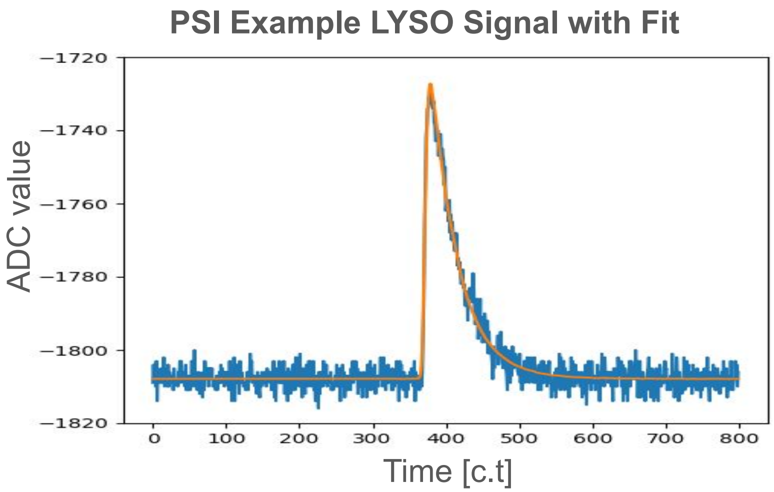
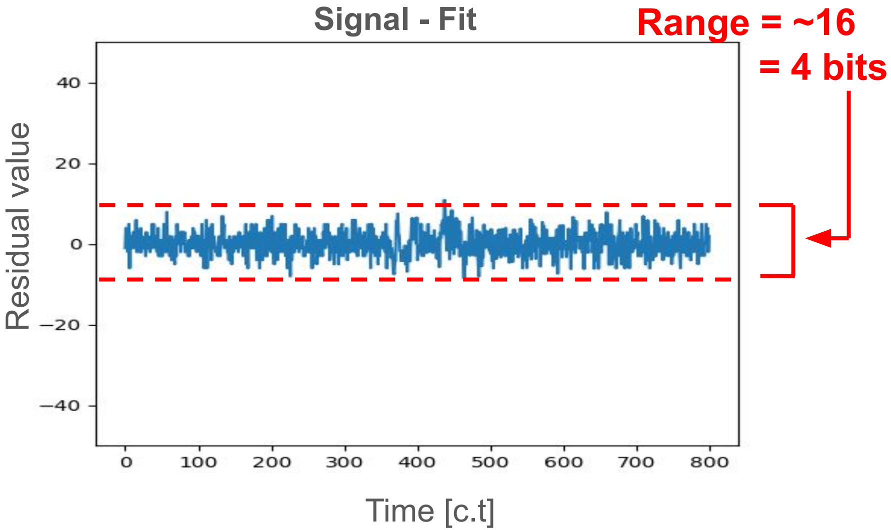
Asymmetric Numeral Systems (ANS) for Compression of Template Fit Data

Jack Carlton
University of Kentucky

Template Fitting

- Can construct a continuous template for our traces $T(t)$
- Can fit traces using template:
  $f(t) = A \cdot T(t - t_0) + B$
- Storing unfit traces takes ~12 bits per ADC sample
- Storing residuals takes ~4 bits per ADC sample
- By fitting and compressing, we can expect compress the data by a **factor of ~3** for "perfect" fits
  - Effective compressing random noise





Theoretical Best Compression

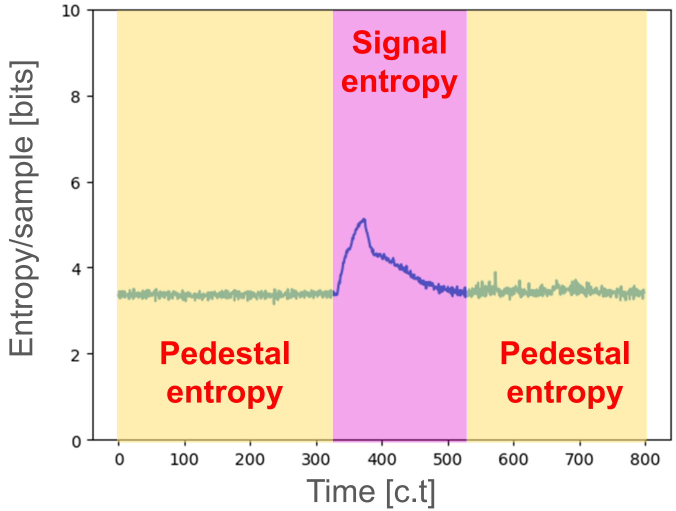
- For lossless compression, the best possible compression rate is the entropy rate
- Entropy rate of pedestal part of signal is **3.4 bits per ADC sample**
  - A perfect fit would reduce signal to pedestal noise
- Thus with perfect fit + compression true perfect compression factor is **~3.5**
  - Highly dependent on size of noise

