An optimal Bayesian solution to the CMB delensing problem

Marius Millea

with

Ethan Anderes  Ben Wandelt

Nordita July 21, 2017
How do we optimally delense future CMB data to obtain the best possible estimates of $r$?
CMB “fields”

\[ f \equiv (T, Q, U) \]

Lensing potential

Cosmo params

Data

\[ P(f, \phi, r \mid d) \]
CMB “fields” \( f \equiv (T, Q, U) \)

Lensing potential \( \mathcal{P}(f, \phi, r | d) \)

Cosmo params

Data
All current analyses are based on this
Currently near-optimal but will be sub-optimal for next-gen noise levels
Carron & Lewis (2017), Hirata & Seljak (2003) give algorithm to maximize this lensing potential of CMB “fields” $f \equiv (T, Q, U)$.

All current analyses are based on this. Currently near-optimal but will be sub-optimal for next-gen noise levels.

$$\hat{\phi}(L) = \int dl_1 W(l_1, l_2) d(l_1)^* d(l_2)$$

$$P(f, \phi, r | d)$$

$$P(\phi | r, d) = \int df P(f, \phi | r, d)$$

Carron & Lewis (2017), Hirata & Seljak (2003) give algorithm to maximize this...
Why is sampling/minimizing $P(f, \phi \mid d)$ such a hard problem?
Why is sampling/minimizing $\mathcal{P}(f, \phi | d)$ such a hard problem?
Why is sampling/minimizing $\mathcal{P}(f, \phi | d)$ such a hard problem?
Why is sampling/minimizing $\mathcal{P}(f, \phi \mid d)$ such a hard problem?

So, as pointed out by Anderes et al. 2015, it's very beneficial to reparametrize,

$$\mathcal{P}(\tilde{f}, \phi \mid d) = \mathcal{P}(f(\tilde{f}), \phi \mid d) \left| \frac{df}{d\tilde{f}} \right|$$

where $\tilde{f} = \mathcal{L}(\phi)f \implies \left| \frac{df}{d\tilde{f}} \right| = 1/|\mathcal{L}(\phi)|$
What is the determinant of lensing?
What is the determinant of lensing?

- Infinite resolution: lensing is a remapping (i.e. permutation) so $\det |\mathcal{L}(\phi)| = 1$
What is the determinant of lensing?

- Infinite resolution: lensing is a remapping (i.e. permutation) so \( \det |\mathcal{L}(\phi)| = 1 \)
- This is not the case when we have *pixelization*. Consider the Taylor series approx:
What is the determinant of lensing?

- Infinite resolution: lensing is a remapping (i.e. permutation) so \( \det |\mathcal{L}(\phi)| = 1 \)
- This is not the case when we have pixelization. Consider the Taylor series approx:

\[
\tilde{f}(x) = f(x + \nabla \phi(x)) \approx [1 + \nabla \phi(x) \cdot \nabla + \ldots] f(x)
\]

\[\mathcal{L}(\phi)\]
What is the determinant of lensing?

- Infinite resolution: lensing is a remapping (i.e. permutation) so \( \text{det } |\mathcal{L}(\phi)| = 1 \)
- This is not the case when we have \textit{pixelization}. Consider the Taylor series approx:

\[
\tilde{f}(x) = f(x + \nabla \phi(x)) \approx \left[1 + \nabla \phi(x) \cdot \nabla + \ldots \right] f(x)
\]

\[
L(\phi)
\]

Matrix representation of \( L(\phi) \) for 16x16 1’ pixel TEB maps for 7\textsuperscript{th} order Taylor series approximation

\[
\log(\text{abs}(L(\phi)_{ij}))
\]
What is the determinant of lensing?

- Infinite resolution: lensing is a remapping (i.e. permutation) so $\det |\mathcal{L}(\phi)| = 1$
- This is not the case when we have *pixelization*. Consider the Taylor series approx:

$$\tilde{f}(x) = f(x + \nabla \phi(x)) \approx [1 + \nabla \phi(x) \cdot \nabla + \ldots] f(x)$$

Matrix representation of $\mathcal{L}(\phi)$ for 16x16 1’ pixel TEB maps for 7th order Taylor series approximation

$$\log(\text{abs}(\mathcal{L}(\phi)_{ij}))$$

not close to 1!

$$\det |\mathcal{L}(\phi)| = 1.9 \times 10^{-9}$$
What is the determinant of lensing?

