Electronics Engineering
A.Y. 2024/2025
Learning in Communications

Structured LISTA for Multidimensional Harmonic Retrieval

Andrea Comastri, Marco Mura

Contents

Chapter 1

Introduction

This report presents the work carried out to replicate the numerical experiments described in the paper *"Structured LISTA for Multidimensional Harmonic Retrieval"* by Rong Fu, Yimin Liu, Tianyao Huang, and Yonina C. Eldar, published in 2021 [1]. The paper addresses the Multidimensional Harmonic Retrieval (MHR) problem and proposes a structured LISTA-Toeplitz network. This network constrains the mutual inhibition matrices to have a Toeplitz (or block-Toeplitz) structure and replaces matrix-vector multiplications with linear convolutions, thereby significantly reducing the number of learnable parameters in large-scale MHR problems.

The rest of this report is organized as follows:

- **Chapter 2** provides the theoretical background of the MHR problem and summarizes the key aspects of the proposed LISTA-Toeplitz algorithm. The standard methods ISTA, FISTA, and LISTA are first introduced, highlighting their strengths and limitations. Then, the motivation for LISTA-Toeplitz is discussed, together with its implementation and advantages.
- **Chapter 3** describes the code implementation and reports the numerical results obtained from the replication of the experiments presented in the paper. Both 1D and 2D harmonic retrieval problems are considered, and the performance of LISTA-Toeplitz is compared against ISTA, FISTA, and LISTA in terms of NMSE and hit rate. The figures confirm the efficiency and accuracy of LISTA-Toeplitz, even when trained with limited samples or tested under off-the-grid conditions.
- **Chapter 4** provides final comments and discusses the comparison between our replicated results and those reported in the original paper. In particular, it is shown that the outcomes match closely, thereby validating both the correctness of our implementation and the effectiveness of the LISTA-Toeplitz network proposed in the paper.

Chapter 2

Structured LISTA for MHR: Theoretical Background

This chapter summarizes the paper *"Structured LISTA for Multidimensional Harmonic Retrieval"* by Rong Fu, Yimin Liu, Tianyao Huang and Yonina C. Eldar [1], highlighting the most relevant aspects related to the implementation of the proposed LISTA-Toeplitz algorithm. Section 2.1 introduces the MHR problem and some standard algorithms (ISTA, FISTA, and LISTA) used to address it, emphasizing their limitations. Section 2.2 explains the motivation for LISTA-Toeplitz and briefly discusses its implementation and advantages. Most mathematical details will be omitted.

2.1 Multidimensional Harmonic Retrieval: ISTA, FISTA and LISTA

The harmonic retrieval problem consists of recovering the frequencies and amplitudes of harmonic signals from observed time-domain samples. Multidimensional harmonic retrieval (MHR) extends this to multidimensional frequency models. In many applications, it is desirable to minimize the number of Nyquist samples required. Compressed sensing can be leveraged to tackle such a challenge, especially when the number of harmonics to be recovered is small. In this framework, MHR can be formulated as a linear decoding problem:

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{w}, \tag{2.1}$$

where $\boldsymbol{y} \in \mathbb{C}^N$ are the obtained sub-Nyquist samples, $\boldsymbol{\Phi} \in \mathbb{C}^{N\times M}$ is the observation matrix, $\boldsymbol{x} \in \mathbb{C}^M$ is the sparse spectral representation of the unknown sinusoids, and $\boldsymbol{w} \in \mathbb{C}^N$ is additive noise. Since $\boldsymbol{\Phi}$ is usually underdetermined, sparsity priors are required for recovery.
In many practical settings, the observation matrix can be expressed as

$$\boldsymbol{\Phi} = \boldsymbol{R}\boldsymbol{\Psi}, \tag{2.2}$$

where $\boldsymbol{\Psi}$ is a Fourier matrix and $\boldsymbol{R}$ is a row-subsampling operator that selects only a subset of Fourier coefficients. This reflects the fact that the signal is observed at sub-Nyquist rate. In this compressed sensing framework, recovery of $\boldsymbol{x}$ is still possible thanks to its sparsity in the Fourier domain.

ISTA

The Iterative Shrinkage Thresholding Algorithm (ISTA) estimates $\boldsymbol{x}$ by solving a regularized regression problem of the form

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{2.3}$$

where λ is the regularization parameter controlling the sparsity penalty characterized by the ℓ_1 norm. The iterations are given by

$$\boldsymbol{x}^{(t+1)} = S_{\lambda/L}\left(\boldsymbol{x}^{(t)} + \frac{1}{L}\Phi^H(\boldsymbol{y} - \Phi\boldsymbol{x}^{(t)})\right), \tag{2.4}$$

where $S_\theta(\cdot)$ denotes the soft-thresholding operator and L is the Lipschitz constant of $\Phi^H\Phi$. ISTA is simple and accurate but converges slowly, typically at a rate $O(1/k)$.

FISTA

To accelerate ISTA, Beck and Teboulle proposed the Fast Iterative Shrinkage Thresholding Algorithm (FISTA) [2]. It introduces a momentum term, performing the shrinkage not on $\boldsymbol{x}^{(t)}$ itself but on a linear combination of the last two iterates. This modification preserves the simplicity of ISTA while improving the convergence rate to $O(1/k^2)$, making it much faster in practice.

