Overview of Analysis Methods for CMB Data

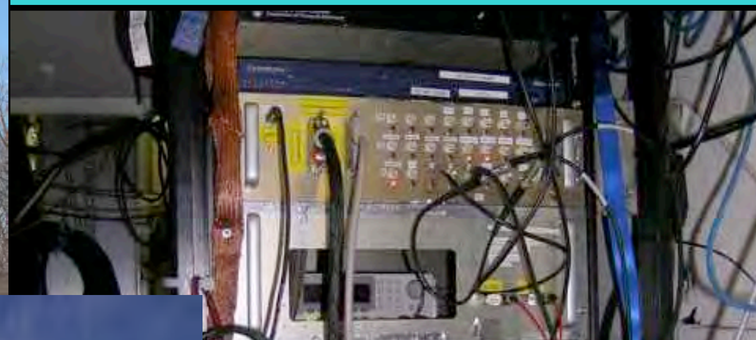
Suzanne Staggs
Princeton University

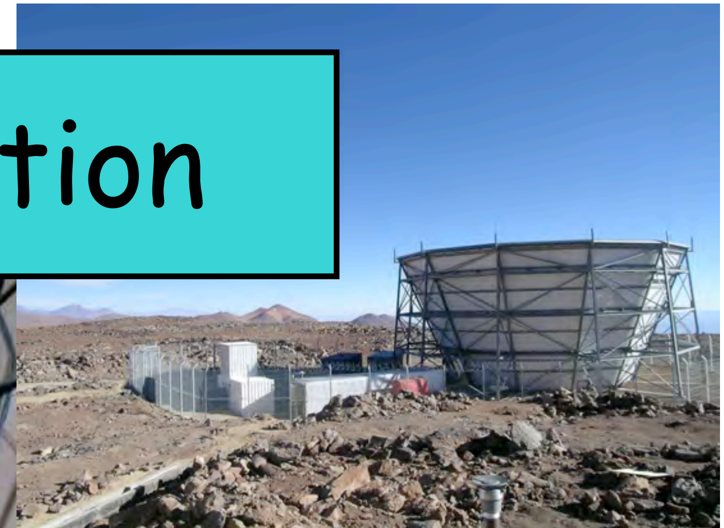
Gunma, Japan; 11 Feb 09

PIPELINE FLOW

- Data Collection
- Data Selection (cutting bad data)
- Data Filtering
- Data Calibration (relative calibration)
- Noise Estimation
- Map Estimation
- Foreground Corrections
- Power Spectrum Estimation
- Cosmological Parameter Estimation

Data Collection





•The output of the first few steps of the pipeline is a time-ordered data vector, often called a timestream or TOD. It comes with telescope position data, since the detectors were scanning across the celestial sphere.

•The complication of multiple detectors is neglected in most of what follows (their TODs are usually concatenated).

•We will use d_t for the t^{th} sample of a TOD.