Entropy Rate Formula
(First order, exact for independent residuals)

$$H(X_i) = \sum_{\text{traces}} p(X_i) \log_2 (p(X_i))$$

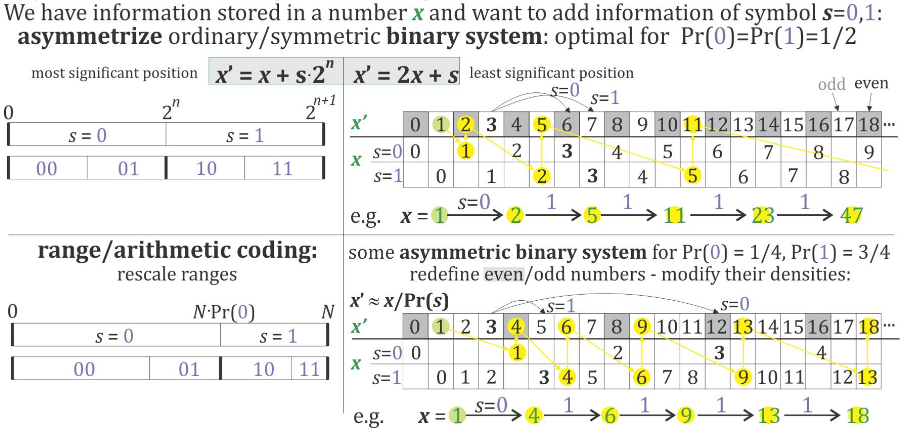
$$X_i \equiv \text{Random variable for } i^{\text{th}} \text{ ADC sample}$$

Entropy Rate of PSI Test Beam Data After Fitting



What is Asymmetric Numeral Systems (ANS)?

- Entropy encoding compression method
- Manages a state that represents encoded data
  - Can separate into many states for GPU parallelization
  - Better at encoding “fractional bits” than other methods like Huffman or Rice-Golomb
- Wikipedia Link



Alphabet and probabilities

$$\mathcal{A} = \{s_1, \ldots, s_k\}, \qquad p_i = \Pr(s_i), \qquad \sum_{i=1}^{k} p_i = 1$$

Finite precision model

$$F = 2^m, \qquad f_i \in \mathbb{N}, \qquad \sum_{i=1}^{k} f_i = F, \qquad \frac{f_i}{F} \approx p_i$$

Cumulative frequencies

$$C_i = \sum_{j<i} f_j$$

ANS state

$x \in \mathbb{N}$

Encoding update (rANS)

$$x' = \left\lfloor \frac{x}{f_s} \right\rfloor F + (x \bmod f_s) + C_s$$

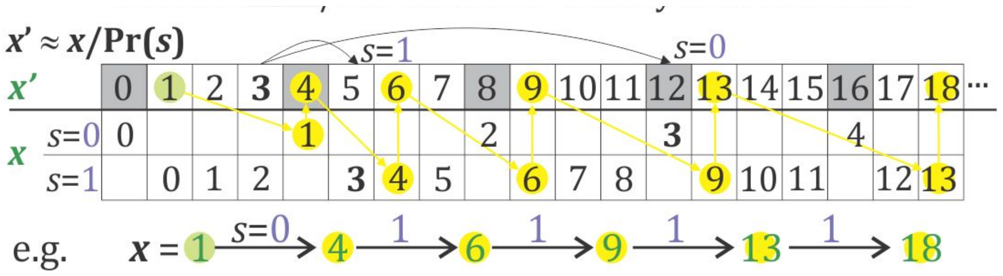
Decoding

$$r = x' \bmod F, \qquad s \text{ such that } C_s \le r < C_s + f_s,$$

$$x = f_s \left\lfloor \frac{x'}{F} \right\rfloor + (r - C_s)$$

What is Table Asymmetric Numeral Systems (tANS)?