LISTA

LISTA (Learned ISTA) [3] takes a different approach: instead of fixing the parameters of ISTA, it unfolds a fixed number T of iterations into a neural network of T layers and learns the parameters $\{\boldsymbol{W_e}^{(t)}, \boldsymbol{W_g}^{(t)}, \boldsymbol{\theta}^{(t)}\}$ with $t = 0, 1, ..., T$ from data:

$$\boldsymbol{x}^{(t+1)} = S_{\theta^{(t)}}\left(\boldsymbol{W_e}^{(t)}\boldsymbol{y} + \boldsymbol{W_g}^{(t)}\boldsymbol{x}^{(t)}\right). \tag{2.5}$$

Here $\boldsymbol{W_e}^{(t)} \in \mathbb{C}^{M\times N}$ plays the role of a learned encoding operator (analogous to $\frac{1}{L}\boldsymbol{\Phi}^H$), while $\boldsymbol{W_g}^{(t)} \in \mathbb{C}^{M\times M}$ acts as the learned inhibition matrix (analogous to $\boldsymbol{I} - \frac{1}{L}\boldsymbol{\Phi}^H\boldsymbol{\Phi}$). The thresholds $\boldsymbol{\theta}^{(t)}$ are also learned for each layer. LISTA converges in very few iterations and is significantly faster than ISTA and FISTA. However, it has drawbacks:

- The matrix $\boldsymbol{W_g}$ requires $O(M^2)$ parameters, making the method expensive in memory and computation.
- For large-scale MHR problems, storing and training such a network is impractical.
- Training requires a large number of labeled samples to avoid overfitting.

Fig. 2.1 illustrates a LISTA network structure with T layers.

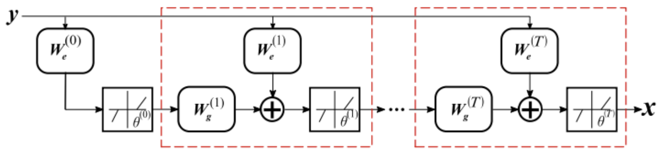

Figure 2.1: Block diagram of LISTA, in which the red dashed-line boxes indicate the process of one ISTA iteration.

2.2 From LISTA to LISTA-Toeplitz

A key observation is that in MHR the Gram matrix $\boldsymbol{\Phi}^H\boldsymbol{\Phi}$ naturally exhibits a Toeplitz (1D) or doubly block-Toeplitz (2D) structure. Consequently, also the matrix $\boldsymbol{W_g}$ inherits this structure. Instead of storing all its entries, $\boldsymbol{W_g}$ can be parameterized by a reduced set of coefficients: a vector $\boldsymbol{h}$ in 1D, or a small matrix $\boldsymbol{H}$ in 2D. In the 1D case, $\boldsymbol{h^{(t)}} \in \mathbb{C}^{(2M-1)}$ contains the coefficients that define each diagonal of the Toeplitz matrix $\boldsymbol{W_g}$, so that the entire matrix is uniquely determined by this single vector. In the 2D case, $\boldsymbol{H} \in \mathbb{C}^{(2M_1-1)\times(2M_2-1)}$($M_1$ and M_2 denote the cardinality of the grid sets in the first and second dimension) collects the coefficients that specify each block of the doubly block-Toeplitz structure, again allowing reconstruction of $\boldsymbol{W_g}$ from far fewer parameters. This reduces the degrees of freedom from $O(M^2)$ to $O(M)$ (1D), and from $O(M^4)$ to $O(M^2)$ (2D).

The Toeplitz structure of $\boldsymbol{\Phi}^H\boldsymbol{\Phi}$ in MHR arises because $\boldsymbol{\Phi}$ is obtained from a row-subsampled Fourier matrix. Multiplying $\boldsymbol{\Phi}^H$ by $\boldsymbol{\Phi}$ amounts to forming correlations between Fourier atoms. These correlations depend only on the difference between frequency indices, not on their absolute positions. This shift-invariance property is what makes the Gram matrix Toeplitz in 1D and doubly block-Toeplitz in 2D.

Implementation via Convolution

Thanks to the Toeplitz structure (1D), the product $\boldsymbol{W_g x}$ can be rewritten as a convolution:

$$\boldsymbol{W_g x} = \boldsymbol{h} * \boldsymbol{x}. \tag{2.6}$$

Thus, each LISTA layer becomes

$$\boldsymbol{x}^{(t+1)} = S_{\theta^{(t)}}\left(\boldsymbol{W_e}^{(t)}\boldsymbol{y} + \boldsymbol{h}^{(t)} * \boldsymbol{x}^{(t)}\right). \tag{2.7}$$

In the 2D case, the same idea applies, with the multiplication realized as a 2D convolution $\boldsymbol{H} * \boldsymbol{X}$ after reshaping the vector $\boldsymbol{x}$ into a matrix, i.e.:

$$\boldsymbol{x}^{(t+1)} = S_{\theta^{(t)}}\left(\boldsymbol{W_e}^{(t)}\boldsymbol{y} + vec(\boldsymbol{H}^{(t)} * \boldsymbol{X}^{(t)})\right). \tag{2.8}$$

Fig. 2.2 illustrates a LISTA-Toeplitz network structure with T layers in a 1D scenario.

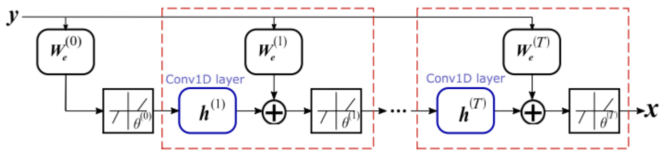

Figure 2.2: Block diagram of 1D LISTA-Toeplitz. The red dashed-line boxes indicate each layer of LISTA-Toeplitz, corresponding to the whole process of an ISTA iteration. Blue boxes represent convolutional layers, which highlight the modification of LISTA-Toeplitz over standard LISTA.

Advantages

This reformulation brings significant benefits:

- **Reduced complexity**: from quadratic to linear in 1D, and from quartic to quadratic in 2D.
- **Computational efficiency**: convolutions can be efficiently computed, even via FFT, reducing complexity from $O(M^2)$ to $O(M \log M)$.
- **Scalability**: LISTA-Toeplitz remains applicable to large-scale MHR problems, unlike standard LISTA.
- **Comparable accuracy**: simulations show similar or better performance compared to LISTA, with much fewer parameters.

In summary, LISTA-Toeplitz exploits the natural structure of the MHR problem to obtain a network that is both lighter and efficient, while retaining high recovery accuracy in both 1D and 2D harmonic retrieval.