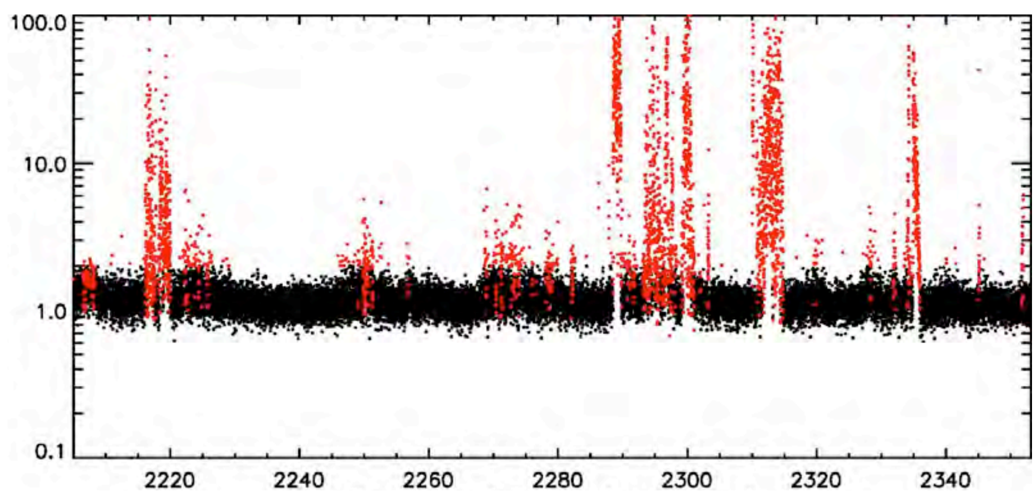
Data Selection

- Cut data for cause; examples:
  - Bad weather as indicated by external sensors: snow, high partial water vapor
  - Bad weather as indicated by data: eg, unusually high 1/f from atmosphere
  - Instrument problems (telescope not moving, for example)
  - Cosmic ray hits & other glitches
  - Miscellaneous but impartial reasons
- Optionally, fill in gaps in the timestreams with fake data with the same noise properties as the real data

Data Selection

- Caveat: must make sure not to impose a harsh cut on the data variance itself, since the CMB contributes to the measured variance
- Bad weather cutting in the CAPMAP polarization experiment
  - data were acquired by directing the telescope beam in a ring around the NCP.
  - the continuously changing elevation lead to a sinusoidal pickup of the atmospheric emission in the intensity detectors.
  - The intensity* data were fitted to a model of this scan-synchronous signal; when the χ^2 to the model exceeded a threshold, the data were cut.

*The intensity data, not the polarimetery data!



Data Filtering

- This takes different forms for different experiments
- Some experiments highpass filter the timestream data
- Others remove scan-synchronous modes
  - In CAPMAP, the polarization channels also picked up a small amount of atmosphere as the telescope scanned its circle; this pickup was removed with a five parameter fit to the ring variable.
- Some experiments have used a notch filter to remove, eg, a 60 Hz line visible in the power spectrum of the timestream
- All this filtering has two effects:
  - The overall experimental sensitivity is reduced
  - Maps are missing modes (can think of this as literally missing Fourier modes), so maps from different experiments may look different
  - Naturally, the filtering must be accounted for in estimating the power spectrum

Data Calibration

- Many experiments get their final calibrations from the WMAP temperature anisotropy as one of the last steps in the pipeline
- Nonetheless, pre-calibration is done for several reasons
  - The basic idea is to convert from DAC units to physical units
  - Allows for sanity checks in the data (is the noise of the expected magnitude?)
  - Allows for time-variations in the responsivity to be removed
  - Puts the output from different detectors into comparable units so they can be binned together in later steps
- Pre-calibration can proceed in various ways
  - TES bolometers are easily calibrated into pW units based on their bias configuration
  - Astrophysical sources (eg, planets) of known (RJ) temperature can be measured through the data-taking season
  - Many experiments have stable internal reference signals (noise pulses, or internal blackbodies -- often a bolometer run in reverse!)

PIPELINE FLOW

- Data Collection
- Data Selection (cutting bad data)
- Data Filtering
- Data Calibration (relative calibration)
- Noise Estimation
- Map Estimation
- Foreground Corrections
- Power Spectrum Estimation
- Cosmological Parameter Estimation

Noise Estimation

- At this point, the pipeline has produced the final timestream vector d_t
- Must estimate the intrinsic detector noise properties very well to distinguish noise from the CMB signal!
- We will model the data as

$$d_t = P_{tp}\Delta_p + n_t,$$

where Δ_p is the sky signal in pixel p (which we want to estimate next!), P_{tp} is the (sparse) pointing matrix, and n_t is the TOD (timestream) noise we wish to estimate

- In particular, we want to find the matrix

$$N_{tt'} = \langle n_t n_{t'} \rangle.$$

- For a single detector, this is the noise correlation function. The diagonal is the variance of the data as a function of time. A slow detector time constant, eg, correlates nearby time samples, leading to offdiagonal terms confined to a band near the diagonal.

Noise Estimation: Finding $N_{tt'}=\langle n_t n_{t'}\rangle$.

