slopesLinearMMSE class in OOMAO

Yoshito Ono

Version 1.0 complied on Apr. 5th, 2017

Abstract

The *slopesLinearMMSE* class can estimate the phase distortion in direction(s) of science target(s) by a (tomographic) wavefront reconstruction from slopes measured by Shack-Hartmann WFS(s) and estimate the phase distortion in direction(s) of science target(s). This document gives you a short summary of a mathematical background of the reconstructor, its functionalities and how to use it.

Contents

1 Definitions and Equations

1.1 MMSE tomographic reconstructor

A vector of measured slopes, $\hat{\boldsymbol{s}}$, from multiple WFSs are considered as

$$\hat{\boldsymbol{s}} = \boldsymbol{M}_{\boldsymbol{s}} \boldsymbol{W}_{\boldsymbol{s}} \left(\boldsymbol{s} + \boldsymbol{\eta} \right), \tag{1}$$

where $\boldsymbol{s}$ is a actual slope vector on a square pupil without noise, $\boldsymbol{\eta}$ is a vector of the measurement noise, $\boldsymbol{W}_{\boldsymbol{s}}$ is a block diagonal matrix defining slope pupil masks and $\boldsymbol{M}_{\boldsymbol{s}}$ is a block diagonal matrix removing tip/tilt/focus from slopes. The k^{th} diagonal block of $\boldsymbol{M}_{\boldsymbol{s}}$ is expressed as

$$\boldsymbol{M}_{\boldsymbol{s},\boldsymbol{k}} = \boldsymbol{I} - \boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}} \boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}}^{-1} \tag{2}$$

where $\boldsymbol{I}$ is an identity matrix and $\boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}} = [\boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}}^{\boldsymbol{x}}; \boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}}^{\boldsymbol{y}}]$. Each column in $\boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}}^{\boldsymbol{x}}$ and $\boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}}^{\boldsymbol{y}}$ has a Zernike mode of tip/tilt/focus in a x and y slope space, respectively. Each diagonal block of $\boldsymbol{W}_{\boldsymbol{s}}$ is a diagonal matrix with value of 1 for a valid measurement and 0 for an invalid measurement. The MMSE tomographic reconstructor for high-orders, $\boldsymbol{R}$, which estimates a phase distortion in a science direction with a

Figure 1: Schematic images of discrete measurement models.

tip/tilt/focus removal, can be given with the following criteria

$$\boldsymbol{R} = \min_{\boldsymbol{R}} \left\langle ||\boldsymbol{M_s M_\phi \phi} - \boldsymbol{R\hat{s}}||^2 \right\rangle, \tag{3}$$

where $\boldsymbol{M_\phi}$ and $\boldsymbol{W_\phi}$ are matrices removing tip/tilt/focus and masking for phases, respectively, and $\boldsymbol{\phi}$ is a phase distortion on a square pupil in a science direction. The estimated phase is $\hat{\boldsymbol{\phi}} = \boldsymbol{M_s M_\phi \phi}$. The final expression of $\boldsymbol{R}$ is written as

$$\boldsymbol{R} = \boldsymbol{M_\phi W_\phi} \left\langle \boldsymbol{\phi s}^T \right\rangle \boldsymbol{W_s^T M_s^T} \left\{ \boldsymbol{M_s W_s} \left(\left\langle \boldsymbol{ss}^T \right\rangle + \left\langle \boldsymbol{\eta\eta}^T \right\rangle \right) \boldsymbol{W_s^T M_s^T} \right\}^{-1}, \tag{4}$$

where $\left\langle \boldsymbol{\phi s}^T \right\rangle$, $\left\langle \boldsymbol{ss}^T \right\rangle$ and $\left\langle \boldsymbol{\eta\eta}^T \right\rangle$ represent phase-slope, slope-slope and noise covariance matrices, respectively, and these matrices are computed in an off-line process. Then, the phase in a science direction is estimated as $\hat{\boldsymbol{\phi}} = \boldsymbol{R\hat{s}}$. Instead of computing $\boldsymbol{R}$ explicitly, $\hat{\boldsymbol{\phi}}$ is usually estimated with iteratively to reduce the computational burden for inverting covariance matrices in the off-line process, as

$$\boldsymbol{M_s W_s} \left(\left\langle \boldsymbol{ss}^T \right\rangle + \left\langle \boldsymbol{\eta\eta}^T \right\rangle \right) \boldsymbol{v} = \boldsymbol{\hat{s}} \tag{5}$$

$$\hat{\boldsymbol{\phi}} = \boldsymbol{M_\phi W_\phi} \left\langle \boldsymbol{\phi s}^T \right\rangle \boldsymbol{v}. \tag{6}$$