Chapter 3

Code Implementation and Numerical Results

To evaluate the effectiveness of the LISTA-Toeplitz network proposed in [1], in comparison with the original LISTA network and conventional iterative algorithms such as ISTA and FISTA, we reproduce most of the numerical experiments presented in the paper. Both 1D and 2D harmonic retrieval problems are addressed using synthetic data. The chapter is organized as follows: Section 3.1 describes the implementation of the code, while Section 3.2 define the figures of merit used to assess the quality of the recovery; Sections 3.3 and 3.4 report and discuss the results obtained on synthetically generated data for 1D and 2D harmonic retrieval problems, respectively.

3.1 Code Implementation

All simulations were carried out using Python. The required libraries and their respective versions are listed in the `requirements.txt` file located inside the code folder. In this section, we address some of the most significant aspects of the code implementation.

Complex-Valued Network Implementation

Most off-the-shelf deep learning frameworks, such as the one we used (PyTorch), are designed for real-valued networks. As described in the paper, matrix multiplications and nonlinear operations are therefore rewritten in terms of their real-valued counterparts.

As an example, we report below the implementation of the complex soft-thresholding operator, which is shared between every algorithm. In the paper, the operator is defined as (formula (24)):

$$\Re\{S_\theta([\boldsymbol{x}]_i)\} = \Re\{[x]_i\}\left(1 - \frac{\theta}{\max\left(|[\boldsymbol{x}]_i|, \theta\right)}\right), \tag{3.1}$$

$$\Im\{S_\theta([\boldsymbol{x}]_i)\} = \Im\{[\boldsymbol{x}]_i\}\left(1 - \frac{\theta}{\max\left(|[\boldsymbol{x}]_i|, \theta\right)}\right). \tag{3.2}$$

In practice, this operation scales each entry of x depending on its magnitude, shrinking coefficients below the threshold θ to zero while reducing the others proportionally.

The corresponding Python implementation is:

```python
def _complex_soft_threshold(self, x, theta):
    # x: [batch, 2*m] (Re, Im)
    x_real, x_imag = x[:, :self.m], x[:, self.m:]
    mag = torch.sqrt(x_real**2 + x_imag**2 + 1e-12)
    scale = torch.clamp(1 - theta / (mag + 1e-12), min=0.0)
    x_real = x_real * scale
    x_imag = x_imag * scale
    return torch.cat([x_real, x_imag], dim=1)
```

Listing 3.1: Complex soft-thresholding operator in PyTorch

This implementation directly corresponds to equations 3.1 and 3.2, where the real and imaginary parts of the complex input are processed separately. The magnitude is computed, the shrinkage factor is applied, and the two components are concatenated back to form the thresholded complex vector.

Synthetic Data Generation

Data generation is a crucial step for training both the LISTA and LISTA-Toeplitz networks. The following function was used to generate the training set for these networks, as well as the test set used to evaluate and compare all algorithms, including ISTA and FISTA. For clarity, we only report the 1D case, since the 2D implementation is very similar.

```python
def create_data_set(H, n, m, k, N=1000, batch_size=512, signal_dev=1,
    noise_dev=0.01): # bs 512

    ### Initialization ###
    y = torch.zeros(N, n, dtype=H.dtype)
    x = torch.zeros(N, m, dtype=H.dtype)

    ### Create signals ###
    for i in range(N):
        ### Create a k-sparsed signal x ###
        index_k = np.random.choice(m, k, replace=False)

        if H.dtype == torch.complex128:
            peaks_real = (signal_dev/np.sqrt(2)) * np.random.randn(k)
            peaks_imag = (signal_dev/np.sqrt(2)) * np.random.randn(k)
            peaks = peaks_real + 1j*peaks_imag

            x[i, index_k] = torch.from_numpy(peaks).to(x)

            # y = Hx+w
            y[i, :] = H @ x[i, :] +  (noise_dev/np.sqrt(2)) * (torch.randn
    (n) + 1j*torch.randn(n))

        else:
            peaks = signal_dev * np.random.randn(k)

            x[i, index_k] = torch.from_numpy(peaks).to(x)

            # y = Hx+w
            y[i, :] = H @ x[i, :] +  noise_dev * torch.randn(n)

    simulated = SimulatedData(y=y, H=H, x=x)
    data_loader = Data.DataLoader(dataset=simulated, batch_size=batch_size
    , shuffle=True)

    return data_loader
```

Listing 3.2: Function used to generate synthetic data

Each sample is generated by first creating a k-sparse signal $\boldsymbol{x}$ with Gaussian-distributed nonzero entries, and then forming the observation $\boldsymbol{y}$ according to the linear model in equation 2.1. The function also provides the option to generate an off-the-grid dataset, allowing us to test the algorithms under non-ideal conditions where the spectral components do not lie exactly on the assumed frequency grid. This makes the evaluation more robust and realistic.

In addition, we define a dedicated class that inherits from `torch.utils.data.Dataset`. This class returns tuples of the form $(\boldsymbol{y}, \boldsymbol{H}, \boldsymbol{x})$, making it straightforward to handle both the observations and their corresponding sparse representations during training and evaluation.

```python
class SimulatedData(Data.Dataset):
    def __init__(self, y, H, x):
        self.y = y
        self.x = x
        self.H = H

    def __len__(self):
        return self.y.shape[0]

    def __getitem__(self, idx):
        y = self.y[idx, :]
        H = self.H
        x = self.x[idx, :]
        return y, H, x
```

Listing 3.3: Class to handle datasets

Training

The training procedure used by LISTA and LISTA-Toeplitz follows a standard supervised learning setup. The function `train` takes as input the model, the training and validation data loaders, and the number of epochs. The optimization is performed using stochastic gradient descent (SGD) with momentum, while a step learning rate scheduler decreases the learning rate after a fixed number of epochs.

During each epoch, the model is set to training mode and updated using mini-batches from the training loader. The prediction $\hat{\boldsymbol{x}}$ is computed via the forward pass, and the mean squared error (MSE) between $\hat{\boldsymbol{x}}$ and the ground truth $\boldsymbol{x}$ is used as the loss function. The optimizer then performs backpropagation to update the network parameters.