- For low signal to noise experiments, the TOD on short time scales is dominated by n_t, so estimation is straightforward
  - (ie, the variance over a time short compared to the motion of the telescope beam on the sky gives an estimate of n_t, and $\langle n_t n_{t'}\rangle \sim \langle d_t d_{t'}\rangle$ for $t \neq t'$)
- For high signal to noise experiments, an iterative approach works -- a crude estimate of $N_{tt'}$ is used to estimate Δ_p which is then removed from d_t and then $N_{tt'}$ is re-estimated.
- For many methods it is important that the noise be stationary. This can limit the length of the TOD which can be effectively mapped (but then maps can be added together).

Noise Estimation: Finding $N_{tt'}=\langle n_t n_{t'} \rangle$.

- In some cases, the data filtering can be handled directly by adjusting the noise matrix
  - For example, in CAPMAP, the scan-synchronous model included fits to sin(ωt) and cos(ωt) every 21 seconds
  - We generated s_t, a vector of length N_{TOD} (corresponding to a few thousand hours) filled with a unit-amplitude discretely sampled sine wave (at ω) 21 seconds long, followed by zeros.
  - We then set $N_{tt} = N_{tt} + \varepsilon^{-1} \langle s_t s_{t'} \rangle$, with ε>>0, effectively giving those modes zero weight (but leaving the matrix formally invertible). We then repeated for the cosine, and for each 21-second interval.
  - In this example, only elements near the diagonal (ie, within 21 seconds of it) are impacted

PIPELINE FLOW

- Data Collection
- Data Selection (cutting bad data)
- Data Filtering
- Data Calibration (relative calibration)
- Noise Estimation
- Map Estimation
- Foreground Corrections
- Power Spectrum Estimation
- Cosmological Parameter Estimation

Mapmaking I

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

- We want a LINEAR ESTIMATOR s_p for Δ_p
- If noise is stationary and Gaussian, then:

$$P(d|\Delta) \propto \exp(-\chi^2/2)$$

$$\chi^2 \equiv \sum_{tt'} n_t^T (N_{tt'})^{-1} n_{t'}$$

$$= (d-P\Delta)^T (N_{tt'})^{-1} (d-P\Delta)$$

- Maximize the probability by minimizing χ^2 : $\left.\frac{\partial \chi^2}{\partial \Delta_p}\right|_{\Delta_p = s_p}$
- The result is:

$$S = (P^T N^{-1} P)^{-1} P N^{-1} d$$

Mapmaking II

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P N^{-1} d$

- Moreover, the noise covariance matrix of the estimated map S is the first factor above: $C = (P^T N^{-1} P)^{-1}$
- So, we're all done EXCEPT … these matrix dimensions are huge! N_{tt} is $N_{TOD} \times N_{TOD}$.
- For ACT, for example, in 15 minutes at 400 Hz for 1000 detectors, $N_{TOD} \sim 4 \times 10^8$
- In that same time, ACT maps about 8×10^4 pixels (~10 deg^2 with 0.75′ resolution): mapmaking is a HUGE compression of the data!
- Matrix inversion goes like $O(N^3)$ … even for a petaflop machine, one inversion would take 10^{10}s ~ 1000 yrs

Mapmaking III

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P N^{-1} d$

So, how to solve efficiently?

- Clever algorithms specific to the scan and sky coverage, using linear algebra tricks and perhaps extra approximations (eg, WMAP, CAPMAP)
- Quick & dirty approximations calibrated by Monte Carlo simulations (roughly the MASTER method, used by, eg, Boomerang, QUAD)
- Iterative solutions

Mapmaking: MASTER

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P N^{-1} d$

- ONE APPROACH:
  - With σ^2 a constant and I the identity matrix, write:

    $$N_{tt} \approx \sigma^2 I$$

  - Find the 'naïve map' from binning the data into pixels:

    $$(s_p) \approx ({P_{pt}}^T P_{tp})^{-1} {P_{pt'}}^T d_{t'} \qquad (({P_{pt}}^T P_{tp})^{-1} = (n_{obs})^{-1})$$

  - Deal with the consequences later: analyze many realistic simulations with the same noise properties as d_t to correct for the effects of $N_{tt} \approx\neq \sigma^2 I$.
  - See Hivon et al, 2002, ApJ 567, 2 for the original application

Mapmaking: Iterating I

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P\, N^{-1} d$

- ANOTHER APPROACH:
  - Multiply the left and right sides of the equation to obtain:

    $$P^T N^{-1} P S = P\, N^{-1} d$$

  - The right hand side = 'the noise-weighted map' -- if N were diagonal, then $N^{-1} d_t$ = the datum times its weight at t…
  - PS produces a synthetic timestream (for an iterative estimate of S) -- so the left side is the noise-weighted synthetic map
  - This equation has the form Ax=b, with x the unknown!
  - Use the firepower of iterative linear algebra solutions

Mapmaking: Iterating II

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map s_p estimate: $P^T N^{-1} P S = P N^{-1} d$

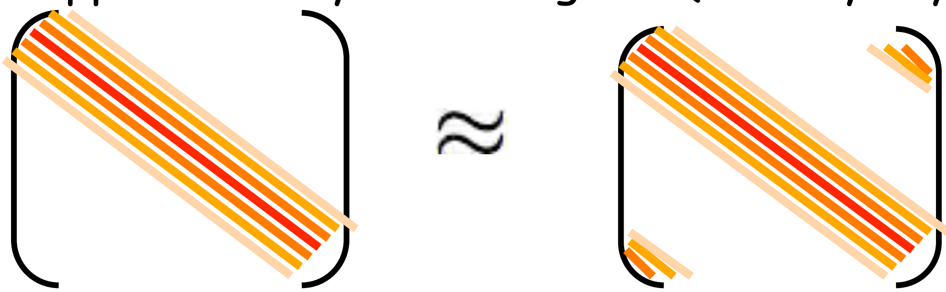
- Let the matrix $A = P^T N^{-1} P$,
- Let the vector $x = s$ and the vector $b = P N^{-1} d_t$
- Then solve Ax=b. Did this recasting solve anything?
  - We still have to deal with N^{-1} but now we will use iterative linear algebra techinques to avoid also having to find A^{-1}.
  - Luckily if the noise is STATIONARY and NOT CORRELATED AT LONG TIME SCALES we can apply a great trick to calculate, eg,

$$w = N^{-1} d$$

  - Upcoming trick from E. L. Wright 1996, astro-ph/9612006.

Mapmaking: Iterating III

- Solving $w = N^{-1} d$ (via $Nw=d$):
  - If the noise is not correlated at long time scales, then N (not its inverse) is approximately band diagonal (& always symmetric):



  - And, we can enforce that it be CIRCULANT as shown on the right with only a small effect on the ends of the timestream. (CIRCULANT means each row is the previous row right-shifted by 1)
  - Let n(t) be the first row of N:

$$Nw=n(t-t')w(t') = w*n\text{: a convolution!}$$

  - So we can go to Fourier space and use the convolution theorem:

$$\mathcal{F}\{w * n\} = \mathcal{F}\{w\} \cdot \mathcal{F}\{n\}$$

Mapmaking: Iterating IV

- Solving $w = N^{-1} d$ (via $Nw=d$):
  - $Nw = w*n$, so $\mathcal{F}\{w*n\} = \mathcal{F}\{w\}\cdot\mathcal{F}\{n\} = \mathcal{F}\{d\}$
  - We can rearrange this: $\mathcal{F}\{w*n\} = \mathcal{F}\{w\}\cdot\mathcal{F}\{n\}$
  - And finally conclude that:

$$N^{-1}d = w = \mathcal{F}^{-1}\{\mathcal{F}\{d\}/\mathcal{F}\{n\}\}$$

  - So, we can calculate A and b without too much pain, and then use a linear algebra technique to solve Ax=b (the mapping equation.)
  - A good method is the Preconditioned Conjugate Gradient method (PCG), described in this context in Dore et al, 2001, A&A 374, 358

