CDAP_report_PARIS

January 21, 2024

1 Introduction

Surface load can deform the lithosphere and change its stress properties. Some studies have shown that the large amount of water dump during typhoons and hurricanes can lead to significant deformation of the lithosphere (e.g. Milliner, 2018). The aim of this project is to use modelled surface displacements to invert the load distribution of the water dumped by the Hilary Hurricane in California, in order to determine whether the actual load distribution can realistically be inverted. If so, the inverted load distribution may then in a second stage be use to compute displacements in any part of the medium and thus provide us the strain and stress changes in the medium induced by the water load.

Several mathematical expressions exist to model the displacements from surface loading. In our implementation we chose the model from Marmo and d'Urso (2013), which provides an analytical solution to model the displacement at any location of the medium induced by a polygonal source with any number of vertices in a Cartesian coordinate system, leaving us easy options for meshing.

2 Method

2.1 Forward problem

2.1.1 Equations

The expression from d'Urso and Marmo allows to model the displacement in three directions (x, y and z) of a point at any location in space that is generated by a source of polygonal shape (with any number of vertices) and of constant load. For a given source and location, the three components of the displacement dx, dy and dz can be computed with the following equations:

$$\begin{pmatrix} dx \\ dy \end{pmatrix} = \frac{p_z}{4\pi\mu} \sum_{i=1}^{n} \begin{pmatrix} y_{i+1} - y_i \\ x_i - x_{i+1} \end{pmatrix} \left(\frac{\mu}{\lambda + \mu} S_{1i} + S_{2i} \right) \tag{1}$$

$$dz = \frac{p_z}{4\pi\mu} \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i) \{ S_{2i} - \frac{\mu}{\lambda + \mu} [S_{3i} - z\alpha(\mathbf{0})] \} \tag{2}$$

Where μ and λ are the Lame parameters, assumed to be constant in the medium for the inversion, and p_z is the load of the source. Because the mathematical developpment behind all the other parameters is quite heavy, we will not enter in the details and will simply note that there is a linear relationship between the displacements dx, dy and dz and the source load p_z. Because the relationship between the load p_z and the displacements is linear, the displacements $\mathbf{d}$ recorded in

3D at all by each station due to the contribution of discrete sources of constant load $\mathbf{p}$ can be expressed as

$$\mathbf{Gp} = \mathbf{d} \tag{3}$$

Where G is a matrix of size [N,M], where N is the number of measurements (3 times the number of stations) and M is the number of discrete sources, *i.e.* the number of parameters. This linear system is at the core of the inversion process detailed in the inverse problem section.

2.1.2 Implementation

The implementation of the forward problem is based on few functions which perform the integrals that were not detailed in the Equations section, and which I translated from the Matlab implementation of d'Urso and Marmo to Python. The functions to compute these integrals are available in the module **load2disp.py** located in the **software** directory. This module also contains the function *load2disp* which computes the displacement recorded at coordinates x, y and z, induced by a polygonal source of any shape and constant load. In order to compute the displacements from a more complex source with non-constant load, one can split the original source into multiple sources of different load and loop over their contribution to the recorded displacement, thanks to the linearity of the problem.

Alternatively, one can build the $\mathbf{G}$ matrix, allowing to compute the displacements with the matrix product : $\mathbf{Gp} = \mathbf{d}$. While this approach requires to have more memory to store explicitly the matrix, it allows one to save it and then use it for the inversion. The function to build $\mathbf{G}$ is available in the **disp2load.py** module located in the **software** folder as the function *build_G*.

2.2 Inverse problem

2.2.1 Equations

Solving a linear inverse problem differs depending on its well-posedness and its conditionning. For our study we focus on the case where the problem is overdetermined, that is to say that there are more data than parameters to invert for, which is a realistic scenario in highly instrumented regions, which is the case for California. For an overdetermined problem, the Least-square solution to our problem is written as :

$$(\mathbf{G^T G})^{-1}\mathbf{G^T d} = \mathbf{p} \tag{4}$$

However, the GPS stations are not spread uniformly across California, leading to regions with low station density (hence low resolution), and due to noise the inversion tends to be unstable, therefore some physics must be added through regularization. The general form of the solution to minimize the misfit $\mathbf{J}$ then becomes :

$$\mathbf{p} = (\mathbf{G^T G} + \mathbf{C m^{-1}})^{-1}\mathbf{G^T d}, \tag{5}$$

where Cm^{-1} is an inverse covariance matrix. Throughout this project three different regulariations were investigated, namely the total variation (TV), the Laplacian and the inverse Gaussian. All

three regularizers aims at smoothing the inverted load distribution, to ensure that there is no large load difference between closeby regions.

- Laplacian regularization : penalty on $||\nabla \mathbf{p}||^2$ which acts as an inverse covariance matrix and ensures that neighbours loads are not too different

$$J_\Delta = min(||\mathbf{Gp} - \mathbf{d}||^2 + \lambda||\nabla \mathbf{p}||^2) \tag{6}$$

$$\mathbf{p} = (\mathbf{G^T G} + \ \ )^{-1}\mathbf{G^T d} \tag{7}$$

- Gaussian Regularization : use the inverse of a Gaussian matrix $\mathcal{G}^{-1}$ as the inverse covariance matrix for the regularization

$$\mathbf{p} = (\mathbf{G^T G} + \ \mathcal{G}^{-1})^{-1}\mathbf{G^T d} \tag{8}$$

- Total variation : penalty on $||\nabla \mathbf{p}||_1$

$$J_{TV} = min(||\mathbf{Gp} - \mathbf{d}||^2 + \lambda||\nabla \mathbf{p}||_1) \tag{9}$$

With total variation, the inversion becomes non-linear because of the absolute value, hence for this specific regularization, we use the soft thresholding algorithm from Mallat, 2009, which is detailled in the algorithm section.

In these previous expressions the value of λ is theorically obtained through L-curve, which is obtained by plotting norm of the misfit as a fucntion of the norm of the regularizer. In this case the value of λ is taken as the value which leads to the best trade-off between fitting the data and minimizing the regularizer, which in our implementation is based on the gradient of the load distribution, and therefore occurs when the load distribution is smooth. This optimal trade-off between the two terms is typically found by choosing the value of λ that is located in the elbow of the L-curve. In practice, to find this value we therefore rely on the second derivative of the misfit as a function of the regularizer term, since it theorically displays a peak at the location of the break point of the elbow. This peak comes from the fact that the L-curve has an almost constant slope before and after the break point, hence its derivative with respect to the regularizer contains 2 constant values separated by a region with a large slope located at the break point of the elbow. By derivating a second time, the regions where the first derivative was constant become null and a peak appears in the region with a large slope.

While Laplacian and Gaussian regularization remains in the field of linear inversion, it is still possible to apply non-linear inversion algorithms to invert for the surface load. The advantage of using a non-linear local optimization algorithm is that it allows to easily add bounds to the inverted solution, which is of great interest in the case study of Hilary Hurricane, since part of the sources to invert the load of are actually located in the Pacific ocean. In the case of a non-linear inversion, an initial guess p_0 is used and improved iteratively so that $f(p_k) < f(p_{k-1})$ where $f(p_k)$ is the misfit at iteration k. For that purpose, in the general form, at each iteration a descent direction Δp is computed as :

$$\Delta p = -Q_k \nabla f(p_k), \tag{10}$$

where Q_k is an approximation of the inverse Hessian and $\nabla f(p_k)$ is the gradient of the misfit function. Then at each iteration p_k is computed as :

$$p_k = p_{k-1} + \alpha_k \Delta p, \tag{11}$$

where α_k, the step size, is determined with a linesearch backtracking algorithm. Throughout this project, we investigated 3 non-linear inversion algorithms; the steepest descent ($Q_k = I$), the Non-Linear Conjugate Gradient (NLCG) and the Limited-BFGS (L-BFGS). Their algorithms are described in the next section, and their implementation is available in the **disp2load.py** module located in the **software** folder.

2.2.2 Algortihms

Linesearch algorithms Non-linear inversion implies iterative convergence toward the set of parameters that minimizes the misfit. Such iterative process requires a step size α adequatly chosen to ensure convergence toward a minimum. For that purpose, the value of α should ideally be determined based on the Wolfe condtions with a backtracking line search algorithm. However, the curvature condition, which provides the lower bound for α, cannot be fulfilled in some occurences. In this case, the step size should be obtained with a backtracking line search that solely enforces the Armijo condition (sufficient decrease). Below are the algorithms to perform the regular line search and the line search that solely enforces the Armijo condition.

Wolfe conditions linesearch Given α_0

set $c_1 \longleftarrow 0.0001$ and $c_2 \longleftarrow 0.9$

set $k \longleftarrow 0$

set $cond2 \longleftarrow False$

while $cond2$ is $False$

if $f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$

if $\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$

$cond2 \longleftarrow True$

else

$\alpha_{k+1} \longleftarrow 10 \times \alpha_k$

end if

else

$\alpha_{k+1} \longleftarrow \alpha_k / 2$

end if

$k \longleftarrow k + 1$

end while

Armijo condition linesearch Given α_0

set $c_1 \longleftarrow 0.0001$

set $k \longleftarrow 0$

set $cond \longleftarrow False$

while $cond$ is $False$

if $f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$

$cond \longleftarrow True$

else

$\alpha_{k+1} \longleftarrow \alpha_k / 2$

end if

$k \longleftarrow k + 1$

end while

Inversion algorithms

Steepest descent Given x_0 and ϵ

Evaluate $\nabla f_0 = \nabla f(x_0)$;

Set $k \longleftarrow 0$;

while $\nabla f_k^T \nabla f_k \geq \epsilon$

Evaluate ∇f_k;

$p_k \longleftarrow -\nabla f_k$

Compute α_k and set $x_{k+1} = x_k + \alpha_k p_k$

$k \longleftarrow k + 1$

end (while)

Non-Linear Conjugate gradient (NLCG) using the Fletcher-Reeves method Given x_0 and ϵ

Evaluate $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0)$;

Set $p_0 \longleftarrow -\nabla f_0, k \longleftarrow 0$

while $p_k^T p_k \geq \epsilon$

Compute α_k and set $x_{k+1} = x_k + \alpha_k p_k$

Evaluate ∇f_{k+1};

$\beta_{k+1} \longleftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$;

$p_{k+1} \longleftarrow -\nabla f_{k+1} + \beta_{k+1} p_k$;

$k \longleftarrow k+1$

end (while)

Limited-memory BFGS (L-BFGS) Choose starting point x_0, integer $m > 0$;

$k \longleftarrow 0$;

repeat

$q \longleftarrow \nabla f_k$;

if $k = 0$

$p_k \longleftarrow -q$

else

for $i = k-1, k-2, ..., k-m$

$\alpha_i \longleftarrow \rho_i s_i^T q$;, where $s_k = x_{k+1} - x_k$, $\rho_k = \frac{1}{y_k^T s_k}$ and $y_k = \nabla f_{k+1} - \nabla f_k$,

$q \longleftrightarrow -\alpha_i y_i$;

end(**for**)

$\gamma_k \longleftarrow \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$

$H_k^0 \longleftarrow \gamma_k I$

$r \longleftarrow H_k^0 q$;

for $i = k-m, k-m+1, ..., k-1$

$\beta \longleftarrow \rho_i y_i^T r$;

$r \longleftarrow r + s_i(\alpha_i - \beta)$;

end(**for**)

stop with result $H_k \nabla f_k = r$

Compute $p_k \longleftarrow -r$

Compute $x_{k+1} \longleftarrow x_k + \alpha_k p_k$, where α_k is chosen to satisfy the Wolfe conditions;

if $k > m$

Discard the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage;

Compute and save $s_k \longleftarrow x_{x+1} - x_k$, $y_k = \nabla f_{k+1} - \nabla f_k$;

$k \longleftarrow k+1$;

until convergence

Soft thresholding Choose starting point x_0 and coefficient ξ for the step size γ;

$k \longleftarrow 0$;

repeat

$\gamma \longleftarrow \frac{\xi}{max(|\mathbf{G^T G}|)}$

$\bar{x}_k \longleftarrow x_k + \gamma(\mathbf{G^T} d_{obs} - \mathbf{G^T G} d_{cal})$,

for i=1, 2, … n

$x_{k+1}[i] \longleftarrow \bar{x}_k[i] max\left(1 - \frac{\gamma\lambda}{|\bar{x}_k[i]|}, 0\right)$

$k \longleftarrow k + 1$

until convergence

2.2.3 Implementation

All these inversion algorithms are implemented in the **disp2load.py** module located in the **software** folder. The *disp2load* function inside the module serves as a wrapper for the functions dedicated to each algorithm.

3 Results

3.1 Benchmarking the forward problem

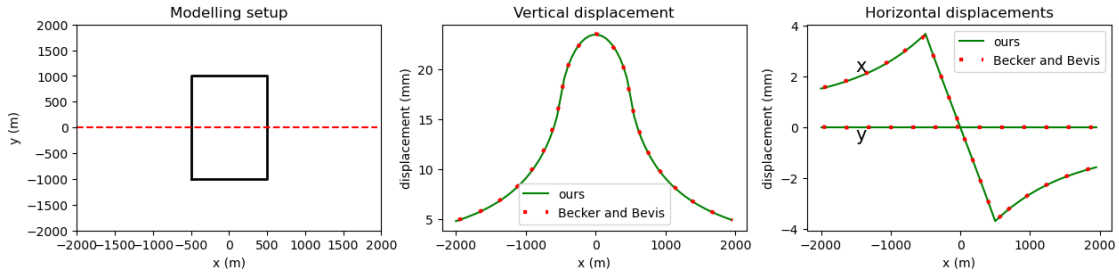
To verify that our python implementation of the forward problem is working, we first need to replicate examples from the litterature, and compare the result of our model to them.

3.1.1 1D verificaiton with the results from Becker and Bevis, 2003

We first try to replicate the displacements generated along a 1D profile from a single source as was done in Becker and Bevis, 2003.

We can see that the modelled displacements with our implementation perfectly fit the solution from Becker and Bevis, 2003, which shows that our implementation works as expected with a single simple source.

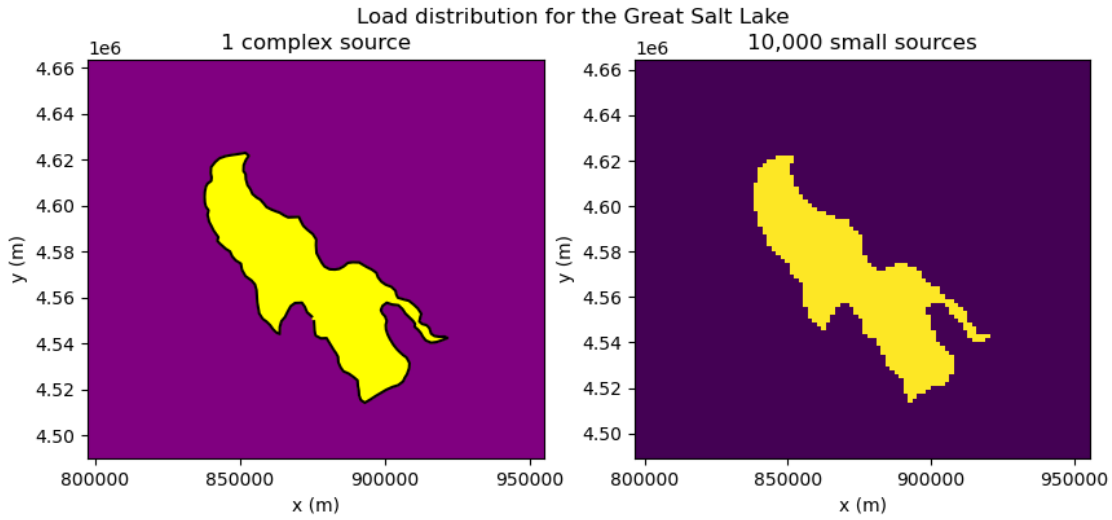
[2]: `<matplotlib.legend.Legend at 0x7f9bb07010d0>`



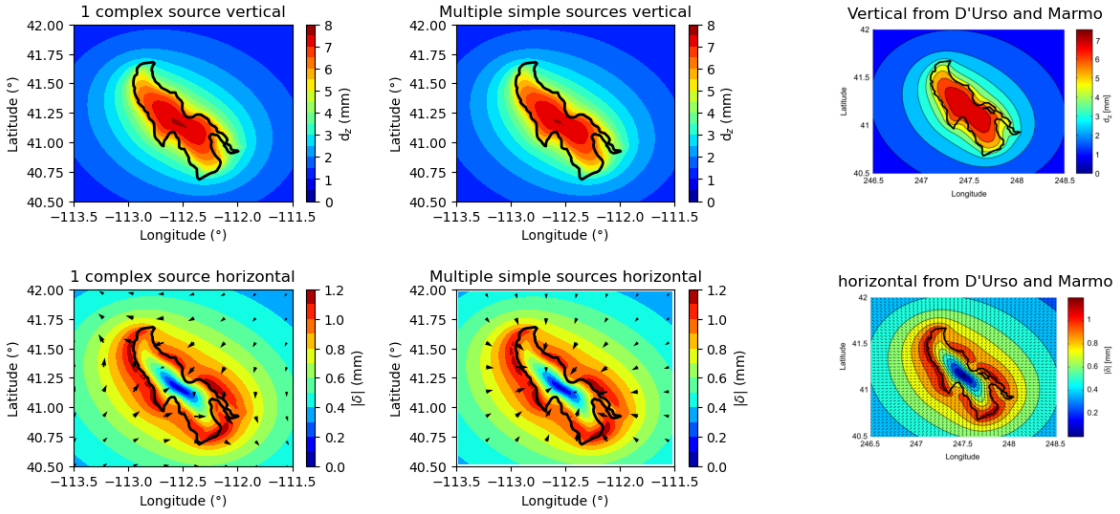
3.1.2 2D verification with the results from d'urso and Marmo

To verify that our implementation also works for measurements in 2D and with multiple sources, we now try to replicate the result obtained in D'Urso and Marmo, 2013 with two different approaches. For that purpose, we use the same parameters, that is to say E = 75GPa, $\nu = 0.25$ and a water, $\rho_{water} = 1150$ kg/m^3 and a constant water level of 1 m, and we use the same complex source shape corresponding to the The Great Salt Lake in Utah. While we use the same source shape as in D'Urso and Marmo, we create the load distribution with 2 approaches; on one hand we make a load distribution where there is a single complex source and and on the other hand we make a load distribution composed for 10,000 small rectangular sources, where sources that are inside the borders of the lake have a constant load equivalent to 1 m of water, while the sources outside of the lake have a null load. This is illustrated by the figure below.

