On the covariance matrix sampler

Zheng Zhang

May 13, 2025

1 Introduction

Given an n-dimensional Gaussian realisation $\boldsymbol{s}$, the covariance matrix follows the following distribution

$$p\left(\mathbf{S}|\boldsymbol{s}\right) \propto \frac{1}{|\mathbf{S}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{s}-\bar{\boldsymbol{s}})^T\mathbf{S}^{-1}(\boldsymbol{s}-\bar{\boldsymbol{s}})\right) \tag{1}$$

It is generally impossible to sample a rank-n covariance matrix with a single realisation of the n-dimensional vector, since it is statistically underdetermined. However, it is usually the case that there is significant degeneracy in $\mathbf{S}$, and the number of degrees of freedom of $\mathbf{S}$ may be less than n. For these cases, we can sample $\mathbf{S}$ with $\boldsymbol{s}$ using the following strategy:

1. Separate the independent variables.
2. Group the independent and identically distributed variables.
3. Estimate the variance of each distribution using different realisations.
4. Sample the covariance matrix

2 Sampling 21 cm covariance

- Independent variables:

  We define a linear operator U which describes $\boldsymbol{s}$ as linear combinations of the the modes in comoving Fourier space:

  $$\boldsymbol{s} = U\tilde{\boldsymbol{s}}. \tag{2}$$

  The elements of $\tilde{\boldsymbol{s}}$ are the Fourier coefficients for the corresponding comoving Fourier mode. Each coefficient $\tilde{\boldsymbol{s}}_j$ is an independent random variable. For convenience, we refer to the wave vector associated with $\tilde{\boldsymbol{s}}_j$ as $\boldsymbol{k}_j$.

- The covariance matrix of $\tilde{\boldsymbol{s}}$ is denoted by $\tilde{\mathbf{S}}$

  $$\tilde{\mathbf{S}} \equiv \langle \tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T \rangle \tag{3}$$

  which is diagonal and

  $$\tilde{\mathbf{S}}_{jj} = P(\boldsymbol{k}_j) = P(|\boldsymbol{k}_j|). \tag{4}$$

- Independent and identically distributed variables:

  $$\left\{\tilde{\boldsymbol{s}}_j \,\middle|\, |\boldsymbol{k}_j| = k\right\}. \tag{5}$$

  The size of the set is denoted as N_k.

- The distribution of $\tilde{\mathbf{S}}$ is given by

$$\begin{aligned} p\left(\tilde{\mathbf{S}}|\tilde{\boldsymbol{s}}\right) &\propto \frac{1}{|\tilde{\mathbf{S}}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\tilde{\boldsymbol{s}}^T\tilde{\mathbf{S}}^{-1}\tilde{\boldsymbol{s}}\right) \\ &= \prod_k [P(k)]^{-\frac{N_k}{2}} \exp\left(-\frac{1}{2}\frac{\sigma_k^2}{P(k)}\right) \end{aligned} \tag{6}$$

where σ_k^2 is effectively a variance estimation with $\tilde{\boldsymbol{s}}$:

$$\sigma_k^2 = \sum_{|\boldsymbol{k}_j|=k} \tilde{\boldsymbol{s}}_j^* \tilde{\boldsymbol{s}}_j. \tag{7}$$

The second line of eqn (6) actually gives a distribution of $p\left(P(k)|\tilde{\boldsymbol{s}}\right)$. And we can sample $P(k)$ by drawing a realisation of the inverse Gamma distribution.