- Infinite resolution: lensing is a remapping (i.e. permutation) so $\det |\mathcal{L}(\phi)| = 1$
- This is not the case when we have pixelization. Consider the Taylor series approx:

$$\tilde{f}(x) = f(x + \nabla \phi(x)) \approx \left[ 1 + \nabla \phi(x) \cdot \nabla + \ldots \right] f(x)$$

Matrix representation of $\mathcal{L}(\phi)$ for 16x16 1’ pixel TEB maps for 7th order Taylor series approximation

$$\log(\text{abs}(\mathcal{L}(\phi)_{ij}))$$

$\det |\mathcal{L}(\phi)| = 1.9 \times 10^{-9}$

Additionally, the variation of the determinant with $\phi$ is significant.
A solution: LenseFlow
A solution: \textbf{LenseFlow}

Define \( f_t(x) \equiv f(x + t\nabla \phi(x)) \) \hspace{1cm} \text{s.t.} \hspace{1cm} \begin{align*}
    f_{t=0}(x) &= f(x) \\
    f_{t=1}(x) &= \tilde{f}(x)
\end{align*}
A solution: LenseFlow

Define \( f_t(x) \equiv f(x + t\nabla \phi(x)) \) \quad \text{s.t.} \quad f_{t=0}(x) = f(x) \quad f_{t=1}(x) = \tilde{f}(x)

One can show \( f_t \) obeys an ODE “flow” equation

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot [1 + t\nabla \nabla \phi(x)]^{-1} \cdot \nabla f_t(x)
\]
A solution: **LenseFlow**

Define \( f_t(x) \equiv f(x + t\nabla \phi(x)) \) s.t. \[
\begin{align*}
  f_{t=0}(x) &= f(x) \\
  f_{t=1}(x) &= \tilde{f}(x)
\end{align*}
\]

One can show \( f_t \) obeys an ODE “flow” equation

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot \left[ 1 + t\nabla \nabla \phi(x) \right]^{-1} \cdot \nabla f_t(x)
\]

- To *lense* a map, just run the ODE from \( t=0 \) to \( t=1 \)
A solution: 

**LenseFlow**

Define \( f_t(x) \equiv f(x + t \nabla \phi(x)) \)  

subject to \( f_{t=0}(x) = f(x) \)  
\( f_{t=1}(x) = \tilde{f}(x) \)

One can show \( f_t \) obeys an ODE “flow” equation

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot [1 + t \nabla \nabla \phi(x)]^{-1} \cdot \nabla f_t(x)
\]

- To *lense* a map, just run the ODE from \( t=0 \) to \( t=1 \)
- To *delense* a map, just run it backwards from \( t=1 \) to \( t=0 \)
A solution: \textbf{LenseFlow}

Define \( f_t(x) \equiv f(x + t \nabla \phi(x)) \) s.t. \[
\begin{align*}
  f_{t=0}(x) &= f(x) \\
  f_{t=1}(x) &= \tilde{f}(x)
\end{align*}
\]

One can show \( f_t \) obeys an ODE “flow” equation

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot [1 + t \nabla \nabla \phi(x)]^{-1} \cdot \nabla f_t(x)
\]

- To \textit{lense} a map, just run the ODE from \( t=0 \) to \( t=1 \)
- To \textit{delense} a map, just run it backwards from \( t=1 \) to \( t=0 \)
- This operation provably has determinant = 1
A solution: **LenseFlow**

Define \( f_t(x) \equiv f(x + t \nabla \phi(x)) \) s.t.

\[
\begin{align*}
    f_{t=0}(x) &= f(x) \\
    f_{t=1}(x) &= \tilde{f}(x)
\end{align*}
\]

One can show \( f_t \) obeys an ODE “flow” equation

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot \left[ 1 + t \nabla \nabla \phi(x) \right]^{-1} \cdot \nabla f_t(x)
\]

- To *lense* a map, just run the ODE from \( t=0 \) to \( t=1 \)
- To *delense* a map, just run it backwards from \( t=1 \) to \( t=0 \)
- This operation provably has determinant = 1

(In practice we use 4\(^{th}\) order Runge-Kutta with 7 time-steps.)
LenseFlow vs. Taylor series

Differences between the two which lead to different determinants
Ok, let’s maximize & sample!

The algorithm we devise is a *coordinate descent*
Ok, let’s maximize & sample!

The algorithm we devise is a coordinate descent

\[ -2 \ln \mathcal{P}(\tilde{f}, \phi \mid d) = \]

\[ = (d - \tilde{f})^\dagger C_n^{-1} (d - \tilde{f}) + \tilde{f}^\dagger \mathcal{L}(\phi)^{-\dagger} C_f^{-1} \mathcal{L}(\phi)^{-1} \tilde{f} + \phi^\dagger C_\phi^{-1} \phi \]

- likelihood
- prior on \( f \)
- prior on \( \phi \)
Ok, let’s maximize & sample!

