



# Fast Inversion of Hyperspectral Observations of Green House Gases (GHG) using Gaussian Locally Linear Mapping (GLiMM)

**SFTP 2025 – Grasse**

Eric Tatulli, Silvere Gousset, Sylvain Doute, Luc Meyer

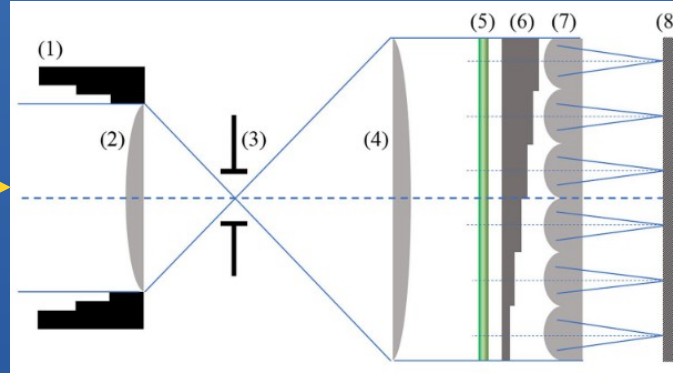
IPAG, UGA/CNRS

# Nanocarb: principles

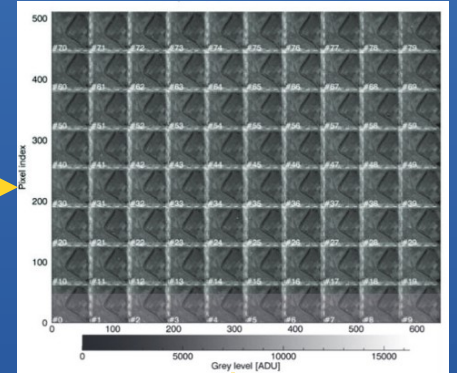
Scenery



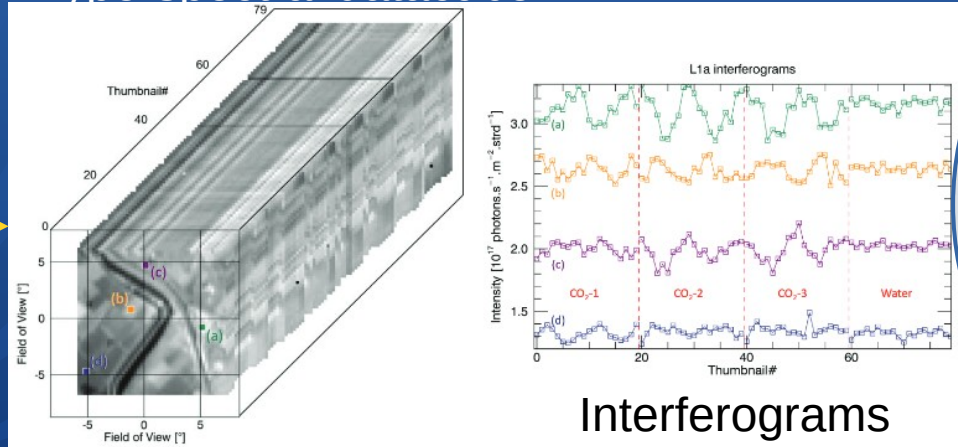
Hyperspectral Instrument



Thumbnails



Hyperspectral datacube



LM iterative  
algorithm

Atmospheric  
Inversion

**GLIMM!**

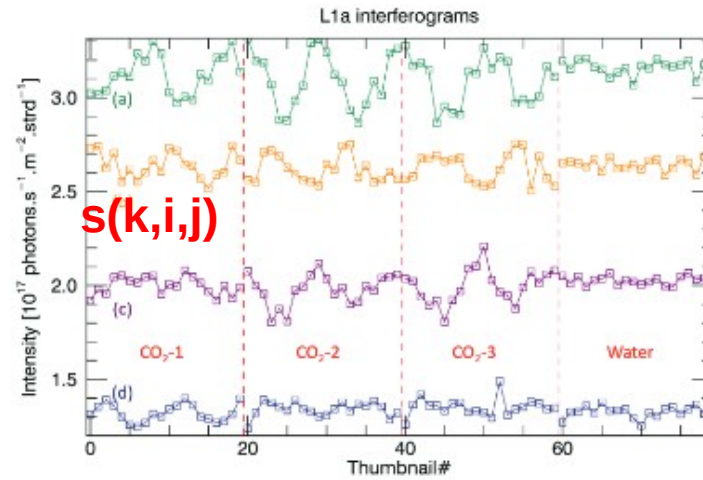
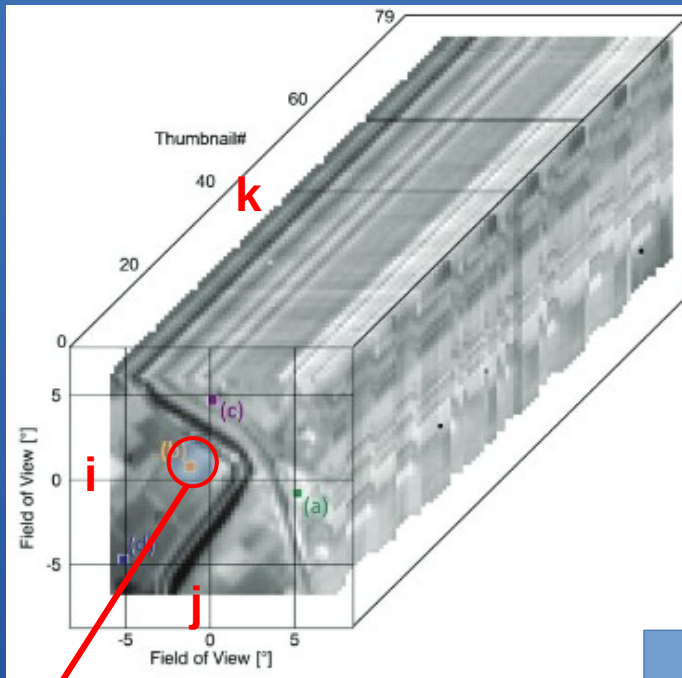
**Albedo**

$X_{CO_2}$

$X_{CH_4}$

$X_{H_2O}$

...



Instrumental  
transfer function

Luminance

$$S^k_{(i,j)} \text{ (ADU)} = \tau \cdot K \int \eta^k_{(i,j)}(\sigma) \cdot L_{(i,j)}(\sigma) d\sigma + \epsilon$$

$$y = \mathbf{F}(x) + \epsilon$$

Albedo,  $X_{CO_2}$ , ( $X_{CH_4}$ ,  $X_{H_2O...}$ )

# Classical Inversion

- Levenberg-Marquardt (LM) iterative inversion

– Modeling:  $\mathbf{y} = \mathbf{F}(\mathbf{x}) + \boldsymbol{\varepsilon}$

Measurement  
noise and  
covariance

– Metrics:

$$\chi^2 = (\mathbf{y} - \mathbf{F}(\mathbf{x}))^T \mathbf{S}_e^{-1} (\mathbf{y} - \mathbf{F}(\mathbf{x})) + (\mathbf{x} - \mathbf{x}_a)^T \mathbf{S}_a^{-1} (\mathbf{x} - \mathbf{x}_a),$$

a priori and  
covariance

– Linearization around  $\mathbf{x}_a$

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_a) + \left. \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \right|_{\mathbf{x}_a} (\mathbf{x} - \mathbf{x}_a)$$

