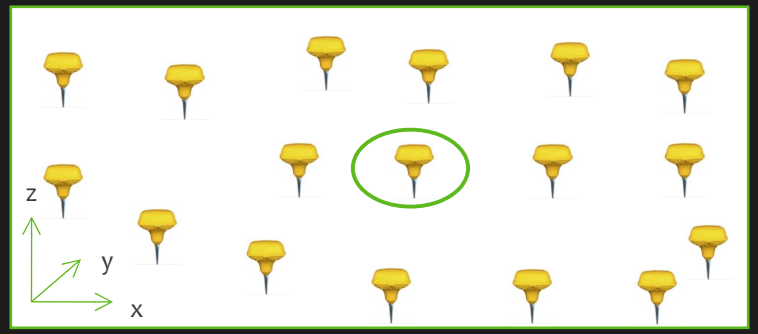
Active noise cancellation

Francesca Badaracco

Active noise cancellation

Sensor array



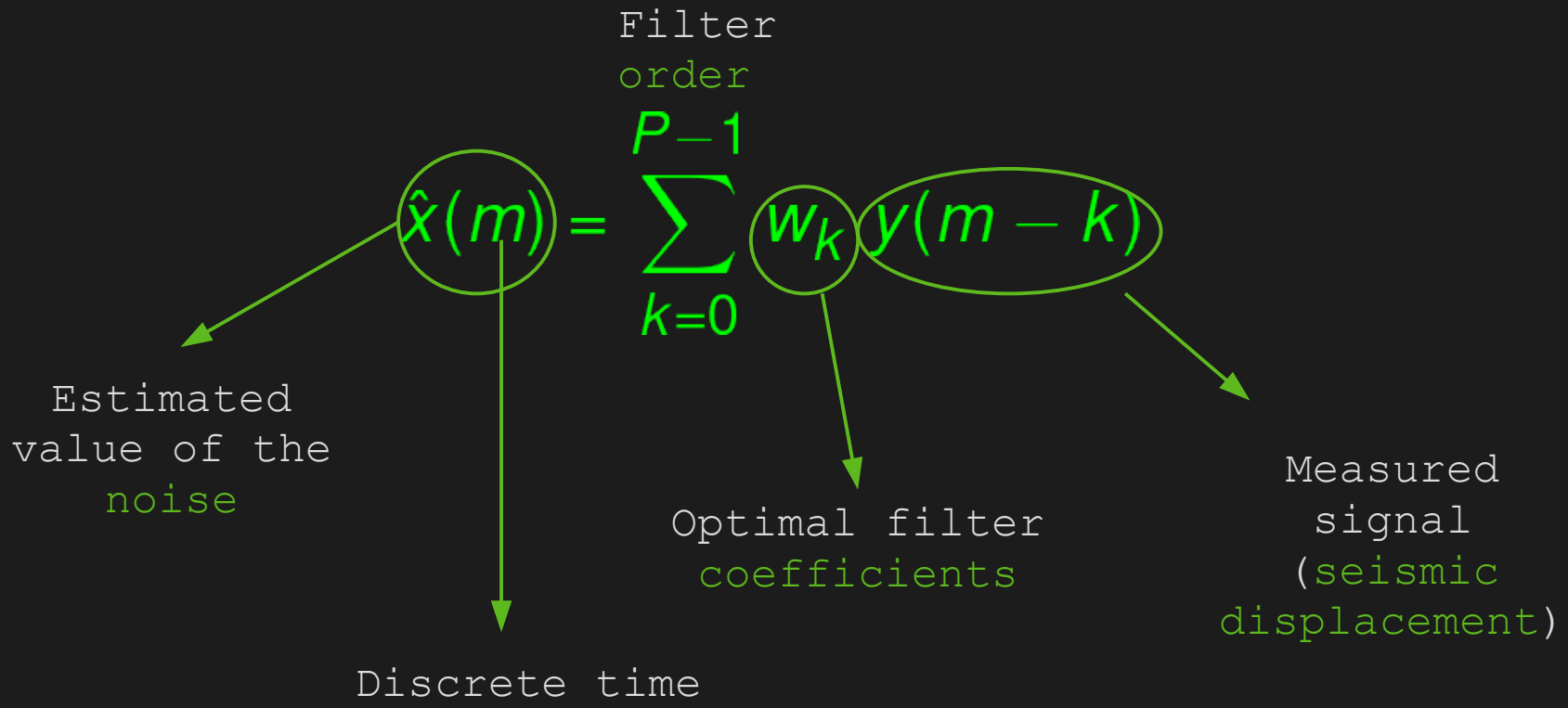
Filter

Noise estimate

Linear Filter is the way:

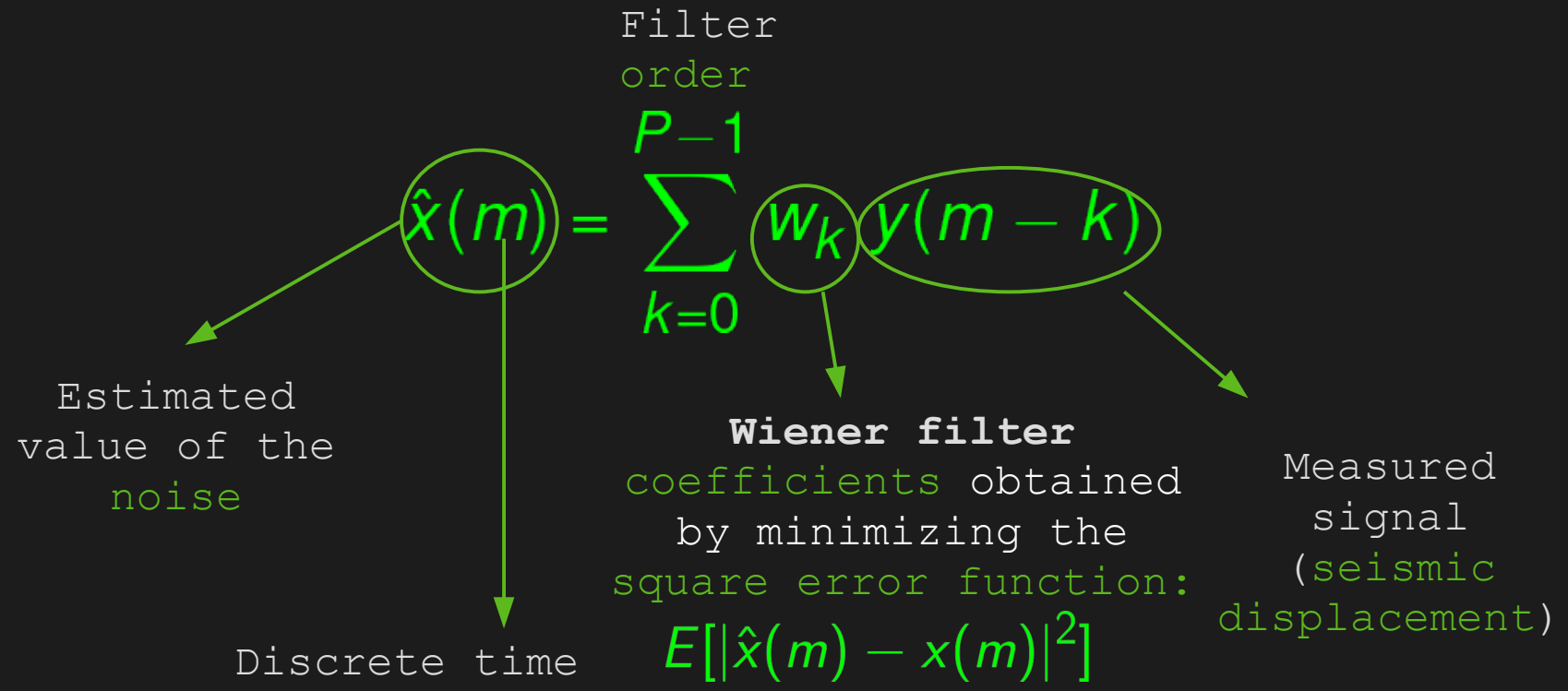
Assumptions:

- **Linear** relationship



WHAT WE USED TO DO:

In GW field you might have heard about **Wiener Filter:**



WHAT WE USED TO DO:
Array optimization with **Wiener Filter:**

Wiener filter to perform a NN cancellation (**time domain**):

