Abstract

Sometimes a mapmaker is worth a whole paper.
Keywords: DTF matrix

Cosmoglobe DR3. II. Mapmaking with interferograms

D. J. Watts[*] A. I. S. Martins

October 10, 2025

1 Introduction

The basic idea is as follows. If you are looking at a smooth emitter with spectrum $S(\nu)$, an interferogram essentially lets you take the Fourier transform.

$$I(d) = \int_0^\infty S(\nu)\cos(2\pi d\nu)\,\mathrm{d}\nu$$

where d is the moving mirror position.[1] This can be recast as a linear operator, F (called so because it is the real part of a Fourier transform), and allows us to write

$$I_d = F_{d\nu}S_\nu.$$

Note that we've implicitly discretized things here, which is of course an approximation.

When FIRAS observed, it would collect data as it was pointing at a position on the sky; let's call this p, which could stand for *p*ixel or *p*osition. What is the spectrum at the point on the sky? Well, if we have a sky model, we can write this as

$$S_\nu(p) = \sum_c a_{c,p} f_c(\nu \mid \beta_p)$$

where c indexes all the components we care about; mostly thermal dust, the CMB, and the CIB, and β_p is the generalized (potentially) spatially-dependent spectral parameters, $f_c(\nu \mid \beta_p)$ the frequency dependence, and $a_{c,p}$ a component map on the sky. This also can be cast as a series of matrix equations. One way of looking at it is this;

$$S_{\nu p} = M_{\nu p,c}a_{c,p}$$

where M takes us from a component amplitude space to a spectrum space.

Where we could get tripped up now is that this spectrum itself has a position dependence; this is where we would the pointing matrix P, where

$$S_{\nu t} = P_{tp}S_{\nu p}.$$

We are now doing the physicist thing where different objects have the same variable names. We must fix this for clarity's sake later.

[*]Corresponding author: D. Watts; duncan.watts@astro.uio.no

[1]We are assuming here that the internal calibrator on FIRAS has been taken care of already.

Now the fun thing is that FIRAS is moving while the interferogram is being taken. Let's pretend that we don't even want to know what the components are, we just want maps; now $S_{\nu,p}$ is the object we want to recover.

$$I_d = F_{dt} P_{tp} S_{p\nu}$$

In words, d is the distance of the mirror, t is the time that we observe, p is the position the telescope is looking at.

Because there is always noise, let us add noise;

$$I_d = F_{dt} P_{tp} S_{p\nu} + n_t$$

There are two different ways to go here. One is, for each I_d, we perform the inverse Fourier transform, then do some mapmaking.

$$I_d \to F^{-1} I_d \to \hat{S}_{p\nu} = \frac{1}{N_\mathrm{obs}} \sum F^{-1} I_d$$

There's a bit more to it, but not much. There is of course noise weighting that can be used as well.

Another way to look at this is through the standard mapmaking equation with a twist;

$$\mathsf{I} = \mathsf{FPS} + \boldsymbol{n}$$

Here, given F, P, I, and $\mathsf{N} = \langle \boldsymbol{n}^T \boldsymbol{n} \rangle$, there is a maximum likelihood solution;

$$[(\mathsf{FP})^T \mathsf{N}^{-1} \mathsf{FP}] \hat{\mathsf{S}} = (\mathsf{FP})^T \mathsf{N}^{-1} \mathsf{I}$$

Incidentally, S looks a bit scary at first because it is a matrix. In practice, we can reshape it so that instead of a $N_p \times N_\nu$ matrix, we can stack columns to obtain one $N_p N_\nu$-length vector.

In principle, this is it. There are other advantages here as well; some things, like the apodization function and the optical transfer function like to push the signal to zero. Can we avoid this here?

This should roughly reduce to the usual approach, if we pretend that P is invertible;

$$\begin{aligned} \hat{\mathsf{S}} &= [(\mathsf{FP})^T \mathsf{N}^{-1} \mathsf{FP}]^{-1} (\mathsf{FP})^T \mathsf{N}^{-1} \mathsf{I} \\ &\approx (\mathsf{FP})^{-1} \mathsf{I} \end{aligned}$$

which completely agrees with the idea from before.

Where this idea can really help is with the averaging; 4–12 IFG's are averaged on board, so the true data model is a bit more complex;

$$I_d = \sum_i (F_{dt_i} P_{t_i p} S_{p\nu} + n_{t_i})^{2}$$

This can create some blurring in the scanning direction, and potentially splitting in regions with high gradients. It is not clear to me yet how this was avoided in the original FIRAS pipeline.