Jacobian  
K

# Classical Inversion

- Levenberg-Marquardt (LM) iterative inversion

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \left[ (1 + \gamma) \mathbf{S}_a^{-1} + \mathbf{K}_i^T \mathbf{S}_e^{-1} \mathbf{K}_i \right]^{-1} \left( \mathbf{K}_i^T \mathbf{S}_e^{-1} (\mathbf{y} - \mathbf{F}(\mathbf{x}_i)) - \mathbf{S}_a^{-1} (\mathbf{x}_i - \mathbf{x}_a) \right)$$

- Hypothesis

- Initial values of X and associated covariance  $\mathbf{S}_a$
- Iteration stop criterion

- References

- Rodgers, C. D. (2000). *Inverse methods for atmospheric sounding: theory and practice*, World scientific.
- Herbin et al. (2013); Dogniaux et al. (2020)

# GLiMM principles

- Bayesian approach (gaussian mixtures)
  - X and Y are linked by combination of **K affine transformations**  $\tau_k$ 
    - Modeled by a missing variable Z such that  $Z = k$  if Y is the image of X by  $\tau_k$

$$Y = \sum_{k=1}^K \mathbb{I}_{Z=k} (\mathbf{A}_k X + \mathbf{b}_k + \epsilon_k)$$

$\mathbb{I}_{Z=k}$

Indicator function

$$p(Y = \mathbf{y}, X = \mathbf{x}; \boldsymbol{\theta}) = \sum_{k=1}^K p(Y = \mathbf{y} | X = \mathbf{x}, Z = k; \boldsymbol{\theta})$$

$$p(X = \mathbf{x} | Z = k; \boldsymbol{\theta}) p(Z = k; \boldsymbol{\theta}).$$

Vector of model parameters

$$\boldsymbol{\theta} = \{\mathbf{c}_k, \boldsymbol{\Gamma}_k, \pi_k, \mathbf{A}_k, \mathbf{b}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$$

$$p(X = \mathbf{x} | Z = k; \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}; \mathbf{c}_k, \boldsymbol{\Gamma}_k)$$

Normal law

$$p(Z = k; \boldsymbol{\theta}) = \pi_k,$$

Uniform law

## References

- Deleforge et al (2015), *High-dimensional regression with gaussian mixtures and partially-latent response variables*, Stat. Comput.
- Kluger (2021);

# GLiMM Inversion

## Forward Modeling

$$p_G(\mathbf{y}|\mathbf{X} = \mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^K \eta_k(\mathbf{x}) \mathcal{N}(\mathbf{y}; \mathbf{A}_k \mathbf{x} + \mathbf{b}_k, \boldsymbol{\Sigma}_k)$$

$$\text{with } \eta_k(\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}; \mathbf{c}_k, \boldsymbol{\Gamma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}; \mathbf{c}_j, \boldsymbol{\Gamma}_j)}$$

Estimation through  $N_{\text{train}}$  simulations of observations

## Inverting

$$p_G(\mathbf{x}|\mathbf{Y} = \mathbf{y}, \boldsymbol{\theta}^*) = \sum_{k=1}^K \eta_k^*(\mathbf{y}) \mathcal{N}(\mathbf{x}; \mathbf{A}_k^* \mathbf{y} + \mathbf{b}_k^*, \boldsymbol{\Sigma}_k^*)$$

$$\text{with } \eta_k^*(\mathbf{y}) = \frac{\pi_k \mathcal{N}(\mathbf{y}; \mathbf{c}_k^*, \boldsymbol{\Gamma}_k^*)}{\sum_{j=1}^K \pi_j^* \mathcal{N}(\mathbf{y}; \mathbf{c}_j^*, \boldsymbol{\Gamma}_j^*)}$$

Needs a grid of X parameters:  
*Sobol distribution*

## Predicting

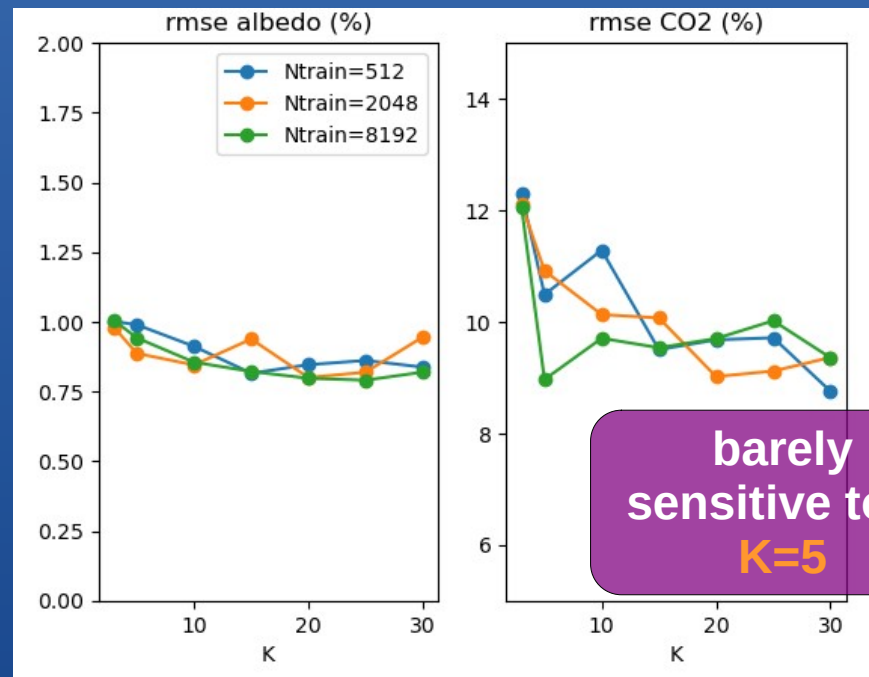
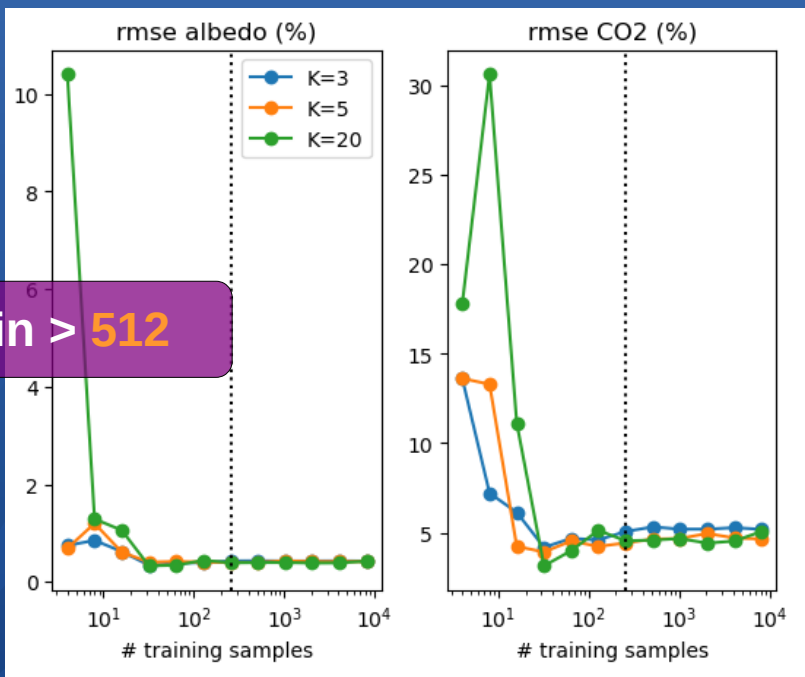
$$\bar{\mathbf{x}}_G(\mathbf{y}) = \mathbb{E}_G[\mathbf{X}|\mathbf{Y} = \mathbf{y}, \boldsymbol{\theta}^*] = \sum_{k=1}^K \eta_k^*(\mathbf{y}) (\mathbf{A}_k^* \mathbf{y} + \mathbf{b}_k^*)$$