At the end of each epoch, the scheduler updates the learning rate, and the model is evaluated on the validation set in evaluation mode. The average training and validation losses are recorded for each epoch, enabling monitoring of convergence and generalization performance.

The function returns the validation losses across all epochs, which can later be used for performance analysis and visualization.

```python
def train(model, train_loader, valid_loader, num_epochs=50):

    # Initialization
    optimizer = torch.optim.SGD(
        model.parameters(),
        lr=7e-6,
        momentum=0.9,
        weight_decay=0,
    )
    scheduler = torch.optim.lr_scheduler.StepLR(
        optimizer, step_size=50, gamma=0.1
    )
    loss_train = np.zeros((num_epochs,))
    loss_test = np.zeros((num_epochs,))

    # Main loop
    for epoch in range(num_epochs):
        model.train()
        train_loss = 0
        for step, (b_y, b_H, b_x) in enumerate(train_loader):
            x_hat, _ = model.forward(b_y)
            loss = torch.sum(torch.abs(x_hat - b_x) ** 2)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            model.zero_grad()
            train_loss += loss.data.item()
        loss_train[epoch] = train_loss / len(train_loader.dataset)
        scheduler.step()

        # validation
        model.eval()
        test_loss = 0
        for step, (b_y, b_H, b_x) in enumerate(valid_loader):
            # b_y, b_H, b_y = b_y.cuda(), b_H.cuda(), b_x.cuda()
            x_hat, _ = model.forward(b_y)
            test_loss += torch.sum(torch.abs(x_hat - b_x) ** 2)
        loss_test[epoch] = test_loss / len(valid_loader.dataset)

    return loss_train, loss_test
```

Listing 3.4: Function used to train the LISTA models

Parameters and Parameter Initialization

The parameters of the MHR problem include M, the number of samples of the sparse signal $\boldsymbol{x}$ (denoted M_1 and M_2 in the 2D case), and N, the number of observation samples of the signal $\boldsymbol{y}$. The sparsity level k specifies the number of non-zero entries in $\boldsymbol{x}$, while σ^2 represents the noise power affecting the latter. The parameter T_{LISTA} defines the number of layers in both the LISTA and LISTA-Toeplitz networks. T_{ISTA} is the number of iterations that ISTA and FISTA perform. The regularization parameter λ plays a critical role in all algorithms, as it directly influences convergence speed and the overall recovery accuracy. N_{tr} denotes the number of training samples used in LISTA and LISTA-Toeplitz; this choice is closely related to the number of parameters the network must learn. To minimize the latter, the parameters of both the LISTA and LISTA-Toeplitz networks are kept constant across layers, except for the threshold parameters, which are initialized as $\boldsymbol{\theta}^{(t)} = \frac{\lambda}{L}$. The number of epochs of the training procedure is configurable too. Finally, N_{test} indicate the number of samples used to test the different algorithms.

For parameter initialization, the dictionary is first estimated from the training data. Given training pairs $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^{N_{tr}}$, we form the observation and label matrices as

$$\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{N_{tr}}], \quad \boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{N_{tr}}].$$

A coarse estimate of the dictionary is then computed as

$$\hat{\boldsymbol{\Phi}} = \boldsymbol{Y}(\boldsymbol{X}^H \boldsymbol{X})^{-1} \boldsymbol{X}^H.$$

Using this estimate, the parameter W_e in LISTA and LISTA-Toeplitz is initialized as

$$\boldsymbol{W_e} = \frac{1}{L} \hat{\boldsymbol{\Phi}}^H,$$

where $L = \lambda_{\max}(\hat{\boldsymbol{\Phi}}^H \hat{\boldsymbol{\Phi}})$ is the largest eigenvalue of the Gram matrix. Other parameters such as $\boldsymbol{W_g}$ in LISTA, or the convolutional filters $\boldsymbol{h}$ and $\boldsymbol{H}$ in LISTA-Toeplitz (for 1D and 2D, respectively), are initialized to zeros in the experiments, although random initialization is also possible.

It is important to highlight that the initialization of both the MHR problem parameters and the network-specific parameters (in LISTA, LISTA-Toeplitz, and the iterative algorithms) is essential, as it strongly influences the behavior and performance of the algorithms.

3.2 Figures of Merit

Before discussing the results of the numerical experiments, we first introduce the two metrics used to assess the reconstruction quality of the different methods in both the 1D and 2D settings. The first one is the normalized mean squared error (NMSE), which measures the mean squared error of the recovered signal $\hat{\boldsymbol{x}}$ normalized by the power of the ground truth signal $\boldsymbol{x}$:

$$\mathrm{NMSE} = \mathbb{E}\left[\frac{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2}{\|\boldsymbol{x}\|^2}\right]. \tag{3.3}$$

The second metric is the hit rate, defined as the percentage of correctly recovered components among the K nonzero entries of $\boldsymbol{x}$. Specifically, for each nonzero entry $[x]_i$, the recovery is considered correct if $i \in S_K(\hat{\boldsymbol{x}})$, where $S_K(\hat{\boldsymbol{v}})$ denotes the set of indices corresponding to the K largest-magnitude entries of a vector $\boldsymbol{v} \in \mathbb{C}^M$.

3.3 1D MHR Numerical Results

In the following we list all the numerical experiments we carried out in the 1D case, specifying the parameter initialization used and commenting the results.

Recovered NMSE of LISTA and LISTA-Toeplitz for $T = 2$ to 10 under Different Noise Levels

$M = 128,\ N = 64,\ k = 4,\ N_{tr} = 3000,\ N_{test} = 100,\ N_{epoch} = 50,\ T = 2 : 10,\ \sigma^2 = -40 : -15dB,\ \lambda = 0.02$

We evaluate the performance of LISTA and LISTA-Toeplitz with different numbers of layers, ranging from $T = 2$ to $T = 10$, and under different noise power levels σ^2, varying from -40 dB to -15 dB. The recovered NMSE of each network is reported in Fig. 3.1.