- You can define the behavior of ANS (for fixed parameter choices) in a lookup table
- This removes the need for operations, rather each stage is a table lookup
  - Division expensive on GPU
- The cost is needing to store a table
  - This can be built one time based on measured probabilities of samples



Why ANS is good at being parallelized

- Two example approaches:
  - Block parallelism
  - Interleaved parallelism
- Both make it so:
  - One thread holds one state
  - Each state has an identical update rule
  - No synchronization is required across threads
- These make parallelization much more natural than other methods
  - Ex. Rice-Golomb, Huffman

Sequential structure of a single stream

$x_{i+1} = T(x_i, s_i) \quad \Rightarrow \quad$ single dependency chain

Parallelization principle

Maintain K independent states $x^{(0)}, \ldots, x^{(K-1)}$

Block-level parallelism

Partition data into blocks $B_0, \ldots, B_{K-1}$

Encode each block with its own ANS state

Interleaved (round-robin) parallelism

For symbol index $i = 0, 1, 2, \cdots : \quad j = i \bmod K$

$$x^{(j)} \leftarrow T\left(x^{(j)},\, s_i\right)$$

Explicitly:

$$s_0 \to x^{(0)},\ s_1 \to x^{(1)},\ \ldots,\ s_{K-1} \to x^{(K-1)},$$
$$s_K \to x^{(0)},\ s_{K+1} \to x^{(1)},\ \ldots,\ s_{2K-1} \to x^{(K-1)},$$
$$s_{2K} \to x^{(0)},\ s_{2K+1} \to x^{(1)},\ \ldots$$

Things to test/research questions

- At some point Sean had a template fitter for 2023 PSI LYSO waveforms
    - Can we easily get a sample of LYSO data to template fit to produce compressible integer residuals?
- Assuming we have integer residuals, these are the things we want to compress with ANS
    - What data throughput (in GB/s) can we expect using tANS?
    - What compression factor can we expect using tANS?
    - How do the above depend on choice of parameters (ex. Total frequency size F, which is related to the size of the table)
        - Expectation: as F goes up, compression factor improves but performance suffers
    - Compare compression factor to Shannon Entropy, which is the theoretical limit Sean computed for 2023 PSI LYSO data
        - Other useful metrics?

Auxiliary Slides

Should we delta encode?

- **Probably not**

- Delta encoding hurts compression if data has small covariance

- As a result, you should only delta encode if there is a positive correlation

Delta encoded expectation

non-delta encoded expectation

$$\mathbb{E}\left[(r_i - r_{i-1})^2\right] < \mathbb{E}\left[r_i^2\right]$$

$$\mathbb{E}\left[(r_i - r_{i-1})^2\right] = \mathbb{E}\left[r_i^2\right] + \mathbb{E}\left[r_{i-1}^2\right] - 2\,\mathbb{E}[r_i\, r_{i-1}$$

$$\mathbb{E}\left[(r_i - r_{i-1})^2\right] < \mathbb{E}\left[r_i^2\right] \iff \mathrm{Cov}(r_i, r_{i-1}) > 0$$

Alternatively, look at the correlation coefficient to see “how much” delta encoding will help or hurt

$$\rho = \frac{\mathbb{E}[r_i\, r_{i-1}]}{\sqrt{\mathbb{E}\left[r_i^2\right]\,\mathbb{E}\left[r_{i-1}^2\right]}} > 0$$

j.carlton@uky.edu

Template Fitting - How to store residuals

Model prediction from stored fit parameters

$$\hat{y}(t;\boldsymbol{\theta}) = A\,T(t-t_0) + B, \qquad \boldsymbol{\theta} = (A, t_0, B)$$

True data (integer ADC samples)

$$y(t) \in \mathbb{Z}$$

Define integer prediction via deterministic rounding

$$\tilde{y}(t) = \lceil \hat{y}(t;\boldsymbol{\theta}) \rceil$$

Define integer residual

$$r(t) = y(t) - \tilde{y}(t)$$

Lossless reconstruction

$$y(t) = \tilde{y}(t) + r(t)$$

Equivalently, using the floor identity

$$r(t) = \lfloor y(t) - \hat{y}(t;\boldsymbol{\theta}) \rfloor \in \mathbb{Z}$$

Floor/Ceiling identity (for $y \in \mathbb{Z},\ x \in \mathbb{R}$):

$$\lfloor y - x \rfloor = y - \lceil x \rceil$$

Therefore

$$y(t) = \lceil \hat{y}(t;\boldsymbol{\theta}) \rceil + \lfloor y(t) - \hat{y}(t;\boldsymbol{\theta}) \rfloor = \lceil \hat{y}(t;\boldsymbol{\theta}) \rceil + r(t)$$

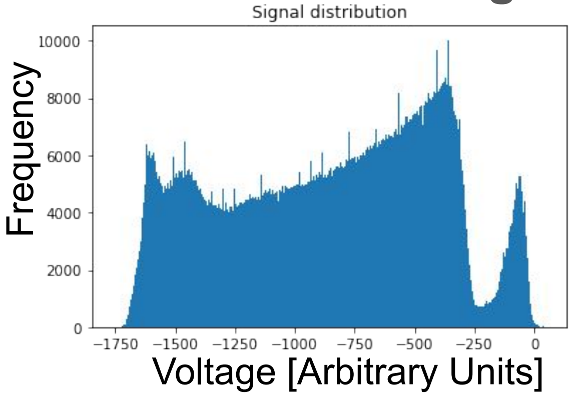
which is exact and lossless.

