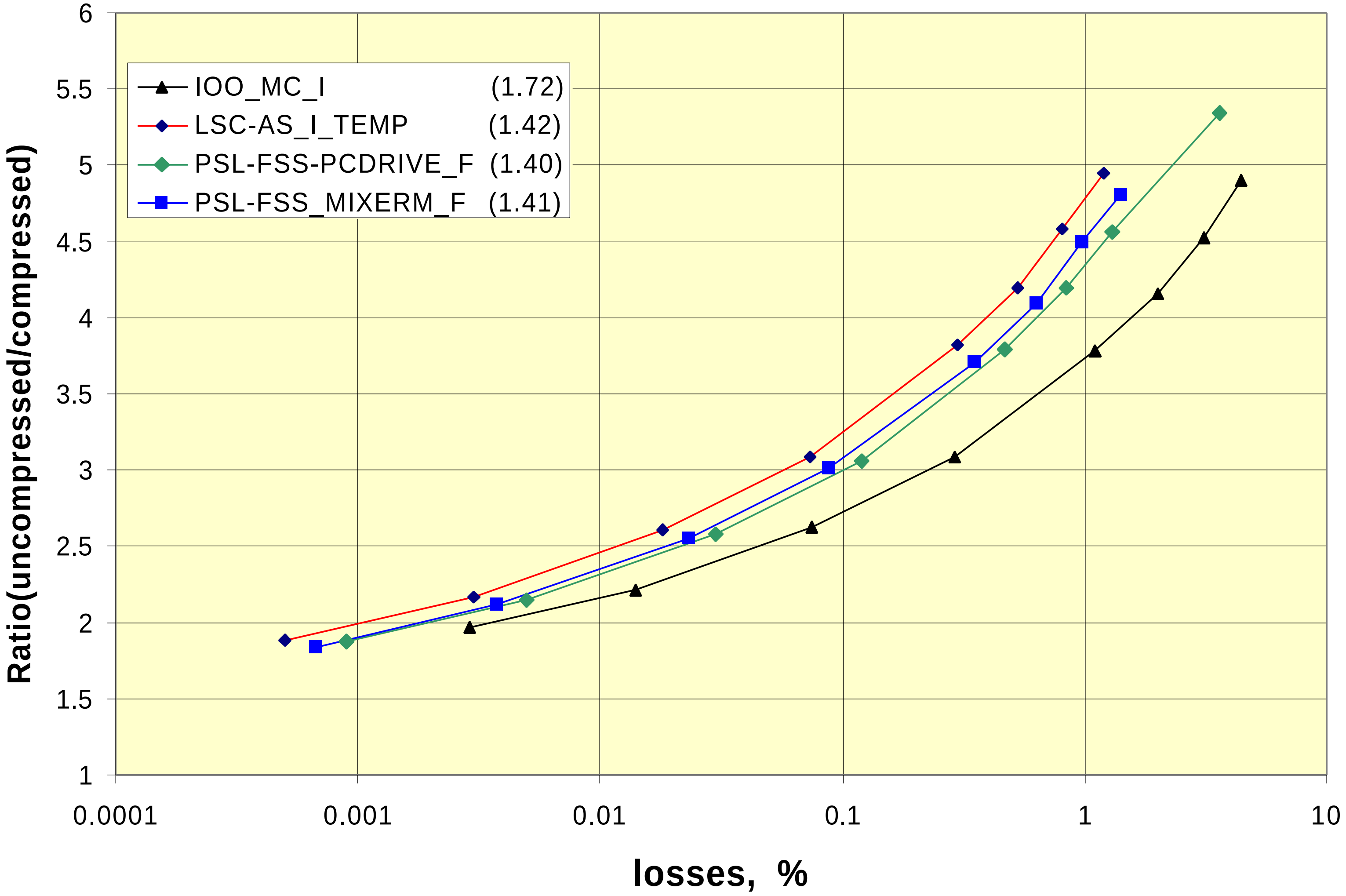
- No correlation between δ & x for (1,2,3)
- *quasi-lossless* compression (CR ~3) for ε < 0.1%