In the first step, a linear inverse problem in Eq.(5) is solved iteratively to estimate $\boldsymbol{v}$, and, then, $\hat{\boldsymbol{\phi}}$ are computed by Eq.(6) from $\boldsymbol{v}$.

1.2 Covariance Matrix and Measurement Models

The slope-slope covariance, $\left\langle \boldsymbol{ss}^T \right\rangle$, consists of $2N_{gs} \times 2N_{gs}$ 2-level Recursive Block Toeplitz (2RBT) matrices [1] in cases including only NGSs or only LGSs (not for a NGSs-LGSs mixed case, like MOSAIC), where N_{gs} is the number of GSs. In a 2RBT matrix, only limited number of elements have an unique value and the rest of the elements can be expressed by these unique values. It means that we need to compute only the unique values to get the covariance matrices.

In order to compute slope-slope covariance matrices theoretically, we have to define a measurement model. Here, we assume a Shack-Hartmann WFS (SH-WFS). A measured slope from a sub-aperture of a SH-WFS corresponds to an averaged phase gradient over a sub-aperture,

$$s_{i,j}^x = \frac{1}{S_{\text{sub}}} \int_{-d_{\text{sub}}/2}^{d_{\text{sub}}/2} \int_{-d_{\text{sub}}/2}^{d_{\text{sub}}/2} \frac{\partial \phi(x_{i,j}+x, y_{i,j}+y)}{\partial x} dxdy \tag{7}$$

$$s_{i,j}^y = \frac{1}{S_{\text{sub}}} \int_{-d_{\text{sub}}/2}^{d_{\text{sub}}/2} \int_{-d_{\text{sub}}/2}^{d_{\text{sub}}/2} \frac{\partial \phi(x_{i,j}+x, y_{i,j}+y)}{\partial y} dxdy \tag{8}$$

where d_{sub} is sub-aperture size, $S_{\mathrm{sub}} = d_{\mathrm{sub}}^2$ is area of a sub-aperture and $x_{i,j}$ and $y_{i,j}$ is a coordinate of the center of a sub-aperture indexed by (i, j). The slope-slope covariance with this model in Eq.(7) and Eq.(8) are given using the Fourier transform [1]. Hereinafter, we call this model as the FFT model.

In the *slopesLinearMMSE* class, two different measurement models are also implemented to reduce a computational time for creating covariance matrices in the off-line process. The first one is a Hudgin-like discrete model and the other is a Fried discrete model. In the Hudgin-like model, a measurement is defined by a phase difference between 2 discrete points, as shown in Fig.1, as

$$s_{i,j}^x = \frac{1}{d_{\mathrm{sub}}}\left[\phi\left(x_{i,j} + \frac{d_{\mathrm{sub}}}{2}, y_{i,j}\right) - \phi\left(x_{i,j} - \frac{d_{\mathrm{sub}}}{2}, y_{i,j}\right)\right] \tag{9}$$

$$s_{i,j}^y = \frac{1}{d_{\mathrm{sub}}}\left[\phi\left(x_{i,j}, y_{i,j} + \frac{d_{\mathrm{sub}}}{2}\right) - \phi\left(x_{i,j}, y_{i,j} - \frac{d_{\mathrm{sub}}}{2}\right)\right]. \tag{10}$$