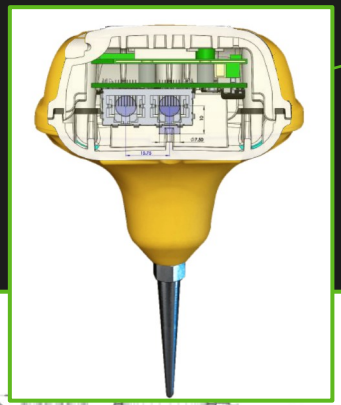
$$\hat{x}(m) = \sum_{k=0}^{P-1} w_k \, y(m-k)$$

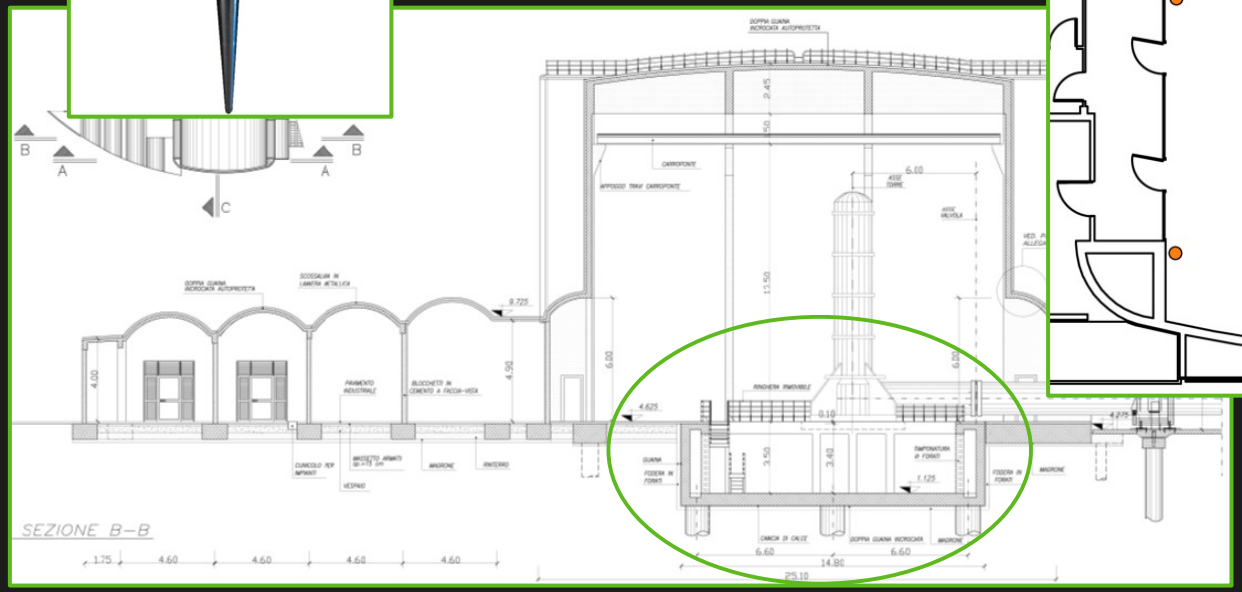
$$R(\omega) = 1 - \frac{\vec{C}_{sn}^{\dagger} \mathbf{C}_{ss}^{-1} \vec{C}_{sn}}{C_{nn}}$$

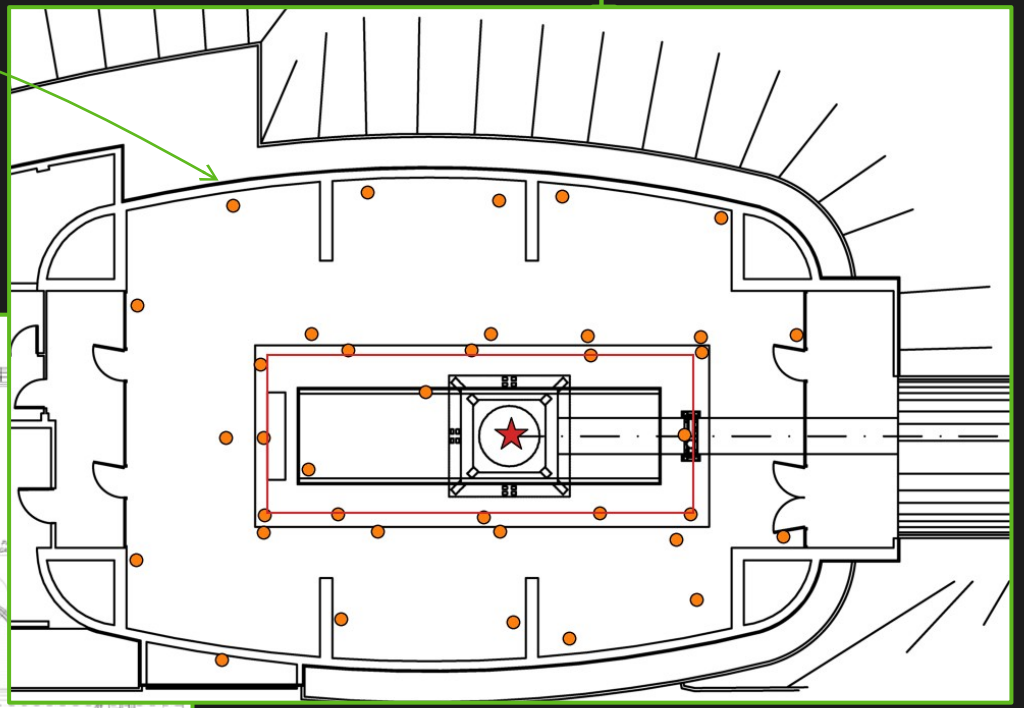
Wiener filter performances (**frequency domain**):

REMEMBER!!!

What if the seismic field is not homogeneous and isotropic?

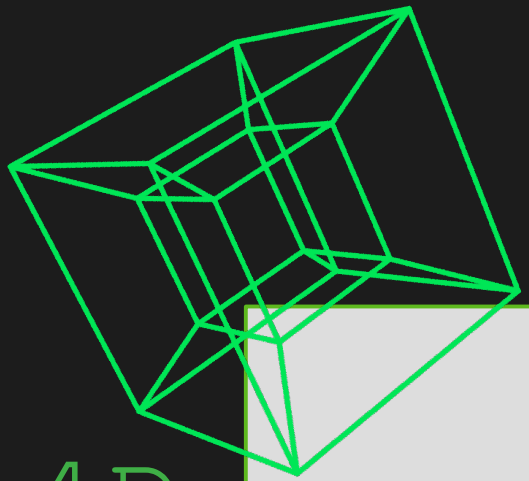






$C_{ss}(x_1, y_1, x_2, y_2)$ is a 4D function!

Curse of dimensionality:

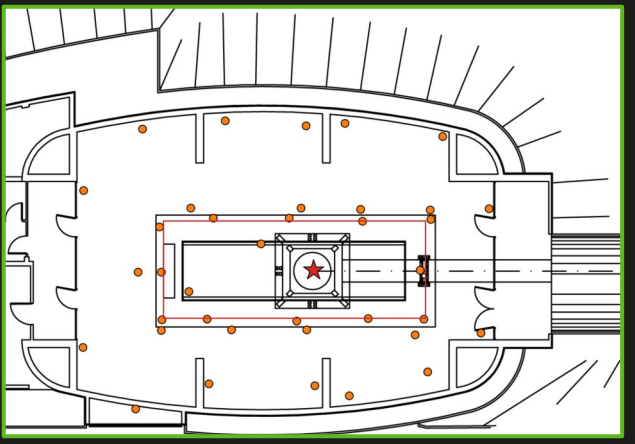


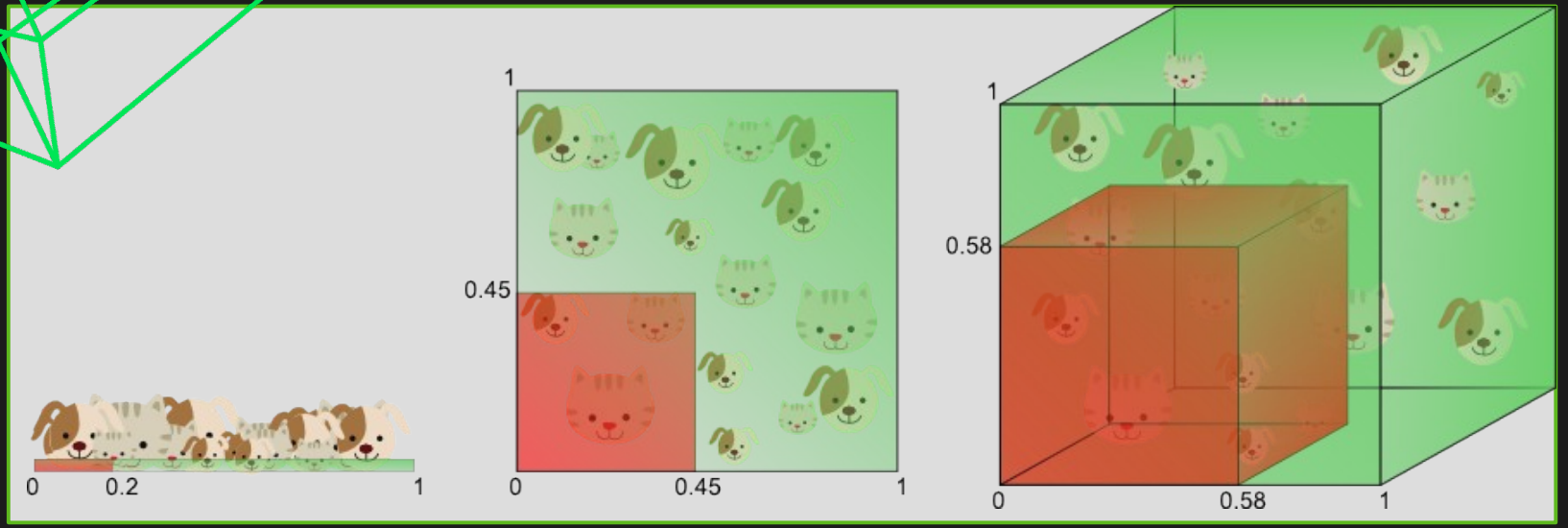
4D

$\rho_{2D} = 0.30$

$\rho_{4D} = 0.05$

$\rho_{4D\_Regular\_grid} = 29$





https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

$$C_{ss}(x_1,y_1,x_2,y_2) = \langle (FFT^*\{s(x_1,y_1)(\omega)\}\ FFT\{s(x_2,y_2)(\omega)\}) \rangle$$

ith seismometer's data stream (1 hour, for example)

FFT$_1$

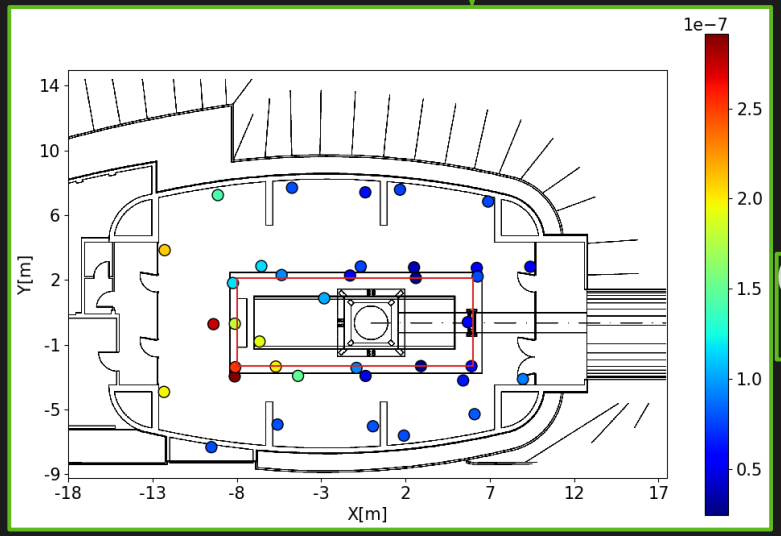
FFT$_2$

N segments with
50% overlapping

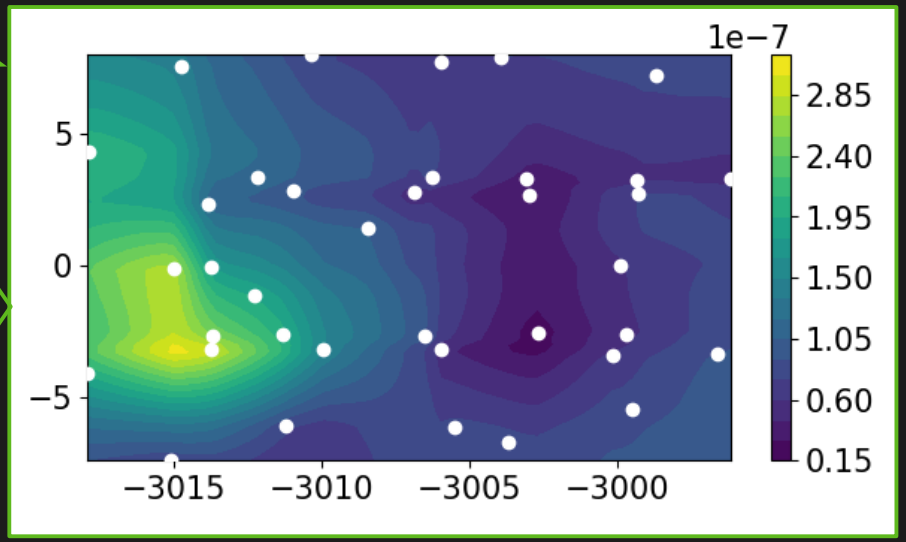
...

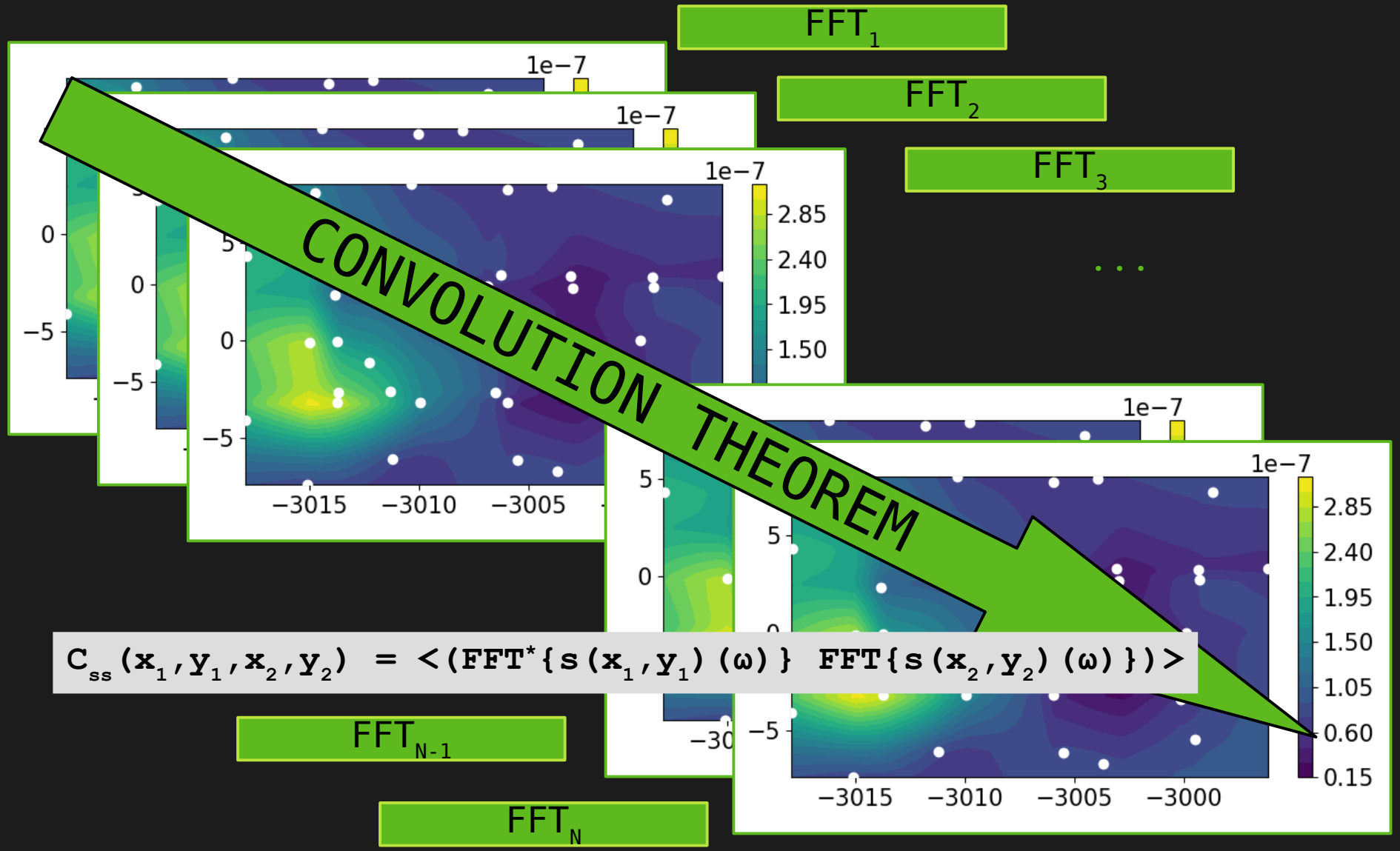
FFT$_{N-1}$

FFT$_N$

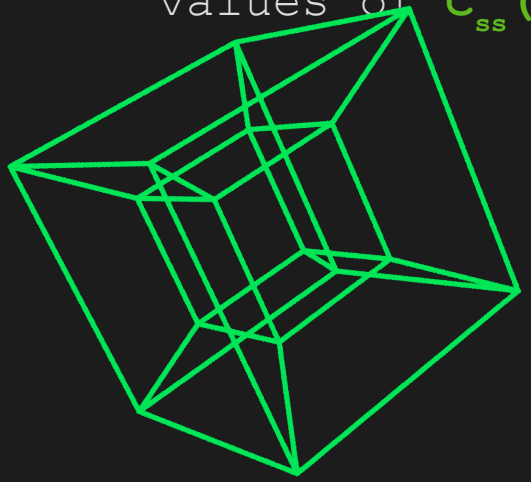


Gaussian Process

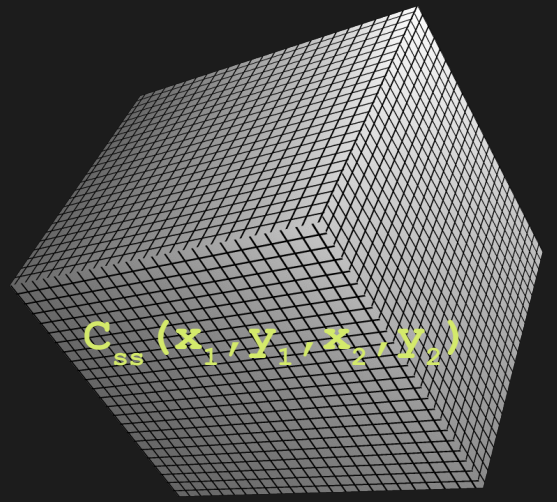


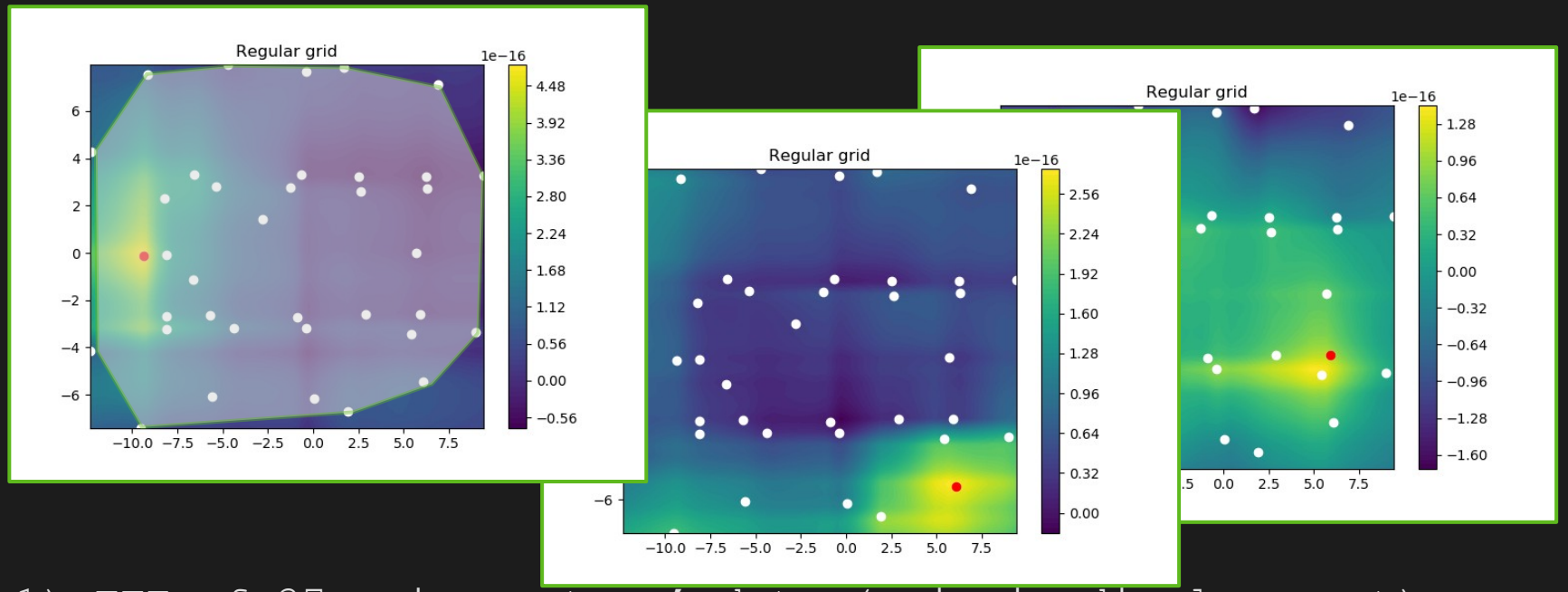

FFT1
FFT2
FFT3
CONVOLUTION THEOREM
FFTN-1
FFTN
1e−7
2.85
2.40
1.95
1.50
1.05
0.60
0.15
−3015
−3010
−3005
−3000
$C_{ss}(x_1,y_1,x_2,y_2) = \langle (FFT^*\{s(x_1,y_1)(\omega)\}\ FFT\{s(x_2,y_2)(\omega)\}) \rangle$

Every point of $C_{ss}(x_1, y_1, x_2, y_2)$ in the 4D space is calculated as before → We can virtually sample as many values of $C_{ss}(x_1, y_1, x_2, y_2)$ as we want, **wherever** we want.



Virtual Sampling + Linear interpolation: we created a **surrogate model** of $C_{ss}(x_1, y_1, x_2, y_2)$



