Robust wavefront sensing in harsh turbulence conditions

Michel Tallon[1], Eric Thiébaut[1], Maud Langlois[1], Bernard Gelly[2], Richard Douet[2], Clémentine Béchet[3], Loïc Denis[4]

[1] *CRAL, Lyon France*

[2] *THEMIS, La Laguna, Canary Islands*

[3] *Pontificia Universidad Catolica de Chile*

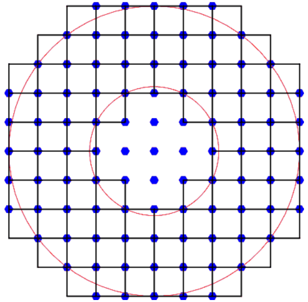
[4] *Laboratoire Hubert Curien, Saint-Etienne, France*

Outline

- Robust WFS → extract slopes and covariance of slopes in real time
- Contents
  - Why ? / context
  - How ? / recipes
    - errors on pixel values
    - errors on slope measurements
  - Real-time slope covariances & control algorithm
  - Conclusion

Context / data from Themis Solar telescope

Shack-Hartmann wavefront sensor at the telescope

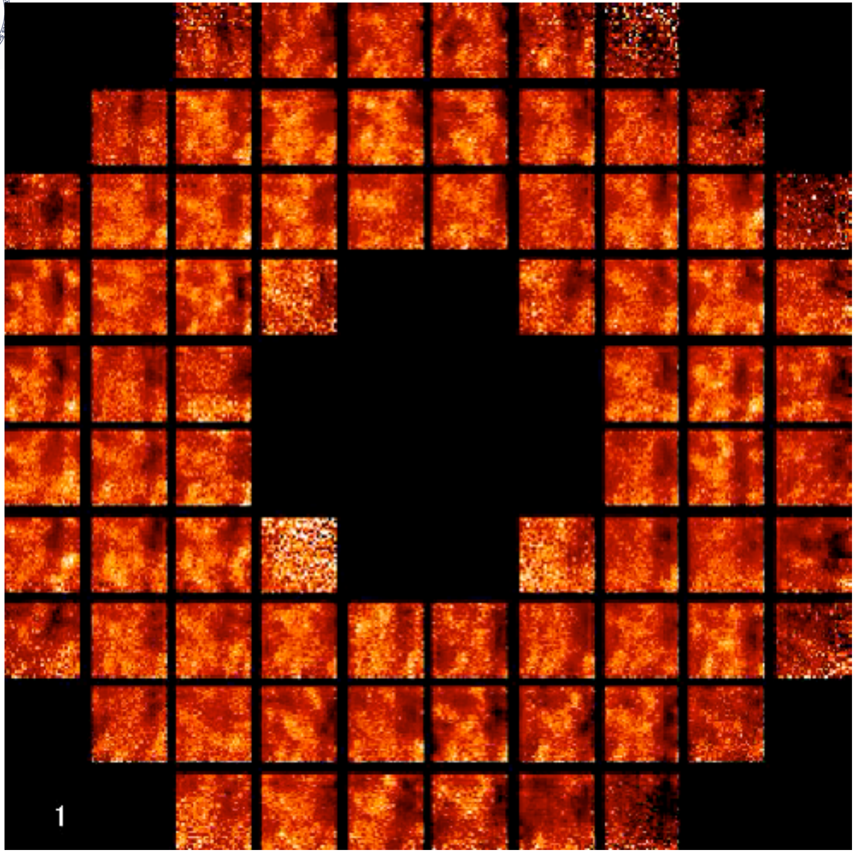


- 76 subapertures (10x10)
- 97 actuators (11x11, ALPAO)
- 1 kHz

- Constraints
  - low cost
  - AO runs unsupervised
- Opportunity to implement new methods of reconstruction and control
  - RTC = standard PC + suitable software
  - Do the best we can do