# single iFOV observations

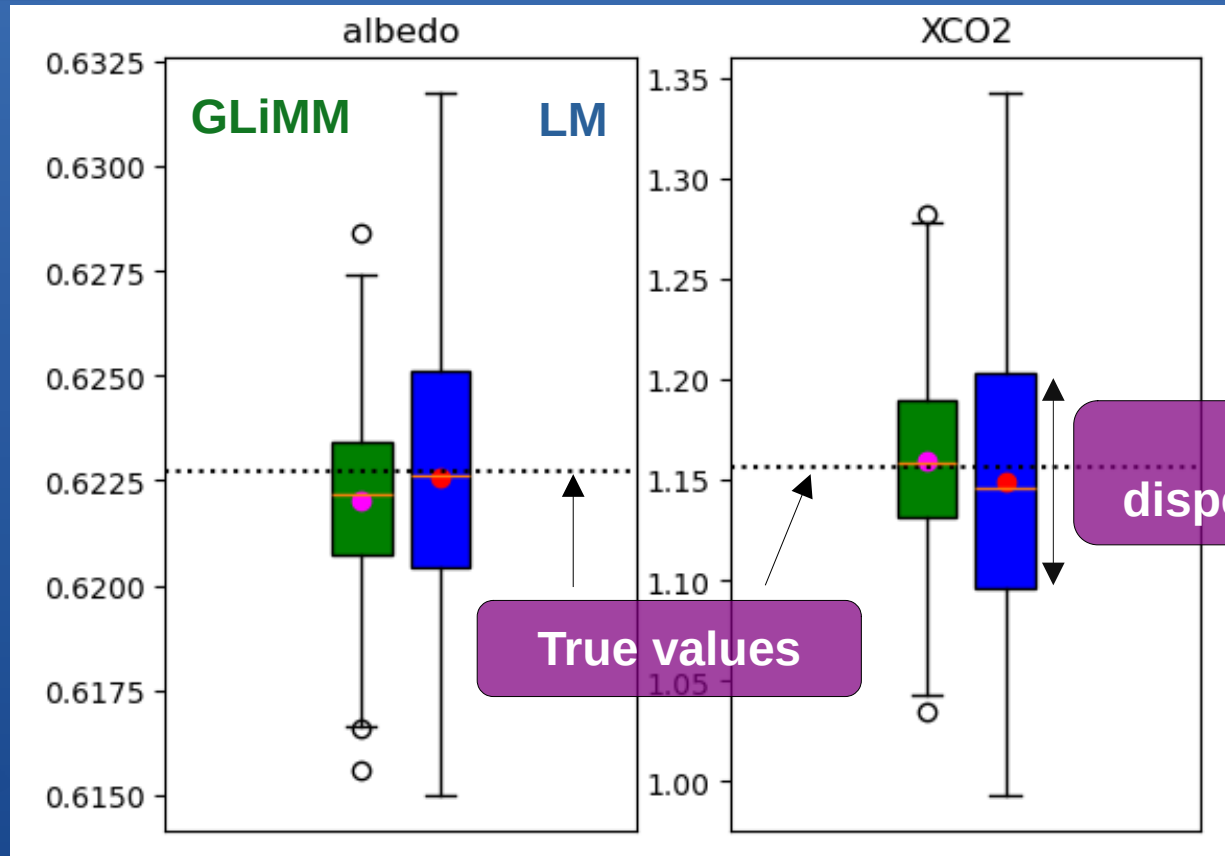
- **Modeling - Training**
  - $Y$  = simulated interferograms,  $X$  = [CO2 scale factor, albedo]
  - Atmospheric model (4AOP) + Instrumental model (Nanocarb)
  - Aerosols: assumed perfectly known (dedicated SPEX instrument)
  - GLiMM input parameters [K, Ntrain]
  - X grid: CO2 [0.8-1.4], albedo [0.1-0.8]
- **Testing: N=100 random draws of noise (photon + detector)**
- **Comparing with Levenberg Marquardt Inversion**
  - $x_a = [1.0, \text{true albedo}]$ ,  $\sigma_a = [0.1, 0.1]$

# GLiMM settings

- Choosing **K** and **Ntrain** (in terms of RMSE)