[2]Or averaged; not important right now

1.1 Motivation

Using a simple binned mapmaker for a traditional CMB experiment is not optimal. However, due to the Fourier nature of an FTS, using a simple binned mapmaker actually biases the results. We show this in Figure 1, where we can see that the low signal-to-noise regions are not well recovered. Leaving even simple white noise untreated will lead to a noise floor when Fourier transformed.

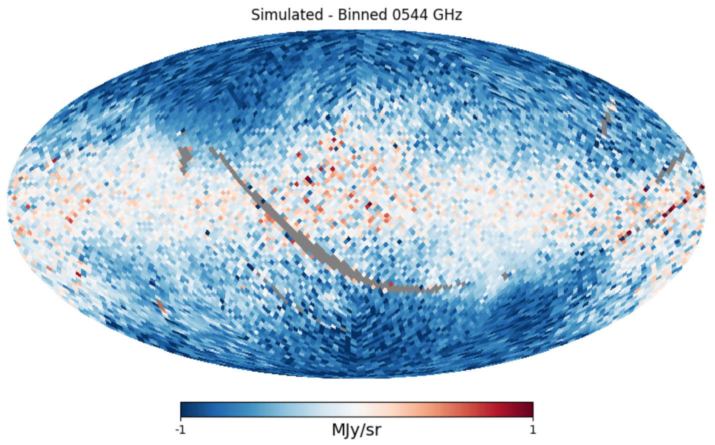


Figure 1: Difference between the simulated data (without noise) and the output of the binned mapmaker. The binned mapmaker is not able to recover the full signal, which leads to a bias in the output.

2 Regarding the Fourier transformations

Everything useful about the Fourier transform, the DTF and specifically the real DTF: here.

2.1 Matrix Inversion Method

So we want to start with the simplest case, where we are able to start from

$$d = Pm + n, \tag{1}$$

and through the maximum likelihood method get to

$$P^T N^{-1} P\hat{m} = p^T N^{-1} d \tag{2}$$

and simply invert the left hand side to get

$$\hat{m} = (P^T N^{-1} P)^{-1} P^T N^{-1} d. \tag{3}$$

Now, it's important to define P, which, for starters, will simply be the DFT matrix, which we will call W. Also, let's go even simpler and say that simply

$$m = P^{-1} d. \tag{4}$$

Now, what is P^{-1}? This guy explains it pretty well.

A Fourier transform is defined as

$$\tilde{x}[k] = \sum_{n=0}^{N-1} e^{-j\frac{2\pi kn}{N}} x[n], \tag{5}$$

which shows that the DFT matrix operator can be writen as

$$W_{kn} = e^{-j\frac{2\pi kn}{N}}, \tag{6}$$

where we W_{kn} is the kth element of the nth row of W. j is of course the imaginary unit (DSP engineers... it was always gonna have to come to them).

The inverse Fourier transform is defined as

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[k] e^{+j\frac{2\pi kn}{N}}, \tag{7}$$

which then similarly defines the inverse DFT matrix as

$$W^{-1}_{nk} = \frac{1}{N} e^{+j\frac{2\pi kn}{N}}, \tag{8}$$

which we can relate to the DFT matrix via matrix operations:

$$\begin{align} W^{-1}_{nk} &= \frac{1}{N}(e^{-j\frac{2\pi kn}{N}})^* \tag{9} \\ &= \frac{1}{N}(W)^*_{nk} \tag{10} \\ &= \frac{1}{N}(W^{*^T})_{kn}, \tag{11} \end{align}$$

which is just the hermitian conjugate, perfect!

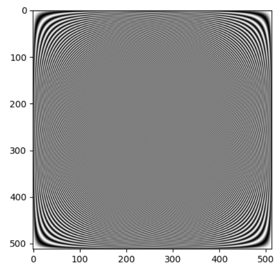
2.2 The curse of using Fourier transforms on real-valued input

This would be perfect and very easy if we were using complex-valued input, where the full N points of the transform are relevant. However, our signal is real-valued, and carrying around a copy of the transform is not very useful and only gives way to more calculations. One would think that we could build a real DFT matrix (i.e. a matrix operator that is equivalent to the real Fourier transform). As such, we need to build a matrix that is $N/2 + 1$ by N, which, of course, is not square, and therefore we lose the nice unitary (up to a normalization factor) properties of the DFT matrix. Still, we can try to work it out on the same principle that the inverse operator is the same as the transpose operator (up to a normalization factor) and see what happens. This is probably the questionable assumption that I'm making.