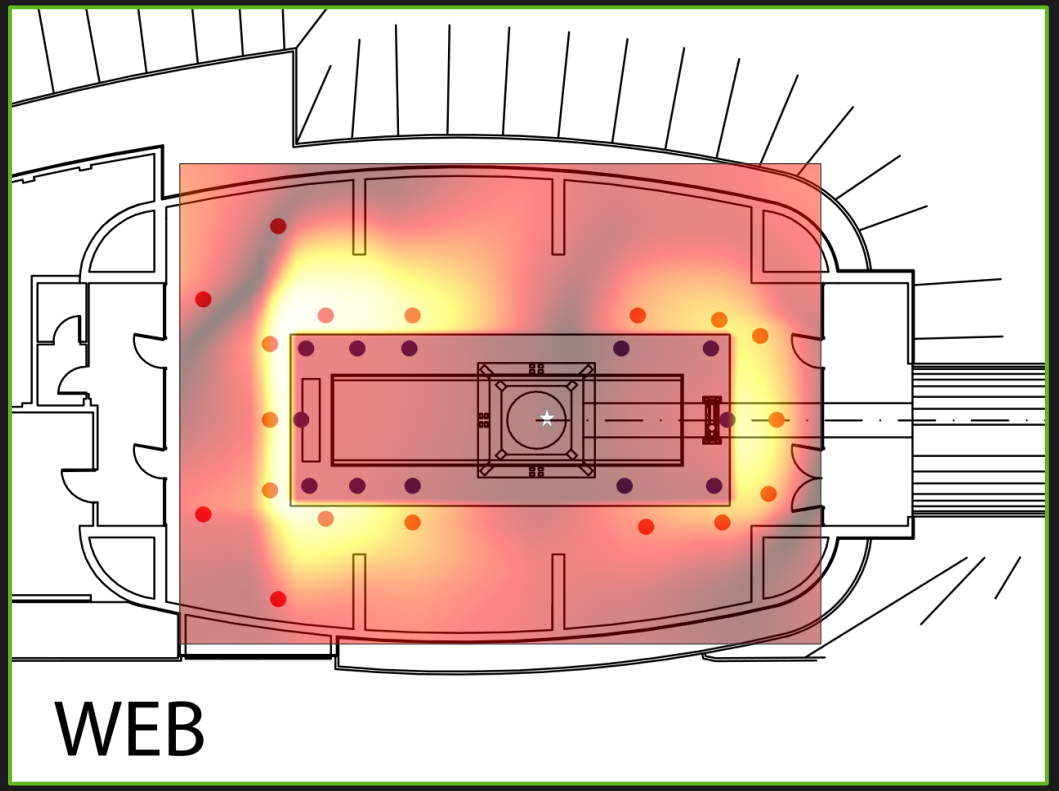


1) FFT of 37 seismometers' data (seismic displacement) → **2D gaussian process** at a frequency f_0: **Convolution theorem** → **surrogate model** of Css:

$$C_{ss}(x_1,y_1,x_2,y_2) = \langle (FFT^*\{s(x_1,y_1)(\omega)\}\ FFT\{s(x_2,y_2)(\omega)\}) \rangle$$

2) Css Sampling → **4D Linear Interpolation on a Regular grid** (it's faster)→ **Css** & **Csn** (integrated with Simpson method)

Summarizing: the residual will depend on the **frequency**, the **number of sensors** and on their **positions**:
In 2D we have **2N coordinates**, where N is the number of the sensors.



Data collection with provisional array

↓

Positions optimization

↓

Optimal array deployment

↓

Subtraction pipeline (applying Wiener filter)

But… wait!

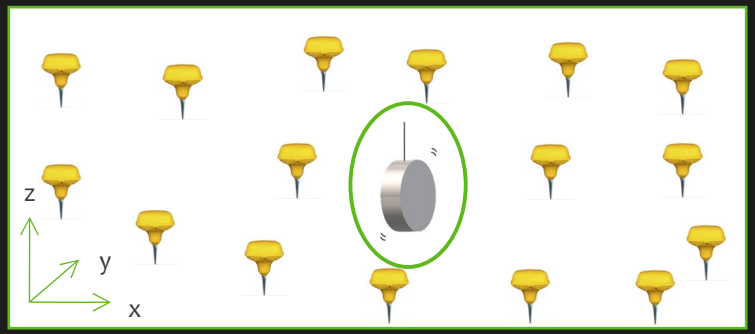
GW detectors vs. **LGWA**

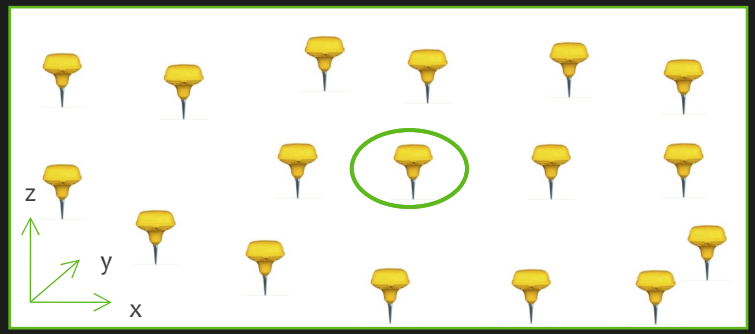
Target sensor: the GW detector, its signal contains:
h(t) = gw(t) + noise(t)

Witness sensors: the seismic sensors, their signal contain:
s(t) = noise(t)

Target sensor: the seismic sensors, their signal contain:
s(t) = gw(t) + noise(t)

Witness sensors: the seismic sensors, their signal contain:
s(t) = gw(t) +noise(t)

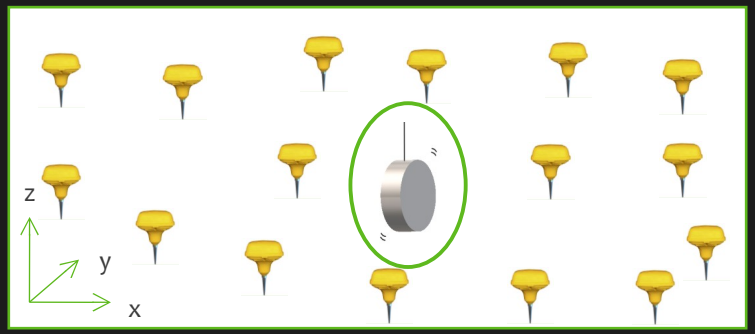




But… wait!

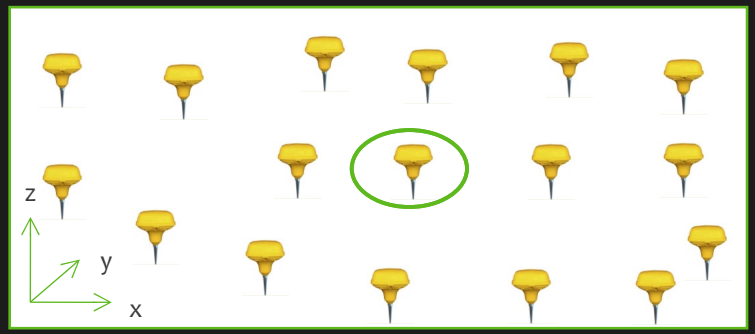
GW detectors vs. **LGWA**

Here, we want to minimize the noise in the target, therefore the array must be such that it collects as much as possible noise, i.e. we want to **minimize the square error**



Here, we want to maximize the signal and simultaneously minimize the noise, i.e. we want to **maximize the SNR**!

A different filter coefficient optimization (not anymore a Wiener Filter!). Array position from SNR maximization



Conclusion

- Remember! There are 2 optimizations:
  - To get the optimal filter coefficients (it decides the filter)
  - To get the optimal positions given a certain filter
- We can follow the procedure used for Virgo to find the LGWA optimal array, but we need to consider a few things:
  - We will use a **different filter** (→ different cost function: SNR instead of square error)
  - We will **not** have any **provisional array:**
    - Therefore, we should rely on seismic **simulations and a full Bayesian approach to Gaussian Process Regression***
    -  What about using **robots** to move the array and get a few measurements before the array optimization? 

* Providing priors to Bayesian array optimization for the Sardinian candidate site of the Einstein Telescope Authors: T. Andric DOI: 10.1393/ncc/i2022-22183-7

Something else to take into consideration:

Data analysis: **blind source separation techniques:**

For example: Independent Component Analysis (ICA)

Goal: separate 2 signal, the GW one from the seismic noise.

ICA assumes that your signals are **independent** and **non-gaussian**. Then it finds the original sources **maximizing the independence** between the possible signals that compose the measured mixture. However it cannot reconstruct the amplitude of the signals.