	ε, %	CF
1	0.13	3.4
2	1.2	5.0
3	4.6	6.7
4	23.	14.3





compression vs losses

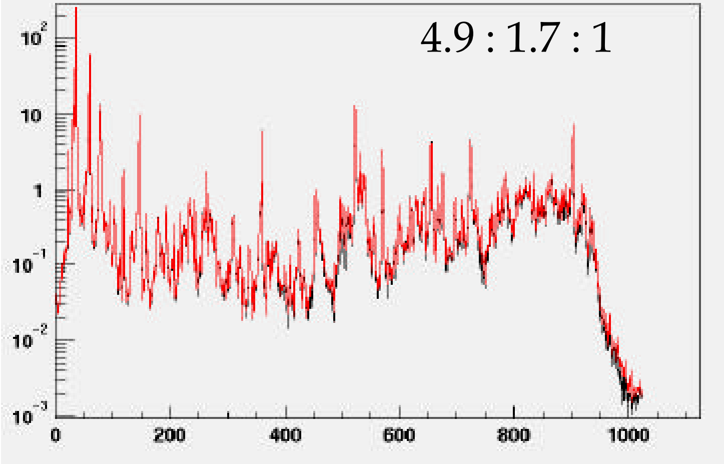


S.Klimenko

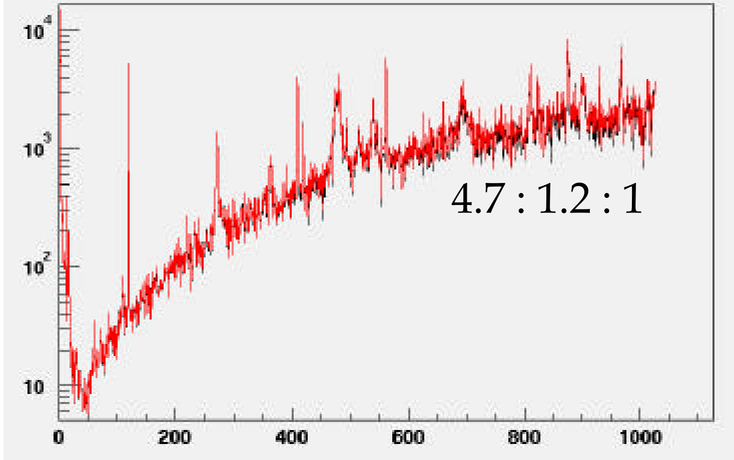
Examples (black - original, red - compressed)