[3]: `Text(0.5, 0.98, 'Load distribution for the Great Salt Lake')`



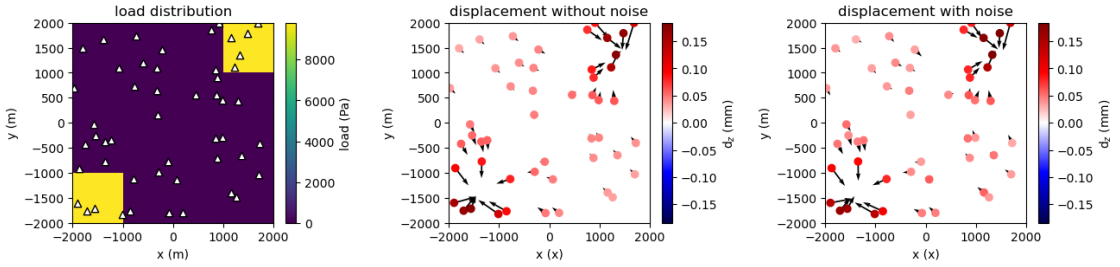
Using these two different load distributions we model the displacements and compare them to the ones modelled by D'Urso and Marmo (figure below). In the following figure, the top panels represent the modelled vertical displacements, while the colormap of the bottom panels represent the norm of the horizontal displacements, and the vectors their direction. We can see that the displacements computed using a single complex source and 10,000 small simple sources is identical, showing that the contribution of a complex source distribution to the displacement can be approximated by summing the contribution of multiple small sources. In addition, one can notice that the modeled displacements are identical to the ones from the litterature, which confirms that our implementation is working properly.



3.2 Benchmarking the inverse problem

3.2.1 Simple load distribution with low noise

Now we have verified that the forward model is working as intended, we can use it to model displacements, which can then be used to benchmark the inversion. For that purpose we first start with a simple load distribution and by applying very small noise to the data (normal distribution with a standard deviation equal to 5 % of the maximum displacement recorded for each component). The figure below shows the simple load distribution with the location of the virtual stations (green triangles), as well as the modelled displacements induced by the sources. For the displacements maps, the colormap represents the vertical displacement, while the arrows indicate the horizontal displacements.



Using the modelled displacements with added noise we now invert the load distribution with the Laplacian, Gaussian and TV regularizations. In order to find the optimal value of λ (and σ for inversion with a Gaussian), we create the L-curves, and to verify that the L-curve lead to the optimal we also compute the norm of the residual between the inverted load distribution and the one used to compute the displacements ($||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$), which is illustrated by the figure below. In this figure, each column is for a different regularization approach, the top row displays the norm on the error between the true load distribution and the inverted one, the second row represents

the L-curves and the third row represents the second derivative of the L-curve with respect to the norm of the regularizer, which is used to determine the value of λ associated with the elbow of the L-curve.

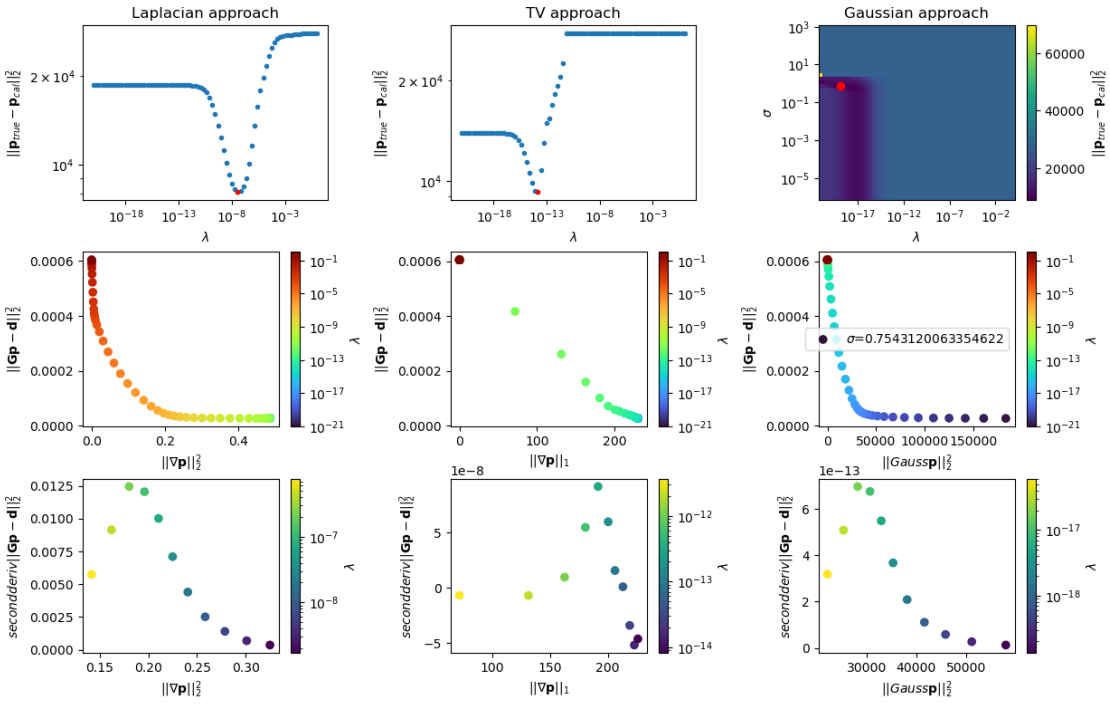
In order to assess the closeness of the value of λ derived from the L-curve to the optimal one provided by the the minimum of $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$, we define the logarithmic error as :

$$\epsilon = |\log 10(\frac{\lambda_{L-curve}}{\lambda_{optimal}})| \tag{12}$$

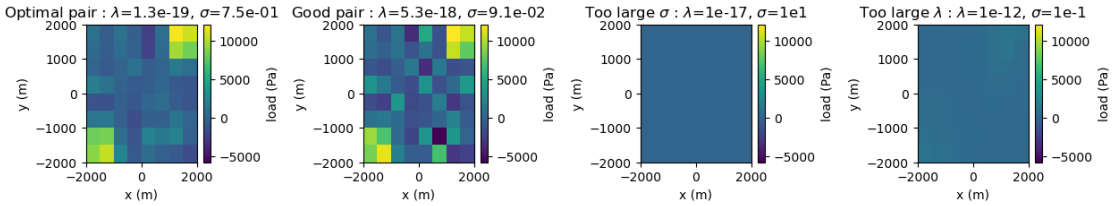
We choose this error over a more common relative error because in some occurences the value of λ estimated with the L-curve can be smaller than the optimal one by several orders of magnitude, which results in underestimated error. With our expression of the logarithmic error, a good estimate of λ with the L-curve leads to an error close to 0, while if either $\lambda_{optimal}$ or $\lambda_{L-curve}$ is larger than the other the error will increase accordingly.

With the Laplacian and TV approaches we find very similar results where the logarithmic error between $\lambda_{L-curve}$ and $\lambda_{optimal}$ is between 1 and 2. On the other hand, looking at the top panel with the Normalized Gaussian approach, one can see that for σ ranging from $10^{-7} to 10^{0}$, the optimal value of λ remains constant, and then when σ becomes too large, the minimun disappears and the error remains large for any λ. One can also notice that for a standard deviation σ_{critic} in between these regions, the error $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ actually decreases significantly and $\lambda_{optimal}$ actually shifts slightly. When choosing σ as the optimal one and using the L-curve to estimate λ, we find a logarithmic error greater than 2, which means that the L-curve was not able to find a value of λ with the same effectiveness as with the other methods.

```
log error on $\lambda$ with Laplacian approach : 1.836238745045882
log error on $\lambda$ with TV approach : 1.3291139240506329
log error on $\lambda$ with Gaussian approach : 2.1265822784810133
```



In order to understand the impact of λ and σ on the inversion we show in the following figure the results of the inversion with various parameter pairs. For that purpose we specifically focus on 4 pairs; the optimal pair ($\sigma_{optimal}$ and $\lambda_{optimal}$), a "good" pair (not optimal σ but optimal λ for the selected value of σ), a pair with σ too large and a pair with λ too large (uniform blue region in previous figure). As one can see, when either σ or λ is too large, the inversion is unstable and the load distribution is null. This result can be interpreted as being the result of excessive smoothing; as when σ increases the width of the Gaussians that serve as the inverse covariance matrix also increases. With the "good" pair of σ and λ we find that the result of the inversion is not great considering the low noise level in the data and that the optimal value of λ was used. On the bright side, when choosing the optimal value of σ, and the optimal value of λ associated, the inverted load distribution is quite good. This result reveals that in order to accurately invert the load distribution, one needs to use the optimal standard deviation as well as the optimal regularization coefficient.



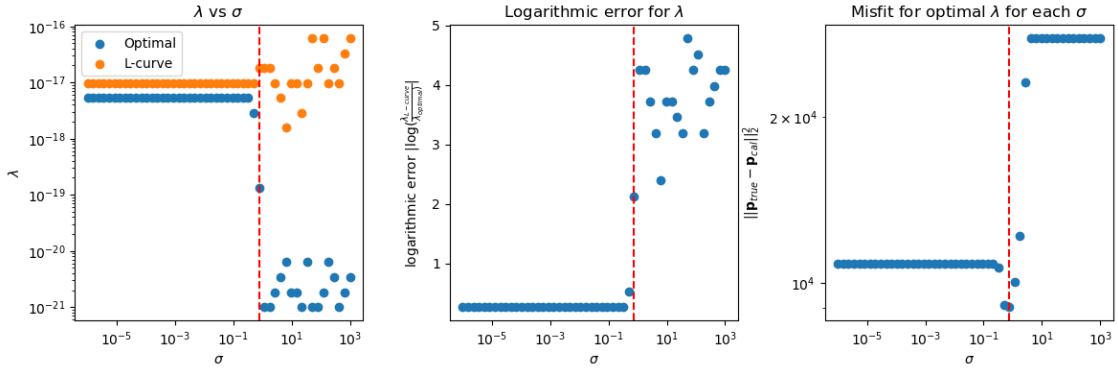
Considering that $\sigma_{optimal}$ is close to σ_{critic}, and that it is possible to tell when the inversion is unstable due to σ being too large (null load distribution), it seems reasonnable to consider the largest value of σ that still leads to a stable inversion as $\sigma_{optimal}$. In addition, since $\lambda_{optimal}$ remains constant for small values of σ and starts slightly shifting once approaching $\sigma_{optimal}$, one may estimate $\sigma_{optimal}$ based on the evolution of the predicted λ as a function of σ. Because in practice one does not know $\lambda_{optimal}$ we must investigate the ability of the L-curve to correctly estimate $\lambda_{optimal}$ for large values of σ, in order to make sure that we actually observe the constant $\lambda_{optimal}$ for small σ, and the shift in $\lambda_{optimal}$ once $\sigma_{optimal}$ is reached.

For that purpose we represent in the left panel of the figure below $\lambda_{optimal}$ and $\lambda_{L-curve}$ as a function of σ, as well as the logarithmic error in the central panel. From these, it appears that when σ is small, its value of does not actually impact the logarithmic error (both $\lambda_{optimal}$ and $\lambda_{L-curve}$ remain constant and the logarithmic error is small), while when σ is larger than $\sigma_{optimal}$, the L-curve starts overestimating $\lambda_{optimal}$, where the larger σ, the more $\lambda_{optimal}$ is overestimated overall.

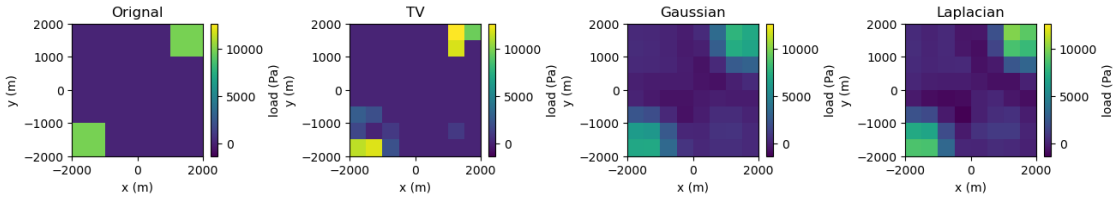
The right panel shows the norm of the error between the true and inverted load distribution as a function of σ, where for each σ the error is taken at the optimal λ (same idea as for the "good" pair in the above figure). In this figure we can clearly see that $\sigma_{optimal} \approx 0.75$ (red dashed vertical line in all panels), and comparing with the left panel we can note that for this σ, the L-curve starts predicting a value for λ different than for $\sigma \leq \sigma_{optimal}$. This shows that the L-curve can allow to estimate $\sigma_{optimal}$, by plotting the evolution of $\lambda_{L-curve}$ as a function of σ. However, at $\sigma_{optimal}$, $\lambda_{L-curve}$ diverges from $\lambda_{optimal}$, making the L-curve unable to find the optimal parameters pair.

```
The lowest error between the true and inverted load distributions is obtained
for $\sigma$ =  0.7543120063354622
For sigma = 0.7543120063354622, the logarithmic error of lambda is
2.1265822784810133
```

[8]: `<matplotlib.lines.Line2D at 0x7f9ba5d91610>`

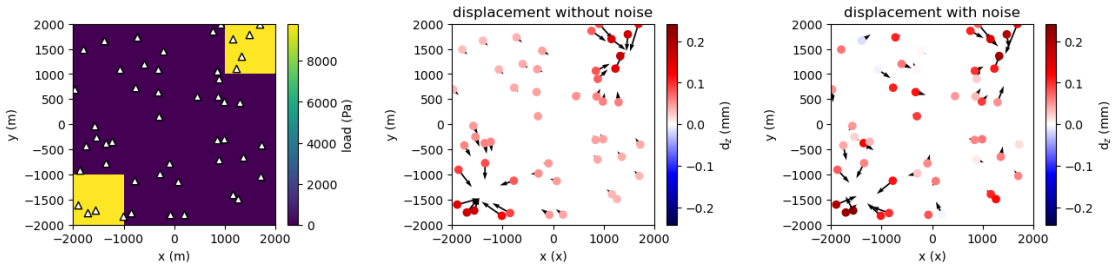


The result of the inversion using $\lambda_{L-curve}$ with the Laplacian, TV and Gaussian regularization approaches is shown in the following figure. We can see that all approaches reproduce fairly well the original load distribution. One can note that while Laplacian finds the most realistic load distribution at the sources (pixels of the sources have more uniform values), total variation better constrains spacially the sources (null load in the central region). Meanwhile, the load distribution inverted with the Gaussian approach is fairly similar to the one derived from the Laplacian, but with worse estimate of the load (consistant with the higher logarithmic error on $\lambda_{L-curve}$ for the Gaussian regularization).