The algorithm we devise is a coordinate descent

\[-2 \ln P(\tilde{f}, \phi \mid d) = \]

\[\tilde{f} \text{ step : a Wiener filter} \]

\[= (d - \tilde{f})^\dagger C_n^{-1} (d - \tilde{f}) + \tilde{f}^\dagger \mathcal{L}(\phi)^{-\dagger} C_f^{-1} \mathcal{L}(\phi)^{-1} \tilde{f} + \phi^\dagger C_\phi^{-1} \phi\]

\[\begin{align*}
\text{likelihood} \\
\text{prior on } f \\
\text{prior on } \phi
\end{align*}\]
Ok, let’s maximize & sample!

The algorithm we devise is a *coordinate descent*

\[ -2 \ln P(\tilde{f}, \phi \mid d) = \]

\[ = (d - \tilde{f})^\dagger C_n^{-1} (d - \tilde{f}) + \tilde{f}^\dagger L(\phi)^{-\dagger} C_f^{-1} L(\phi)^{-1} \tilde{f} + \phi^\dagger C_\phi^{-1} \phi \]

\(\tilde{f}\) step: a Wiener filter

\(\phi\) step

\(\phi\) step

likelihood

prior on \(f\)

prior on \(\phi\)
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, $r=0.05$
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, $r = 0.05$
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, r=0.05
Starting point: \( \phi = 0 \)

Simulated data with: 1uK-arcmin noise, \( r=0.05 \)
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, $r=0.05$
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, $r=0.05$
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, $r=0.05$
Starting point: $\phi = 0$

Simulated data with: 1uK-arcmin noise, $r=0.05$

Step 1

Step 3

Step 30
30min on 1 single multi-core CPU for these 2500deg$^2$
1024x1024, 3 arcmin pixels
Masking works too

(Only affects the Wiener filter step which needs more conjugate gradient steps => 4 hours)
Masking works too

(Only affects the Wiener filter step which needs more conjugate gradient steps => 4 hours)
Masking works too

(Only affects the Wiener filter step which needs more conjugate gradient steps => 4 hours)
Masking works too

(Only affects the Wiener filter step which needs more conjugate gradient steps => 4 hours)
What about $r$?

For now, a slightly simplified preview: $\mathcal{P}(f, \phi, r | d)$

Samples of:
Conclusions

• We can maximize $\mathcal{P}(f, \phi, r \mid d)$
• Sampling is coming up and I’ve given you a preview of it
• Looking forward to more improvement, application to data, and feedback from the community (see our paper soon!)
LenseFlow determinant
LenseFlow determinant

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot [\mathbb{1} + t \nabla \nabla \phi(x)]^{-1} \cdot \nabla f(x) \tag{pt}
\]
LenseFlow determinant

\[ \frac{df_t(x)}{dt} = \nabla \phi(x) \cdot [1 + t \nabla \nabla \phi(x)]^{-1} \cdot \nabla f(x) \]

\[ \mathcal{L}(\phi) = [1 + \varepsilon p_{t_n} \cdot \nabla] \cdots [1 + \varepsilon p_{t_0} \cdot \nabla] \]
LenseFlow determinant

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot [1 + t \nabla \nabla \phi(x)]^{-1} \cdot \nabla f(x)
\]

\[
\mathcal{L}(\phi) = [1 + \varepsilon p_{t_n} \cdot \nabla] \cdots [1 + \varepsilon p_{t_0} \cdot \nabla]
\]

\[
\log \det [1 + \varepsilon p_t \cdot \nabla] = \varepsilon \text{Tr} [p_t \cdot \nabla] + \mathcal{O}(\varepsilon^2)
\]
LenseFlow determinant

\[
\frac{df_t(x)}{dt} = \nabla \phi(x) \cdot \left[ 1 + t \nabla \nabla \phi(x) \right]^{-1} \cdot \nabla f(x)
\]

\[
\mathcal{L}(\phi) = [1 + \varepsilon p_{t_n} \cdot \nabla] \cdots [1 + \varepsilon p_{t_0} \cdot \nabla]
\]

\[
\log \det [1 + \varepsilon p_t \cdot \nabla] = \varepsilon \text{Tr} [p_t \cdot \nabla] + \mathcal{O}(\varepsilon^2)
\]

So for LenseFlow \( \det |\mathcal{L}(\phi)| = 1 \) so we can ignore it!
LenseFlow

Taylor series

Differences between two which lead to different determinants