# Inversion results

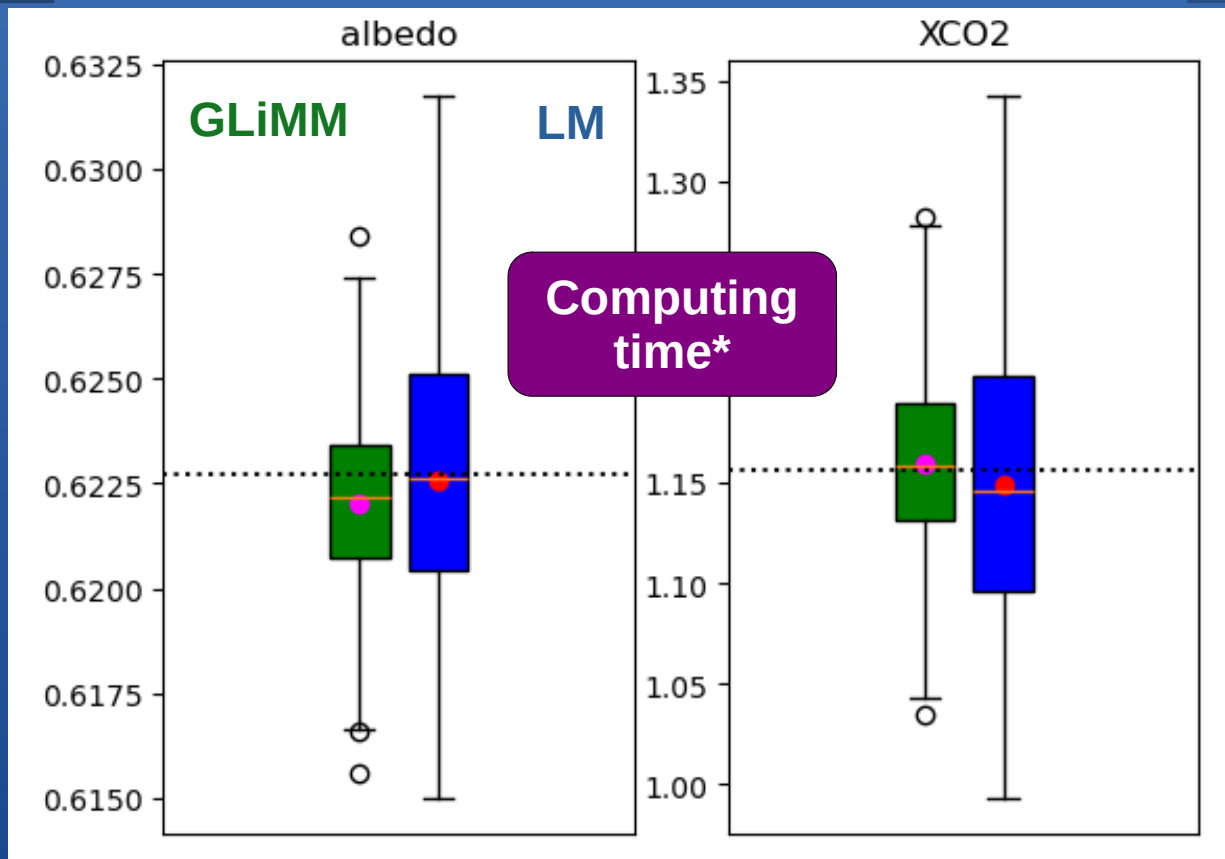


# Inversion results

GLiMM  
~ 0.4s

Running  
grid of  
simulations  
+ training

~6 min



LM  
~ 15min

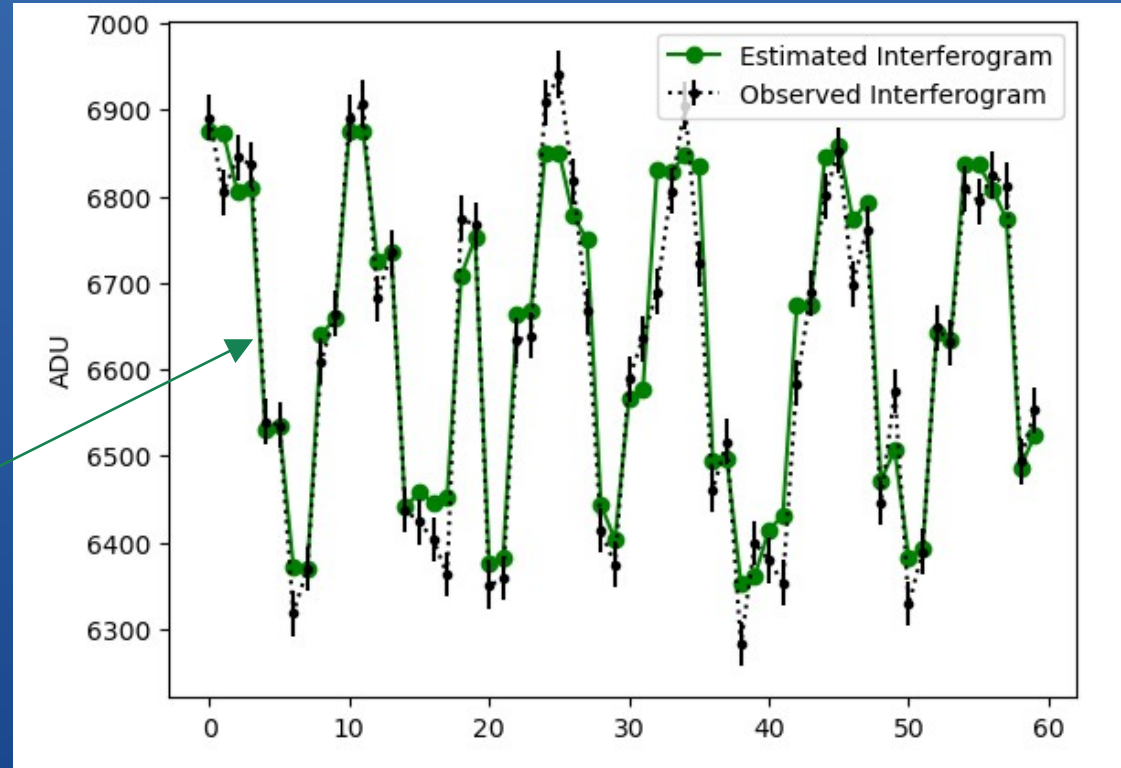
\* PC Intel Core i7-13800H CPU @ 1.4GHz – 30 GB of RAM

# GLiMM Forward Model

Estimated X

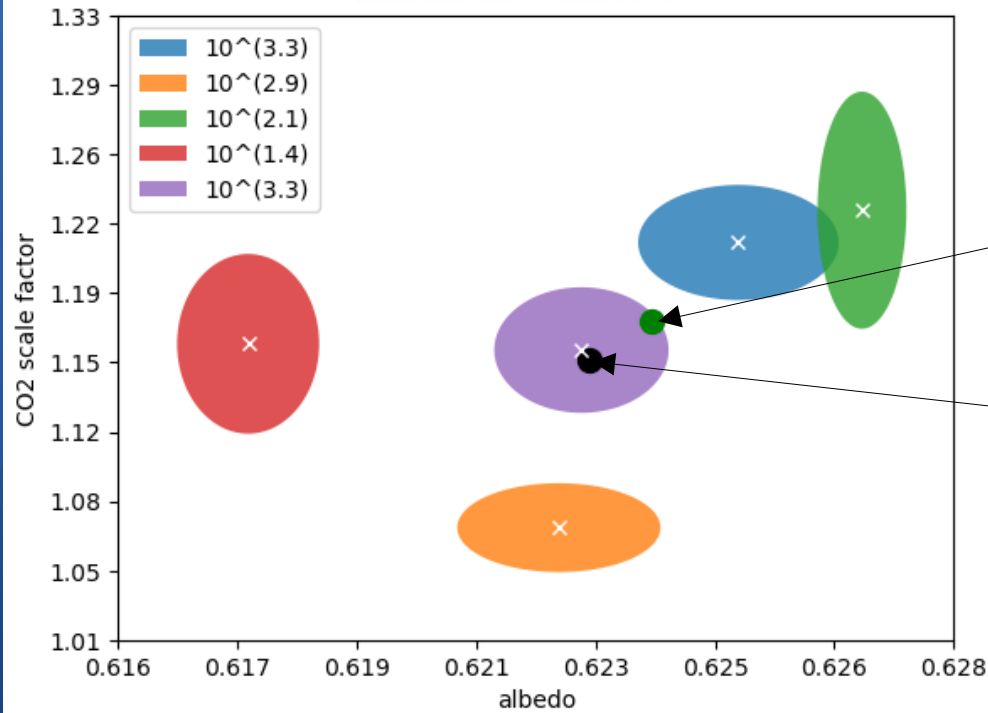
$$\mathbb{E}[Y|X = \mathbf{x}] = \sum_{k=1}^K \alpha_k(\mathbf{x}) (\mathbf{A}_k \mathbf{x} + \mathbf{b}_k)$$