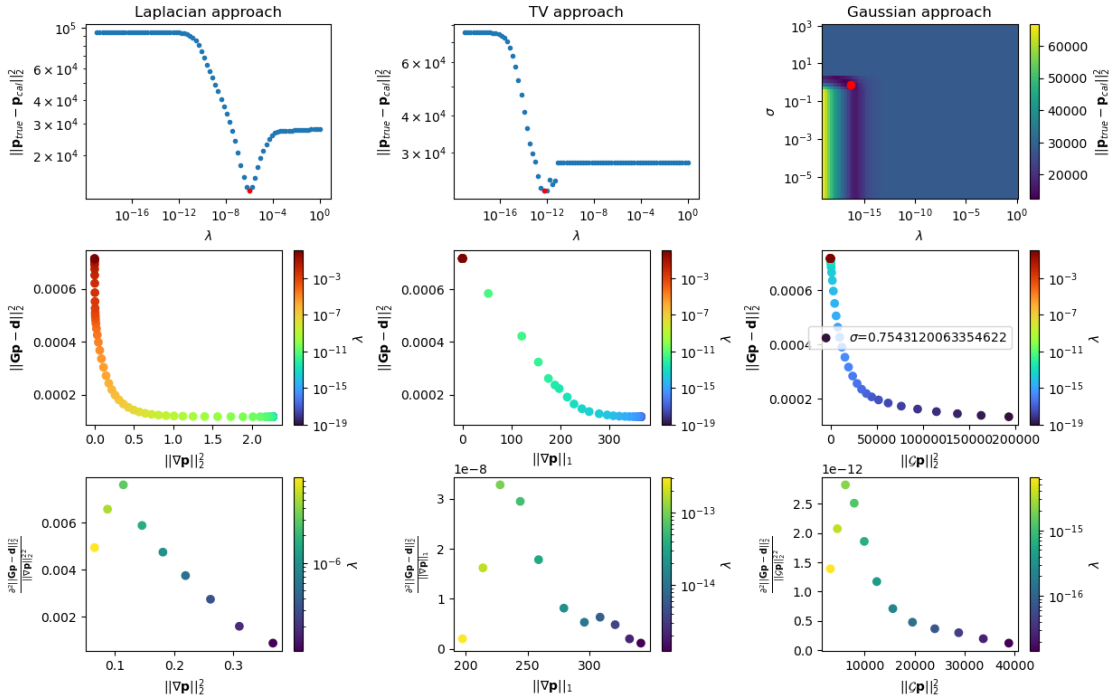
3.2.2 Simple load distribution with large noise

Since we are ultimately interested in the performance of the inversion scheme with low Signal-to-Noise Ratio (SNR), we also investigate the behavior of these approaches with a larger noise level. For that purpose we use the same load distribution and the same station locations as before, and solely change the standard deviation of the noise to 20% of the maximum displacement recorded along each component.



With the increased noise level on the displacements we find that the logarithmic error between $\lambda_{L-curve}$ and $\lambda_{optimal}$ is lower for all three regularization approaches than it is with lower noise level. The logarithmic error on λ for the Laplacian and TV approaches is lower than 1, and with the Gaussian approach it is lower than 2. One can also note that for all approaches $\lambda_{optimal}$ is larger with the increased noise level than it is for a low noise level, which is the expected behavior (stronger smoothing required when higher noise level). On the other hand, for the Gaussian regularization, we can also notice that $\sigma_{optimal}$ remains the same with large and small noise.

```
log error on $\lambda$ using the L-curve with Laplacian approach :
0.4810126582278489
log error on $\lambda$ using the L-curve with TV approach : 0.7215189873417724
log error on $\lambda$ with Gaussian approach : 1.6835443037974702
```



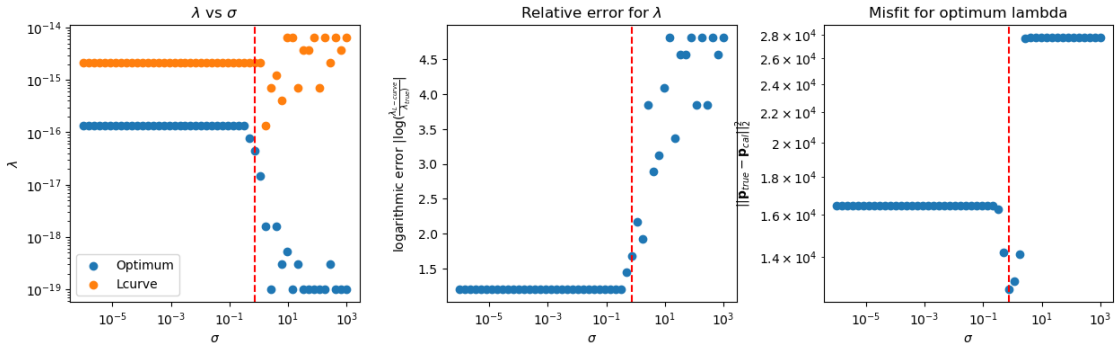
Since we observe similar behavior of the Gaussian approach with large noise level to the one with small noise level (unstability of inversion for large σ, constant $\lambda_{optimal}$ for small σ and small shift of

$\lambda_{optimal}$ at $\sigma_{optimal}$), we conduct the same analsysis to determine wether the evolution of $\lambda_{L-curve}$ can be used to derive $\sigma_{optimal}$ in the following figure.

In the left and central panels, we can see that with this higher noise level, while $\lambda_{L-curve}$ remains constant for low σ, it is significantly different than the optimal one (large logarithmic error) even for small values of σ. In addition, one can note that unlike with low noise level, $\lambda_{L-curve}$ does not start shifting exactly at $\sigma_{optimal}$, but starts shifting for $\sigma \geq \sigma_{optimal}$. Based on these observations it appears that for large noise level, one cannot guess $\sigma_{optimal}$ based on the last value of $\lambda_{L-curve}$ that is identical to the previous ones. However, since $\lambda_{L-curve}$ is constant for small σ, we may still estimate $\sigma_{optimal}$ as being the value above which the inversion is unstable and below which $\lambda_{L-curve}$ is constant as a function of σ.

We previously noted that the logarithmic error of λ is smaller for large noise level than for low noise level. Based on the constantly larger logarithmic error at low σ for large noise level, it is clear that the smaller logarithmic error measured simply comes from the fact that $\lambda_{L-curve}$ remains constant for larger σ which implies that if we chose an even larger noise level the constant logarithmic error at small σ could actually be larger than the one with low noise level. Hence, the L-curve is actually less effective to estimate $\lambda_{optimal}$ with large noise.

```
The lowest error between the true and inverted load distributions is obtained
for $\sigma$ =  0.7543120063354622
For sigma = 0.7543120063354622, the log error of lambda is 1.6835443037974702
```

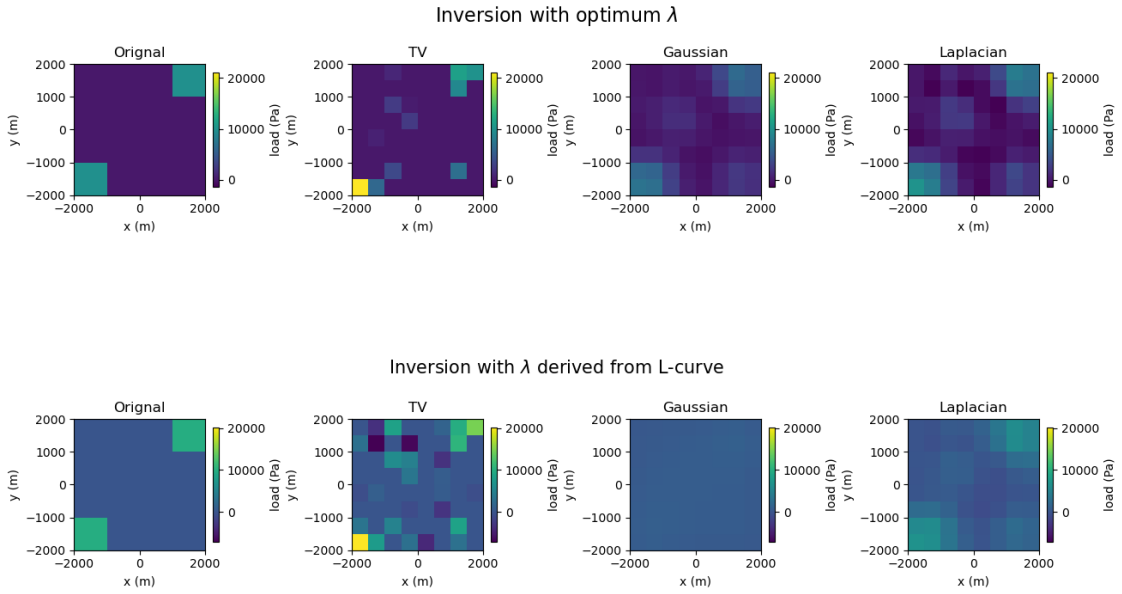


Below we display the result of the inversion for the Laplacian, the TV and the Gaussian, using either $\lambda_{optimal}$ or $\lambda_{L-curve}$.

We can see that with large noise, the total variation starts to perform poorly, as the spacial extent of the sources is not well constrained even with the optimal values of σ and λ. On the other hand, one can see that the Gaussian and Laplacian perform similarly, with the Gaussian performing slightly better than the Laplacian (no negative load in the central region).

When using the values of $\lambda_{L-curve}$, Total Variation performs even worse, and the Gaussian approaches perform starts performing poorly because $\lambda_{L-curve}$ is quite far from $\lambda_{optimal}$. Meanwhile, because $\lambda_{L-curve}$ is quite close to $\lambda_{optimal}$ for the Laplacian regularization (log error of 0.48); it still performs quite well.

From this observation it seems that Laplacian is to favor for large noise level, however Gaussian approach may still be relevant if we find a reliable way to assess find $\lambda_{optimal}$ with this regularization.



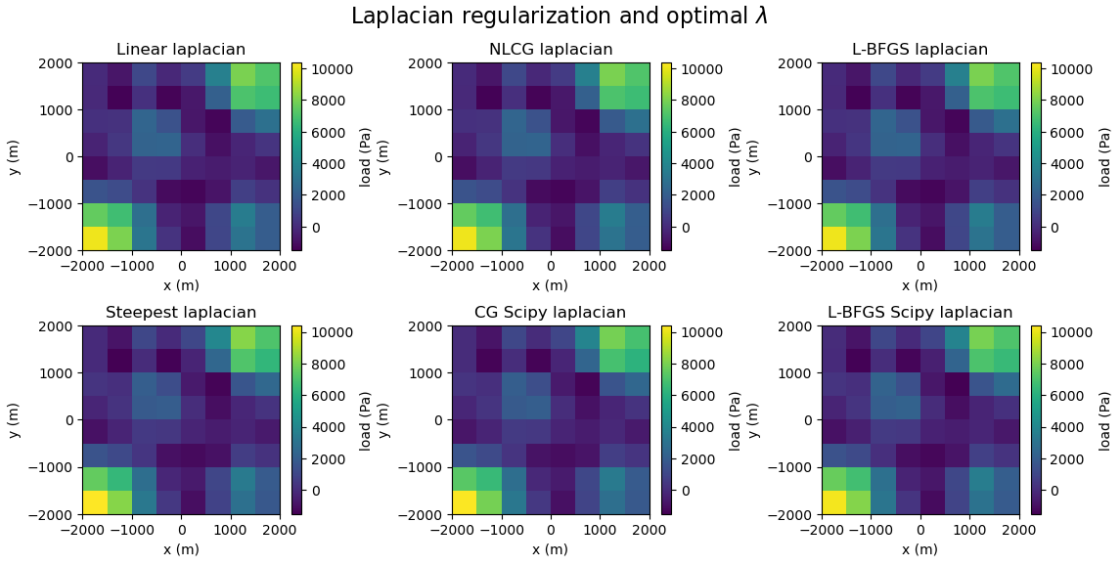
3.2.3 Simple load distribution with strong noise and non-linear approach based on Laplacian and Gaussian

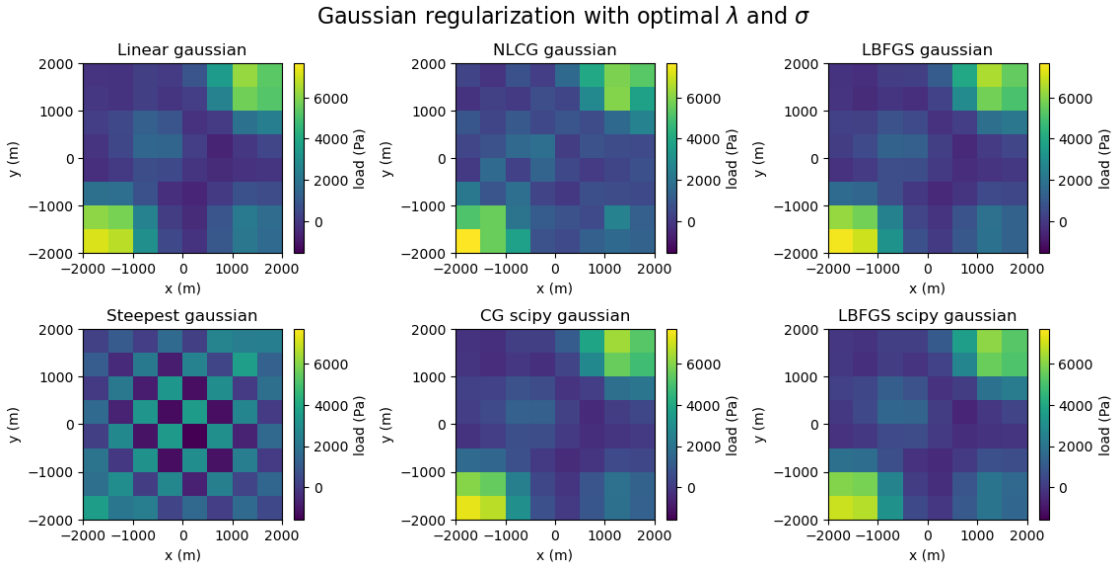
For Laplacian and Gaussian approaches we solved a linear system to invert the load distribution. We can also use non-linear approaches, which then allow to easily add constraints to some regions of the load distribution.

We first compare the result of the inversion non-linear local optimization algorithms to the result of the linear inversion to verify that their results are similar. The following figure displays the result of the inversion for the various algorithms for Laplacian and Gaussian regularization. We also include the result of the inversion some Scipy implementations, to verify that our implementations work properly in case the result of the local optimization differs from that of the linear inversion.

For the Laplacian regularization, we see that all algorithms converged to the same load distribution as the one obtained by linear inversion.

For the Gaussian regularization, we find that the non-linear inversion algorithms all lead to different results. While the steepest descent yields an unstable result, and our NLCG implementation leads to a result somewhat different than the linear inversion, the three other implementations (Scipy's CG and L-BFGS and our L-BFGS) all lead to the same load distribution as the linear inversion. Because there is no Scipy equivalent to the NLCG algorithm, it is difficult to know whether the difference between our NLCG inversion and the linear is due to a bug or to the algorithm itself, however considering that with the Laplacian regularization it yielded the correct load distribution, it is unlikely that it is due to a bug. The same remark can be made for the steepest descent implementation.





3.2.4 Improving the result with a bounded region

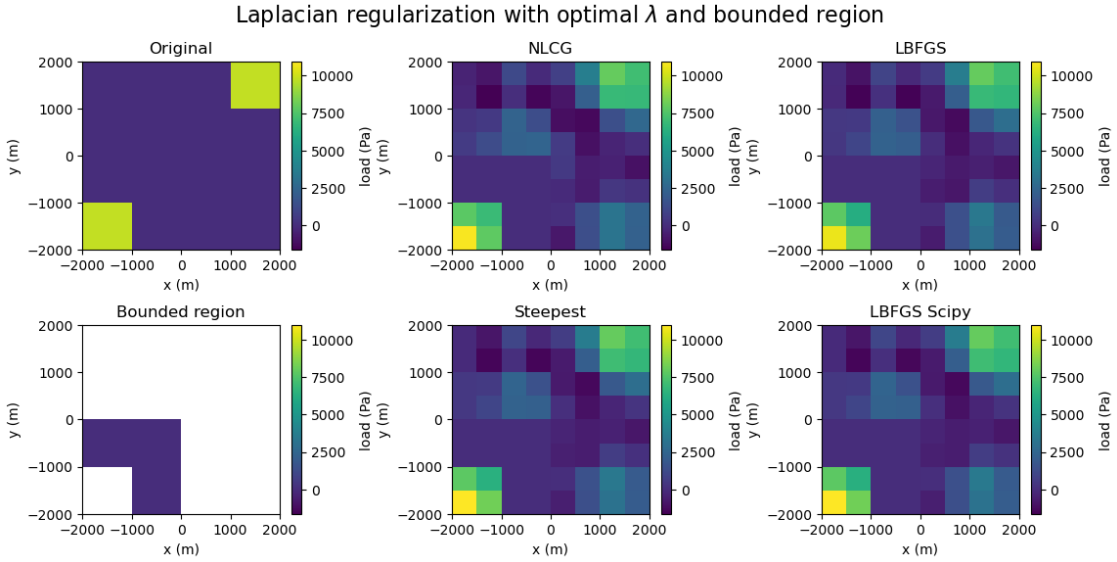
Since we have shown that various local optimization algorithms can be used to obtain the same result as with the linear inversion, we now add bounds to a specific region to see if it can improve the result of the inversion.