- Ø 90 cm Solar telescope
- Tenerife, Canaria Islands
- Altitude 2400m

Context / harsh conditions …

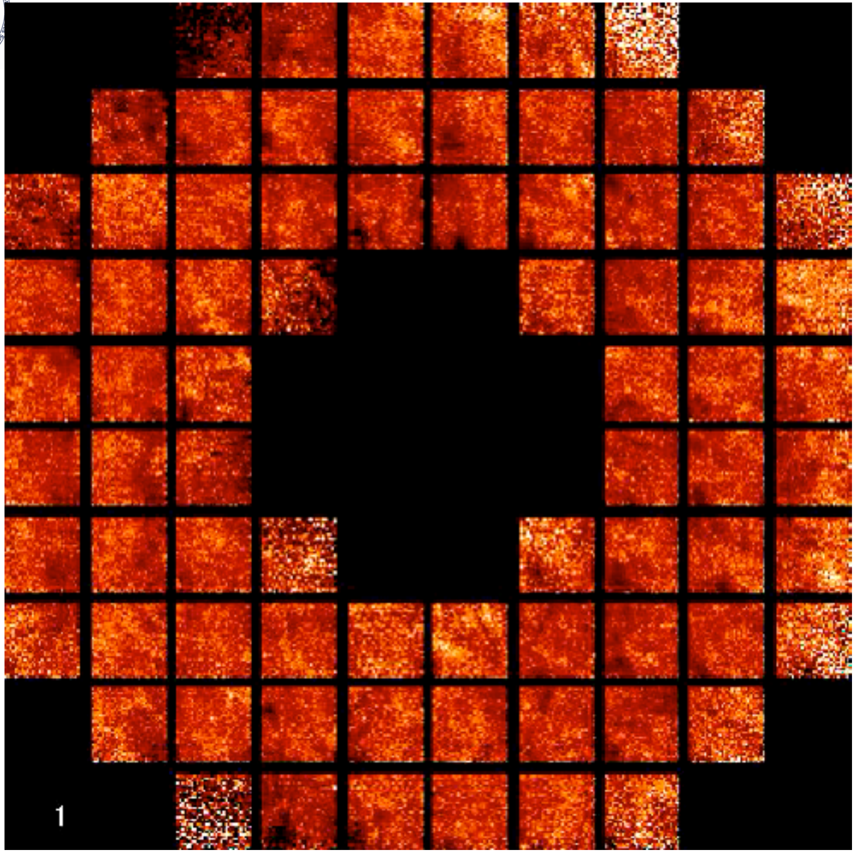


- AO in the visible
- Day time (median r_0 : 4.6 cm)
- Sensing on granulation ~ 2% contrast
- Fast spatial and temporal variations in the pupil

- structures in the sub-images
  => accuracy not isotropic
  - similar to laser guide star elongation
  - structures change at the minute scale

- Field-of-view 10" / 30 pix
- here : 100 frames @ 1 kHz

Context / harsh conditions ...



- AO runs unsupervised
  => "robustness"
- key point: get the AO system informed
  – errors on the WFS measurements
- Field-of-view 10" / 30 pix
- here : 100 frames @ 1 kHz

Context / recipe

- Focus on robustness
  - promptly adapt to varying conditions
  - unsupervised + best for any conditions => auto-calibration
  - adapt to evolutions in the system
    - interaction matrix (differential pointing)
    - decentering of the pupil (derotator)

- Recipe
  - estimate the errors on the pixel values
  - estimate the errors (and their covariance) on slopes
  - take them into account in the computation of the commands.

- Presentation focused on wavefront sensing.

Detector preprocessing / pixel values

- Model (*fixed exposure time*)

$$r_i = \frac{t_i\,\phi_i}{g_i} + b_i + n_i$$

- r_i raw pixel i [ADU]
- t_i total transmitance [e⁻/ph]
- ϕ_i photons
- g_i gain [e⁻/ADU]
- b_i bias [ADU]
- n_i noise [ADU]

- 3 types of calibration ***set of*** frames
  - *dark:* $\phi_i = 0$ no incident flux
  - *flat:* $\mathrm{E}(\phi_i) = \mathrm{E}(\phi^{\text{flat}})$ same average flux on all the pixels
  - *static:* $\mathrm{Var}(t_i\,\phi_i) = \mathrm{E}(t_i\,\phi_i)$ constant flux on each pixel

- Calibrated pixel values

$$d_i = \alpha_i (r_i - \beta_i)$$

$$\mathbb{E}(d_i) = \mathbb{E}(\phi_i)/\mathbb{E}(\phi^{\text{flat}})$$

=>

$$\beta_i = \mathbb{E}(r_i^{\text{dark}})$$

$$\alpha_i = \frac{1}{\mathbb{E}(r_i^{\text{flat}}) - \mathbb{E}(r_i^{\text{dark}})}$$