Fit of the  
interferogram



# GLiMM backward model

$$p_G(\mathbf{x}|\mathbf{Y} = \mathbf{y}, \theta^*)$$



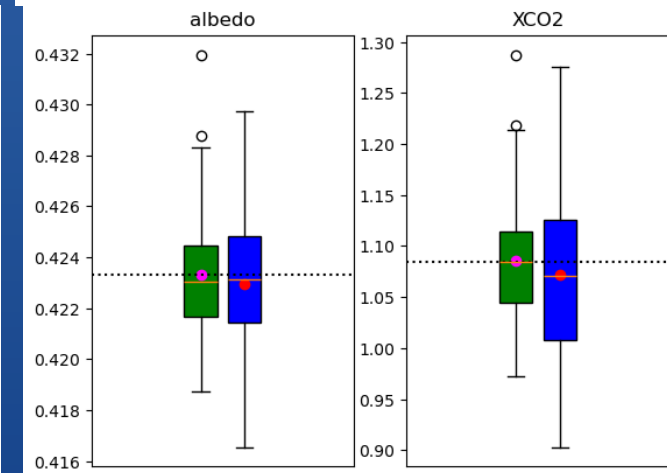
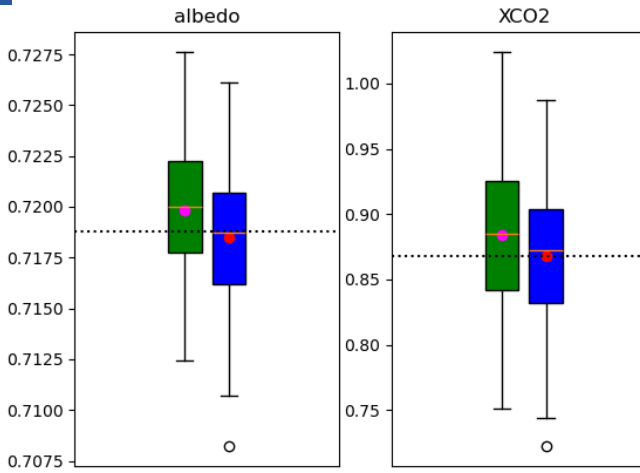
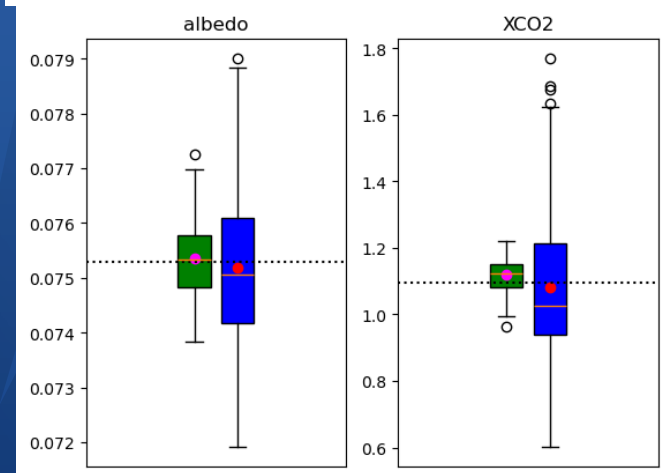
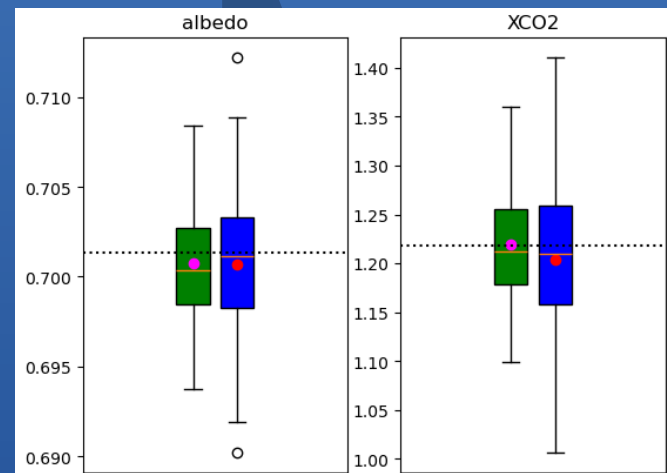
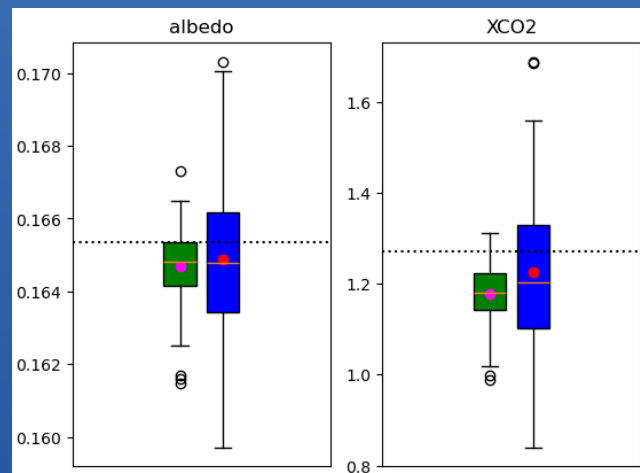
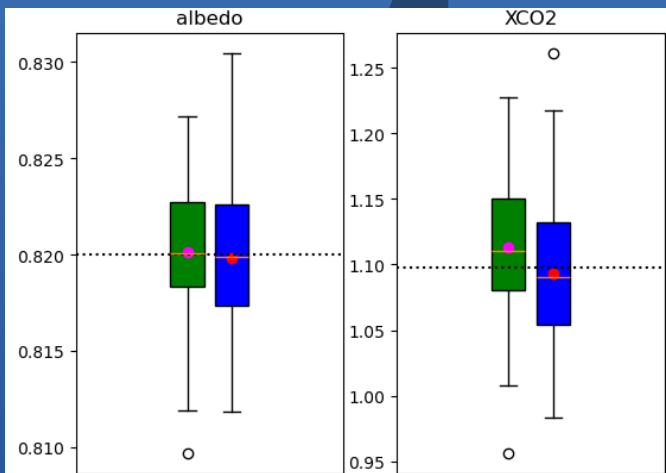
Estimated value

$$\bar{\mathbf{x}}_G(\mathbf{y}) = \mathbb{E}_G[\mathbf{X} | \mathbf{Y} = \mathbf{y}, \theta^*] = \sum_{k=1}^K \eta_k^*(\mathbf{y}) (\mathbf{A}_k^* \mathbf{y} + \mathbf{b}_k^*).$$