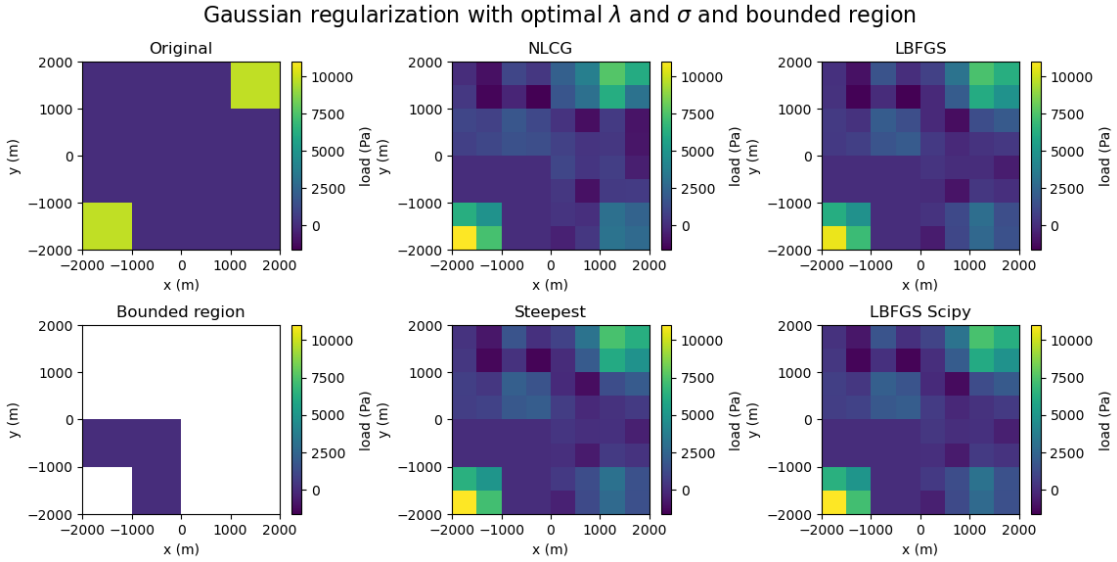
Because we add bounds we need to re-evaluate $\lambda_{optimal}$ and $\sigma_{optimal}$. We estimate these parameters by computing the L2 norm of the difference between the true load and the inverted one using linear inversion, as was previously done. We find that both Laplacian and Gaussian regularization have a different $\lambda_{optimal}$ and $\sigma_{optimal}$ with added bounds.

The optimal value of λ with Laplacian and bounded region is : 9.712740198471168e-07
The optimal value of σ and λ with Gaussian and bounded region are : 0.49417133613238384 and 7.692649574879162e-17 respectively.

For the bounded region we selected the region around the bottom left source, and forced a null load in this region (see bounded region plot in the figures below).

On one hand with the Laplacian regularization, we do not notice clear improvement of the inverted load distribution apart from the improvement in the bounded region. On the other hand, with the Gaussian regularization, clear improvement in the inversion can be observed for steepest descent and NLCG algorithms, since they are able to converge to the same load distribution as the other algorithms.



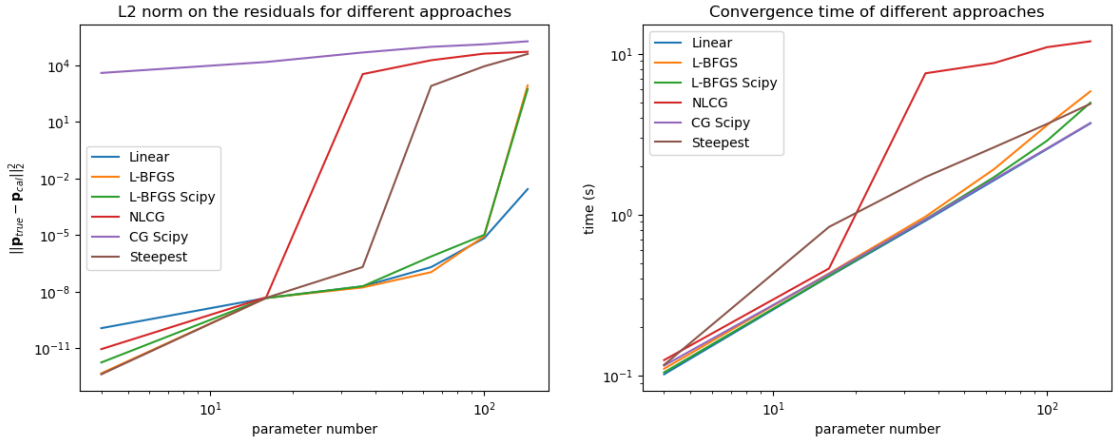


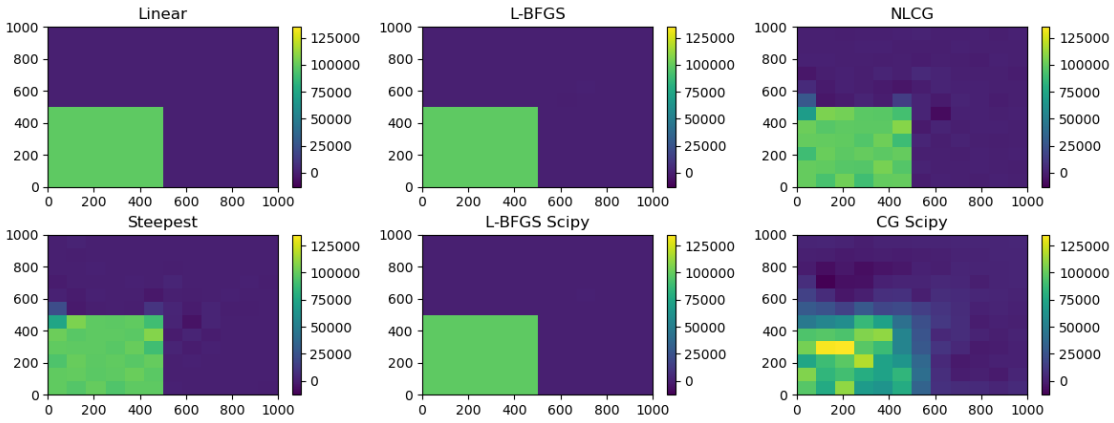
3.2.5 Simple load distribution convergence time analysis for different inversion approaches

Since all non-linear local optimization algorithms appear to work properly and lead to the same result, we need to find the algorithm that performs best in most situations, in order to use the best algorithm to perform inversions with significantly larger number of parameters to invert for. For that purpose we perform a time convergence analysis, where we fix the maximum number of iterations to 10,000.

The first figure below shows the result of the analysis. In the left-hand side panel is represented the L2 norm between the true load distribution and the inverted one as a function of the number of parameters to invert for. In the right-hand side panel, is represented the computation time required to perform the inversion as a function of the number of parameters. For the tested number of sources we find that the linear inversion is systematically the fastest and also yields the closest load distribution. While Scipy's CG algorithm reveals itself as being as fast as the linear inversion, it also systematically leads to the highest error. Our L-BFGS implementation and the one from Scipy are the overall best performing non-linear inversion algorithms, as they have the lowest error and the overall lowest run times. The overall worst performing algorithm seems to be the NLCG, as it has the overall second largest error and is also the slowest overall.

The second figure shows the result of the inversion for 144 parameters with a single source at the bottom left corner and no added noise to the data. On this figure we can clearly see that only the L-BFGS algorithm yields to the same load ditribution as the linear inversion.

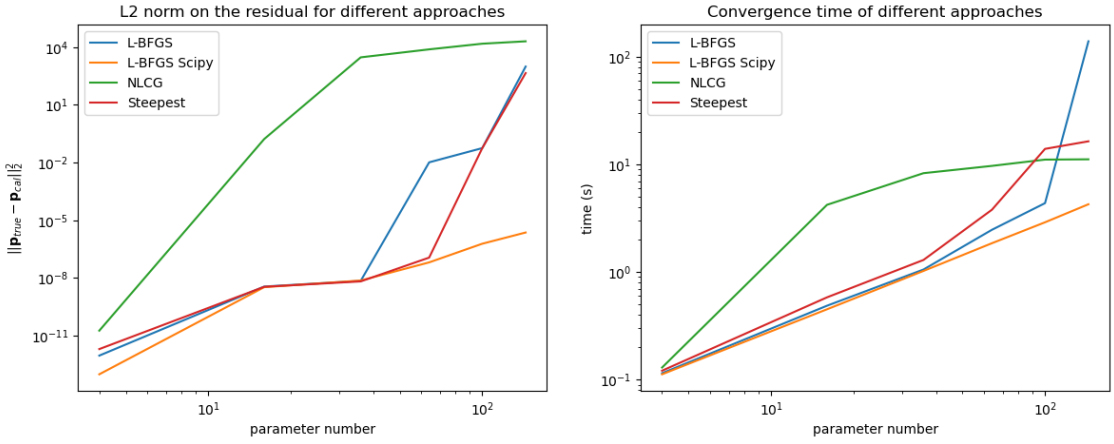


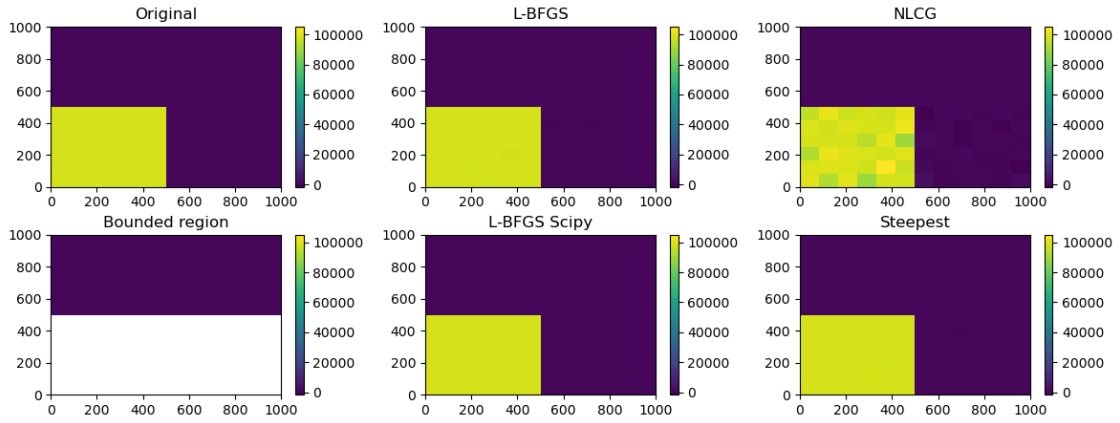


3.2.6 Simple load distribution convergence time analysis for different inversion approaches and with a bounded region

We also perform the same analysis with added bounds, again locking the maximum number of iterations to 10,000. The aim of this analysis is to determine whether L-BFGS still performs best with a bounded region, as well as to see if the bounds impact the convergence time.

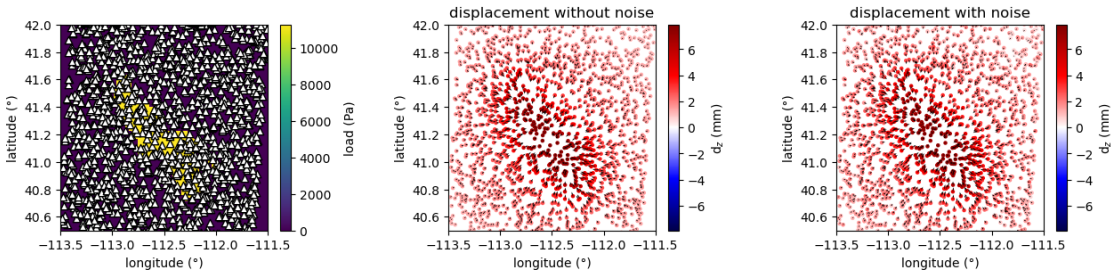
When adding a bounded region at the top of the domain, we find that NLCG still leads to the largest error, while our L-BFGS and Steepest descent implementations have similar error, and which is substantially larger than the error obtained with Scipy's L-BFGS. As for the run time, we can notice that the NLCG is the overall slowest, but surprisingly our L-BFGS implementation becomes much slower to invert for 144 parameters. The convergence time of all of our implementations seems to be significantly impacted by the provided bounds, which is not the case of for scipy's implementation. This result may imply that our implementation of the projection of the bounds is not as efficient as Scipy's (slower convergence of our implementation).





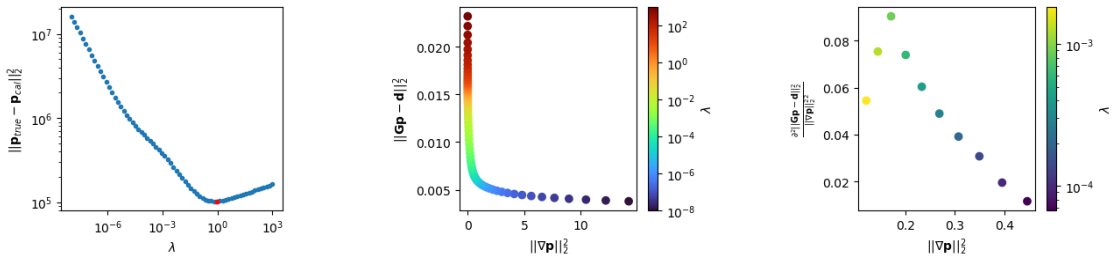
3.2.7 Inversion a of a complex load distribution with small noise and with/without constraints

Because we are interested in inverting a complex load distribution, we need to assess the result of the inversion with a complex source shape. For that purpose we take the example of the Great Salt Lake with its complex source shape and model the displacements at random locations before adding a small noise level to them (standard deviation equal to 5% of the maximum displacement recorded by the selected component), which is shown in the figure below.



We first perform the linear inversion with a wide range of values of λ to create the L-curve, and to compute the L2 norm between the true and inverted load distributions $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ (figure below). We can clearly see that the L-curve finds $\lambda_{L-curve}$ between 10^{-4} and 10^{-3}. However, the L2 norm on the difference between the true and inverted load, in the left panel, shows that $\lambda_{optimal} \approx 1$. This shows that the effectiveness of the L-curve to estimate $\lambda_{optimal}$ is also impacted by the complexity of the load distribution.

```
logarithmic error on $\lambda$ from the L-curve with the Laplacian
regularization is : 3.028985507246376
```

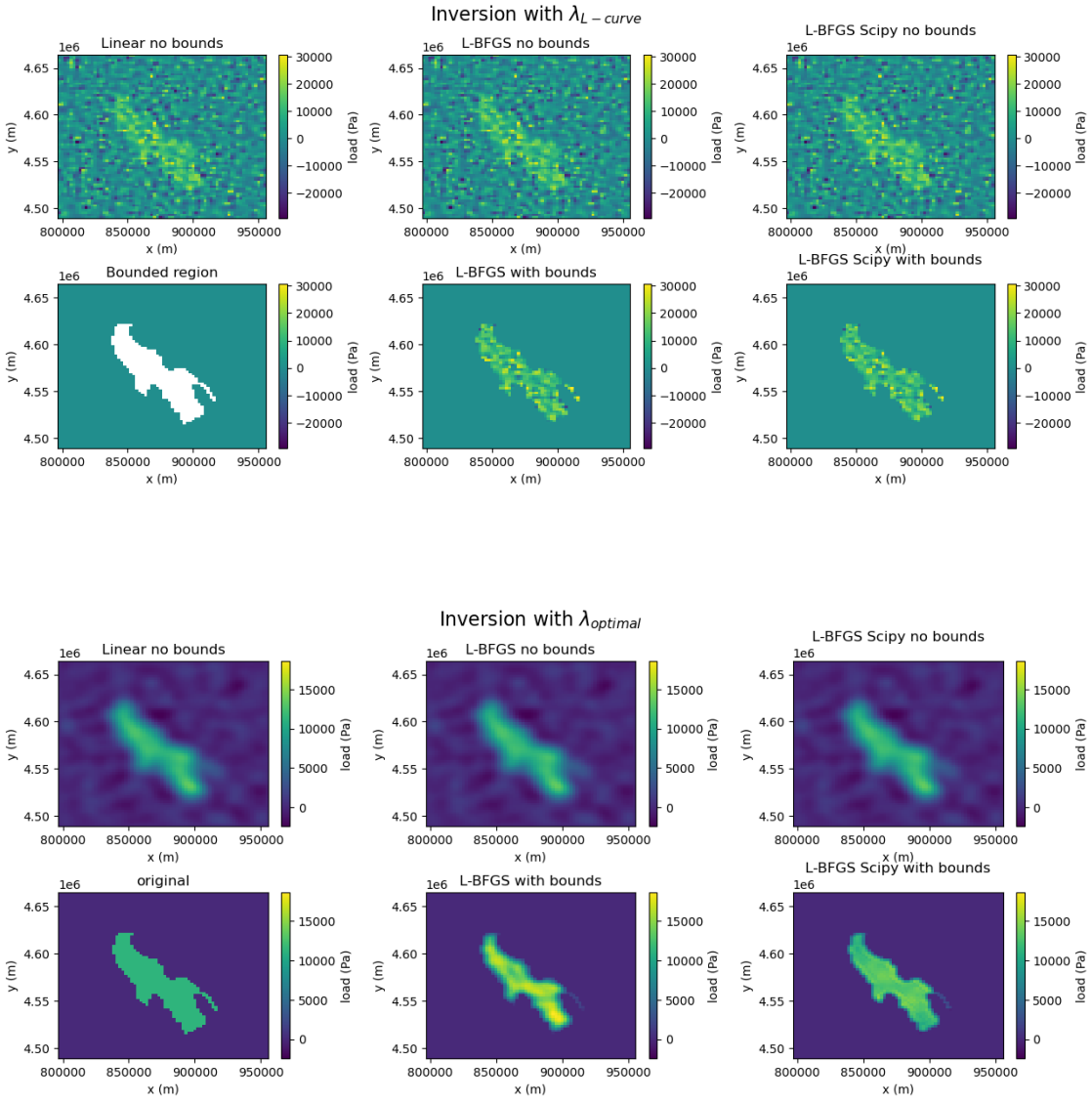


Since the L-curve fails to correctly estimate $\lambda_{optimal}$ (large logarithmic error) we look at the result of the inversion using both $\lambda_{L-curve}$ and $\lambda_{optmial}$. Moreover, since the convergence analysis shows that overall the linear inversion as well as the L-BFGS algorithm lead to the best result (small error in the inversion and short computation time), we perform the inversion of the complex load distribution with these approaches. The resulting figures load distributions with $\lambda_{L-curve}$ and $\lambda_{optmial}$ are in the figures below.