In the Fried model, a measurement is defined by a phase difference between averages of 2 edge points on the left and right side of sub-aperture, as shown in Fig.1, as

$$\begin{aligned} s_{i,j}^x = \frac{0.5}{d_{\mathrm{sub}}}\Bigg[&\phi\left(x_{i,j} + \frac{d_{\mathrm{sub}}}{2}, y_{i,j} + \frac{d_{\mathrm{sub}}}{2}\right) + \phi\left(x_{i,j} + \frac{d_{\mathrm{sub}}}{2}, y_{i,j} - \frac{d_{\mathrm{sub}}}{2}\right) \\ &-\phi\left(x_{i,j} - \frac{d_{\mathrm{sub}}}{2}, y_{i,j} + \frac{d_{\mathrm{sub}}}{2}\right) - \phi\left(x_{i,j} - \frac{d_{\mathrm{sub}}}{2}, y_{i,j} - \frac{d_{\mathrm{sub}}}{2}\right)\Bigg] \end{aligned} \tag{11}$$

$$\begin{aligned} s_{i,j}^y = \frac{0.5}{d_{\mathrm{sub}}}\Bigg[&\phi\left(x_{i,j} + \frac{d_{\mathrm{sub}}}{2}, y_{i,j} + \frac{d_{\mathrm{sub}}}{2}\right) + \phi\left(x_{i,j} - \frac{d_{\mathrm{sub}}}{2}, y_{i,j} + \frac{d_{\mathrm{sub}}}{2}\right) \\ &-\phi\left(x_{i,j} + \frac{d_{\mathrm{sub}}}{2}, y_{i,j} - \frac{d_{\mathrm{sub}}}{2}\right) - \phi\left(x_{i,j} - \frac{d_{\mathrm{sub}}}{2}, y_{i,j} - \frac{d_{\mathrm{sub}}}{2}\right)\Bigg]. \end{aligned} \tag{12}$$

The slope-slope covariance for the both discrete model can be computed by the combination of the phase structure function, D_ϕ, so much faster than the FFT model.

In the case with only NGSs, the phase-slope covariance matrix consists of $2N_{gs}$ 2RBT matrices for a system. In a case with LGSs, however, a pupil size of a SH-WFS becomes smaller with altitude due to the cone effect of LGS, and hence, a spatial sampling of slope changes as well. On the other hand, a spatial sampling of phase in a science direction doesn't change since the science target is at an infinite altitude. This difference in spatial sampling between slopes from LGSs and phases in a science direction breaks the Toeplitz structure in the phase-slope covariance, except for a ground layer. In order to keep the Toeplitz structure in the phase-slope covariance, a phase-slope covariance should be computed for each layer with a same spatial sampling as slope for the phase grid, as shown in the

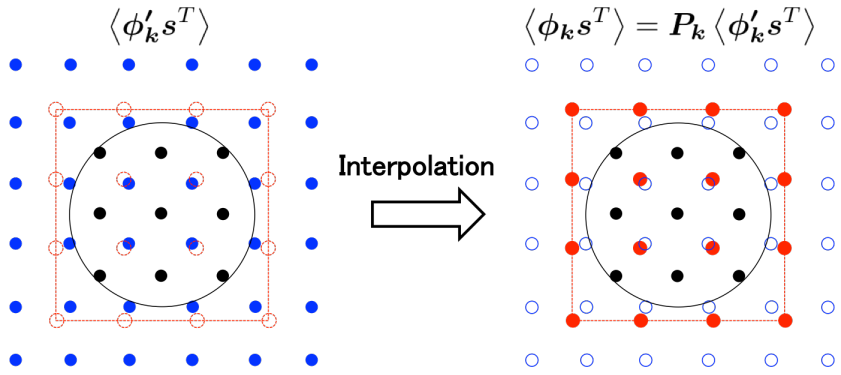

Figure 2: Schematic images of computation for a phase-slope covariance matrix. The black points represents points defining slopes and the red points are for phases. The blue points represents a resampled phase grid with the same spatial sampling as one for slopes.

left panel of Fig.2, and, then, the phase at the actual sampling is computed by an interpolation from *resampled* phase-slope covariance. Therefore, Eq.(6) can be rewritten as

$$\hat{\boldsymbol{\phi}} = \boldsymbol{M}_{\boldsymbol{\phi}} \boldsymbol{W}_{\boldsymbol{\phi}} \boldsymbol{P} \begin{bmatrix} \left\langle \boldsymbol{\phi}'_1 \boldsymbol{s}^T \right\rangle \\ \left\langle \boldsymbol{\phi}'_2 \boldsymbol{s}^T \right\rangle \\ \vdots \\ \left\langle \boldsymbol{\phi}'_{N_l} \boldsymbol{s}^T \right\rangle \end{bmatrix} \boldsymbol{v}. \tag{13}$$

where $\left\langle \boldsymbol{\phi}'_k \boldsymbol{s}^T \right\rangle$ is a resampled 2RBT phase-slope covariance matrix for the k^{th} layer and $\boldsymbol{P}$ is a bi-linear interpolation matrix. Each $\left\langle \boldsymbol{\phi}'_k \boldsymbol{s}^T \right\rangle$ has $2N_{gs}$ 2RBT matrices.