Mapmaking: Solving for More

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P N^{-1} d$

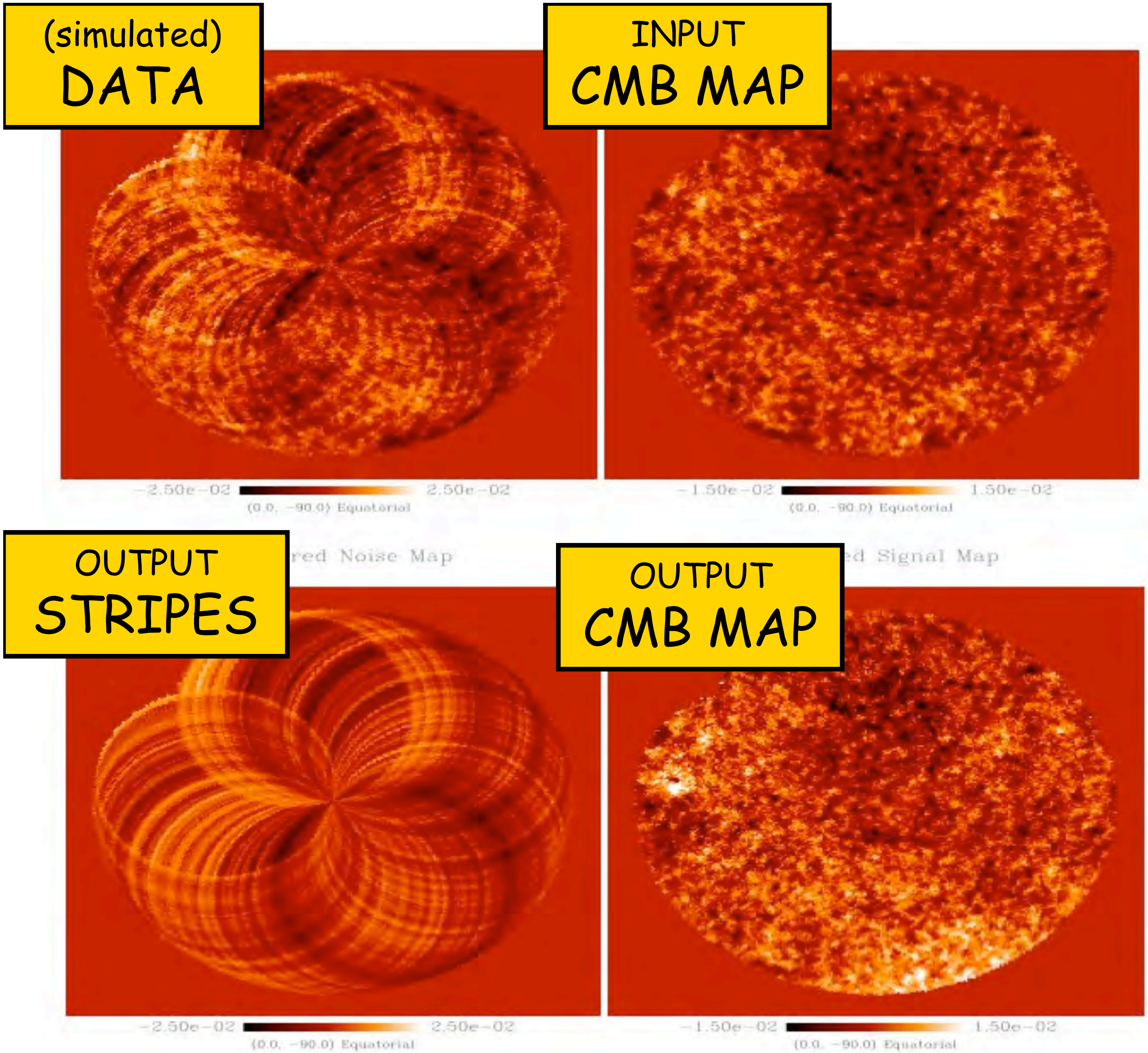
What if you don't know the full form of n_t?

- For example, there might be atmospheric structure in it, or properties of your receiver you did not anticipate

You can iterate!