So, this is how we define the real DFT matrix:

$$\begin{align} \omega &= e^{-2\pi i/N} \tag{12} \\ W^r_{v_i x_i} &= \omega^{v_i x_i}, \tag{13} \end{align}$$

and this matrix works on the real-valued input as expected, the same as using the numpy.fft.rfft function.


(a) Square DFT.

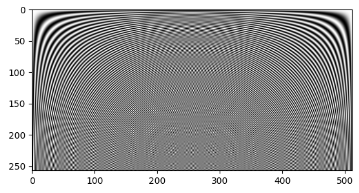

(b) Non-square DFT.

Figure 2: Visual representations of the forward matrix operators.

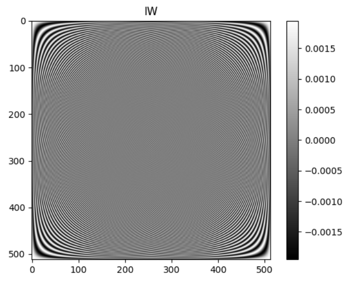
We can also visualise what the operators look like in Figure 2.

Now we need the inverse operator, which then needs to be N by $N/2+1$. If the assumption that the transpose matrix is the inverse operator holds, then building the inverse operator from the definition of the inverse Fourier transform should work. The closest I have been able to come up with is the following definition:

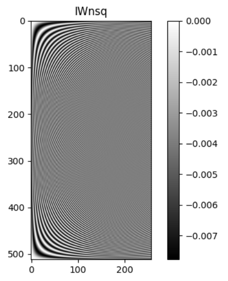
$$\omega = e^{2\pi i/N} \tag{14}$$

$$W^{r^{-1}}_{x_i \nu_i} = \frac{1}{N/2+1}(\omega^{\nu_i x_i} - 1). \tag{15}$$

The matrix operators are shown in Figure 3.



(a) Square inverse DFT.

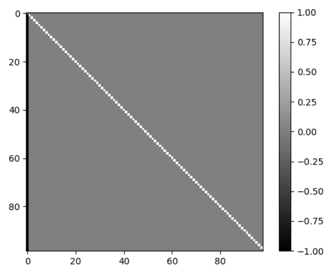


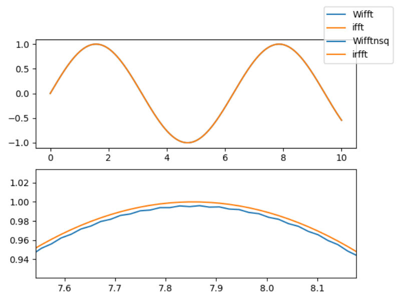
(b) Non-square inverse DFT.

Figure 3: Visual representations of the inverse matrix operators.

Nothing seems obviously wrong with the non-square version of the operator. However, Figure 4 shows what the matrix multiplication of the forward and inverse operators ($W^{r^{-1}}W^r$) looks like (which we expect to be the identity matrix) and the close-up of the difference between the original real-valued input and the output after putting it through our built non-square DFT and inverse DFT operators.

I also tried simply taking the hermitian conjugate of the DFT matrix instead of building the inverse DFT matrix "from scratch" and that seems to wield the same results.


(a) Multiplication of the forward and inverse operators.



(b) Close-up of the difference between the original and the output.

Figure 4: Visual representations of the problems with the non-square inverse DFT.

3 Mapmaking for FTS

When looking at each individual interferogram (and therefore only looking at one pixel), there is a fairly simple matrix formulation of the maximum likelihood mapmaking equation, which which relates the spatial distance in the instrument x with sky frequency ν and reads

$$(F^T_{\nu;x} N^{-1}_{x;x} F_{x;\nu}) m_\nu = F^T_{\nu;x} N^{-1}_{x;x} d_x, \tag{16}$$

where ; in the index separates the two dimensions of the matrix.

Now the standard mapmaking equation relates time t with position in the sky p and reads

$$(P^T_{p;t} N^{-1}_{t;t} P_{t;p}) m_p = P^T_{p;t} N^{-1}_{t;t} d_t. \tag{17}$$

Now in order to be able to treat pixel correlations, we need to put the two together, which I think will look like this:

$$(P^T_{\nu p;\nu} F^T_{\nu;xt} N^{-1}_{xt;xt} F_{xt;\nu} P_{\nu;\nu p}) m_{\nu p} = P^T_{\nu p;\nu} F^T_{\nu;xt} N^{-1}_{xt;xt} d_{xt}. \tag{18}$$