3 Sampling foreground covariance

- Independent variables:
  - The foreground coefficients are denoted as $f_{i,n}$, where i is the index of the pixel and n is the index of the frequency dependent foreground basis function. The tuple of all coefficients is denoted by $\boldsymbol{f}$.
  - For each i we define a vector $\boldsymbol{f}^{(i)}$, which groups all foreground coefficients of the same i. The size of the vector is N_{modes}, the total number of basis functions.
  - As the draft paper explains, $\boldsymbol{f}^{(i)}$ follows a multivariate distribution:

$$\boldsymbol{f}^{(i)} \sim \mathcal{N}(\bar{\boldsymbol{f}}^{(i)}, \mathbf{F}) \tag{8}$$

    where $\mathbf{F}$ is an N_{modes}-by-N_{modes} covariance matrix.
  - Different $\boldsymbol{f}^{(i)}$ vectors are understood as different realisations of the same distribution.
- The conditional probability of $\mathbf{F}$ is given by

$$\begin{aligned} p\left(\mathbf{F}|\boldsymbol{f}\right) &= \prod_i p\left(\mathbf{F}|\boldsymbol{f}^{(i)}\right) \\ &\propto \prod_i \frac{1}{|\mathbf{F}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\boldsymbol{f}^{(i)} - \bar{\boldsymbol{f}}^{(i)}\right)^T \mathbf{F}^{-1}\left(\boldsymbol{f}^{(i)} - \bar{\boldsymbol{f}}^{(i)}\right)\right) \\ &= \prod_i \frac{1}{|\mathbf{F}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\,\mathrm{Tr}\left(\mathbf{F}^{-1}\mathbf{D}^{(i)}\right)\right] \\ &= |\mathbf{F}|^{-\frac{N_{pix}}{2}} \exp\left[-\frac{1}{2}\,\mathrm{Tr}\left(\mathbf{F}^{-1}\tilde{\mathbf{D}}\right)\right] \end{aligned} \tag{9}$$

where

$$\mathbf{D}^{(i)} \equiv \left(\boldsymbol{f}^{(i)} - \bar{\boldsymbol{f}}^{(i)}\right)\left(\boldsymbol{f}^{(i)} - \bar{\boldsymbol{f}}^{(i)}\right)^T \tag{10}$$

and

$$\tilde{\mathbf{D}} \equiv \frac{1}{N_{pix}} \sum_i^{N_{pix}} \mathbf{D}^{(i)} \tag{11}$$

is the scale matrix. The the mean values $\bar{\boldsymbol{f}}^{(i)}$ are estimated from $\boldsymbol{f}^{(i)}$, then the denominator in this equation above should be replaced accordingly with the correct number of degrees of freedom, which in most cases is $N_{pix} - 1$.

- F can then be sampled using the Inverse-Wishart distribution with

$$p = N_{\mathrm{modes}}, \qquad \nu = N_{pix} - p - 1, \tag{12}$$

where p is the size of the scale matrix and ν is the number of degrees of freedom. Note that degrees of freedom must be greater than or equal to the dimension of the scale matrix.

- "**L**" and "**H**" conventions for Cholesky decomposition

$$\mathbf{C} = \mathbf{L}\mathbf{L}^\dagger = \mathbf{H}^\dagger\mathbf{H} \tag{13}$$

They differ in where the † is placed. (Note the difference in `Numpy` and `Scipy` defaults.)

- Consistent **L** conventions:
  - Covariance

$$\mathbf{C} \equiv \mathbf{C}^{\frac{1}{2}}\mathbf{C}^{\frac{1}{2}\dagger} \qquad \boldsymbol{n} = \mathbf{C}^{\frac{1}{2}}\boldsymbol{w} \qquad \boldsymbol{n} \sim \mathbf{N}(0, \mathbf{C}) \tag{14}$$

  - Inverse covariance

$$\mathbf{C}^{-1} \equiv \mathbf{C}^{-\frac{1}{2}}\mathbf{C}^{-\frac{1}{2}\dagger} \qquad \boldsymbol{w} = \mathbf{C}^{-\frac{1}{2}\dagger}\boldsymbol{n} \qquad \boldsymbol{w} \sim \mathbf{N}(0, \mathbf{I}) \tag{15}$$

  - The Cholesky decomposition is not unique. However, given the Cholesky decomposition of a covariance matrix, you can always derive the Cholesky decomposition of its inverse

$$\mathbf{C}^{-\frac{1}{2}} = \mathbf{C}^{-1}\mathbf{C}^{\frac{1}{2}} \qquad \left(\mathbf{C}^{\frac{1}{2}}\right)^{-1} = \mathbf{C}^{-\frac{1}{2}\dagger} \tag{16}$$

  - Note that, given these definitions,

$$\mathbf{C}^{-\frac{1}{2}} \neq \left(\mathbf{C}^{\frac{1}{2}}\right)^{-1} \tag{17}$$

- In the GCR equations, given the above conventions, the $\mathbf{C}^{-\frac{1}{2}}$ term can be understood as coming from $\mathbf{C}^{-1}\mathbf{C}^{\frac{1}{2}}\boldsymbol{w}$

- Abstract:

$$\boldsymbol{d} = \mathbf{M}\boldsymbol{T} + \boldsymbol{n} \qquad\qquad \mathbf{M} = \mathbf{F}^\dagger\tilde{\mathbf{M}}\mathbf{F} \tag{18}$$

- Detailed

$$\begin{aligned}\boldsymbol{d}(x,y,z) &= \sum_{x',y',z'} \mathbf{M}(x,y,z;x',y',z')\boldsymbol{T}(x',y',z') + \boldsymbol{n}(x,y,z) \\ \mathbf{M}(x,y,z;x',y',z') &= \sum_{k_x,k_y,k_z} \mathbf{F}^\dagger(x,y,z;k_x,k_y,k_z)\tilde{\mathbf{M}}(k_x,k_y,k_z)\mathbf{F}(k_x,k_y,k_z;x',y',z')\end{aligned} \tag{19}$$

- Derived linear system with respect to $\tilde{\mathbf{M}}$

$$\boldsymbol{d}(x,y,z) = \sum_{k_x,k_y,k_z} \mathbf{U}(x,y,z;k_x,k_y,k_z)\tilde{\mathbf{M}}(k_x,k_y,k_z) + \boldsymbol{n}(x,y,z) \tag{20}$$

  where

$$\mathbf{U}(x,y,z;k_x,k_y,k_z) = \sum_{x',y',z'} \mathbf{F}^\dagger(x,y,z;k_x,k_y,k_z)\mathbf{F}(k_x,k_y,k_z;x',y',z')\boldsymbol{T}(x',y',z') \tag{21}$$

- By realising the speparable form the Fourier matrices, we obtain

$$\begin{aligned}\mathbf{U}(x,y,z;k_x,k_y,k_z) &= \sum_{x',y',z'} \mathbf{F}^\dagger(x;k_x)\mathbf{F}^\dagger(y;k_y)\mathbf{F}^\dagger(z;k_z)\mathbf{F}(k_x,x')\mathbf{F}(k_y,y')\mathbf{F}(k_z,z')\boldsymbol{T}(x',y',z') \\ &= \mathbf{F}^\dagger(x;k_x)\mathbf{F}^\dagger(y;k_y)\mathbf{F}^\dagger(z;k_z)\tilde{\boldsymbol{T}}(k_x,k_y,k_z) \\ &= \mathbf{F}^\dagger(x,y,z;k_x,k_y,k_z)\tilde{\boldsymbol{T}}(k_x,k_y,k_z)\end{aligned} \tag{22}$$

  The last equation means that $\mathbf{U}$ is the IDFT matrix whose columns are weighted by the Fourier transform of $\boldsymbol{T}$:

```
def DFT_matrix(n):
    # using the default norm
    # check the internal consistency of normalisation in your code
    return np.fft.fft(np.eye(n))

def tensor_product(*matrices):
    result = matrices[0]
    for matrix in matrices[1:]:
        result = np.kron(result, matrix)
    return result

DFT = DFT_matrix(32)
DFT_n3_matrix = tensor_product([DFT, DFT, DFT])

 def m_projector(Temp_cube, DFT_n3_matrix):
     Temp_fft_vec = np.fft.fftn(Temp_cube, axes=(0, 1, 2)).flaten()
     U = DFT_n3_matrix.conj().T * Temp_fft_vec[np.newaxis, :]
     return U # the output is a 2D matrix
```