1.3 Matrix-Vector Multiplication

Generally, in a solver for an linear inverse problem, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, we have to repeat matrix-vector multiplications, $\boldsymbol{A}\boldsymbol{x}$. In the case of Eq.(5), $\boldsymbol{A} = \boldsymbol{M}_{\boldsymbol{s}} \boldsymbol{W}_{\boldsymbol{s}} \left(\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle + \left\langle \boldsymbol{\eta}\boldsymbol{\eta}^T \right\rangle \right)$, $\boldsymbol{x} = \boldsymbol{v}$ and $\boldsymbol{b} = \hat{\boldsymbol{s}}$. Since the noise covariance $\left\langle \boldsymbol{\eta}\boldsymbol{\eta}^T \right\rangle$ is a very sparse matrix, $\left\langle \boldsymbol{\eta}\boldsymbol{\eta}^T \right\rangle \boldsymbol{v}$ can be computed efficiently. Similarly, the masking matrix $\boldsymbol{W}_{\boldsymbol{s}}$ is also sparse. The tip/tilt/focus removing matrix $\boldsymbol{M}_{\boldsymbol{s}}$ is not sparse but $\boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}}$ and $\boldsymbol{T}^{-1}_{\boldsymbol{s},\boldsymbol{k}}$ in each diagonal block are not big matrix ($2n_s^k \times 3$, where n_s^k is the number of valid sub-apertures in the k^{th} SH-WFS). Therefore, matrix-vector multiplication for each diagonal block can be computed as $\boldsymbol{M}_{\boldsymbol{s},\boldsymbol{k}} \boldsymbol{u}_{\boldsymbol{k}} = \boldsymbol{u}_{\boldsymbol{k}} - \boldsymbol{T}_{\boldsymbol{s},\boldsymbol{k}} \boldsymbol{T}^{-1}_{\boldsymbol{s},\boldsymbol{k}} \boldsymbol{u}_{\boldsymbol{k}}$. The remaining part is a matrix-vector computation of $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle \boldsymbol{v}$.

As mentioned above, the slope-slope covariance consists of $2N_{gs} \times 2N_{gs}$ 2RBT matrices, and $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle \boldsymbol{v}$ can be considered as a combination of a 2RBT matrix-vector multiplication. One method is proposed to compute a 2RBT matrix-vector multiplication efficiently in [2]. Using this method allow us to compute a matrix-vector multiplication with $O(N \log N)$ instead of N^2, in which N.

Assume that $\boldsymbol{A}$ is a 2RBT matrix of size $n_{\text{sub}}^2 \times n_{\text{sub}}^2$ and $\boldsymbol{a}$ is a vector of $(2n_{\text{sub}}-1)^2$ unique elements in the 2RBT matrix $\boldsymbol{A}$, and consider to compute a matrix-vector multiplication $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, where $\boldsymbol{x}$ is an input vector of length n_{sub}^2. In the algorithm proposed in [2], this matrix-vector multiplication can be computed as the following steps.

1. Reassign $\boldsymbol{x}$ into a zero-valued vector $\boldsymbol{\beta}$ of length n_{sub}^2 according to some rules.
2. The 1-D Fourier transform of the reassigned vector $\boldsymbol{\beta}$.
$$\tilde{\boldsymbol{b}} = \mathcal{F}\boldsymbol{\beta}. \tag{14}$$
3. An element-wise vector multiplication
$$\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{a}} \cdot \tilde{\boldsymbol{b}}, \tag{15}$$
where $\tilde{\boldsymbol{a}}$ is a 1-D Fourier transform of $\boldsymbol{a}$ which is precomputed in the off-line process.
4. The 1-D inverse Fourier transform of $\tilde{\boldsymbol{c}}$
$$\boldsymbol{c} = \mathcal{F}^{-1}\tilde{\boldsymbol{c}}. \tag{16}$$
5. Reassign $\boldsymbol{c}$ into a vector $\boldsymbol{y}$ of length n_{sub}^2 according to some rules.