3.1 Revised data model and corresponding ML mapmaking equation

After speaking to Sigurd, he came up (somewhat independently) with the following data model:

$$d_i = P_i F T_i m + n, \tag{19}$$

where i for the FIRAS case would correspond to temperature bins. And the corresponding ML mapmaking equation would then be:

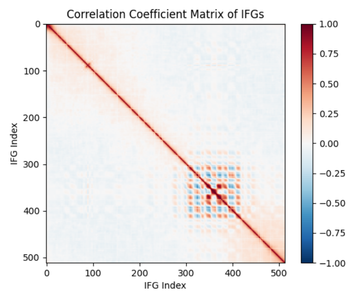
$$\hat{m} = \left(\sum_i T_i^T F^T P_i^T N^{-1} P_i F T \right)^{-1} \sum_i T_i^T F^T P^T N^{-1} d_i, \tag{20}$$

where F here should be cosine transforms.

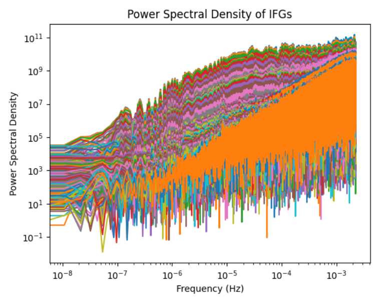
We spoke about it after and mentioned that it actually might make more sense to add the ICAL (and others) back in when the data is in IFG space, which then even can ignore these temperature bins. This would probably mean that we need to do all other low-level IFG processing before entering this ”pipeline” too, like removing the transient reponse from the onboard digital filters, etc.

3.2 Mapmaking without taking into account frequency correlations

For the sake of starting with something simple, we will assume that we have white noise and no frequency correlations (which is not true, the current noise estimate (without taking into account frequency correlations) for FIRAS is shown in Figure 5, which seems to be white + blue noise).



(a) Noise covariance matrix not taking into account frequency correlations (size is 512 x 512). In order to subtract the signal to estimate this, I simply subtracted one IFG from the next and so on.



(b) Noise PSD not taking into account frequency correlations (Fourier transform was made along NIFG). In order to subtract the signal to estimate this, I simply subtracted one IFG from the next and so on.

Figure 5: Noise estimations without taking into account frequency correlations.

I now believe that there is no difference in where we place the (let’s call it) space distance Fourier transform F_x in relation to the pointing matrix in our data model, if we are considering a stop and stare approach (which we are for now), because it is the same to first bin and then take the Fourier transform or vice-versa. Taking this into account, the most efficient way to program is taking the Fourier transform of the following after binning:

$$\hat{m}_p = \frac{\sum_{i\in p} \frac{d_i}{\sigma_i^2}}{\sum_{i\in p} \frac{1}{\sigma_i^2}}. \tag{21}$$

For now we are also assuming an instrument that is not differential, i.e. there is no sutraction of an internal calibration because, even though FIRAS is differential, a general FTS does not need to be. In order to take this into account, the two options are to either consider a completely different equation and Fourier transform as the first step and then subtract in sky frequency space or to leave it as is and subtract in interferogram space.

3.3 Dealing with beam movement while taking interferograms

In FIRAS, and likely for any FTS, the instrument is still moving while taking interferograms (i.e. it does not stop and stare at a single pixel while it is taking one interferogram). In particular, FIRAS averages a number of interferograms onboard before sending them down. For example, for a short scan, the instrument takes one full interferogram, the MTM flys back, another interferogram is taken, the MTM flys back again, and so on, until 16 full interferograms are taken, with a full cycle, for example while taking data in the slow mode (0.8 cm/s velocity of the MTM), taking 55.36 seconds in total. As such, 16 onboard interferograms correspond to 1 interferogram in our data ("telemetered" interferograms), when taking data in the short mode. Timing info is given in https://lambda.gsfc.nasa.gov/data/cobe/firas/ancil/FIRAS_FEX_MTMSWEEP.TXT.

As such, we want to understand, for each telemetered interferogram and their latitude and longitude coordinates, how many and which pixels they correspond to, and how much data is taken corresponding to that pixel. For example, if one telemetered interferogram was taken equally over two pixels, the part of the pointing matrix corresponding to this goes from

$$P = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{22}$$

to

$$P = \begin{pmatrix} 0.5 & 0.5 \\ 0 & 0 \end{pmatrix}. \tag{23}$$

We also know that the pointing solution for each interferogram is given in the middle of taking each telemetered interferogram. (Section 4.2 of the explanatory supplement) The FIRAS line of sight traces a path along a line of approximately constant ecliptic longitude. The beam is of 7 degrees. The beam moves at 3.5 degrees per minute.

References