Detector preprocessing / error on pixel values

- Model (*fixed exposure time*)

$$r_i = \frac{t_i\,\phi_i}{g_i} + b_i + n_i$$

- r_i raw pixel i [ADU]
- t_i total transmitance [e⁻/ph]
- ϕ_i photons
- g_i gain [e⁻/ADU]
- b_i bias [ADU]
- n_i noise [ADU]

- 3 types of calibration ***set of*** frames
  - *dark:* $\phi_i = 0$ no incident flux
  - *flat:* $\mathrm{E}(\phi_i) = \mathrm{E}(\phi^{\text{flat}})$ same average flux on all the pixels
  - *static:* $\mathrm{Var}(t_i\,\phi_i) = \mathrm{E}(t_i\,\phi_i)$ constant flux on each pixel

- Error on pixel values

$$\mathrm{Var}(d_i) \approx \frac{\max(d_i, 0) + v_i}{u_i} \quad \Rightarrow$$

$$u_i = g_i\left(\mathbb{E}(r_i^{\text{flat}}) - \mathbb{E}(r_i^{\text{dark}})\right)$$

$$v_i = g_i\,\frac{\mathrm{Var}(r_i^{\text{dark}})}{\mathbb{E}(r_i^{\text{flat}}) - \mathbb{E}(r_i^{\text{dark}})}$$

$$g_i = \frac{\mathbb{E}(r_i^{\text{stat}}) - \mathbb{E}(r_i^{\text{dark}})}{\mathrm{Var}(r_i^{\text{stat}}) - \mathrm{Var}(r_i^{\text{dark}})}$$

Slopes with their covariances / Ingredients

- Cost function (from *maximum likelihood*)

$$\psi(\boldsymbol{\theta}) = \sum_{k=1}^{n} \left\{ \eta_k \, \|\boldsymbol{d}_k - \alpha_k \, \mathbf{R}_k(\boldsymbol{s}_k).\boldsymbol{r}\|^2_{\mathbf{W}_k} - m_k \, \log \eta_k \right\} + \mu \, \|\mathbf{D}.\boldsymbol{r}\|^2_2$$

- Fit the reference image (model), shifted and rescaled on sub-images
  - Use the weights on the pixels: $\mathbf{W}_k$

- At the same time:
  - Rescales the weights (model is not perfect): η_k
  - Get the best reference image: $\boldsymbol{r}$
  - Smooth (extrapolated) reference image: $\mu \, \|\mathbf{D}.\boldsymbol{r}\|^2_2$

- $\boldsymbol{d}_k$ pixels of subap k
- $\boldsymbol{r}$ reference image
- $\mathbf{R}_k$ shifting operator
- $\boldsymbol{s}_k$ slopes
- α_k rescaling (scintillation)
- $\mathbf{W}_k$ weights on pixels
- η_k rescaling of weights
- m_k number of pixels
- $\mathbf{D}$ finite difference operator
- μ weight on smoothing

Slopes with their covariances / recipe step #1

- Cost function (from *maximum likelihood*)

$$\psi(\boldsymbol{\theta}) = \sum_{k=1}^{n} \left\{ \eta_k \, \|\boldsymbol{d}_k - \alpha_k \, \mathbf{R}_k(\boldsymbol{s}_k).\boldsymbol{r}\|^2_{\mathbf{W}_k} - m_k \, \log \eta_k \right\} + \mu \, \|\mathbf{D}.\boldsymbol{r}\|_2^2$$

- $\boldsymbol{d}_k$ pixels of subap k
- $\boldsymbol{r}$ reference image
- $\mathbf{R}_k$ shifting operator
- $\boldsymbol{s}_k$ slopes
- α_k rescaling (scintillation)
- $\mathbf{W}_k$ weights on pixels
- η_k rescaling of weights
- m_k number of pixels
- $\mathbf{D}$ finite difference operator
- μ weight on smoothing