Then, in order to compute $\boldsymbol{y} = \left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle \boldsymbol{v}$, we have to repeat these steps for all blocks in the slope-slope covariance matrix and sum the results as

1. for $i = 1 : N_{gs}$
2. $\quad \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{x}} = \boldsymbol{y}_{\boldsymbol{j}}^{\boldsymbol{y}} = \boldsymbol{0}$
3. $\quad$ for $j = 1 : N_{gs}$

4. $\quad\quad \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{x}} = \boldsymbol{v}(\ (1 : n_{\text{sub}}^2) + 2n_{\text{sub}}^2(j-1)\ )$
5. $\quad\quad \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{y}} = \boldsymbol{v}(\ (1 + n_{\text{sub}}^2 : 2n_{\text{sub}}^2) + 2n_{\text{sub}}^2(j-1)\ )$
6. $\quad\quad \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{x}} = \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{x}} + \left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{xx} \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{x}} + \left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{xy} \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{y}}$
7. $\quad\quad \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{y}} = \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{y}} + \left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{yx} \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{x}} + \left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{yy} \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{y}}$
8. $\quad$ end
9. $\quad \boldsymbol{y}(\ (1 : n_{\text{sub}}^2) + 2n_{\text{sub}}^2(i-1)\ ) = \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{x}}$
10. $\quad \boldsymbol{y}(\ (1 + n_{\text{sub}}^2 : 2n_{\text{sub}}^2) + 2n_s(i-1)\ ) = \boldsymbol{y}_{\boldsymbol{i}}^{\boldsymbol{y}}$
11. end

where $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{k_1 k_2}$ is a k_1-k_2 slope covariance in the $(i,j)^{\text{th}}$ block matrix in $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle$. Each $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{k_1 k_2} \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{k_2}}$ is a matrix-vector multiplication with a 2RBT matrix, which requires two 1-D Fourier transforms and 1 element-wise vector product. However, we don't need to compute the 1-D Fourier transform of the reassigned $\boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{k_2}}$ every time in the loop. This can be done once before the loop, and, then, we can reuse the 1-D Fourier transform of the reassigned $\boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{k_2}}$ in the loop. In addition, since the Fourier transform is linear operation, we can sum results of the element-wise vector multiplication before the inverse Fourier transform. By taking into account these facts, the number of the Fourier transform in one $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle \boldsymbol{v}$ can be reduced to $2N_{gs}$ from $4N_{gs}^2$. Finally, the algorithm to compute $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle \boldsymbol{v}$ is written as