black - original, red - compressed, lossy : lossless : 1

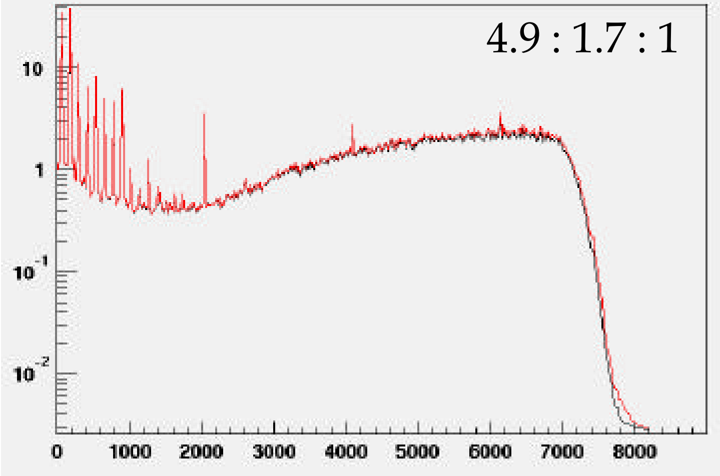
PEM-BSC7-ACCZ



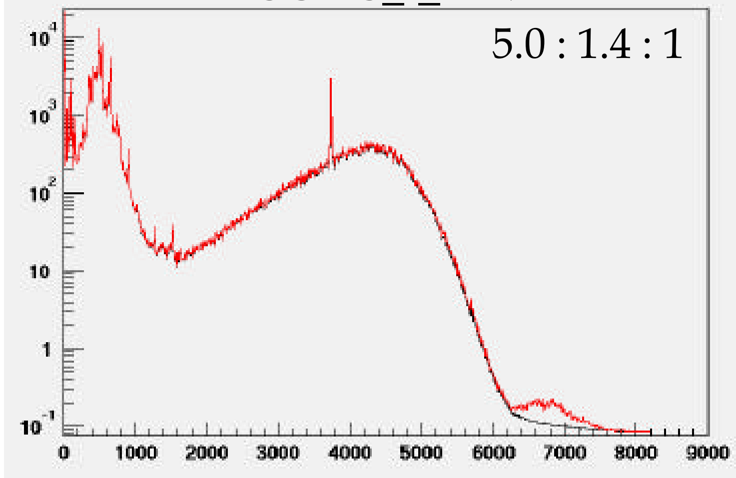
ASC-BS_P



IOO-MC-I



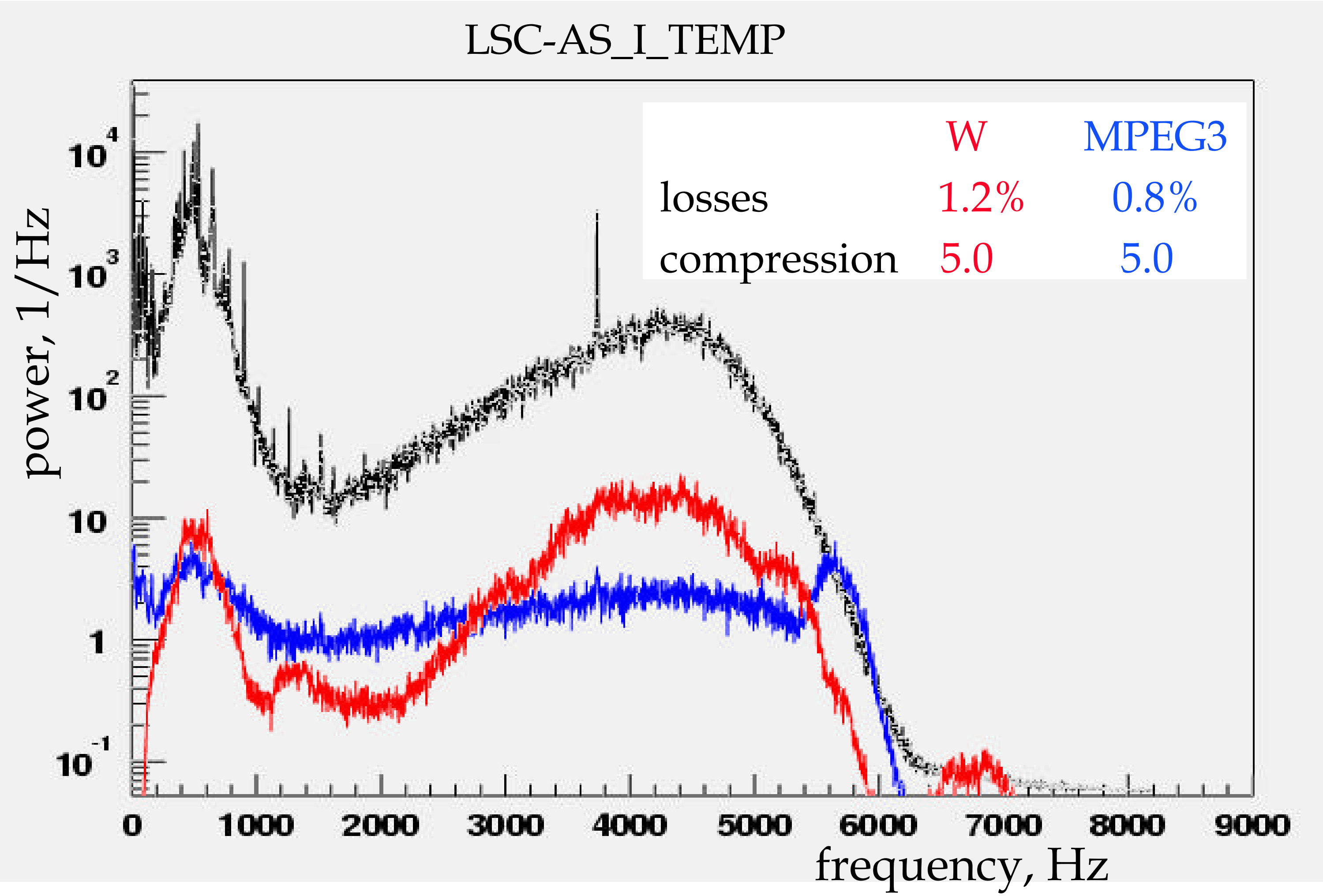
LSC-AS_I_TEMP



S.Klimenko

wavelet vs MPEG3

- MPEG3 - commercial audio compression program



Summary

- Fast and efficient data encoder (*rdc*) optimized for compression of white Gaussian noise is presented.
- Wavelets can be used to de-correlate and reduce data
  - ➢ for lossless compression the lifting wavelet transform that maps integers to integers is used.
  - ➢ for lossy compression the data dynamic range reduction in wavelet domain is used.
- Combination of wavelets and rdc offers a universal tool both for lossless and lossy compression that can be used to compress Level 1 data and generate Level 2-3 data.
- Flexible lossy compression. All options between quasi-lossless compression and decimation are possible.
- UF suggests to develop compression tool specifically for LIGO data.