When using $\lambda_{optimal}$ we find that the inverted load distribution is fairly similar to the original one, even without a bounded region. When adding a bounded region, we find that the result of the LBFGS implementation from scipy improves, while the result from our implementation worsens. This result is likely due to the fact that the bounded region slows the convergence of our implementation, and therefore our 1,000 iterations limit is not sufficient. On the other hand, when inverting the load distribution with $\lambda_{L-curve}$ the resulting distribution is erroneous with any approach, and with and without bounds. This observation is consistent with the fact that there is a large logarithmic error for λ.

```
############### $\lambda$ derived from L-curve ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 457526.79555665335
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 458701.1659516342
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 458701.1659516342
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 162276.30271134834
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 148979.04406398983
#########################################################################"
############### $\lambda$ derived from Optimum ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 102685.65268305948
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 102685.84369934698
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 102685.84369934698
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 108222.36610196909
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
```

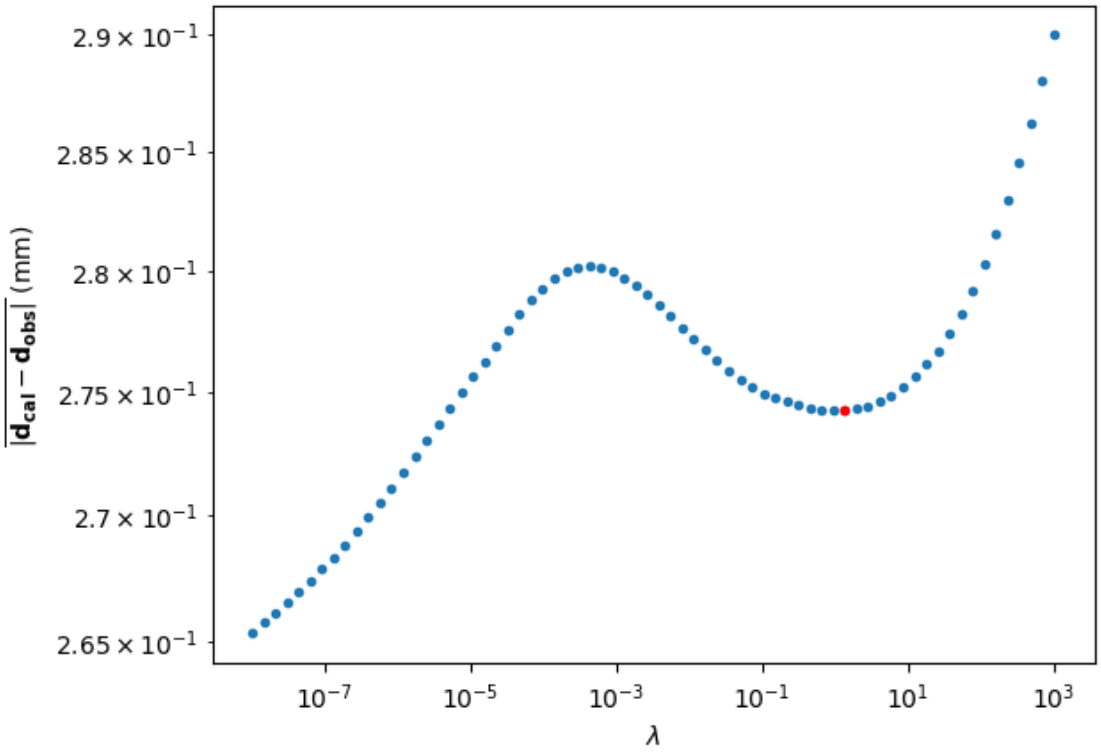
```
constrained LBFGS implementation is : 78138.8005099885
##############################################################################"
```



Because in a real case one does not know $\lambda_{optimal}$, and since the L-curve fails to estimate it, we need to find an alternative criterion.

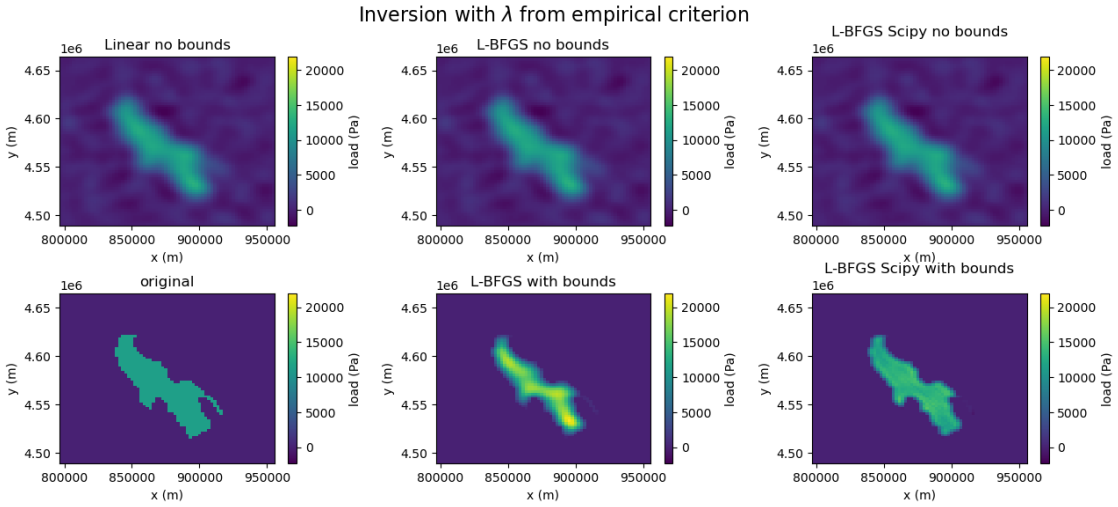
Empirically we find that the average of the absolute value of the residual between the recorded horizontal displacements and the ones modelled with the inverted load distribution $\overline{|\mathbf{d_{cal}} - \mathbf{d_{obs}}|}$ has a local minimum for a λ close to $\lambda_{optimal}$ (logarithmic error on lambda of only 0.15). While this criterion does not appear as trustworthy as the L-curve, because of the existence of multiple local minima, it seems useful to estimate $\lambda_{optimal}$ if we have prior information on the expected value of $\lambda_{optimal}$. In this case $\lambda_{L-curve} \approx 10^{-3}$, and with the empirical criterion we can see that the other minimum for $\lambda \leq 10^{-8}$, which is further from $\lambda_{L-curve}$ than the minimum for $\lambda \approx 1$. Therefore we may be able to choose $\lambda_{empirical}$ from the closest minimum to $\lambda_{L-curve}$.

```
logarithmic error on $\lambda$ with Laplacian approach : 0.1594202898550732
```



Using $\lambda_{empirical}$ we perform the inversion (figure below). As expected, with $\lambda_{empitical}$, the inversion works fairly well even without bounds, as it does with $\lambda_{optimal}$. In addition, similarly to what was noted, the bounded inversion with our implementation of the L-BFGS is worse than the unbounded one, while with scipy's implementation it improves the result.

```
############## $\lambda$ derived from empirical criterion ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 103310.89626433521
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 103319.65338499246
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 103319.65338499246
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 131275.90002048563
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 88700.48794874191
################################################################################"
```



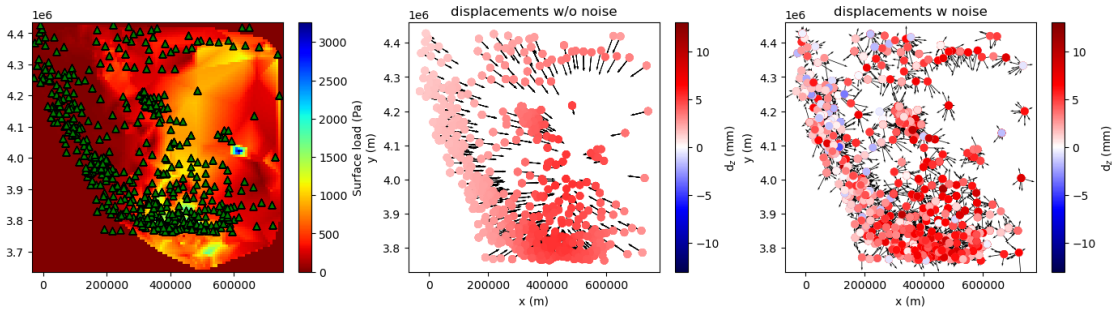
3.3 Case study : Hilary Hurricane

We have studied the result of the inversion with various noise levels, various source complexity and with various algorithms. The objective of this section is to investigate the feasibility of retrieving accurate load distribution of the water dumped during Hilary Hurricane using recorded GPS displacements from California.

To perform this case study, we use the precipitation data from California and assume that the water does not flow, does not evaporate and does not infiltrate the soil. In this case the load can be computed from the precipitation map as $P = \rho_w g h$, where ρ_w si the water density, g is the gravity and h is the height of the water column at a given position. The resulting load distribution is shown in the left-hand side panel of the following figure.

We then model the theoretical displacements recorded by the GPS stations in California (central panel) and add noise to them (right-hand side panel). We add noise with a normal distribution and a standard deviation of 3 mm to the vertical displacements, and of 1 mm to the horizontal displacements, which is the common noise level expected at these stations. Because the modelled displacements are rather small, when adding noise to the data we see that the displacements appear random, especially at small scale. At larger scale, thanks to the large number of stations, one can see that the vertical displacements in the South and in the center are statisically larger than those in the West, which is consistent with the modelled displacement without noise. This observation is of interest as it implies that a large λ will be required to be sensitive to longer wavelength displacements.

One can also notice that the station density is not homogeneous in California; some regions are greatly instrumented, while others, such as in the North-East, are poorly instrumented. This non-uniformity in the station distribution plays a significant role in the resolution of the inverted model, which should also be investigated.



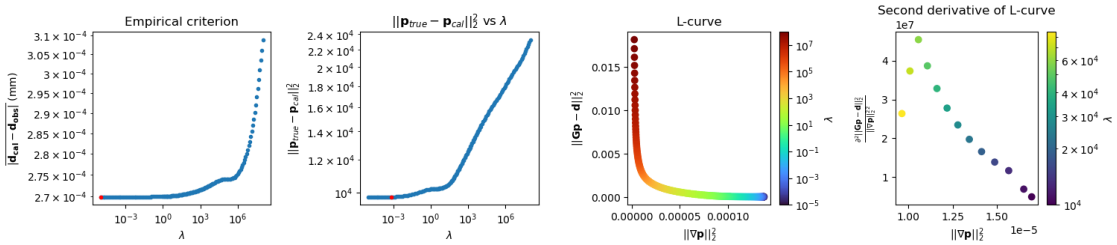
3.3.1 Inversion with Laplacian regularization

Throughout the previous tests we have seen that the Laplacian regularization tends to lead to the best result, therefore we start with the Laplacian regularization.

Without noise Because of the uneven station distribution and of the especially large noise level, we first perform the inversion without added noise, to verify that in an optimal condition we are able to invert the load distribution. Since in previous tests the L-curve did not necessarely estimate correctly $\lambda_{optimal}$, we also compute the norm of the difference between the true and inverted loads $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$, as well as the mean of the absolute value of the residual between the recorded and modelled horizontal displacements $\overline{|\mathbf{d_{cal}} - \mathbf{d_{obs}}|}$.

The figure below shows the results. The norm of the residuals between the load distribution finds that $\lambda_{optimal} \approx 10^{-3}$, however, one can note that the norm actually remains constant for $\lambda \leq \lambda_{optimal}$, which implies that inverting with these λ would lead to similar result. In the the left-hand side panel we see that there is no clear local minimum, hence we use the lowest point to derive $\lambda_{empirical}$, and find it somewhat close to $\lambda_{optimal}$. On the other hand, we find $\lambda_{L-curve} \approx 5 \times 10^4$, which is almost 8 orders of magnitude larger than $\lambda_{optimal}$.

```
loagrithmic error on $\lambda$ with L-curve : 7.944891957550186.
logarithmic error on $\lambda$ with empirical criterion : 1.8227848101265822.
```



Looking at the result of the inversion with the optimal value of λ we find that the inverted load distribution is very similar to the original one, revealing that the station density in California provides sufficient resolution overall. Comparing the result of the linear inversion to the one from unbounded LBFGS we can note that the inversion with the L-BFGS displays some artifacts at the

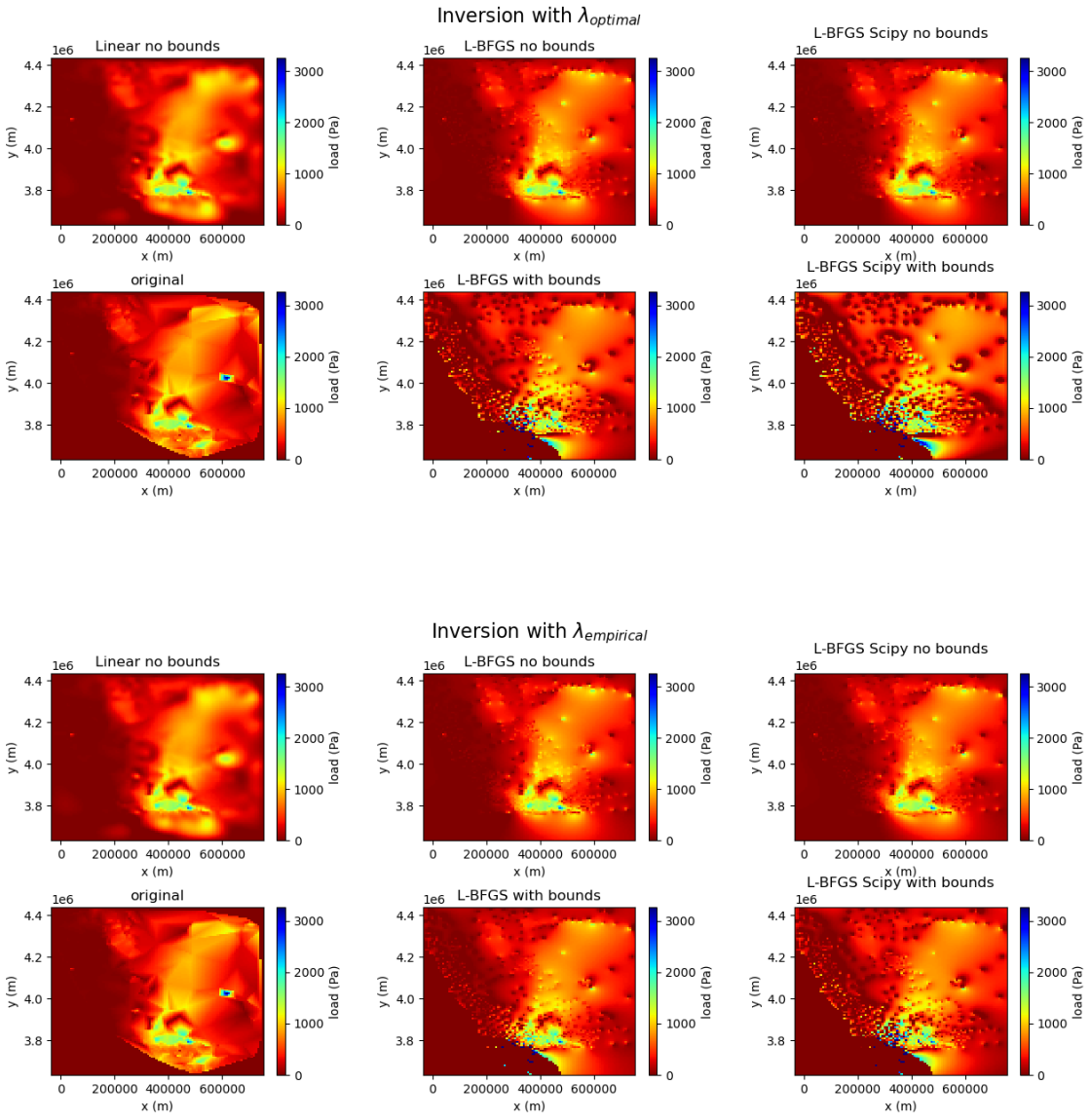
location of the stations, which do not appear with the linear inversion. When constraining the load in the ocean, by setting the load to 0, we find that the inverted load distribution is actually worse with both Scipy's and our implementations. This worsening of the inverted load is likely due to the fact that in the original load distribution map, some regions in the ocean have non-null load (because of the linear interpolation that I performed between station precipitation data to make the precipitation map). Because of this, we force a null load in certain regions where the original load is actually not null, thus leading to the worsening of the inversion. This effect can be spotted in the South-South West region where the inverted load on islands is strongly overestimated, because we forced the load around them to be null. This result illustrates the major impact that using an erroneous bounds can have on the result of the inversion.