1. for $i = 1 : N_{gs}$
2. $\quad \boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{x}} = \text{Reassign}\left\{\boldsymbol{v}(\ (1 : n_s) + 2n_s(i-1)\ )\right\}$
3. $\quad \tilde{\boldsymbol{b}}_{\boldsymbol{i}}^{\boldsymbol{x}} = \mathcal{F}\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{x}}$
4. $\quad \boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{y}} = \text{Reassign}\left\{\boldsymbol{v}(\ (1 + n_s : 2n_s) + 2n_s(i-1)\ )\right\}$
5. $\quad \tilde{\boldsymbol{b}}_{\boldsymbol{i}}^{\boldsymbol{y}} = \mathcal{F}\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{y}}$
6. end
7. for $i = 1 : N_{gs}$
8. $\quad \tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{x}} = \tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{y}} = \boldsymbol{0}$
9. $\quad$ for $j = 1 : N_{gs}$
10. $\quad\quad \tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{x}} = \tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{x}} + \tilde{\boldsymbol{a}}_{\boldsymbol{ij}}^{\boldsymbol{xx}} \cdot \tilde{\boldsymbol{b}}_{\boldsymbol{j}}^{\boldsymbol{x}} + \tilde{\boldsymbol{a}}_{\boldsymbol{ij}}^{\boldsymbol{xy}} \cdot \tilde{\boldsymbol{b}}_{\boldsymbol{j}}^{\boldsymbol{y}}$
11. $\quad\quad \tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{y}} = \tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{y}} + \tilde{\boldsymbol{a}}_{\boldsymbol{ij}}^{\boldsymbol{yx}} \cdot \tilde{\boldsymbol{b}}_{\boldsymbol{j}}^{\boldsymbol{x}} + \tilde{\boldsymbol{a}}_{\boldsymbol{ij}}^{\boldsymbol{yy}} \cdot \tilde{\boldsymbol{b}}_{\boldsymbol{j}}^{\boldsymbol{y}}$
12. $\quad$ end
13. end
14. for $i = 1 : N_{gs}$
15. $\quad \boldsymbol{c}_{\boldsymbol{j}}^{\boldsymbol{x}} = \mathcal{F}^{-1}\tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{x}}$
16. $\quad \boldsymbol{y}(\ (1 : n_s) + 2n_s(i-1)\ ) = \text{Reassign}\{\boldsymbol{c}_{\boldsymbol{j}}^{\boldsymbol{x}}\}$
17. $\quad \boldsymbol{c}_{\boldsymbol{j}}^{\boldsymbol{y}} = \mathcal{F}^{-1}\tilde{\boldsymbol{c}}_{\boldsymbol{i}}^{\boldsymbol{y}}$
18. $\quad \boldsymbol{y}(\ (1 + n_s : 2n_s) + 2n_s(i-1)\ ) = \text{Reassign}\{\boldsymbol{c}_{\boldsymbol{j}}^{\boldsymbol{y}}\}$
19. end

where $\tilde{\boldsymbol{a}}_{\boldsymbol{ij}}^{\boldsymbol{k_1 k_2}}$ is the Fourier transform of the unique elements of $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle_{i,j}^{k_1 k_2}$.

For the matrix-vector multiplication for $\left\langle \boldsymbol{\phi}_k' \boldsymbol{s}^T \right\rangle$ in Eq.(13), the Fourier transform should be done for each layer because each $\left\langle \boldsymbol{\phi}_k' \boldsymbol{s}^T \right\rangle$ has a different size depending on altitude, which means each $\boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{k}}$ is reassigned into a vector of a different length. Therefore, the required number of the 1-D Fourier transform in one $\left\langle \boldsymbol{s}\boldsymbol{s}^T \right\rangle \boldsymbol{v}$ is $4N_{gs}N_l$.

2 Usage

2.1 Input

Slope from SH-WFSs in unit of pixel.

2.2 Output

Phase distortion at a pupil-plane for science direction(s) in unit of m.

2.3 Parameters

Parameter	Status	Description
wfs	Required	*shackHartmann* object
tel	Required	*telescope* object
atmModel	Required	*atmosphere* object
guideStar	Required	*source* object for guide star(s)
mmseStar	Optional	*source* object for science target(s). If you don't set *mmseStar*, the direction of science star(s) are set to the same direction(s) as guide star(s).
noiseVar	Optional	Noise covariance matrix. This parameter can be • a scalar : a diagonal noise covariance matrix with a same noise variance for all measurements • a vector : a diagonal noise covariance matrix with different noise variances for measurements • a matrix The default value is set to 0. The unit of the noise variance should be square of angle of radian [rad^2].
slopesMask	Optional	Logical matrix to define pupil masks for slopes. The size of the matrix should be $2n_{\text{sub}}^2 \times N_{gs}$, and the k^{th} column includes vectorized masks for x and y slopes in the k^{th} SH-WFS. You can define a different mask for each SH-WFSs. If you don't set this parameter, the mask is defined by *validLenslet* of *shackHartmann* object.
covModel	Optional	Model to compute theoretical covariance matrices (see 1.2). There are 3 models, • *'FFT'* : realistic FFT model. • *'Hudgin'* : discrete Hudgin-like 2 points model. • *'Fried'* : discrete Fried 4 points model. The default value is set to *'FFT'*.
isTTRM	Optional	Logical flag to define whether you remove tip/tilt modes from reconstruction. If you set this parameter to true, tip/tilt modes are automatically removed from measured slopes before reconstruction. The tip/tilt modes are removed also from an estimated wavefront. The default value is false.
isFocusRM	Optional	Logical flag to define whether you remove a focus mode from reconstruction. The default value is false.
RTOL	Optional	Relative tolerance for an iterative method. The default value is 10^{-3}.
MAXIT	Optional	Maximum number of iteration for an iterative method. The default value is 30.