Thank you for the attention

$$\delta\rho_{\text{temp}}(\mathbf{r},t) = -\frac{\rho_0}{T_0}\delta T(\mathbf{r},t)$$

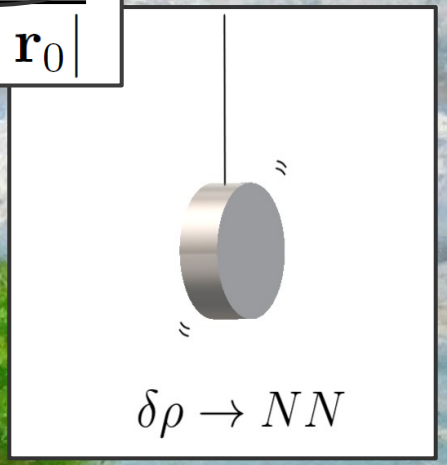
$$\delta\rho_{\text{press}}(\mathbf{r},t) = \frac{\rho_0}{\gamma p_0}\delta p(\mathbf{r},t)$$

Adiabatic index

ATMOSPHERIC NN

$$\delta\phi(\mathbf{r}_0,t) = -G\int dV \frac{\delta\rho(\mathbf{r},t)}{|\mathbf{r}-\mathbf{r}_0|}$$

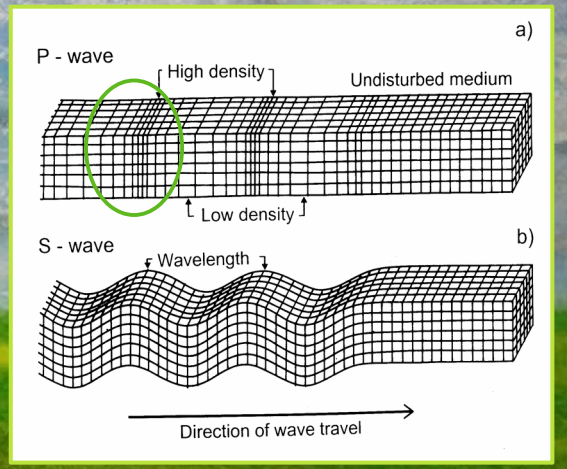
Newtonian Noise (NN):
Perturbation of the gravity field due to a variation in the density (δρ) of the surrounding media.



$$\delta\rho \rightarrow NN$$

$$\delta\rho_{\text{seis}}(\mathbf{r},t) = -\nabla\cdot(\rho(\mathbf{r})\boldsymbol{\xi}(\mathbf{r},t))$$

SEISMIC NN



Residual in frequency domain

$$R(\omega) = 1 - \frac{\vec{C}_{sn}^{\dagger} \mathbf{C}_{ss}^{-1} \vec{C}_{sn}}{C_{nn}}$$

$$C_{sn_i}(\omega) = E[s_i^*(\omega) n(\omega)]$$

i^{th} element of the **vector** containing all the cross power spectral densities of all the seismic sensors with the test mass (containing also the NN)

$$C_{nn}(\omega) = E[n^*(\omega) n(\omega)]$$

Power Spectral Density of the target signal (test mass). It's a **scalar**

$$C_{ss_{ij}}(\omega) = E[s_i^*(\omega) s_j(\omega)]$$

i^{th} element of the **matrix** containing all the cross power spectral densities between all the seismic sensors

What if the seismic field is not homogeneous and isotropic?

Residual in frequency domain

$$R(\omega) = 1 - \frac{\vec{C}_{sn}^{\dagger}\mathbf{C}_{ss}^{-1}\vec{C}_{sn}}{C_{nn}}$$

$C_{sn_i}(\omega) = E[s_i^*(\omega)n(\omega)]$ We can use a model (next slide)

$C_{nn}(\omega) = E[n^*(\omega)n(\omega)]$ We treat it just as a **unknown** constant

$C_{ss_{ij}}(\omega) = E[s_i^*(\omega)s_j(\omega)]$ This is easy: we just need to **collect data**!

What is a Gaussian Process?

Statistic →
Inferring models from data →
Interpretation

Machine Learning →
Learn algorithms to **predict** new data →
Black boxes

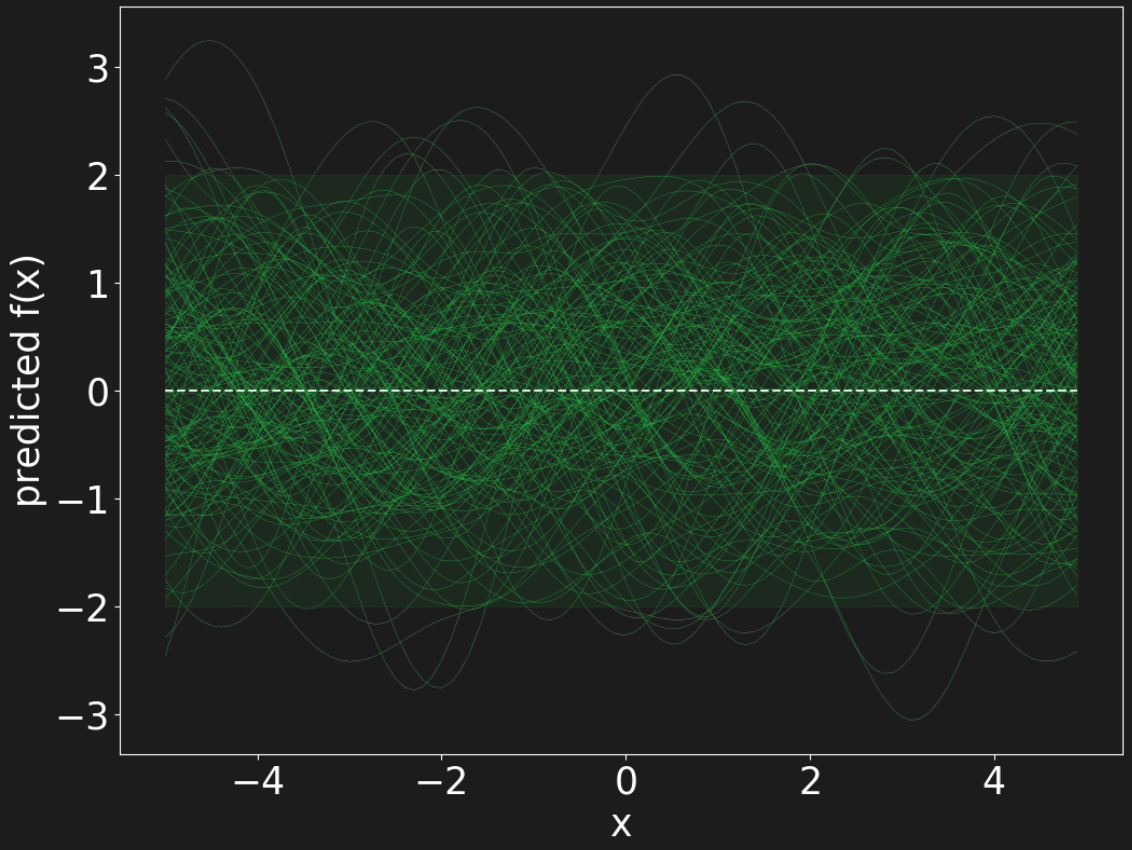
Gaussian Processes →
mathematically equivalent to **known models**
+
Learn from data and can **predict** new values

regression

Kriging
(geophysics)

classification

What is a Gaussian Process?



Gaussian process = a collection of random variables with a joint Gaussian distribution.

Gaussian process over functions = the values taken by a function in a point x_i: $f(x_i) = f_i$ are random variables.

$x_1, x_2, \ldots, x_N \rightarrow f_1, f_2, \ldots, f_N$ with a gaussian joint distribution with mean and covariance:

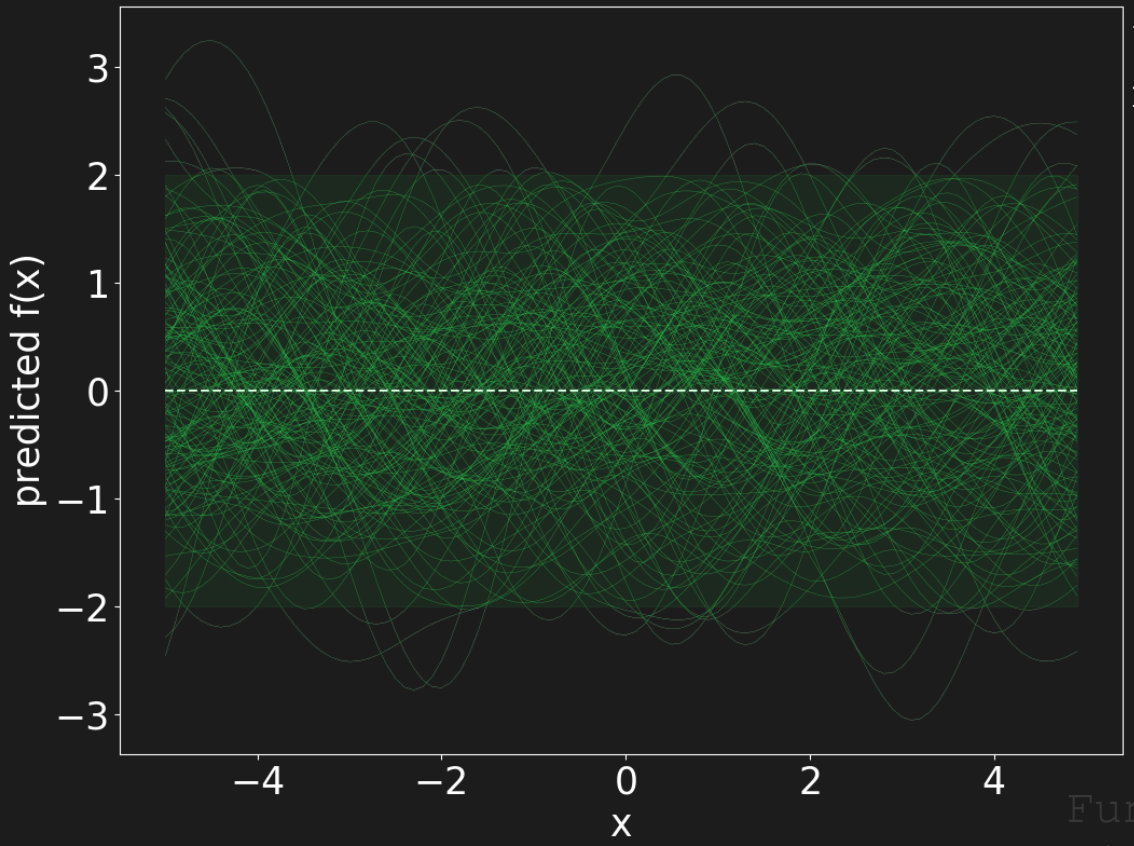
$$m(x) = E[f(x)]$$

$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))]$$

σ_f = signal variance, l = scale

σ_ε = noise variance

What is a Gaussian Process?



We can draw functions from a multivariate normal distribution:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

GP regression takes the form of a Bayesian inference over a "latent function", f(x):

$$y = f(x) + \varepsilon$$

→ Gaussian distributed noise

$$f(x) \sim \mathcal{N}(0, k(x, x'))$$

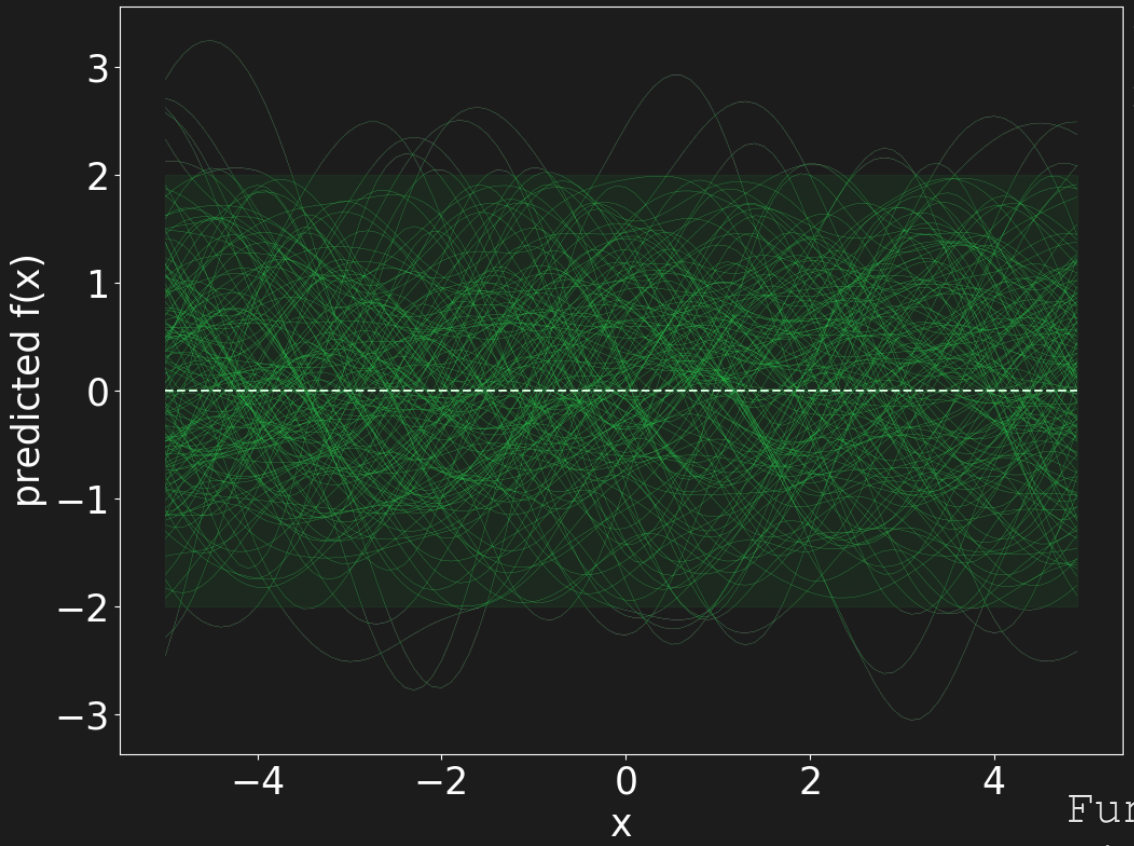
$$k(x_i, x_j) = \sigma_f^2 e^{-\frac{(x_i - x_j)^2}{2l}} - \sigma_\varepsilon \delta_{ij}$$

σ_f = signal variance, l = scale

σ_ε = noise variance

Functions f(x) sampled by a prior with fixed hyper-parameters: σ_f = 1, l = 0 and σ_ε = 0 and zero mean.

What is a Gaussian Process?



We can draw functions from a multivariate normal distribution:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

GP regression takes the form of a Bayesian inference over a "latent function", f(x):

$$y = f(x) + \varepsilon$$

ε → Gaussian distributed noise

$$f(x) \sim \mathcal{N}(0, k(x, x'))$$

$$k(x_i, x_j) = \sigma_f^2 e^{-\frac{(x_i - x_j)^2}{2l}} - \sigma_\varepsilon \delta_{ij}$$

σ_f = signal variance, l = scale

σ_ε = noise variance

Functions f(x) sampled by a prior with fixed hyper-parameters: σ_f = 1, l = 0 and σ_ε = 0 and zero mean.

Prior for the distribution of values f(x):

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

Conditioning with observed data:

$$f_* \mid x_*, \mathbf{x}_0, \mathbf{y}_0 \sim \mathcal{N}(\mu_*, \sigma_*)$$

Predicted value in x_*

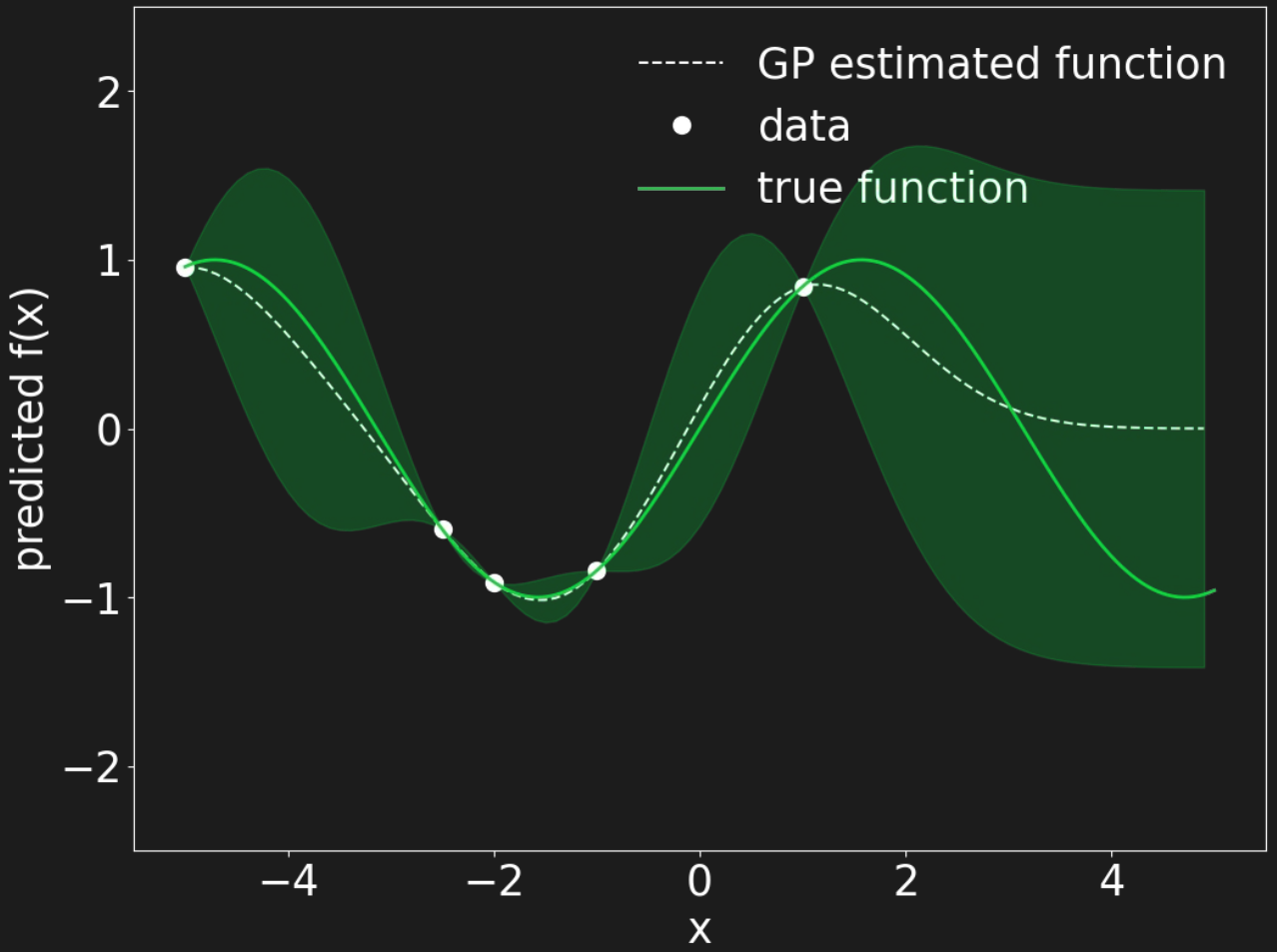
$$\mu_* = k(x_*, \mathbf{x}_0)^T (k(\mathbf{x}_0, \mathbf{x}_0) + \sigma_\varepsilon^2 \bar{\mathcal{I}})^{-1} \mathbf{y}_0$$

$$\sigma_* = k(x_*, x_*) - k(x_*, \mathbf{x}_0)^T (k(\mathbf{x}_0, \mathbf{x}_0) + \sigma_\varepsilon^2 \bar{\mathcal{I}})^{-1} k(x_*, \mathbf{x}_0)$$

Prior covariance

Info that observation gives us about the function

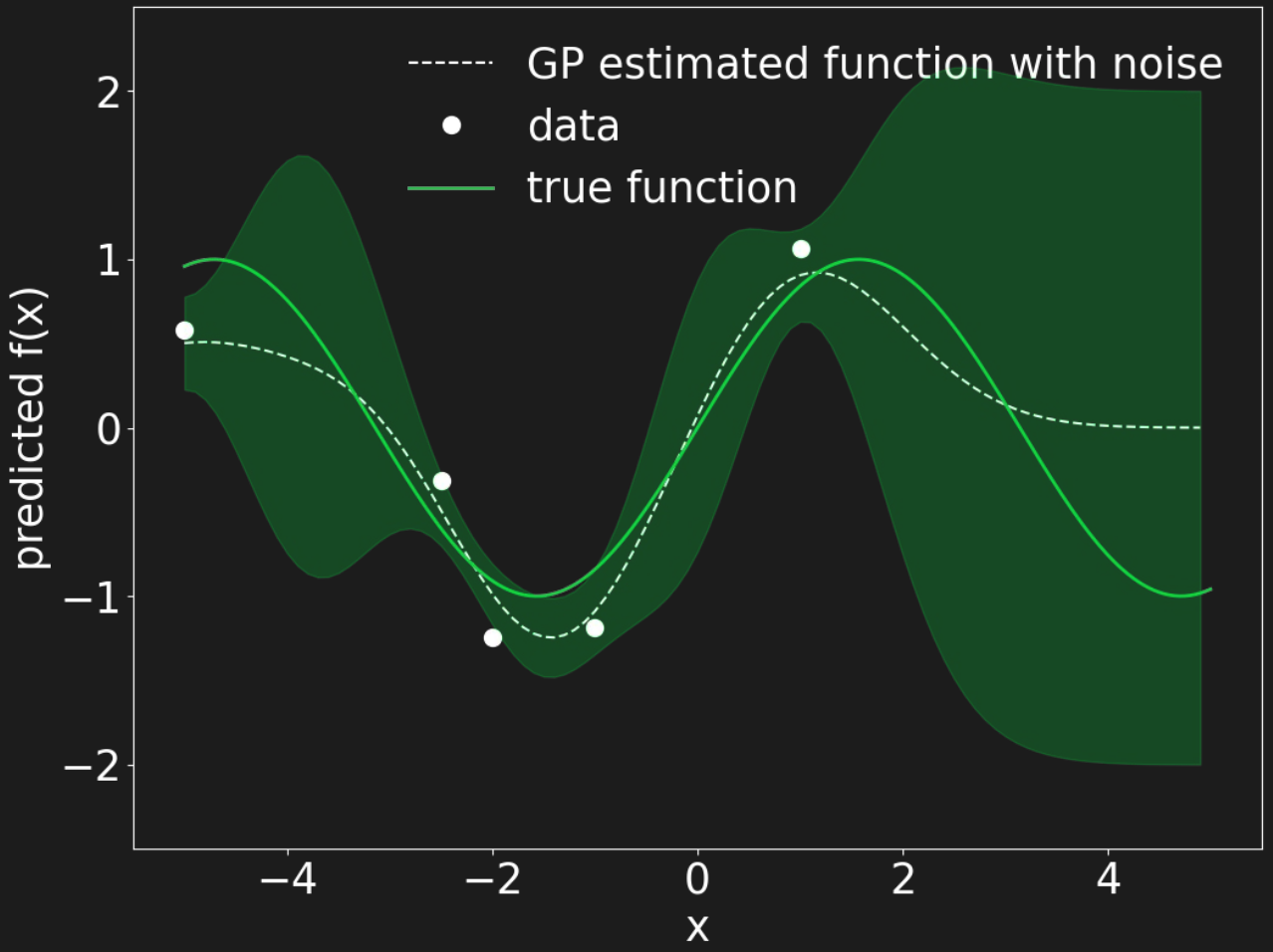
Free noise signal



Posterior obtained from the conditioning of the prior over the white data point. The white dashed curve represents $\mu*(x_*)$ and the shaded area $\pm 2\sigma_*(x_*)$.

The hyper-parameters were fixed: $\sigma_f=1$, $l=0$, $\sigma_\varepsilon=0$.

Noisy signal



Posterior obtained from the conditioning of the prior over the white data point. The white dashed curve represents $\mu_*(x_*)$ and the shaded area $\pm 2\sigma_*(x_*)$.

The hyper-parameters were fixed: $\sigma_f=1$, $l=0$ and $\sigma_\varepsilon=0.4$

Which are the best hyper-parameters?

Parameters: they **define the model** and can be learned from the data (e.g. coefficients of a linear model or the weights in a neural network).

Hyper-parameters: they are **external to the model** and cannot be estimated from the data (like the learning rate for neural networks). However, they can be **optimized** in 2 ways:

Fully Bayesian framework:

→ non-gaussian likelihood
→ rely on **Monte Carlo methods** (computationally ~~expensive~~)

or

Maximizing the log-likelihood:

Optimization + matrix inversion

Gaussian Processes are non-parametric models.

Likelihood: given some parameters, the higer it is, the more likely it will be that we sample that observed data.

$$\log p(\mathbf{y}|\mathbf{x}_0) = -\frac{1}{2}\mathbf{y}^T(k(\mathbf{x}_0,\mathbf{x}_0)+\sigma_\varepsilon^2\bar{\mathcal{I}})^{-1}\mathbf{y} - \frac{1}{2}\log|k(\mathbf{x}_0,\mathbf{x}_0)+\sigma_\varepsilon^2\bar{\mathcal{I}}| - \frac{N}{2}\log 2\pi$$

Data fit:
It decreases monotonically with the length scale (l) → less flexible model → worse fit

Minus complexity penalty:
The simpler the model (big l scale) the bigger it becomes

N=number of training points

Likelihood: try to favour the least complex model able to explain the data (automatic Occam Razor).