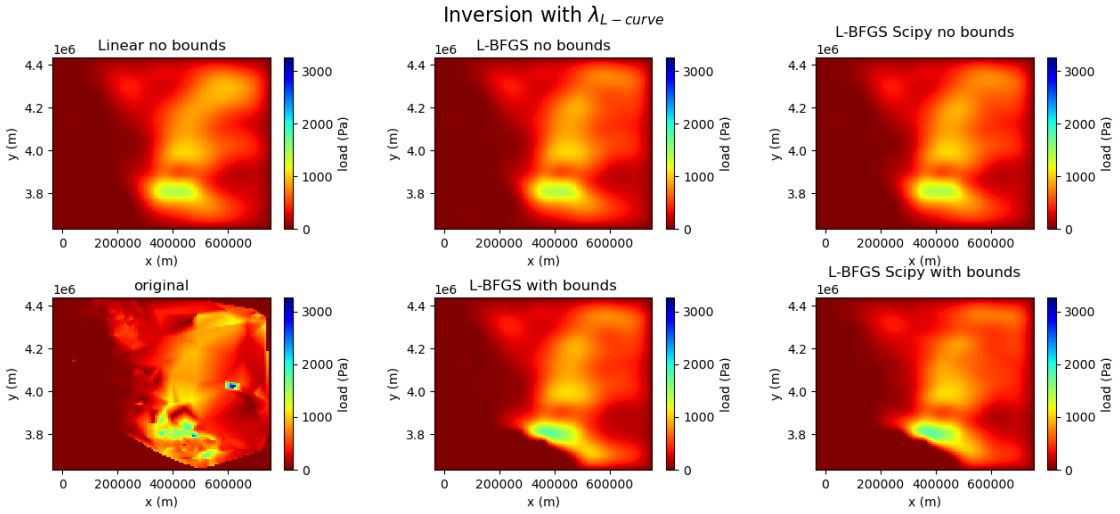
When using $\lambda_{empirical}$ for inversion, the inverted distributions are very similar to the ones inverted with $\lambda_{optimal}$, therefore the same remarks can be made. These inverted loads are are very close to the ones inverted with $\lambda_{optimal}$ because, as was pointed out on the $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ vs λ curve, for $\lambda \leq \lambda_{optimal}$, $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ is almost constant.

On the contrary, when using $\lambda_{L-curve}$, the result of the inversions is extremely different from the one with $\lambda_{optimal}$. Because $\lambda_{L-curve}$ is greatly overestimated the inverted load distributions are very smooth. In addition, unlike with $\lambda_{optimal}$, because $\lambda_{L-curve}$ is large, there are no artifacts with L-BFGS inversion.

```
############## $\lambda$ derived from optimum ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 9752.445691669262
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 13282.707510898004
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 13316.310280663427
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 47578.42273849038
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 57023.99990322123
#########################################################################"
############## $\lambda$ derived from empirical criterion ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 9753.060141902448
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 13704.149184481996
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 13275.365302509608
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 29836.751700099478
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 45109.59453148601
#########################################################################"
############## $\lambda$ derived from L-curve ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 15817.30359992933
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
```

```
unconstrained LBFGS implementation is : 16258.335910103686
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 15906.497449244136
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 18470.48875662657
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 19043.63631816189
##############################################################################"
```



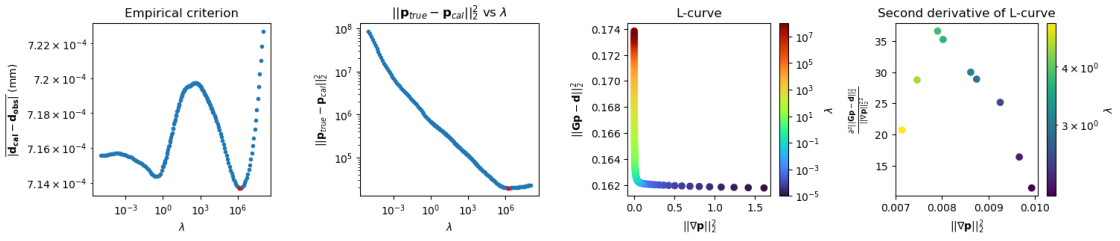


3.3.2 Inversion with noise 1

Since we have shown that without noise we are able to correctly invert the load distribution by using $\lambda_{empricial}$ as the approximation for $\lambda_{optimal}$, we can perform the inversion with the noisy data.

With noisy data, we find that $\lambda_{optimal} \approx 10^6$, which is larger than with noiseless data. This observation is consistent with the one made on the displacements with added noise; due to the large noise level displacements at a small scale appear random, while displacements at large scale are still somewhat consistent with the true displacements. On one hand, the L-curve finds $\lambda_{L-curve} \approx 10^0$, thus strongly underestimates the required smoothing. On the other hand, the empirical criterion displays 2 clear local minima at $\lambda \approx 10^0$ and $\lambda \approx 10^6$. In this case the first minimum is the closest to $\lambda_{L-curve}$, however it is not the closest to $\lambda_{optimal}$, which reveals that the L-curve cannot be used to determine which local minimum leads to the best approximation of $\lambda_{optimal}$.

```
loagrithmic error on $\lambda$ with L-curve : 5.6399437412095645.
logarithmic error on $\lambda$ with empirical criterion : 0.08080808080808044.
```



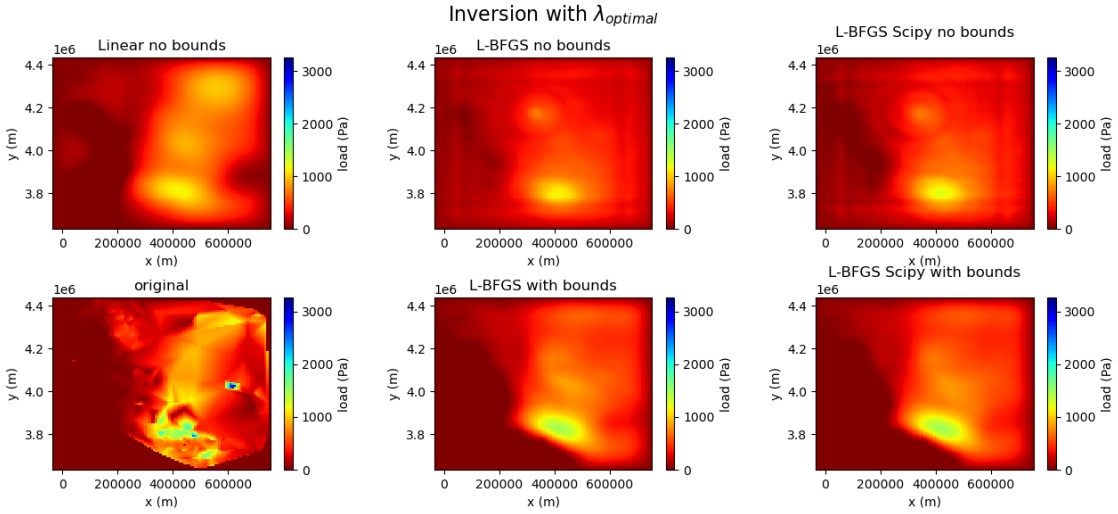
The figures below show the result of the inversion with the values of λ derived previously.

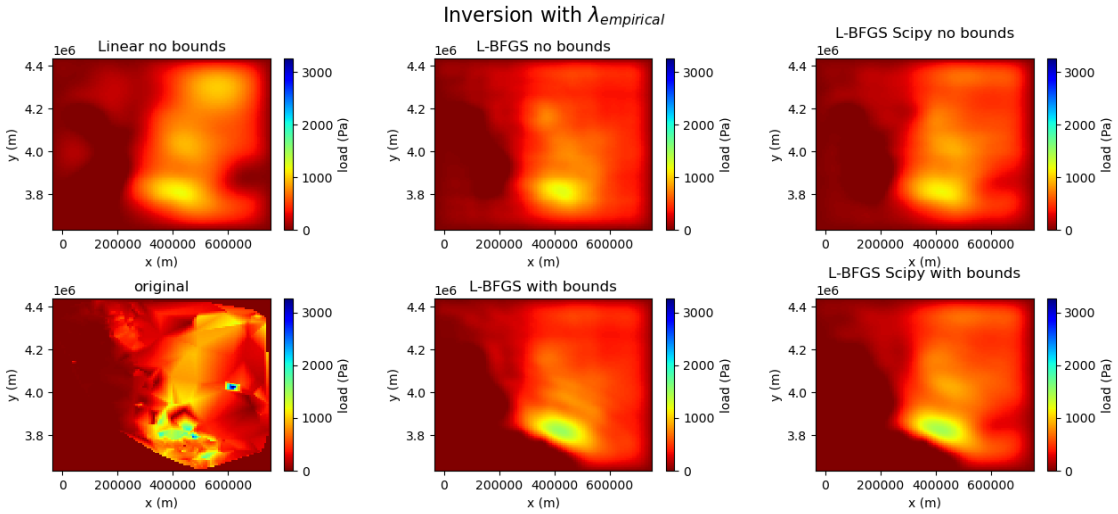
Using either $\lambda_{optimal}$ or $\lambda_{empirical}$ the inverted load distribution is a smoothed version of the original load distribution, as expected with large λ. However, with non-bounded L-BFGS inversions we can

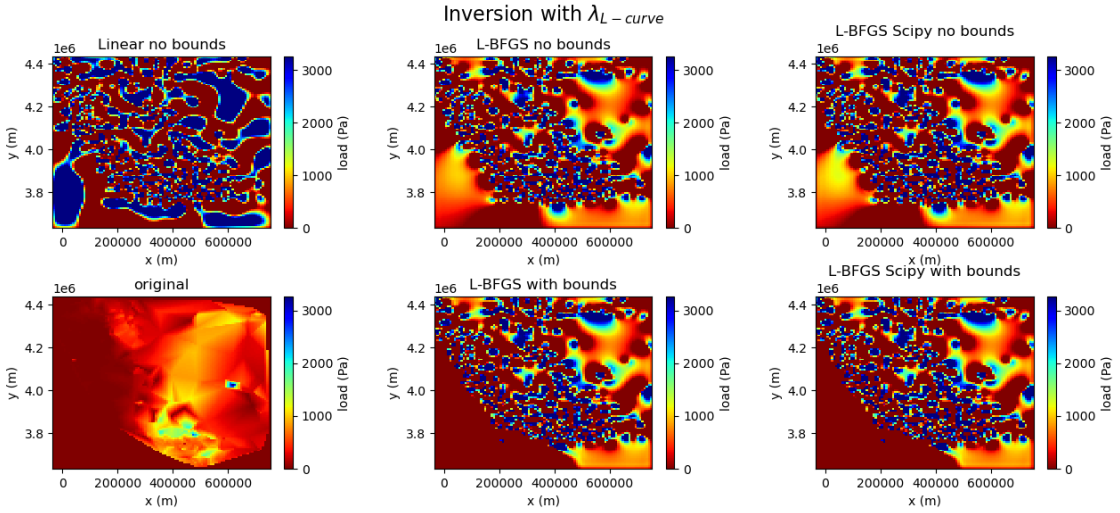
observe an artifact (more striking for $\lambda_{optimal}$ than for $\lambda_{empirical}$), which does not appear with the bounded inversion.

On the other hand, when using $\lambda_{L-curve}$, the result of the inversion is completely erroneous because λ is underestimated.

```
############## $\lambda$ derived from optimum ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 19771.32463025989
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 25954.872576703805
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 24454.585558761024
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 23421.71386699892
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 23663.87229916023
#########################################################################"
############## $\lambda$ derived from empirical criterion ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 19810.19761221672
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 22613.44730796945
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 21070.23622548608
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 24282.57354403735
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 23027.509496116098
#########################################################################"
############## $\lambda$ derived from L-curve ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 497421.32551851467
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 287442.1712854608
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 308357.92967675836
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 293764.60106146225
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 298925.53731129144
#########################################################################"
```


Inversion with $\lambda_{optimal}$
Linear no bounds
L-BFGS no bounds
L-BFGS Scipy no bounds
original
L-BFGS with bounds
L-BFGS Scipy with bounds
1e6
y (m)
x (m)
load (Pa)


Inversion with $\lambda_{empirical}$
Linear no bounds
L-BFGS no bounds
L-BFGS Scipy no bounds
original
L-BFGS with bounds
L-BFGS Scipy with bounds
1e6
y (m)
x (m)
load (Pa)



3.3.3 Inversion with a Gaussian covariance matrix

The inversion using a Laplacian regularization revealed that the L-curve systematically provides a poor estimate of $\lambda_{optimal}$ for this study. Meanwhile, the empirical criterion systematically leads to a local minimum close to the optimal value of $\lambda_{optimal}$, however since there are multiple local minima, there is an ambiguity in the choice of $\lambda_{empirical}$, making this criterion unreliable. The aim of this section is to investigate and compare the ability of the L-curve to provide a good estimate of $\lambda_{optimal}$ when using a the inverse of a Gaussian instead of the Laplacian as the inverse covariance matrix.

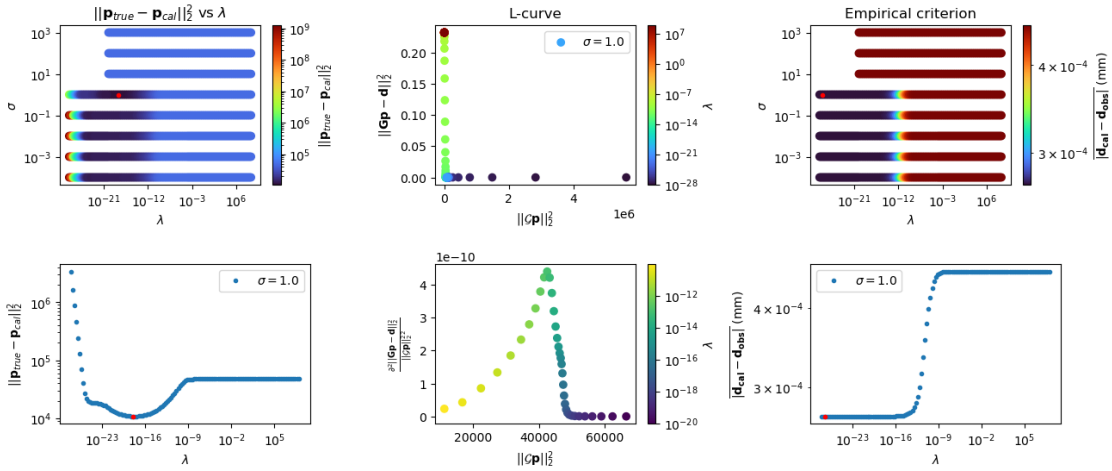
without noise We first perform the inversion without adding noise to the displacements. The figure below shows the result of the inversion for various pairs of λ and σ. Looking at the top left-hand side panel, we note that when σ is too large the inversion becomes unstable (no minimum for $\sigma \geq 10$) and $\sigma_{optimal}$ is at the stability limit, similarly to what was observed in previous sections. Moreover we can observe a slight shift of $\lambda_{optimal}$ for $\sigma_{optimal}$, which is consistent with previous results, and shows that $\sigma_{optimal}$ can be estimated. For $\sigma = \sigma_{optimal}$, $\lambda_{optimal} \approx 10^{-18}$.