Table 1: Parameters for *slopesLinearMMSE* class.

Parameter	Status	Description
isWarmStart	Optional	Logical flag to define whether you use the warm start method. In the warm star method, the reconstructor uses an estimate at a previous frame as an initial for an iterative method to reduce the required number of iteration. The default is true.
tPredict	Optional	Time lag in seconds for a simple prediction. The input value should be non-negative. The default is 0, which means no prediction.
tMulti	Optional	Time delay in seconds for a multi-time step reconstruction []. The input value should be non-negative scalar or vector, like [25 50]. The default is 0, which means no multi-time step reconstruction.
NF	Optional	Tuning parameter for the FFT model. The default is 1024.
sf	Optional	Tuning parameter for the FFT model. The default is 4.

Table 2: Parameters for *slopesLinearMMSE* class.

2.4 Object construction

Listing 1: Full construction

```
R = slopesLinearMMSE(wfs,... % shackHartmann object
    tel,... % telescope object
    atmModel,... % atmosphere object
    guideStar,... % source object for guide star(s)
    'mmseStar',science,... % source object for science target(s)
    'noiseVar', noiseVar,... % noise covariance matrix
    'slopesMask', slopesMask,... % mask for slopes
    'covModel', 'Hudgin',... % model to compute covariance matrices
    'isTTRM', false,... % Flag for removing tip/tilt from a reconstructor
    'isFocusRM', false,... % Flag for removing focus from a reconstructor
    'RTOL', 1e-3,... % Relative tolerance for an iterative method
    'MAXIT', 30,... % Maximum number of iteration for an iterative method
    'isWarmStart', true,... % Flag for using warm start method
    'tPredict', 0,... % time lag for a prediction
    'tMulti', 0,... % time delay for a multi-time step reconstruction
    'NF', 1024,... % a tuning parameter for the FFT model
    'sf', 4); % a tuning parameter for the FFT model
```

2.5 Example for construction and reconstruction

2.5.1 Open loop

Listing 2: Open loop

```
%% build reconstructor
R = slopesLinearMMSE(wfs,... % shackHartmann object
    tel,... % telescope object
    atm,... % atmosphere object
    ast,... % source object for guide star(s)
    'mmseStar',sci,... % source object for science target(s)
    'noiseVar', noiseVar); % noise covariance matrix

%% AO loop
```

```
for k=1:nIteration

    % Atmosphere update
    +tel;

    % propagation
    sci = sci.*tel*dm*cam;
    ast = ast.*tel*wfs;

    % reconstruction
    est = R*wfs.slope;

    % Interaction matrix & update dm
    dm.coefs = iF*est; % iF is a DM influence matrix

end
```

2.5.2 Pseudo open loop

Listing 3: Pseudo open loop

```
%% build reconstructor
R = slopesLinearMMSE(wfs,... % shackHartmann object
    tel,... % telescope object
    atm,... % atmosphere object
    ast,... % source object for guide star(s)
    'mmseStar',sci,... % source object for science target(s)
    'noiseVar', noiseVar); % noise covariance matrix

%% AO loop
for k=1:nIteration

    % Atmosphere update
    +tel;

    % propagation
    sci = sci.*tel*dm*cam;
    ast = ast.*tel*dm*wfs;

    % compute pseudo open loop slope
    polslope = bsxfun(@minus, wfs.slopes, commandMatrix*dm.coefs )

    % reconstruction
    est = R*polslope;

    % Interaction matrix
    coefs = iF*est;

    % update dm
    dm.coefs = (1-gain_pol)*dm.coefs+gain_pol*iF*coefs;

```

```
31 end
```

reference

References

[1] R. Conan, "Fast iterative optimal estimation of turbulence wavefronts with recursive block Toeplitz covariance matrix," Proc. SPIE **9148**, 91480R (2014).

[2] D. Lee, "Fast multiplication of a recursive block toeplitz matrix by a vector and its application," Journal of Complexity **2**, 295–305 (1968).