- For example, solve for $y = S - S_o$ where S_o is your first, possibly naïve estimate of the sky map.
- y is a map of the non-sky garbage in the map! Dore et al 2001 called it the 'stripes' map.

Mapmaking:
Solving for More



A beautfiul example of solving for stripes ($y = S - S_o$) for Planck (in simulations) from Dore et al, 2001.

PIPELINE FLOW

- Data Collection
- Data Selection (cutting bad data)
- Data Filtering
- Data Calibration (relative calibration)
- Noise Estimation
- Map Estimation
- Foreground Corrections
- Power Spectrum Estimation
- Cosmological Parameter Estimation

Power Spectrum Estimation I

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P N^{-1} d$

- (Map S may need to have point sources masked out or other foregrounds removed before this step.)
- ONE APPROACH: Maximum Likelihood
  - Begin with map S and its covariance, $C = (P^T N^{-1} P)^{-1}$
  - THE BIG IDEA: the CMB is not just white noise on all scales; its C_ℓ spectrum also causes correlations in the map. Given the C_ℓ, we can construct $T_{pp'} = \langle \Delta_p \Delta_{p'} \rangle = T_{pp'}\{C_\ell\}$
  - The likelihood of C_ℓ given the map S is:

$$\mathcal{L}(s|\{C_\ell\}) = \frac{\exp[-s^T (T+C)^{-1} s/2]}{\sqrt{(2\pi)^N |T+C|}}$$

Power Spectrum Estimation II

$$\mathcal{L}(s|\{C_\ell\}) = \frac{\exp[-s^T(T+C)^{-1}s/2]}{\sqrt{(2\pi)^N|T+C|}}$$

- To estimate the C_ℓ we maximize the likelihood, but again we have large matrices to invert, so tricks are needed.
- Many exist (eg, signal-to-noise eigenmodes, Bond et al 1998)
- Even so, if you want to estimate 10 or more C_ℓ, it is not feasible to calcculate the likelihood in a 10-dimensional parameter space
- Use Monte Carlo Markov Chains (MCMC) to estimate the likelihood!

Power Spectrum Estimation III

$$d_t = P_{tp}\Delta_p + n_t$$

where d_t = TOD, Δ_p = sky in pixel p, P_{tp} = pointing matrix, & n_t = TOD noise

map estimate: $S = (P^T N^{-1} P)^{-1} P N^{-1} d$

- ANOTHER APPROACH TO ESTIMATE C_ℓ:
  - Take the map S and Fourier transform it (or spherical harmonic transform it)
  - Apodize the map if you need to (multiply by a nice window function so it does not ring)
  - Figure out what you have!
    - Analytically :Psuedo- C_ℓ method, see Wandelt et al 2001
    - By comparing to simulations where you did the same thing (MASTER)
    - REMASTER: correct your crude power spectrum by the factors needed
    - There are many refinements (eg, the TAPER method, Das et al, 2008, arXiv:0809.1092)