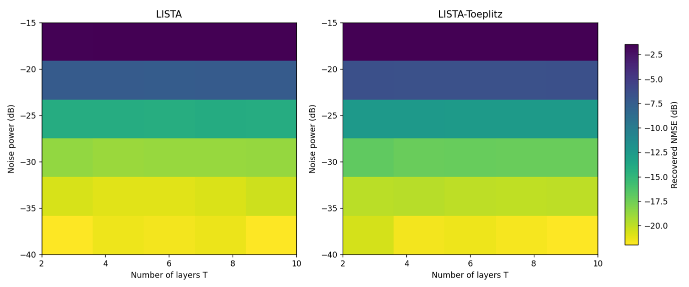

Figure 3.1: Recovered NMSE of LISTA and LISTA-Toeplitz for $T = 2$ to 10 under different noise levels.

As expected, the performance of both networks degrades as the noise power increases. The NMSE values of LISTA and LISTA-Toeplitz are very similar, showing no significant differences. Unlike what is reported in the reference paper, we do not observe a noticeable improvement in performance when increasing the number of layers for a fixed noise level. This behavior can be explained by the fact that we kept the learnable parameters constant across layers, and during training only the final output (after T layers) was optimized.

NMSE of Different Algorithms and Networks in Each Iteration/Layer

$M = 128,\ N = 64,\ k = 4,\ N_{tr} = 3000,\ N_{test} = 100,\ N_{epoch} = 50,\ T = 5,\ \sigma^2 = -40dB$

Inherited from LISTA, the LISTA-Toeplitz network preserves the advantage of a faster convergence rate compared to ISTA and FISTA. The recovered NMSE values at each iteration (for ISTA and FISTA) or at each layer (for LISTA and LISTA-Toeplitz) are reported in Fig. 3.2. In this experiment, we consider a fixed number of layers $T = 5$ and set the noise power to $\sigma^2 = -40$ dB. For ISTA and FISTA, the regularization parameter λ in (2) is chosen as 0.02, 0.1, and 0.5.

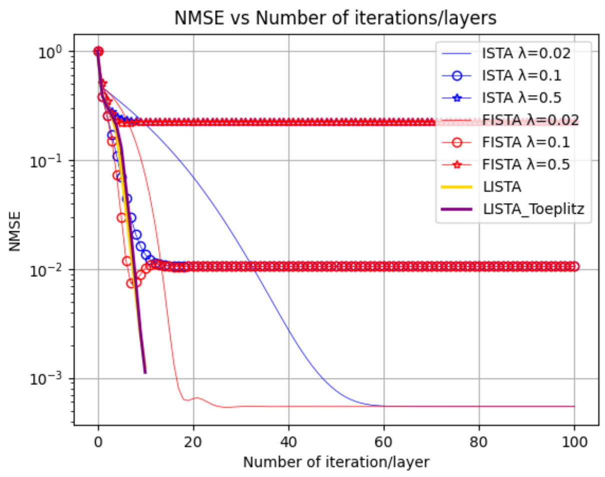

Figure 3.2: NMSE of Different Algorithms and Networks in Each Iteration/Layer with noise power $\sigma^2 = -40dB$

The results highlight that ISTA and FISTA are affected by a fundamental trade-off between reconstruction accuracy and convergence speed, depending on the value of λ: larger values of λ result in faster convergence but poorer recovery performance. In contrast, LISTA and LISTA-Toeplitz are able to learn key parameters, such as λ, through end-to-end training, thereby achieving the best possible recovery performance within a limited computational budget.

Since the total computational cost grows proportionally with the number of iterations, LISTA and LISTA-Toeplitz reach convergence significantly faster than the classical ISTA and FISTA, thus reducing the required computation time.

Limited Training Samples

$M = 128,\ N = 64,\ k = 4,\ N_{tr} = 1500,\ N_{test} = 100,\ N_{epoch} = 100,\ T = 5,\ \sigma^2 = -40dB$

Figure 3.3 illustrates the learning behavior of LISTA and LISTA-Toeplitz when trained with a limited number of samples. Since LISTA contains a larger number of parameters to optimize, it is more prone to overfitting compared to LISTA-Toeplitz. In particular, after approximately 50 epochs (where each epoch corresponds to N_{tr} update steps), LISTA begins to overfit the training data, as evidenced by the stagnation of the validation loss.

In contrast, LISTA-Toeplitz demonstrates better generalization on this dataset, maintaining stable recovery performance even with fewer training samples. This result highlights that exploiting the Toeplitz structure improves sample efficiency during training.

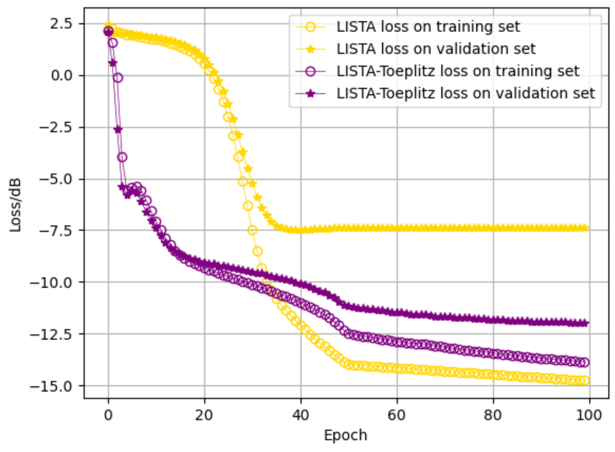

Figure 3.3: Learning curves with limited training samples for LISTA and LISTA-Toeplitz.

On-the-grid vs Off-the-grid

$M = 128$, $N = 64$, $k = 4$, $N_{tr} = 3000$, $N_{test} = 100$, $N_{epoch} = 50$, $T = 5$, $\sigma^2 = -40$ dB

Here we evaluate the recovery performance of ISTA, FISTA, LISTA, and LISTA-Toeplitz in both on-the-grid and off-the-grid scenarios. In the first case (Fig. 3.4), the frequencies of the sinusoids lie exactly on the discretized grid points. In the second case (Fig. 3.5), the true frequencies are shifted by one quarter of the grid spacing, resulting in an off-the-grid mismatch.