- **Linearization** (Thiébaut et al 2018)
  - ≈ matched filter
  - Each sub-aperture k independently

$$\|\boldsymbol{d}_k - \alpha_k \, \mathbf{R}_k(\boldsymbol{s}_k).\boldsymbol{r}\|^2_{\mathbf{W}_k}$$

$$\varphi_k(\boldsymbol{u}_k) = \|\boldsymbol{d}_k - \mathbf{H}_k.\boldsymbol{u}_k\|^2_{\mathbf{W}_k} = \gamma_k - 2\, \boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{u}_k + \boldsymbol{u}_k^{\mathrm{T}} \mathbf{A}_k.\boldsymbol{u}_k,$$

with $\boldsymbol{u}_k = \alpha_k \begin{pmatrix} 1 \\ x_k \\ y_k \end{pmatrix}$

$$\begin{cases} \gamma_k = \|\boldsymbol{d}_k\|^2_{\mathbf{W}_k} \\ \boldsymbol{b}_k = \mathbf{H}_k^{\mathrm{T}}.\mathbf{W}_k.\boldsymbol{d}_k \\ \mathbf{A}_k = \mathbf{H}_k^{\mathrm{T}}.\mathbf{W}_k.\mathbf{H}_k \end{cases}$$

=> $\widehat{\boldsymbol{u}}_k = \mathbf{A}_k^{-1}.\boldsymbol{b}_k$ → (x_k, y_k)

$$\varphi_k(\widehat{\boldsymbol{u}}_k) = \gamma_k - \boldsymbol{b}_k^{\mathrm{T}}.\widehat{\boldsymbol{u}}_k$$

Slopes with their covariances / recipe step #2

- Cost function (from *maximum likelihood*)

$$\psi(\boldsymbol{\theta}) = \sum_{k=1}^{n} \left\{ \eta_k \underbrace{\|\boldsymbol{d}_k - \alpha_k \, \mathbf{R}_k(\boldsymbol{s}_k).\boldsymbol{r}\|^2_{\mathbf{W}_k}}_{= \varphi_k(\widehat{\boldsymbol{u}}_k)} - m_k \, \log \eta_k \right\} + \mu \, \|\mathbf{D}.\boldsymbol{r}\|_2^2$$

- $\boldsymbol{d}_k$ pixels of subap k
- $\boldsymbol{r}$ reference image
- $\mathbf{R}_k$ shifting operator
- $\boldsymbol{s}_k$ slopes
- α_k rescaling (scintillation)
- $\mathbf{W}_k$ weights on pixels
- η_k rescaling of weights
- m_k number of pixels
- $\mathbf{D}$ finite difference operator
- μ weight on smoothing

- **Rescaling of weights**
  – Inform on discrepancy between data and reference image

$$\eta_k = \frac{m_k}{\varphi_k(\widehat{\boldsymbol{u}}_k)}$$

Slopes with their covariances / recipe step #3

- Cost function (from *maximum likelihood*)

$$\psi(\boldsymbol{\theta}) = \sum_{k=1}^{n} \left\{ \eta_k \, \|\boldsymbol{d}_k - \alpha_k \, \mathbf{R}_k(\boldsymbol{s}_k).\boldsymbol{r}\|^2_{\mathbf{W}_k} - m_k \, \log \eta_k \right\} + \mu \, \|\mathbf{D}.\boldsymbol{r}\|_2^2$$

- $\boldsymbol{d}_k$ pixels of subap k
- $\boldsymbol{r}$ reference image
- $\mathbf{R}_k$ shifting operator
- $\boldsymbol{s}_k$ slopes
- α_k rescaling (scintillation)
- $\mathbf{W}_k$ weights on pixels
- η_k rescaling of weights
- m_k number of pixels
- $\mathbf{D}$ finite difference operator
- μ weight on smoothing

- **Covariance matrix of slopes** (α_k, x_k, y_k)

$$\mathbf{C}_k = \frac{1}{\eta_k} \mathbf{J}_k^{\mathrm{T}}.\mathbf{A}_k^{-1}.\mathbf{J}_k$$

for each subaperture *k*

  – $\mathbf{J}_k$: jacobian of non-linear relationship between $\boldsymbol{u}_k$ and (α_k, x_k, y_k)

=> send to controller now

Slopes with their covariances / recipe step #4

- Cost function (from *maximum likelihood*)

$$\psi(\theta) = \sum_{k=1}^{n} \left\{ \eta_k \, \|\boldsymbol{d}_k - \alpha_k \, \mathbf{R}_k(\boldsymbol{s}_k).\boldsymbol{r}\|^2_{\mathbf{W}_k} - m_k \, \log \eta_k \right\} + \mu \, \|\mathbf{D}.\boldsymbol{r}\|_2^2$$