With the empirical criterion (right-hand side panels), the unstable behavior of the inversion is also clear, since for $\sigma \geq 10$, $\overline{|\mathbf{d_{cal}} - \mathbf{d_{obs}}|}$ remains identical for any λ. Despite its similarity with the left-hand side panel, the empirical does not show any clear change in the optimal value of λ for $\sigma = \sigma_{optimal}$ (curve for $\sigma = 1$ is identical to that for $\sigma = 0.1$), meaning that based purely on the empirical criterion one only knows the upper limit for $\sigma_{optimal}$ (when the inversion becomes unstable) if there is no noise in the data. Therefore in this case, in practice one would need to use the L-curve to also assess the lower bound for $\sigma_{optimal}$ by looking at the stability of $\lambda_{L-curve}$ as a function of σ. For $\sigma = \sigma_{optimal}$ (bottom right-hand side panel), the empirical criterion does not show any local minimum, similarly to what was observed with the Laplacian regularization with noiseless data, and therefore we estimate from the overall minimum, yielding $\lambda_{empirical} \approx 10^{-27}$. This estimate of $\lambda_{optimal}$ is almost 10 orders of magnitude smaller than the actual $\lambda_{optimal}$. In addition, one can see that the error between the true and inverted loads actually increases for $\lambda \leq \lambda_{optimal}$ (bottom left-hand side panel), which implies that unlike with the Laplacian regularization,

underestimating $\lambda_{optimal}$ deteriorates significantly the inverted load distribution.

We compute the L-curve and its second derivative to find $\lambda_{L-curve}$ for $\sigma = \sigma_{optimal}$, and find that $\lambda_{L-curve} \approx 10^{-13}$, which corresponds to a logarithmic error of about 5. Hence, even with no noise the L-curve fails to correctly estimate $\lambda_{optimal}$.

```
logarithmic error on $\lambda$ with L-curve : 4.979797979797979.
logarithmic error on $\lambda$ with empirical criterion : 9.498780912574016.
```

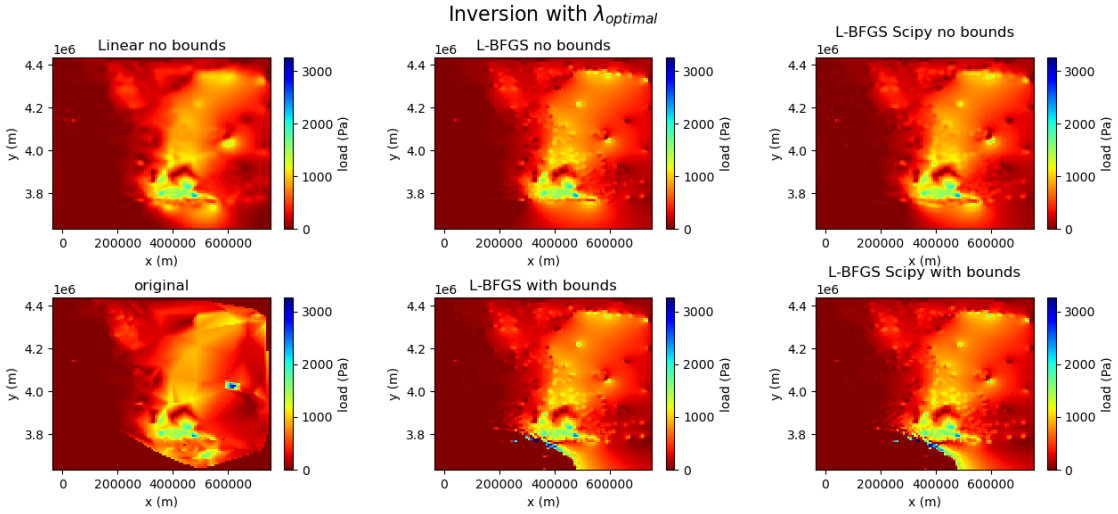


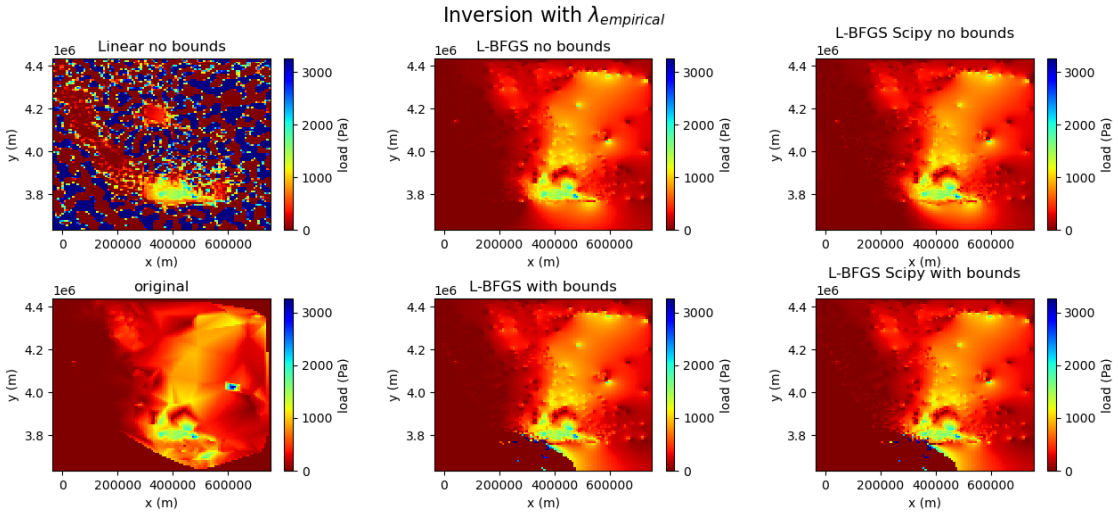
When using $\lambda_{optimal}$ (first figure below) the result of the inversion is similar to the one obtained with the Laplacian regularization; the load distribution inverted by linear inversion is similar to the original one, and the load distributions inverted with L-BFGS algorithms are also similar but also contain artifacts.

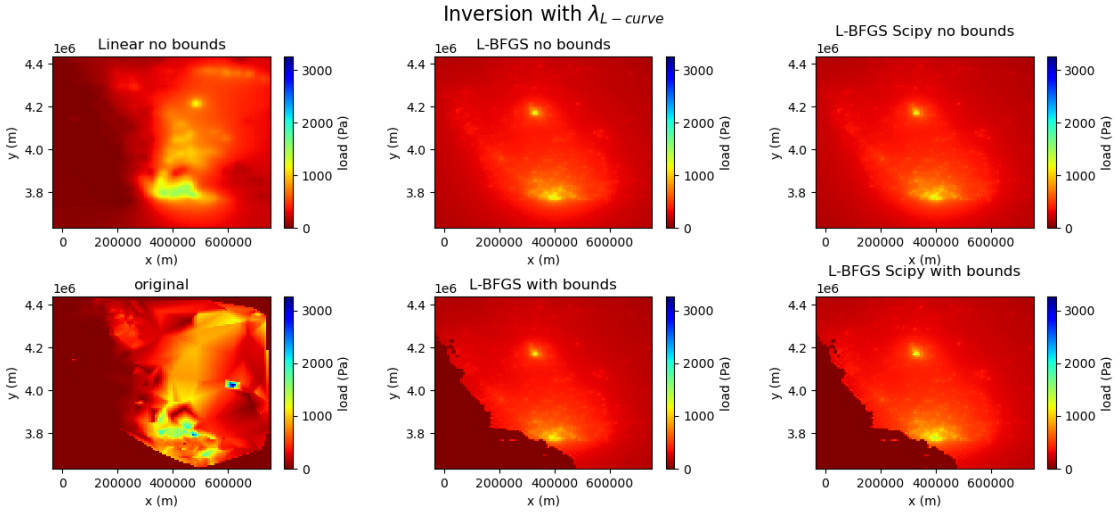
As expected, unlike with the Laplacian regularization, the large underestimation of $\lambda_{optimal}$ by the empirical criterion leads to a completely erroneous inverted load distribution when using linear inversion. However, one can note that the inverted load distribution with the L-BFGS algorithm are actually very similar to the ones inverted with $\lambda_{optimal}$. This is possible because we compute $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ using linear inversions (to save time), and therefore it is likely that if we were to compute $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ using L-BFGS inversions there would be a constant minimum for $\lambda \leq \lambda_{optmal}$ instead of the increase observed with the linear inversion. This shows that all inversion methods do not have the same sensitivity to λ. That result may be valuable to verify that the inverted load distribution is correct when using $\lambda_{empirical}$, since one does not not know from which local minimum to take $\lambda_{empirical}$ from. By comparing the result of different inversion algorithms one could consider that the chosen $\lambda_{empirical}$ is wrong if different algorithms converge to very different load distributions.

When using $\lambda_{L-curve}$, because $\lambda_{L-curve} >> \lambda_{optimal}$, the inverted load distribution with the linear inversion is a smooth version of the original distribution. The result of the L-BFGS inversion is significantly worse than the one with linear inversion. This result and the result from the inversions with $\lambda_{empirical}$ show that no approach systematically performs better than the others, and that comparing the result of the inversion between the linear and non-linear inversions can help to tell if the predicted λ is close to $\lambda_{optimal}$.

```
############## $\lambda$ derived from optimum ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 10833.037442375487
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 13896.311404658763
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 12671.535335450462
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 20899.746023712647
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 21595.23329510373
##########################################################################"
############## $\lambda$ derived from empirical criterion ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 867351.9201095988
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 14036.751677792208
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 13450.404497916064
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 23625.397918854513
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 24988.70594687477
##########################################################################"
############## $\lambda$ derived from L-curve ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 17500.2229841958
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 32281.241427119814
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 32357.65361840373
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 31902.072411078425
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 31204.891587853403
##########################################################################"
```


Inversion with $\lambda_{optimal}$
Linear no bounds
L-BFGS no bounds
L-BFGS Scipy no bounds
original
L-BFGS with bounds
L-BFGS Scipy with bounds
1e6
y (m)
x (m)
load (Pa)


Inversion with $\lambda_{empirical}$
Linear no bounds
L-BFGS no bounds
L-BFGS Scipy no bounds
original
L-BFGS with bounds
L-BFGS Scipy with bounds
1e6
y (m)
x (m)
load (Pa)



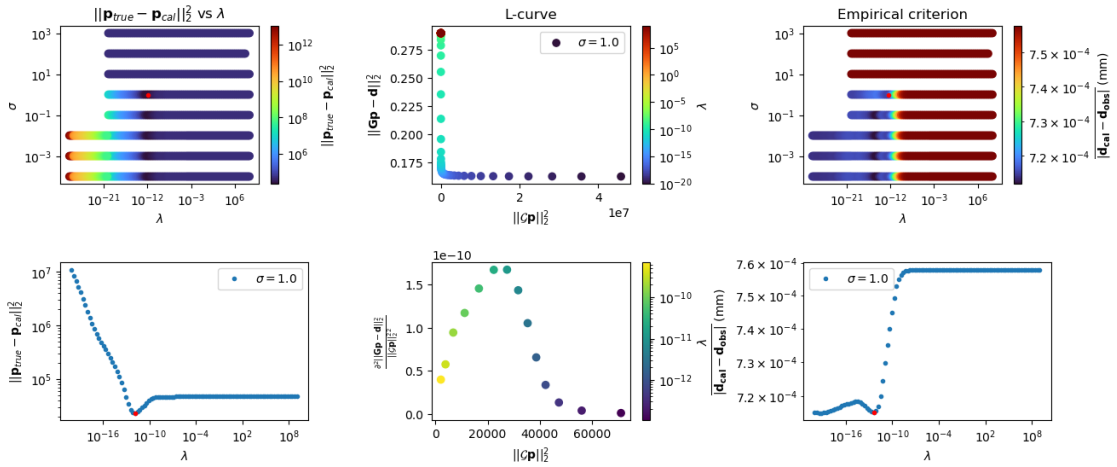
Inversion with noise Without noise in the data we have seen that both L-curve and the empirical criterion failed to approximate $\lambda_{optimal}$. As seen with the Laplacian regularization, the main reason why the empirical criterion fails to predict $\lambda_{optimal}$ when there is no noise in the data comes from the fact that there is no local minimum to extract $\lambda_{empirical}$ from, and so we resort to the lowest point of the curve. The aim of this part is therefore to see if with added noise at least the empirical creterion can correctly predict $\lambda_{optimal}$.

With added noise, we see that $\lambda_{optimal} \approx 10^{-12}$ and $\sigma_{optimal} = 1.0$ (left-hand side panels).

For $\sigma_{optimal}$, the L-curve predicts $\lambda_{L-curve} \approx 10^{-11}$, which is close to $\lambda_{optimal}$ (logarithmic error $<$ 1).

With the empirical criterion (right-hand side panels), due to the noise in the data we actually see a clear change in the location of local minima for $\sigma = \sigma_{optimal}$ (top panel). This reveals that if there is noise in the data, the empirical criterion could potentially be used to estimate $\sigma_{optimal}$. In addition, for $\sigma_{optimal}$, we observe 2 local minima (bottom panel), one of which being at $\lambda \approx 10^{-12}$, very close to $\lambda_{optimal}$.

```
logarithmic error on $\lambda$ with L-curve : 0.8787878787878789.
logarithmic error on $\lambda$ with empirical criterion : 0.5858585858585865.
```



The figures below display the result of the inversion using $\sigma_{optimal}$ with $\lambda_{optimal}$, $\lambda_{empirical}$ and $\lambda_{L-curve}$.

When using $\lambda_{optimal}$, one can notice that the inverted load distribution is significantly different from the one inverted with the Laplacian regularization with the same noise. One can notice that with the Gaussian regularization and the linear inversion, the inverted load contains more small scale fluctuations. In addition, the load inverted with the L-BFGS algorithm differs significantly from the one with linear inversion.

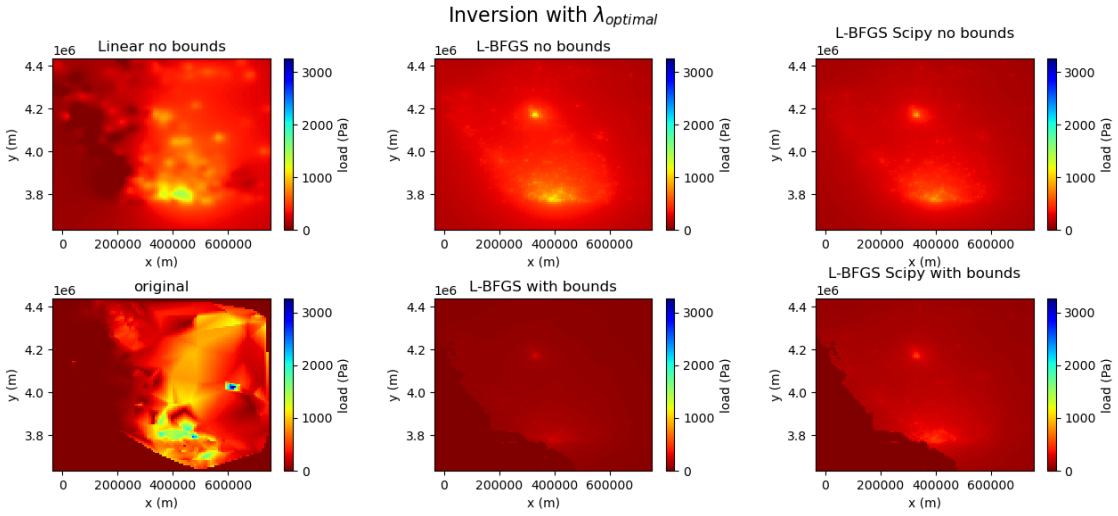
When using $\lambda_{empirical}$, because it is close to $\lambda_{optimal}$, the same remarks can be done.

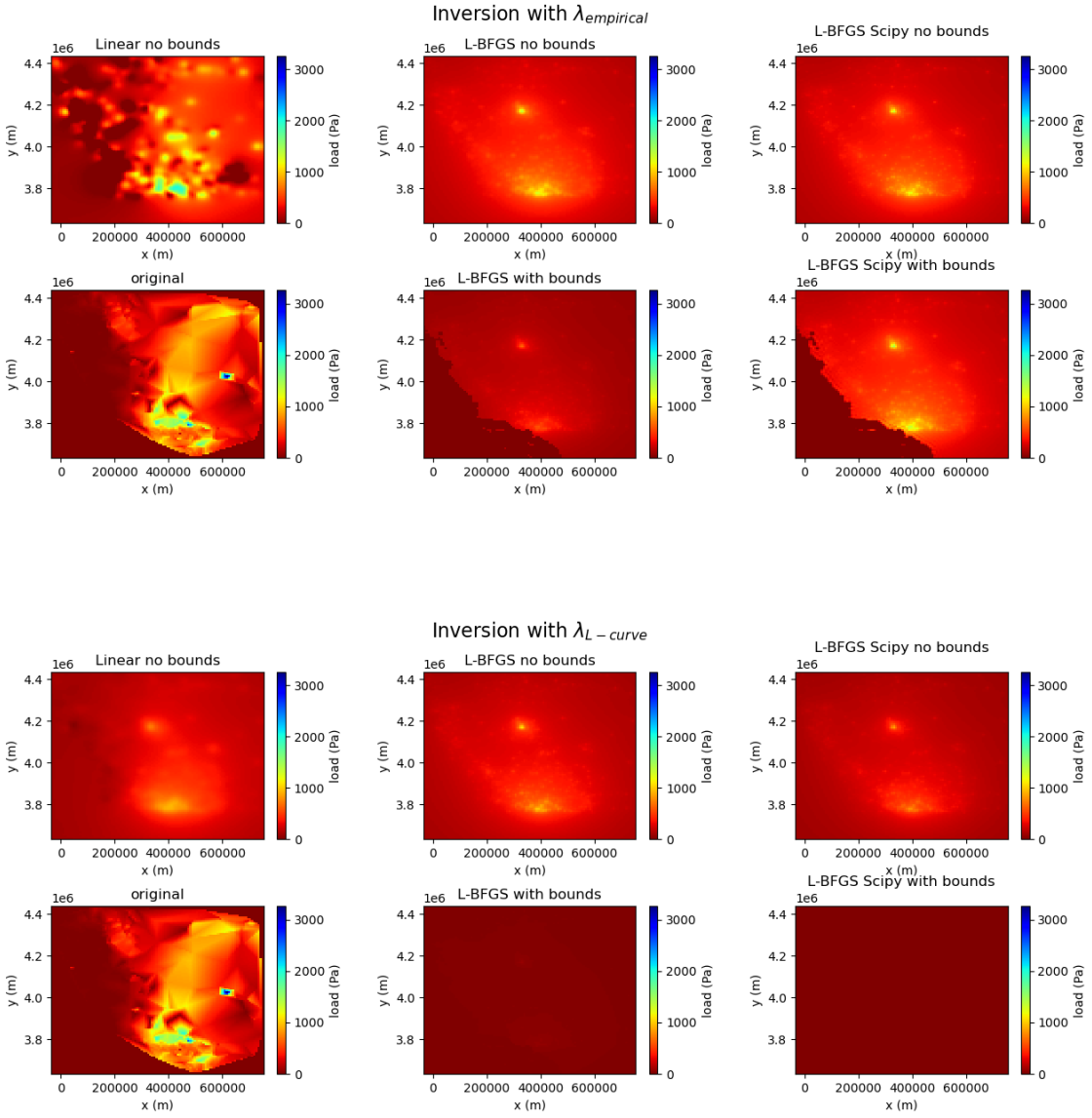
With $\lambda_{L-curve}$, we find that the inverted load distribution with linear inversion differs significantly from the ones with $\lambda_{optimal}$ and $\lambda_{empirical}$, even though $\lambda_{L-curve}$ is rather close to $\lambda_{optimal}$.

While both the L-curve and the empirical criterion provide close estimate of $\lambda_{optimal}$ with the Gaussian regularization, this regularization approach tends to provide worse results than the inversion with the Laplacian using $\lambda_{empirical}$ (larger error with the Gaussian regularization than with the Laplacian inversion). This is highlighted by the fact that with the same noise, the $||\mathbf{p}_{true} - \mathbf{p}_{cal}||_2^2$ error norm with the Laplacian regularization and $\lambda_{empirical}$ is as low as 19810, while with the Gaussian regularization and with $\sigma_{optimal}$ and $\lambda_{optimal}$ it is as low as 23528.

```
############## $\lambda$ derived from optimum ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 23527.954255791312
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 32325.86591641428
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 34341.95732441145
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 43235.189278388956
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 38588.34635175131
#########################################################################"
############## $\lambda$ derived from empirical criterion ###############
```

```
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 27241.92862463793
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 31872.047956352126
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 32245.45559034588
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 37708.350986172234
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 30955.343184576675
##########################################################################"
############## $\lambda$ derived from L-curve ###############
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with Linear inversion
is : 29594.256473637433
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our
unconstrained LBFGS implementation is : 33357.828308961616
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
unconstrained LBFGS implementation is : 35103.498322412466
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with our constrained
LBFGS implementation is : 46398.57028001249
The error $|| \mathbf{p}_{true} - \mathbf{p}_{cal} ||_2^2$ with scipy's
constrained LBFGS implementation is : 47484.121054758405
##########################################################################"
```





3.4 Resolution of the inverted load distribution

In previous sections we investigated the use of Laplacian and Gaussian regularization to invert the closest load distribution to the one that led to the recorded displacements. Depending on the noise level the coefficient λ for optimal regularization varies, and as a result the load distribution is more or less smoothed. This resolution study aims at determining the expected resolution with both approaches as a function of the regularization coefficient λ, and as a function of σ for the Gaussian approach in the case where the volume under the gaussian is normalized (constant λ when σ varies).

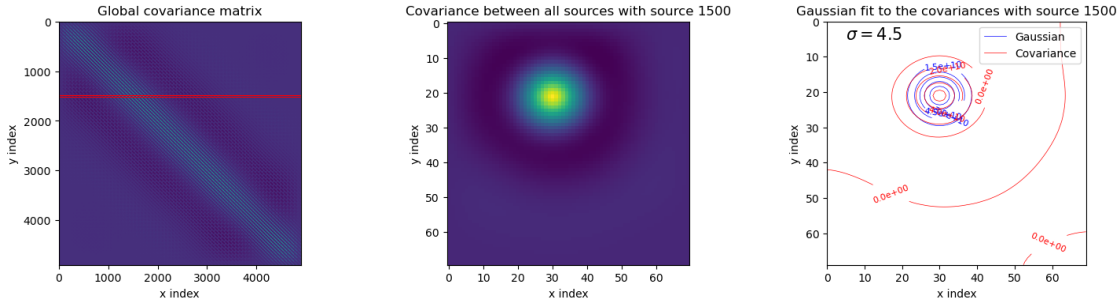
To determine the resolution we use the inverse operator $(\mathbf{G^T G} + \mathcal{G}^{-1})^{-1}$, which behaves as a covariance matrix. The elements in its diagonal contain the variance of the parameters, while the elements off-diagonal are the covariance between the parameters, which can be translated to model resolution.

In this section we will first take a look at the effect of the regularization coefficient on the resolution, and we will then look at the effect of the effect of the uneven station distribution in California on the fluctuation of the resolution.

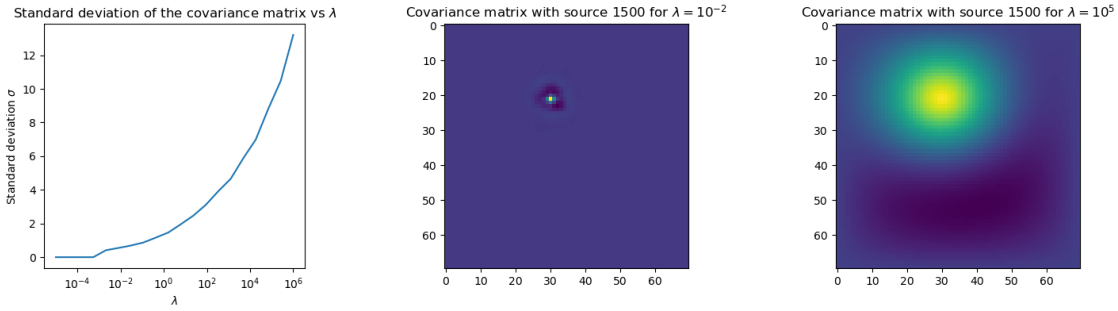
3.4.1 Resolution with the Laplacian

In previous sections we saw that the acceptable values of λ when using the Laplacian as the inverse covariance matrix ranged from 10^{-8} to 10^{6}.

Because we are working with parameters in 2D, covariances that appear distant in the covariance matrix can actually be between two close parameters and thus be large, which explains the appearance of the matrix (left-hand side panel below). Overall one can see that it is diagonal, which tends to indicate that parameters far away are not coupled. When selecting the value of a specific row and reshaping the vector to the shape of the load distribution, we find that the covariances have a shape similar to a 2D Gaussian (central panel in the figure below). We therefore fit it with a Gaussian, in order to determine the standard deviation (right-hand side panel in the figure below). This standard deviation informs us on the distance where the coupling of the parameters becomes negligeable so that we can consider that a parameter is independant of the others. In this case, we find that the best fit is reached for $\sigma = 4.5$, meaning that parameters most of the coupling between the parameters (so the loads) occurs within about 4-5 pixels away from the selected source.

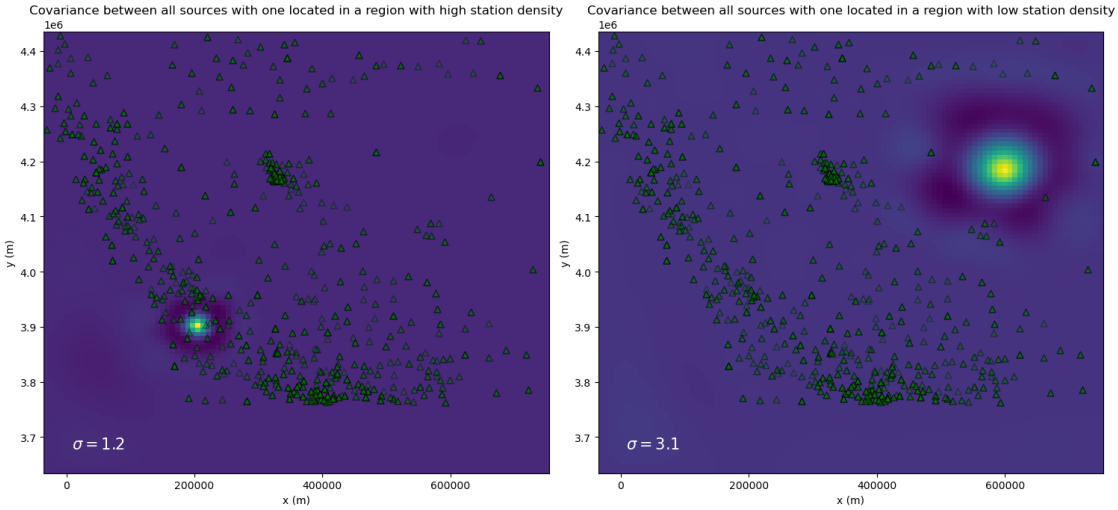


Performing the same Gaussian fit to the covariance of the loads around source 1500 for different values of λ, as expected, we find that when λ increases the standard deviation increases as well, implying that the more we smooth the resolut of the inversion the lower the resolution. On the left-hand side panel of the figure below we can see that for $\lambda \leq 10^{-1}$, the standard deviation σ of the best fitting Gaussian is less than 1 pixel, which implies that the loads of the sources are mostly decoupled; while for large values of λ, the standard deviation reveals that sources 10 pixels away are still coupled. The central and right-hand side panels illustrate the difference between the covariance matrix for $\lambda = 10^{-2}$ and $\lambda = 10^{5}$.



We previously mentionned that the uneven distribution of stations in California is likely to have an impact on the resolution. In order to see the effect of station desnsity on the resolution we display the covariance matrix with a source located in a region with high density of stations (left-hand side panel of the figure below) and with a source located in a region with a low density of stations (right-hand side panel of the figure below).

When looking at the covariance between all sources loads with one located in a region with high/low station density with the same regularization coefficient λ, one can clearly notice the drastic difference in resolution, where the lower the station density the larger the standard deviation.



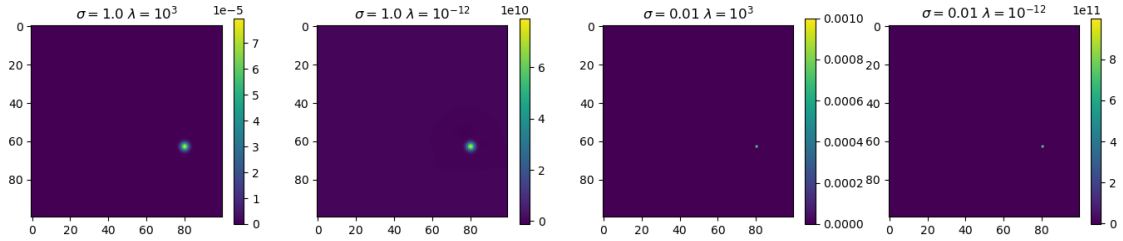
3.4.2 Resolution with the Gaussian

With the normalized Gaussian regularization, we saw that the optimal value of λ is not impacted by the chosen standard deviation of the Gaussian (expect for $\sigma = \sigma_{optimal}$). This is explained by the fact that the volume under the Gaussian being normalized, the strength of the smoothing is only dependant on the value of λ. However, one can observe extremely different inversion results when using Gaussians with various σ while keeping the same value of λ. In addition, we saw with the study of the resolution using the Laplacian regularization that λ can impact the width the area of coupling between sources, therefore it is necessery to investigate the trade-off between σ and

λ for the Gaussian regularization. The figure below shows a comparison between the covariance matrices obtained for the same source 4 differents pairs of σ and λ.

Comparing the coupling of the sources with a reference source for different σ and λ, one can notice that λ does not have an impact on the width of the coupled region, hence on the resolution, and only impacts the amplitude of the covariance between sources; it only conditions the sum of the covariances, so the volume under the Gaussians. Because of that, one can actually "choose" the resolution of their model independently of the strength of the coupling between parameters with the Gaussian approach.

```
The sum of the matrix is  0.0010000000005198292 with $\sigma$ = 1.0 and $\lambda
= 10^{3}$
The sum of the matrix is  0.0009999999999999707 with $\sigma$ = 0.01 and
$\lambda = 10^{3}$
The sum of the matrix is  183388572845.7235 with $\sigma$ = 1.0 and $\lambda =
10^{-12}$
The sum of the matrix is  190429370087.58032 with $\sigma$ = 0.01 and $\lambda =
10^{-12}$
```



4 Difficulties encountered and skills learnt during the project

4.1 Things learnt during the project

During this project I discovered how to model the displacements generated by a complex load distribution at the surface, as well as the various ways to invert it from displacmeent measurements. For this purpose, I had the opportunity to familiarize and implement various non-linear local optimization inversion algorithms, such as the NLCG, and the LBFGS algorithms. In addition, I discovered three ways to regularize the inversion; either with the Laplacian and Gaussian of the parameters or with the Total Variation, which can be implemented using the soft thresholding algorithm. Finally, I also learnt about bound projection, which is used to bound the values of parameters inverted with local optimization algorithms.

4.2 Difficulties encountered

Several difficulties were encountered throughout the project :

- The second Wolfe condition (curvature condition) fails to converge sometimes, therefore we had to implement a second linesearch algorithm that solely enforces the Armijo condition. In

addition, we had to choose a maximum number of iteration above which the linesearch would consider that it fails to converge.

- Estimating $\lambda_{optimal}$ with the L-curve generally does not work if the load distribution is too complex (too many parameters to invert for) and the data is too noisy. Therefore we had to resort to an empirical criterion to estimate $\lambda_{optimal}$, which also brings its shares of problems too, as this criterion presents non unique local minima and no clear method was found to systematically find which minimum leads to the best inversion. Also, with noiseless data, this criterion is even more unreliable as there no longer are minima.
- While the linear inversion with a Laplacian regularization tends to provide the best inversion overall, we have seen that there are still occurences where the non-linear inversion with the L-BFGS performs better since these different approaches have different sensitivity to λ, therefore we cannot always trust the result of the linear inversion.
- Our implementation for bounded inversion seem to severely degrade the convergence of the inversion, while Scipy's implementation does not. As a result, since we limit the number of iterations, the result of the inversion with our and with Scipy's implementation are very different.

5 Conclusion

All our tests lead us to the conlusion that inverting a high resolution surface load distribution due to the water dumped during Hilary Hurricane using GPS displacements is very challenging. Considering that the actual load is likely even smaller than the one used in this case study, because we made the assumption that all the precipitated water remains on the ground, it seems that even if we manage to correctly estimate $\lambda_{optimal}$, the resolution of the inverted load distribution would be too low to accurately model local stress changes in the medium.