Fit's job:
1. Compute fit params + residuals
2. Store fit params
3. send **integer** residuals to compression algorithm

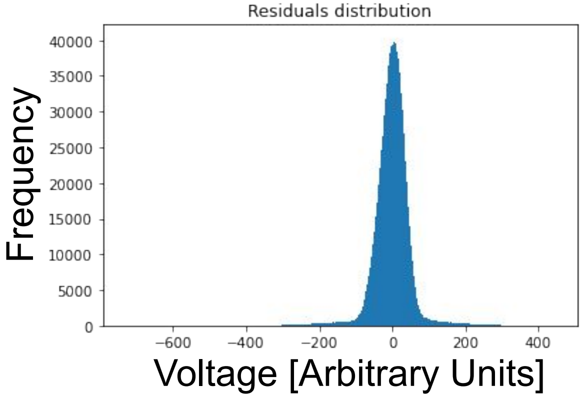
Rounding avoids messy binary float representations of the data

Rounding residuals is still lossless!

No Conditioning



Shape Fit



Higher Order Entropy Estimations

- Assume we have N characters (traces) in our alphabet (data set)
- **Zero order:** each character in alphabet is statistically independent

$$H = \log_2(N)$$

- **First order:** each character in alphabet is statistically independent, p_i is the probability of that character to occur

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

- **Second order:** $P_{j|i}$ is correlation between subsequent characters

$$H = -\sum_{i=1}^{N} p_i \sum_{j=1}^{N} P_{j|i} \log_2(P_{j|i})$$

- **General Model (impractical):** B_n represents the first n characters

$$H = \lim_{n\to\infty} \left[-\frac{1}{n} \sum p(B_n) \log_2(B_n) \right]$$

What is Asymmetric Numeral Systems (ANS)?

Alphabet and probabilities

$$\mathcal{A} = \{s_1, \dots, s_k\}, \qquad p_i = \Pr(s_i), \qquad \sum_{i=1}^{k} p_i = 1$$

Finite precision model

$$F = 2^m, \qquad f_i \in \mathbb{N}, \qquad \sum_{i=1}^{k} f_i = F, \qquad \frac{f_i}{F} \approx p_i$$

Cumulative frequencies

$$C_i = \sum_{j<i} f_j$$

Normalized ANS state

$L = 2^R$ (normalization bound), $\qquad x \in \mathbb{N}, \qquad x \geq L$

Renormalization (emit/consume bits, base $b = 2^B$)

encode: while $x \geq f_s L$, output $(x \bmod b)$, $x \leftarrow \lfloor x/b \rfloor$

decode: while $x < L$, $x \leftarrow b\,x + \text{next bits}$

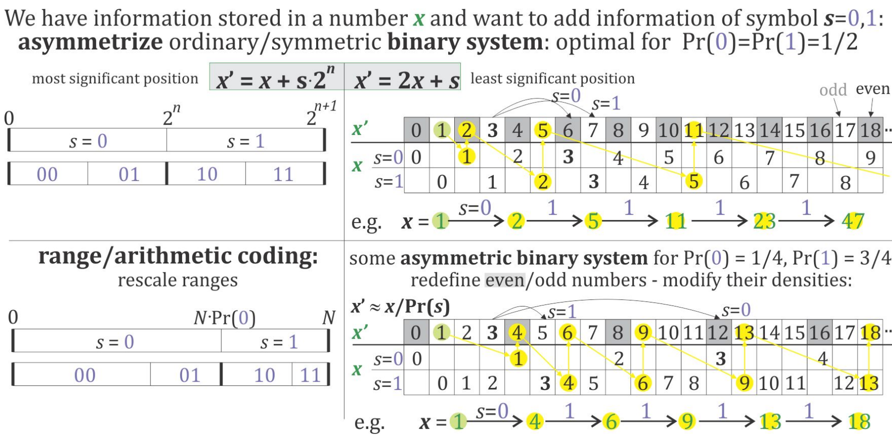
Encoding update (rANS)

$$x' = \left\lfloor \frac{x}{f_s} \right\rfloor F + (x \bmod f_s) + C_s$$

Decoding

$r = x' \bmod F, \qquad s$ such that $C_s \leq r < C_s + f_s$,

$$x = f_s \left\lfloor \frac{x'}{F} \right\rfloor + (r - C_s)$$



This is like a “buffer” between bitstream and the state. Your state, likely encoded in a 64 bit integer, will overflow when encoding or decoding. So you need to “emit” or “consume” bits to an arbitrary long bitstream as that happens.