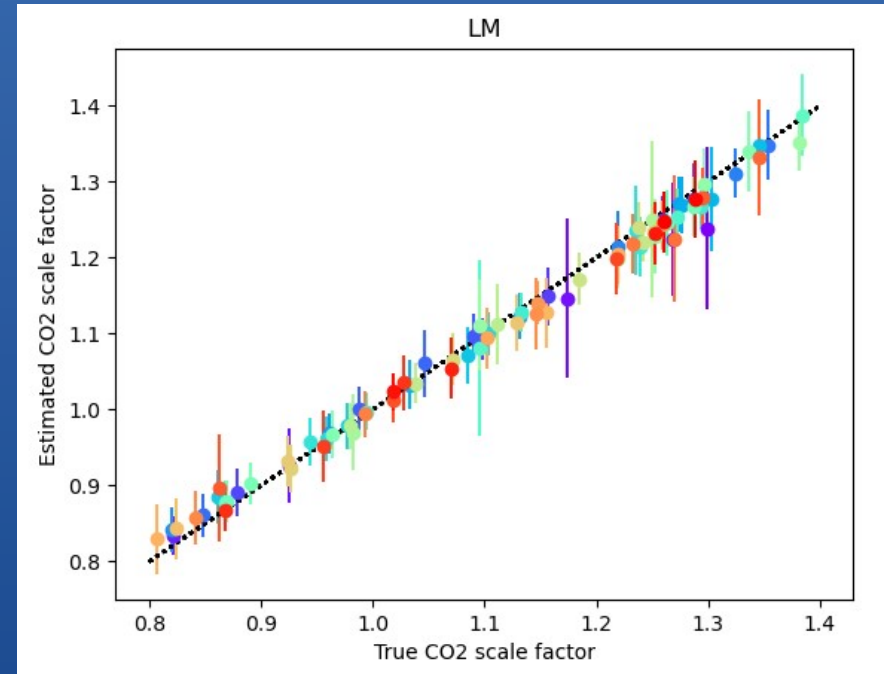
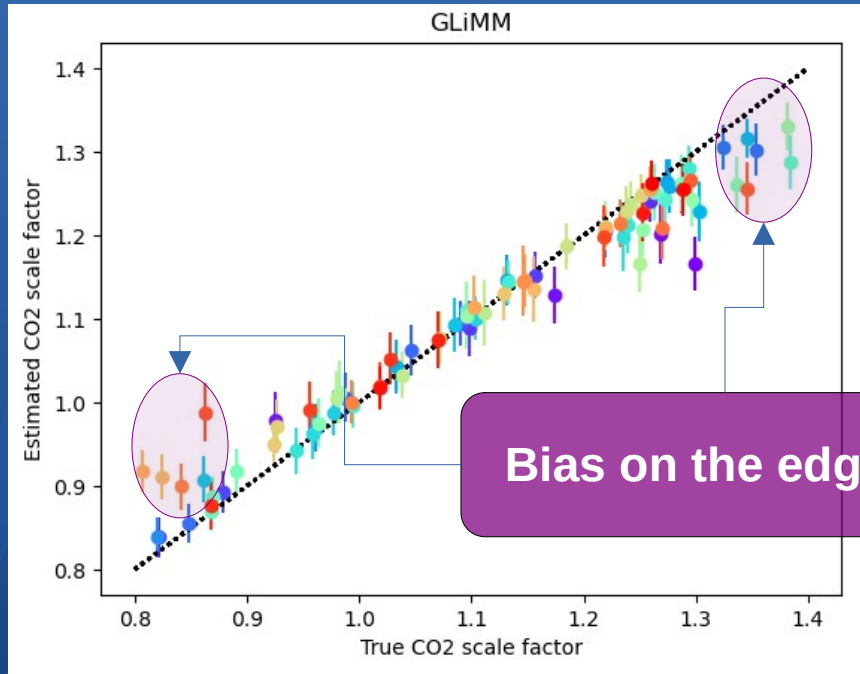
True value

Several outcomes with similar probability  
→ more subtle ways of estimating  $X$

# Random examples



# Spanning the CO2 range



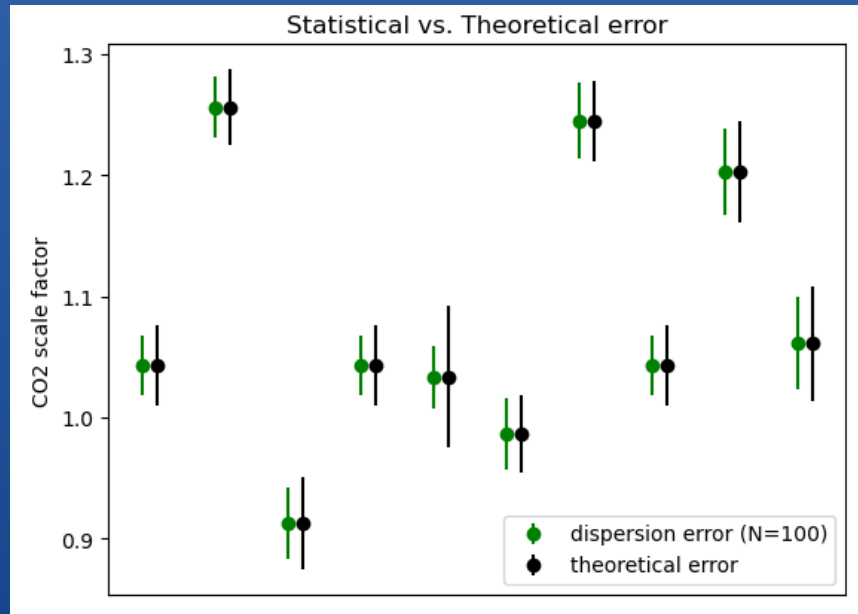
## GLiMM vs. LM at a glance

	Time (100 samples)	Underlying Assumptions	Training set	Bias
LM	~ 15 min	$x_a, S_a$ stop criterion	-	No
GLiMM	~ 0.4 sec (+ 6 min)	grid of parameters K, Ntrain	Yes (> 512)	On edges? (calibratable?)

# Accuracy of CO2 estimates

- Statistical dispersion vs. theoretical error

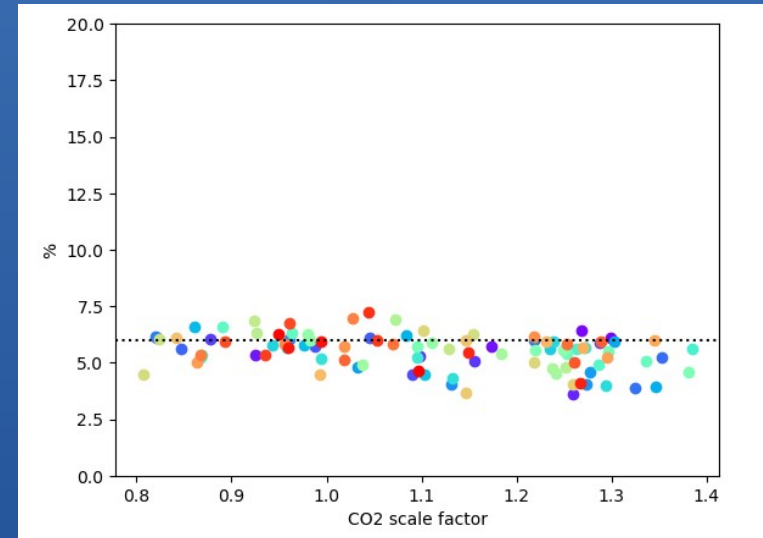
$$\text{Var}(X|Y = \mathbf{y}) = \sum_{k=1}^K \alpha_k^*(\mathbf{y}) [\boldsymbol{\Sigma}_k^* + (\mathbf{A}_k^* \mathbf{y} + \mathbf{b}_k^*)^t (\mathbf{A}_k^* \mathbf{y} + \mathbf{b}_k^*)] - \bar{\mathbf{x}}_G(\mathbf{y})^t \bar{\mathbf{x}}_G(\mathbf{y})$$



GLiMM provides **reliable error bars** with proper order of magnitude

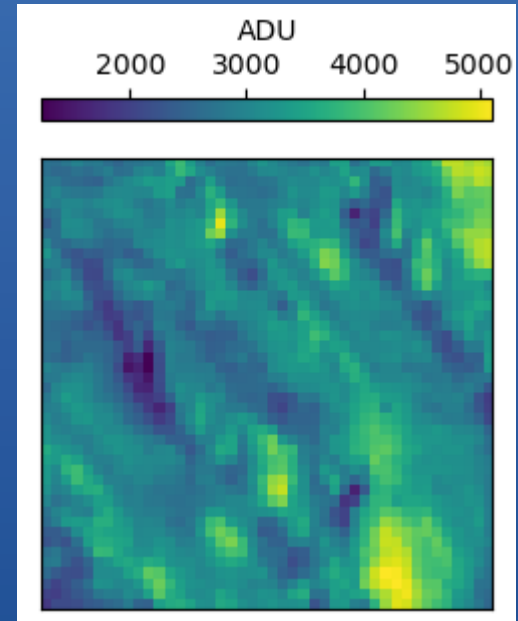
# Accuracy of CO2 estimates

- CO2 accuracy of a single observation
  - $\sigma_{\text{CO}_2}/X_{\text{CO}_2} \sim 6\%$
- 4 ppm accuracy  $\leftrightarrow$  1% in scale factor
  - Need average over  $> 150$  observations (SNR $>2$ )
  - Temporal (1 iFOV over  $\sim 200$  snapshots)
  - Spatial (1 snapshot over 50x50 pixels)



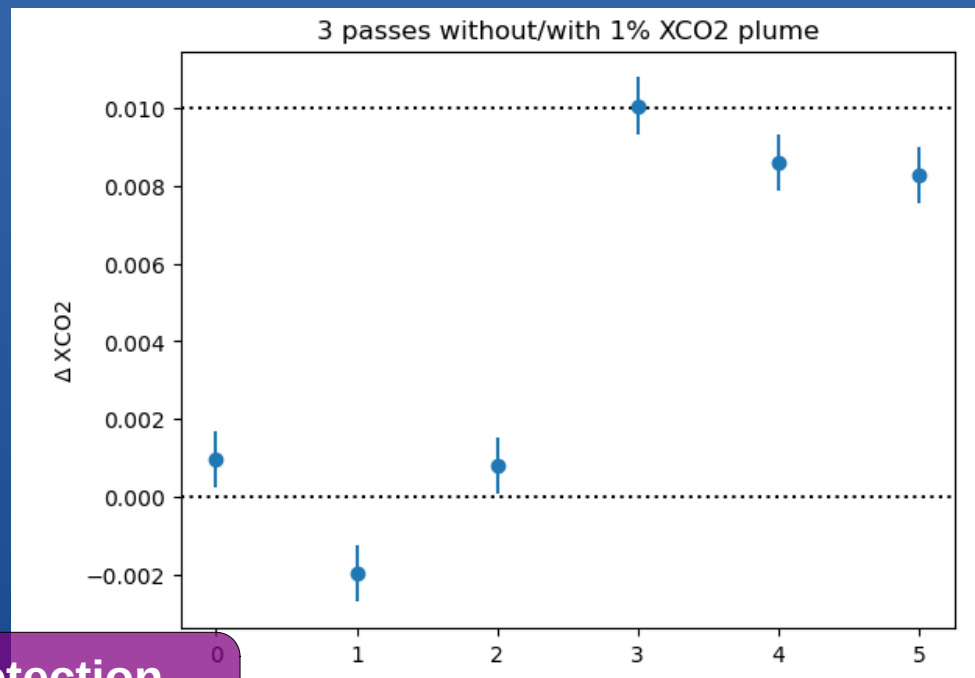
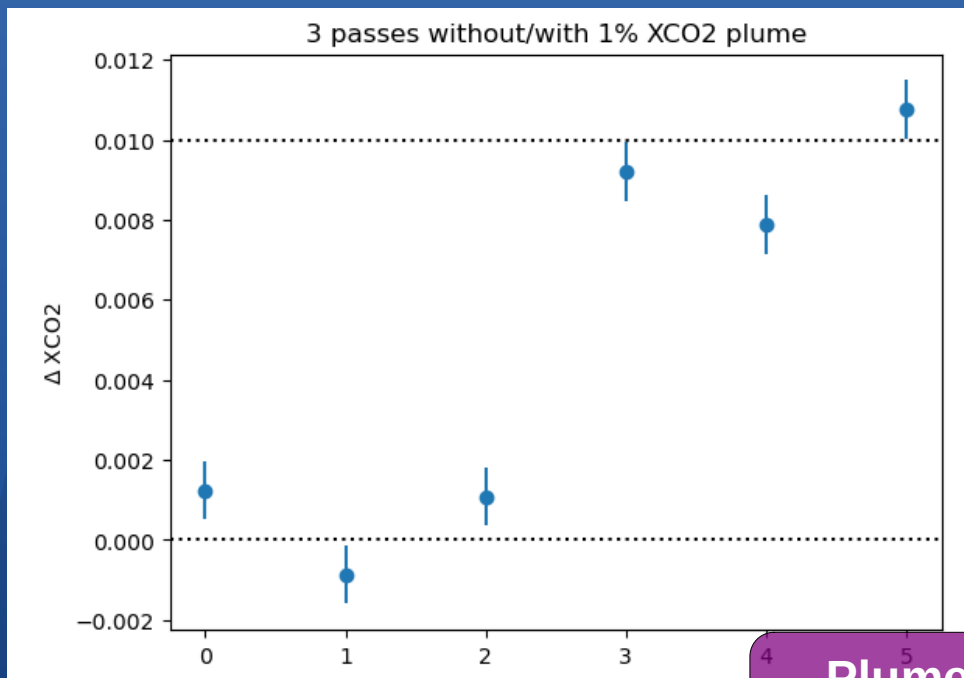
# CO2 Plume Detection

- Flying over a CO2 plume
  - Belchatow 2020 scenery
  - 44 x 44 Snapshots
  - 1% CO2 scale factor variation  
(no plume vs. plume)
- Are we able to detect the plume?
  - Applying GLiMM to each pixel of the image
  - $Y = \text{interferogram} + \text{viewing angle}$ ,  $X = [\text{CO}_2, \text{albedo}]$
  - Airplane passes without then with CO2 plume