- **Get / update the reference image**

$$\psi'(\theta) = \sum_{k=1}^{n} \|\boldsymbol{d}_k - \mathbf{G}_k.\boldsymbol{r}\|^2_{\eta_k \mathbf{W}_k} + \mu \, \|\mathbf{D}.\boldsymbol{r}\|_2^2$$

  – Solution obtained by conjugate gradient method with :

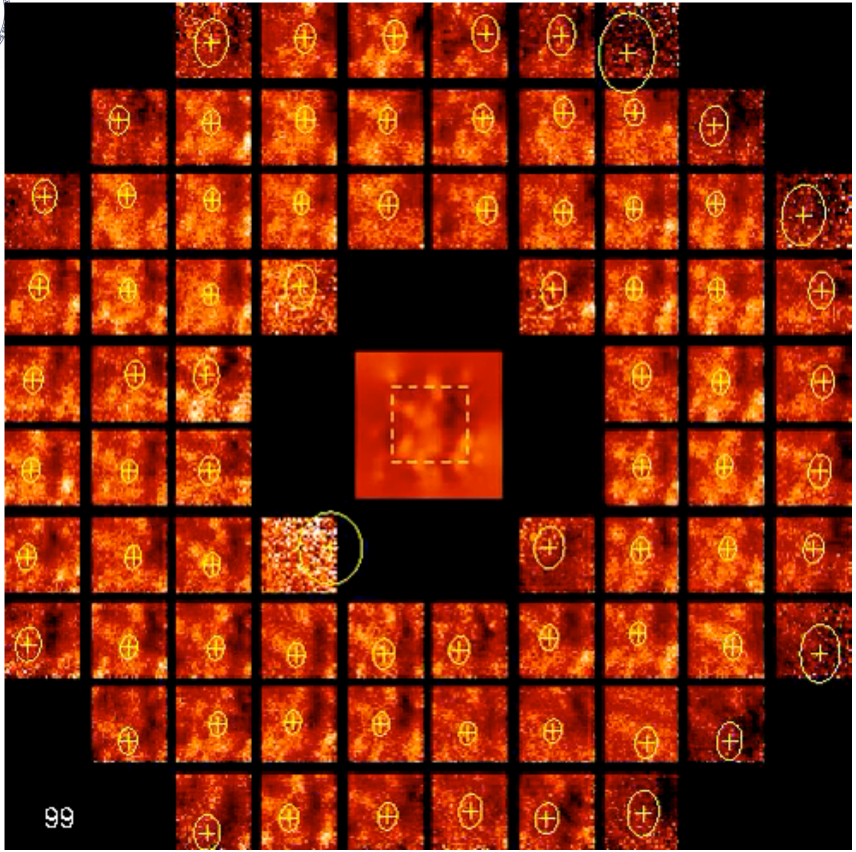
$$\left(\sum_{k=1}^{n} \eta_k \, \mathbf{G}_k^{\mathrm{T}}.\mathbf{W}_k.\mathbf{G}_k + \mu \, \mathbf{D}^{\mathrm{T}}.\mathbf{D} \right) \boldsymbol{r} = \sum_{k=1}^{n} \eta_k \, \mathbf{G}_k^{\mathrm{T}}.\mathbf{W}_k.\boldsymbol{d}_k$$

weighted sum of recentered sub-images

→ ***r*** used for updating reference frame for next frame

- $\boldsymbol{d}_k$ pixels of subap k
- $\boldsymbol{r}$ reference image
- $\mathbf{R}_k$ shifting operator
- $\boldsymbol{s}_k$ slopes
- α_k rescaling (scintillation)
- $\mathbf{W}_k$ weights on pixels
- η_k rescaling of weights
- m_k number of pixels
- $\mathbf{D}$ finite difference operator
- μ weight on smoothing