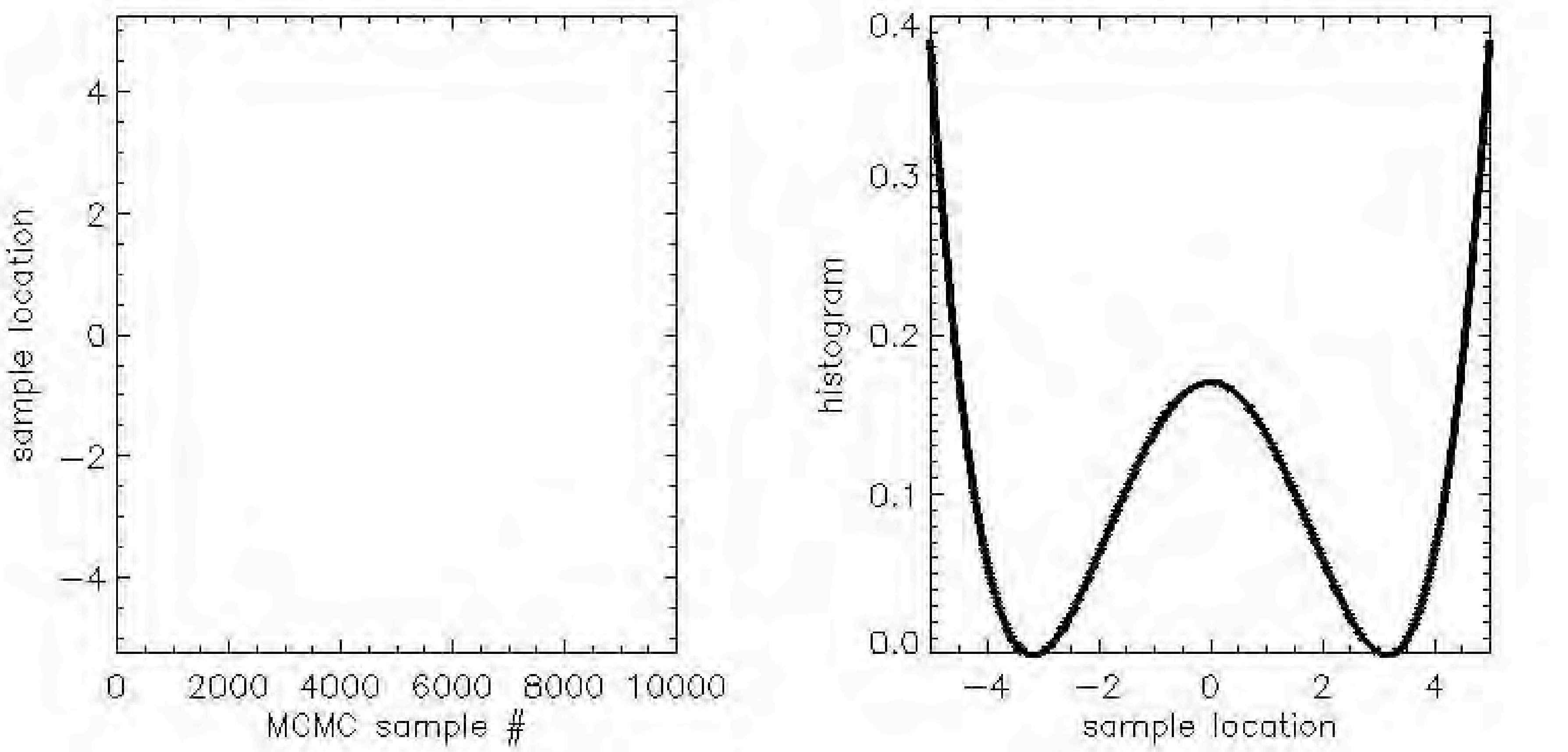
Parameter Estimation & MCMC

- Use maximum likelihood to estimate cosmological parameters given the estimated C_ℓ:
- The large dimensionality of the problem again leads to the need for MCMC
- IDEA: generate many SAMPLES of the multi-dimensional likelihood L for random vectors of the parameters {x}.
- Generate these samples so they end up distributed like the likelihood L itself!
- Markov chain M(x): the value of $M(x_k)$ only depends on $M(x_{k-1})$
- One simple algorithm (Metropolis-Hastings):
  - Begin with a trial distribution (eg, n-dimensional Gaussian) from which you will draw vectors Δx
  - Start with some $\{x_o\}$. At the kth iteration:
    - Generate $x^* = x_k + \Delta x$
    - Calculate $r = L(x^*)/L(x_k)$
    - If $r > 1$, $x_{k+1} = x^*$; else $x_{k+1} = x_k$

MCMC: Eiichiro style

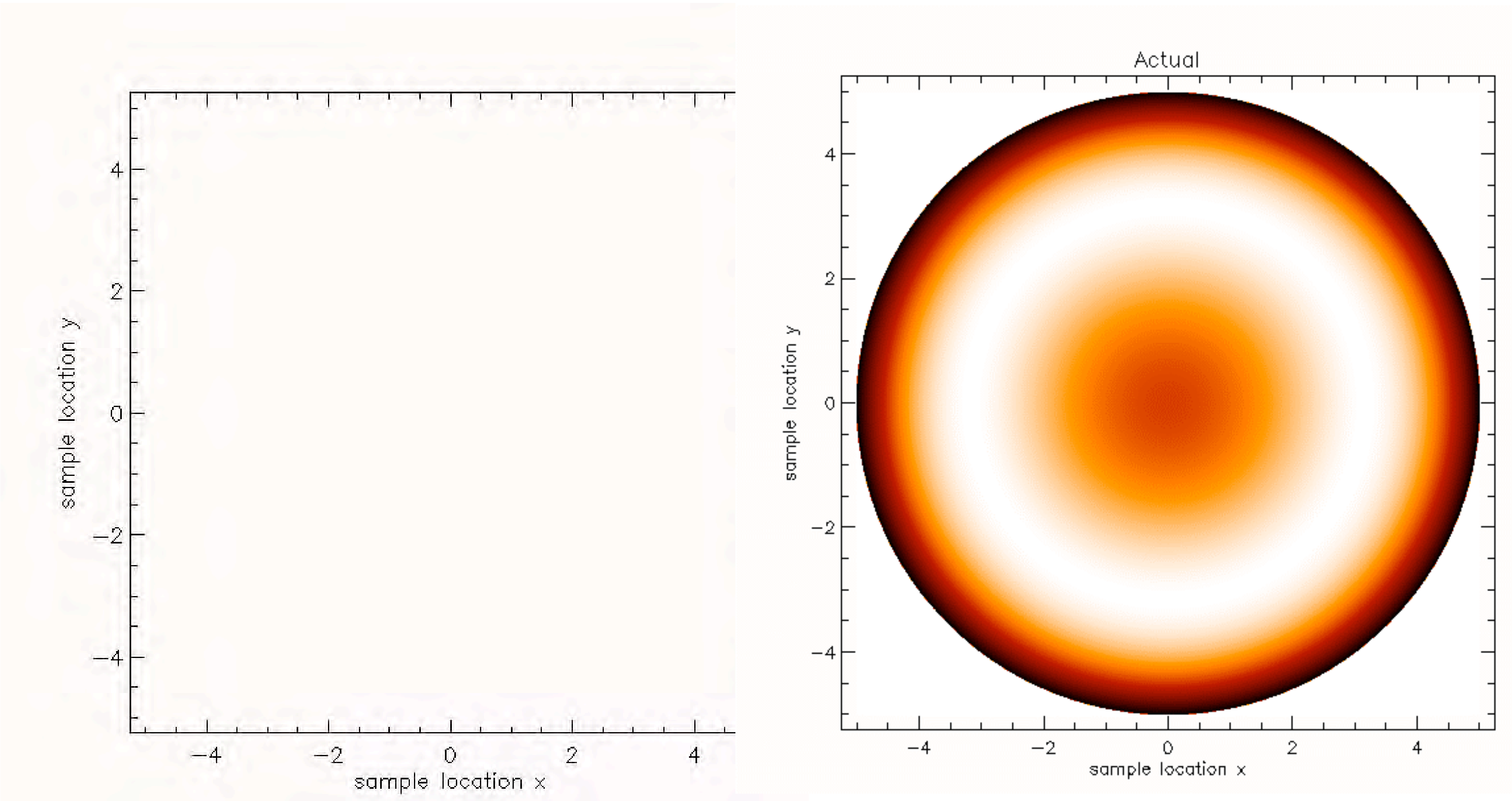
- You stand at some point on the likelihood and you send out feelers in a volume around you. If you find a higher likelihood spot, you go to it.
- If you are near a minimum in the distribution, of course, you will be likely to find such a spot rather quickly; conversely, near a maximum, you will tend to linger.
- Meanwhile, all those attempts are being counted -- so you end up with a lot more samples at the tops of the hills!

MCMC from flat trial



Movie from Lewis Hyatt

MCMC from 2d Gaussian Trial function



Movie from Lewis Hyatt

END

ありがとう。