The results show that all methods are able to recover the targets in the on-the-grid setting. In the more challenging off-the-grid case, however, the mismatch introduces reconstruction errors for all the algorithms.

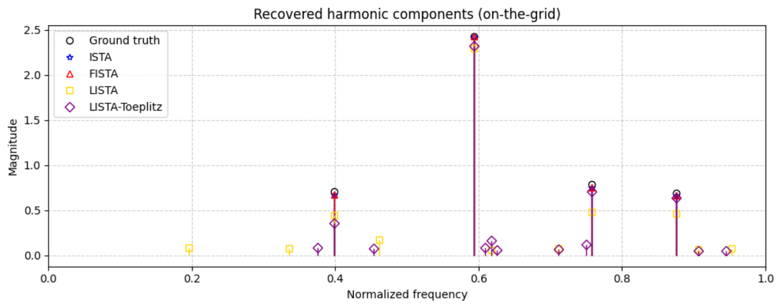

Figure 3.4: Recovered results in the on-the-grid case.

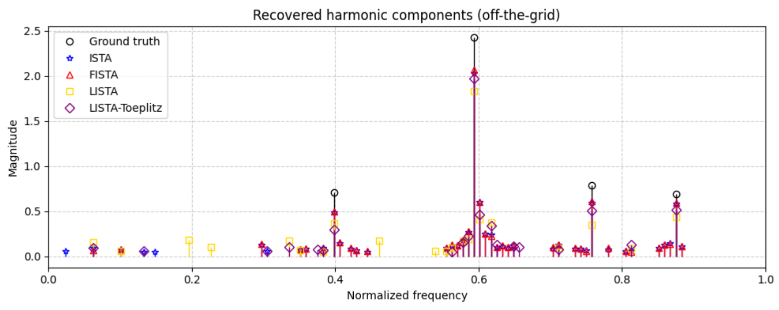

Figure 3.5: Recovered results in the off-the-grid case.

To emphasize the impact of parameter initialization on the recovery performance of the networks, we also report the on-the-grid(Fig. 3.6) and off-the-grid(Fig. 3.7) results obtained when $\boldsymbol{W_g}$ and $\boldsymbol{h}$ are initialized with ideal values ($\boldsymbol{W_g} = I - \frac{1}{L}\hat{\boldsymbol{\Phi}}^H\hat{\boldsymbol{\Phi}}$ and $\boldsymbol{h}$ derived from $\boldsymbol{W_g}$), instead of using the initialization $\boldsymbol{W_g} = 0$ and $\boldsymbol{h} = 0$.

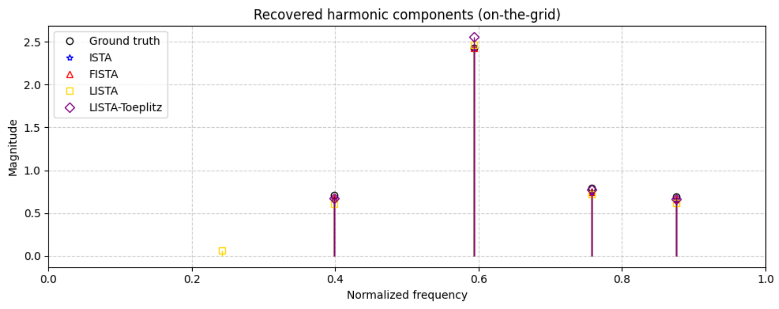

Figure 3.6: Recovered results in the on-the-grid case, ideal initialization.

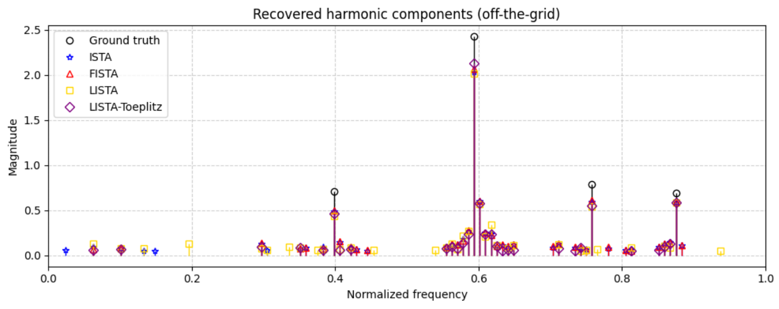

Figure 3.7: Recovered results in the off-the-grid case, ideal initialization.

NMSE and Hit Rate of 1D Harmonic Retrieval Comparison

$M = 128,\ N = 64,\ k = 4,\ N_{tr} = 3000,\ N_{test} = 100,\ N_{epoch} = 50,\ T = 5,\ \sigma^2 = -40$ dB, $\lambda = 0.1$

We compare the performance of ISTA, FISTA, LISTA, and LISTA-Toeplitz in terms of normalized mean squared error (NMSE) and hit rate. The NMSE quantifies the overall recovery accuracy, while the hit rate measures the ability to correctly identify the true frequency components. LISTA and LISTA-Toeplitz were trained using a fixed noise power of $\sigma^2 = -40$ dB, while during testing the noise power was varied from -70 dB to 20 dB. The corresponding curves are reported in Fig. 3.8 and Fig. 3.9.

The results show that LISTA-Toeplitz achieves recovery performance very close to that of ISTA, FISTA, and LISTA across the entire range of noise levels. In particular, both the NMSE and hit rate curves of LISTA-Toeplitz almost overlap with those of the other algorithms, confirming that the Toeplitz structure does not compromise reconstruction accuracy. At the same time, the structural constraint significantly reduces the number of learnable parameters, making the network more efficient.

These findings validate the effectiveness of LISTA-Toeplitz: it provides accuracy comparable to classical iterative methods and to LISTA, while retaining the benefits of lower complexity and faster training.

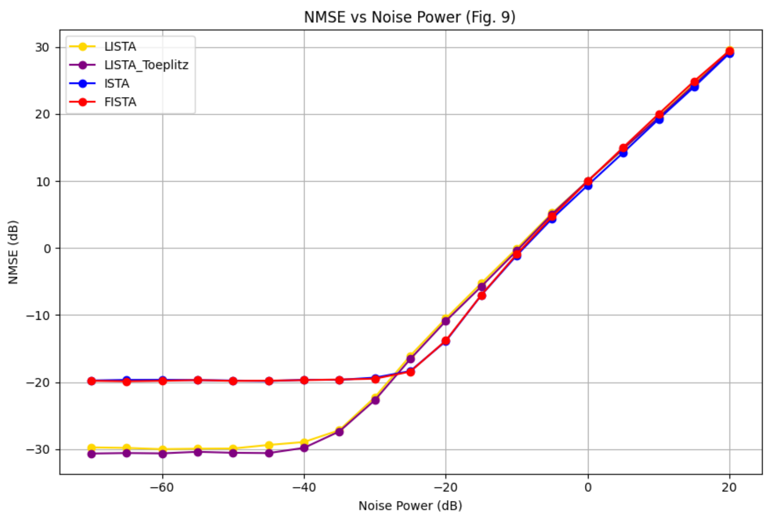

Figure 3.8: NMSE comparison of ISTA, FISTA, LISTA, and LISTA-Toeplitz.

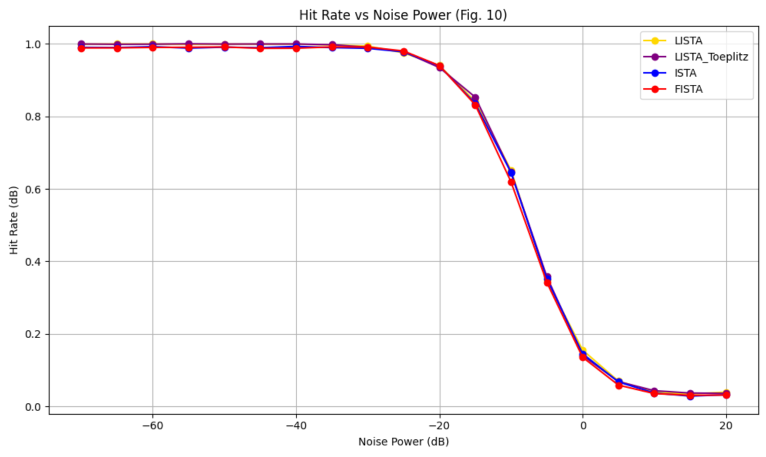

Figure 3.9: Hit rate comparison of ISTA, FISTA, LISTA, and LISTA-Toeplitz.

3.4 2D MHR Numerical Results

Simulations were also conducted to show the performance of LISTA-Toeplitz for 2D harmonic retrieval problems, where reduction of the parameters to learn is even more essential. The numerical experiments we carried out in the 2D case are listed in the following.

Recovered 2D plane of MH components

$M1 = 12$, $M2 = 12$, $N = 64$, $k = 4$, $N_{tr} = 3000$, $N_{test} = 100$, $N_{epoch} = 50$, $T = 5$, $\sigma^2 = -40$ dB, $\lambda = 0.02$

Here evaluate the recovery performance of ISTA, FISTA, LISTA, and LISTA-Toeplitz in a 2D on-the-grid scenarios, i.e. the frequencies of the sinusoids lie exactly on the discretized grid points. As for the 1D case, the results (Fig. 3.10) show that all methods are able to recover the targets in the on-the-grid setting.

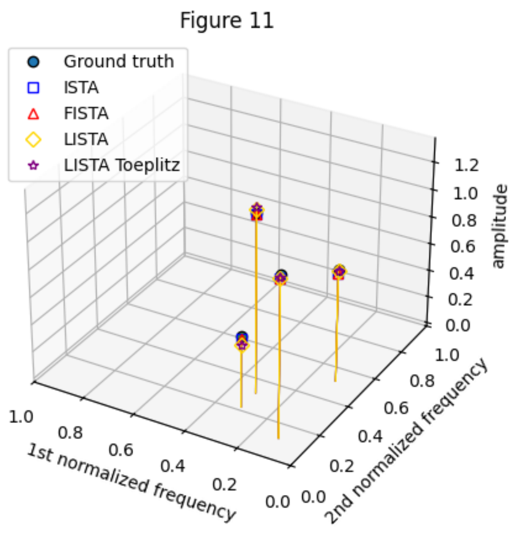

Figure 3.10: Recovered 2D plane of MH components in the on-the-grid setting

NMSE and Hit Rate of 2D Harmonic Retrieval Comparison

$M_1 = 8$, $M_2 = 16$, $M = M_1 \times M_2 = 128$, $N = 64$, $k = 4$, $N_{tr} = 3000$, $N_{test} = 100$, $N_{epoch} = 50$, $T = 5$, $\sigma^2 = -40$ dB, $\lambda = 0.1$

We extend the comparison to the 2D harmonic retrieval problem, evaluating ISTA, FISTA, LISTA, and LISTA-Toeplitz in terms of normalized mean squared error (NMSE) and hit rate. LISTA and LISTA-Toeplitz were trained with a fixed noise power of $\sigma^2 = -40$ dB, while in the test phase the noise power was varied from -70 dB to 20 dB. The resulting performance curves are presented in Fig. 3.11 and Fig. 3.12.

The results confirm that LISTA-Toeplitz provides recovery performance comparable to ISTA, FISTA, and LISTA across all noise levels. Both the NMSE and hit rate curves of LISTA-Toeplitz are very close to those of the other algorithms, which demonstrates that the Toeplitz structure can be exploited without loss of accuracy. At the same time, LISTA-Toeplitz significantly reduces the number of parameters compared to LISTA, yielding a more compact and efficient model.

These findings validate the robustness of LISTA-Toeplitz also in the 2D setting: it retains the high accuracy of classical iterative methods and LISTA, while benefiting from reduced complexity and faster training.

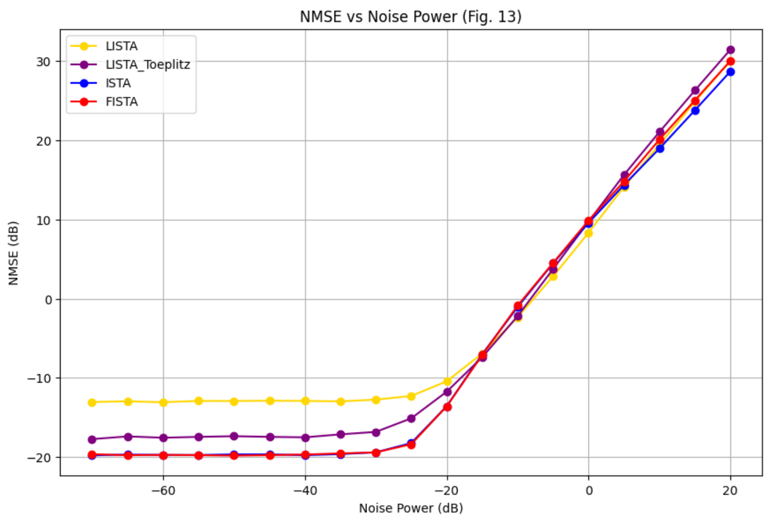

Figure 3.11: NMSE comparison of ISTA, FISTA, LISTA, and LISTA-Toeplitz in 2D harmonic retrieval.

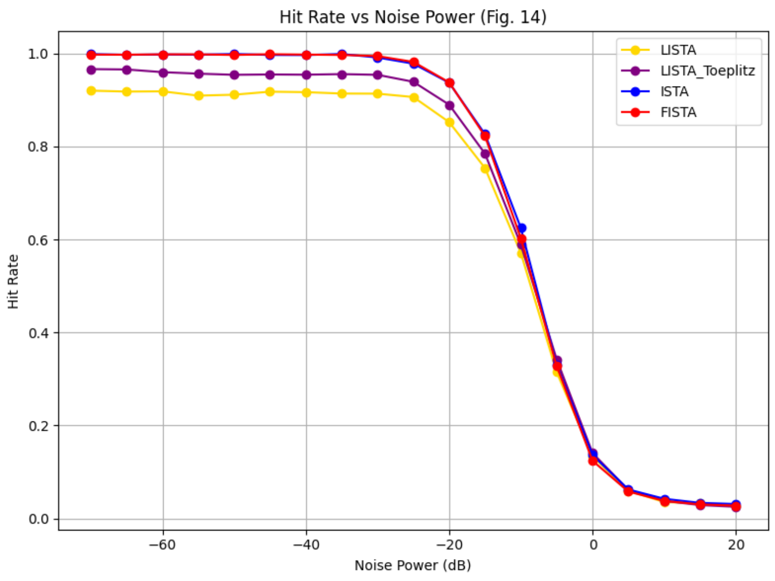

Figure 3.12: Hit rate comparison of ISTA, FISTA, LISTA, and LISTA-Toeplitz in 2D harmonic retrieval.

Chapter 4

Final Comments

We now draw some conclusions regarding the work carried out and the results obtained. The goal of this project was to replicate the numerical experiments presented in the paper *"Structured LISTA for Multidimensional Harmonic Retrieval"* by Rong Fu, Yimin Liu, Tianyao Huang and Yonina C. Eldar [1]. As discussed in Chapter 3, our experiments were designed to mirror those of the original paper as closely as possible.

Before presenting the comparison, a few clarifications are in order. Since the authors did not provide their implementation, our code and experimental setup were built from scratch, based on the descriptions available in the paper. In some cases, slight adjustments to parameters were introduced, mainly to reduce computational complexity and simulation time, given the limited computational resources at our disposal. These changes, however, were carefully chosen so as not to compromise the validity of the results and still allow for a fair comparison. Whenever some information in the paper was unclear or omitted, we made reasonable and consistent assumptions. Finally, for obvious reasons, we did not attempt to replicate the experiments involving real radar data for air target detection, and restricted our study to synthetic simulations.

With these considerations in mind, we can compare our results with those reported in the paper. Overall, the correspondence between the two is very strong. In both the 1D and 2D harmonic retrieval scenarios, our simulations confirm that LISTA-Toeplitz achieves performance nearly identical to that of ISTA, FISTA, and LISTA in terms of NMSE and hit rate, while requiring significantly fewer parameters and reduced training complexity. The characteristic overlap of the recovery curves, observed in both our results and the original ones, validates the claim that imposing a Toeplitz structure does not degrade reconstruction accuracy. Moreover, we also observed that LISTA-Toeplitz consistently benefits from improved sample efficiency and faster convergence, as highlighted in the original work.

Minor discrepancies can be attributed to differences in implementation details and computational constraints. For example, in certain experiments we used fewer training samples compared to the original settings. This choice was justified by the fact that we chose a smaller sparse signal's dimension. Nevertheless, the overall trends of the recovery curves and the relative performance of the algorithms are in excellent agreement with the findings of the paper.

In conclusion, our replication strongly supports the validity of the structured LISTA-Toeplitz network proposed in the original study. The algorithm achieves the same level of reconstruction accuracy as state-of-the-art iterative and learned methods, while offering lower complexity and improved efficiency. These results reinforce the value of structured deep unfolding networks in addressing large-scale multidimensional harmonic retrieval problems.

Bibliography

[1] Tianyao Huang Rong Fu Yimin Liu and Yonina C. Eldar. “Structured LISTA for Multidimensional Harmonic Retrieval”. In: *IEEE TRANSACTIONS ON SIGNAL PROCESSING* 69 (2021), pp. 3459–3472.

[2] Amir Beck and Marc Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *Siam J. Imag. Sci.* 2.1 (2009), pp. 183–202.

[3] K.Gregor and Y.Lecun. ““Learning fast approximations of sparse coding”. In: *Proc. Int. Conf. Int. Conf. Mach. Learn.* (2010), pp. 399–406.