# CO2 Plume Detection

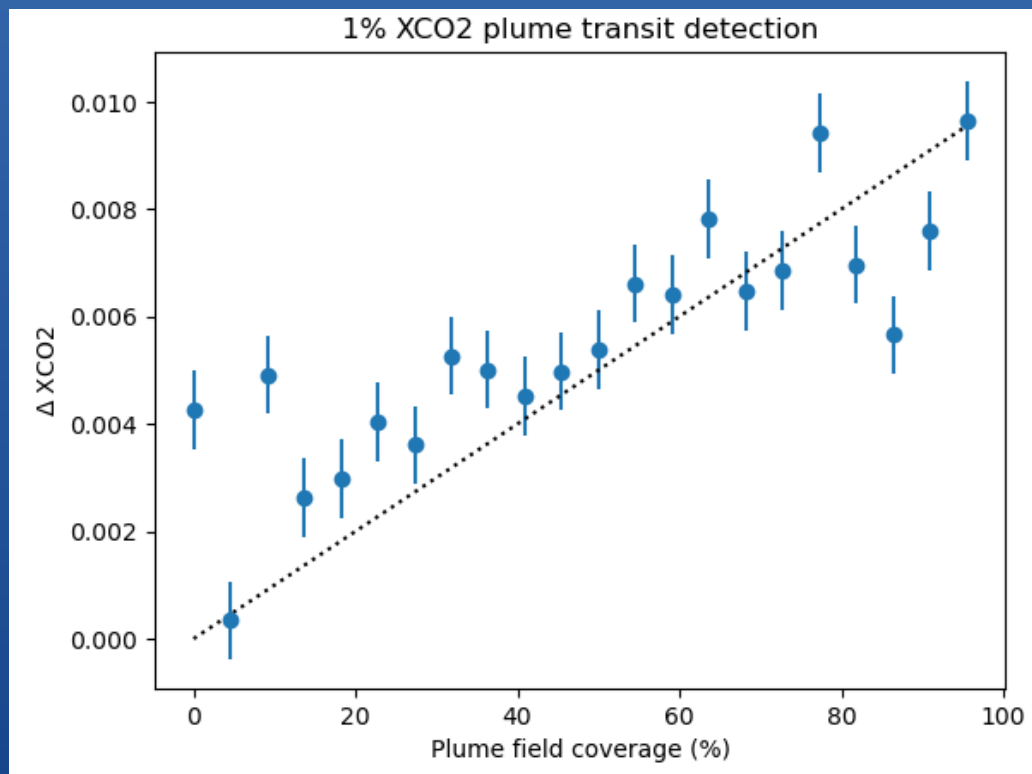
- Scenery with/without CO2 plume



4 Plume Detection  
SNR ~ 3-4

# CO2 Plume Detection

- Smooth transition of the CO2 plume over the scenery



# Conclusions

- **GLiMM machine learning approach**
  - Useful tool for **quick inversion** of complex (non linear, non analytic) models  $Y=F(X) + \varepsilon$
  - Applied to hyperspectral observations of GHG:
    - Obtains similar estimations (and statistics) than LM iterative approach
    - Can handle a large flow of data in a reasonable amount of time
- **Current Limitations - Perspectives**
  - Requires a simulated training set → modeling errors?
  - Need of a unified model for images
  - Need to include aerosols

# Using GLiMM

- Planet-GLLiM
  - <https://gitlab.inria.fr/xllim/planet-gllim>
    - C++ with Python bindings implementation of Gaussian Locally-Linear Mapping.
    - compiled shared library in a **Docker** container
    - Python environment in your browser

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, 'docker.desktop PERSONAL', a search bar, and system icons. The main area is titled 'Containers' and shows a summary of container usage: 'Container CPU usage 113.66% / 2000% (20 CPUs available)' and 'Container memory usage 373.4MB / 7.32GB'. Below this is a search bar and a toggle for 'Only show running containers'. A table lists containers with columns for Name, Container ID, Image, Port(s), CPU (%), Memory, and Actions. One container named 'xllim\_notebook' is highlighted, showing it is running with ID 'a1af9c5f9c26' and image 'xllim\_jupyter: 8888:8888'. The bottom status bar shows 'Engine running', 'RAM 1.36 GB', 'CPU 5.83%', and 'Disk 41.15 GB avail. of 67.32 GB'.

Name	Container ID	Image	Port(s)	CPU (%)	Memory	Actions
xllim_notebook	a1af9c5f9c26	xllim_jupyter: 8888:8888		101.57%	367.9MB	⏏
build	-	-	-	0%	0B / 0B	⏏

The screenshot shows a web browser window displaying a Python code editor. The browser address bar shows 'localhost:8888/lab/tree/glimm\_onepix-testKNechsub.ipynb'. The code editor contains Python code for GLiMM, including imports for xllim, numpy, matplotlib, and scipy. The code defines a function 'getSobolSeq' and a sampling function. The code is as follows:

```
[1]: import xllim
import numpy as np
import matplotlib.pyplot as plt
import json
import logging
logging.getLogger().setLevel(logging.INFO)
import pickle
from scipy.stats import qmc;

[2]: def getSobolSeq(Ldim, Nch, xmin=None, xmax=None):
    sampler = qmc.Sobol(d=Ldim, scramble=False);
    sample = sampler.random_base2(m=int(np.round(np.log2(Nch))));

    if xmin is None:
        xmin=np.zeros(Ldim);
        xmax=np.ones(Ldim);
    else:
        xmin=np.array(xmin);
        xmax=np.array(xmax);

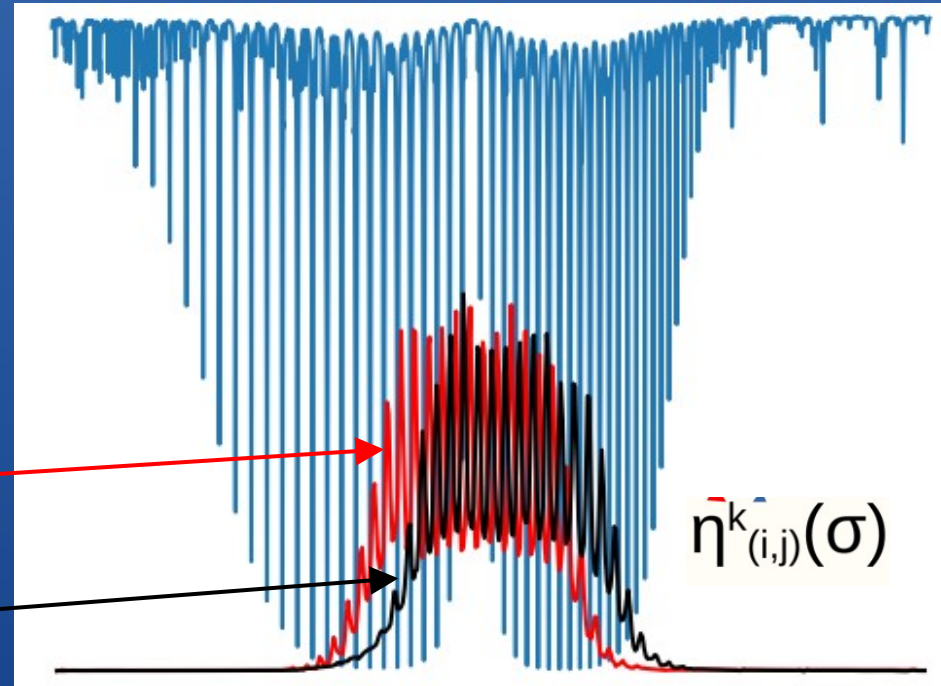
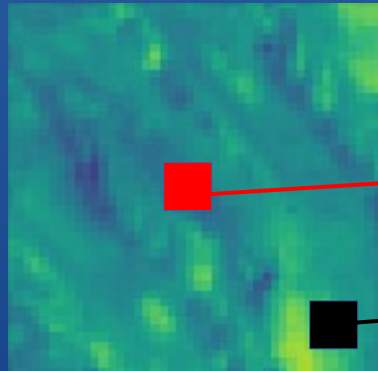
    sample=(sample*(xmax-xmin))+xmin
    return sample;

[3]: Nfiles=64;
Nch=Nfiles*1000;
Ldim=2;
sig_simu=0.01;
#conv=1: sim ADUe
ADUe=2.5;
signoise=30;
Nchsub=50000;

[4]: filename='PKL/RTtrainset_V2_Ldim'+str(Ldim)+'*.pkl';
fd=open(filename, 'rb');
```

# Appendix

- Effect of instrumental transfer function
  - Shift of shape and spectral bandwidth with viewing angle
  - Implicit in GLiMM training
  - Interpolation not efficient



# Appendix

- Effect of instrumental transfer function
  - Shift of shape and spectral bandwidth with viewing angle
  - Implicit in GLiMM training
  - Interpolation not efficient