Results on open-loop data

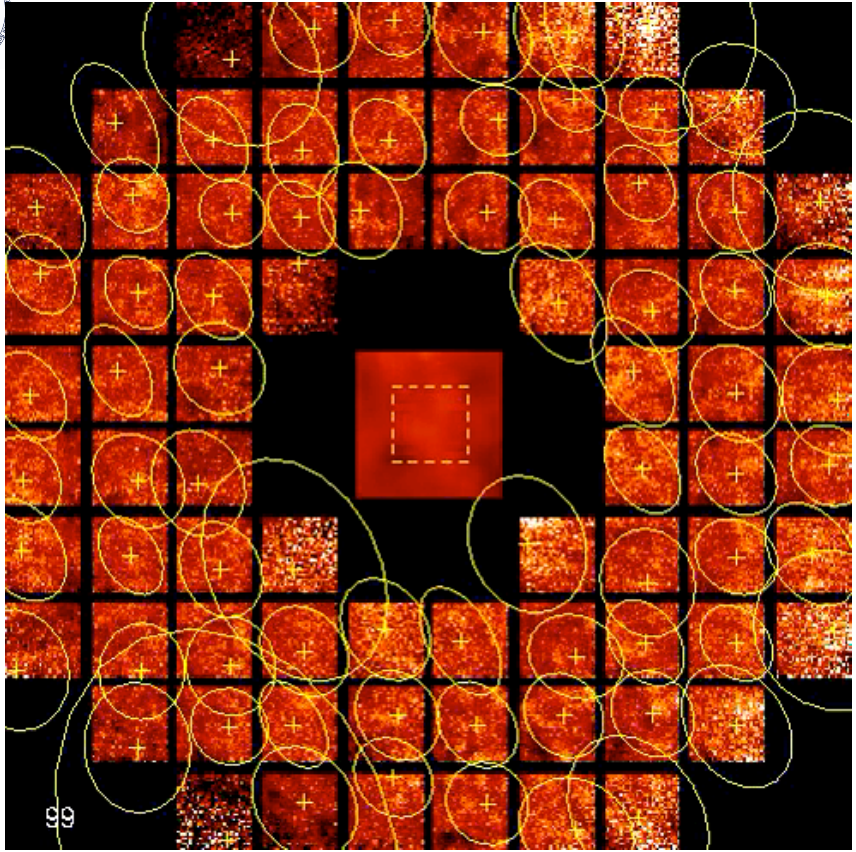


- crosses = x_k, y_k
- ellipse radii = 1 σ
- center = reference image

- comments
  - x_k more accurate than y_k
  - errors larger in subapertures on the edge
  - reference image gets known outside the FoV

- Field-of-view 10" / 30 pix
- here : 100 frames @ 1 kHz

Results on open-loop data



- crosses = x_k, y_k
- ellipse radii = 1 σ
- center = reference image

- comments
  - best accuracy now along the first diagonal
  - errors larger in subapertures on the edge
  - reference image gets known outside the FoV

- Field-of-view 10" / 30 pix
- here : 100 frames @ 1 kHz

Real-time slope covariances → controller

- Usual way to get the commands from the slopes
  - model of the system:

interaction matrix, actuators, slopes, errors

$$\boldsymbol{s} = \mathbf{M}.\boldsymbol{a} + \boldsymbol{e}$$

  - look for: $\widehat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \|\boldsymbol{s} - \mathbf{M}.\boldsymbol{a}\|^2_{\mathbf{C}_e^{-1}}$

=> $\widehat{\boldsymbol{a}} = \left(\mathbf{M}^{\mathrm{T}}.\mathbf{C}_{\boldsymbol{e}}^{-1}.\mathbf{M}\right)^{\dagger}.\mathbf{M}^{\mathrm{T}}.\mathbf{C}_{\boldsymbol{e}}^{-1}.\boldsymbol{s}$ => need to (pseudo) invert $\left(\mathbf{M}^{\mathrm{T}}.\mathbf{C}_{\boldsymbol{e}}^{-1}.\mathbf{M}\right)$ *at each frame*

- Instead: iterative method (*conjugate gradient*, e.g. *Fractal Iterative Method*)

$$\left(\mathbf{M}^{\mathrm{T}}.\mathbf{C}_{\boldsymbol{e}}^{-1}.\mathbf{M} + \mu\, \mathbf{C}_{\boldsymbol{a}}^{-1}\right) \widehat{\boldsymbol{a}} = \mathbf{M}^{\mathrm{T}}.\mathbf{C}_{\boldsymbol{e}}^{-1}.\boldsymbol{s}$$

Conclusions

- Errors on pixel values ➔ covariance of slope errors ➔ iterative reconstructor
  - on-going work…

- Can be used with laser guide star elongation
  - actual covariances instead of modeled ones
  - actual (evolving, truncated) Sodium profile
  - Sodium profile extrapolated (i.e. known outside the truncated field-of-view)

- Other example in CANARY data: