Department of Earth Sciences

Institute of Geophysics

Hybrid Methods for OED

- Research Proposal -

Arnaud Mercier

November 24, 2025

Supervisors:

Prof. Dr. Hansruedi Maurer

Dr. Ben Moseley

1 Motivation

Optimized Experimental Design (OED) for full-waveform inversion (FWI) presents significant challenges due to the complex and high-dimensional nature of seismic data acquisition. Traditional approaches to OED often rely on combinatorial methods, selecting a subset of sources and receivers from a comprehensive survey. Recent advances in scientific machine learning, particularly the development of differentiable physics frameworks such as JAX, offer promising new avenues for tackling OED with greater flexibility by incorporating learnable components and leveraging auto-differentiation.

By employing hybrid approaches, it becomes possible to optimize source and receiver placement in a continuous space rather than a predefined grid. This reduces the risk of converging to local minima associated with combinatorial constraints. Furthermore, the integration of pre-trained models has the potential to accelerate both forward modelling and inversion steps, providing substantial computational speed-ups. These efficiency gains enable faster iteration cycles, allowing for more comprehensive testing and refinement of experimental designs.

Additionally, hybrid approaches enable learnable model parameterisation and data selection strategies, which can adapt to complex geological scenarios and utilize new criteria informed by FWI results.

By merging differentiable physics with optimized design, this proposal aims to develop a framework that overcomes key limitations of conventional methods, offering greater flexibility and generalizability for OED. This proposal outlines the planned steps to investigate hybrid approaches in OED. Preliminary tasks are defined, and four basic test cases are proposed.

2 Problem formulation

The OED and FWI problems can be summarized in Figure 1. The OED process can be defined as by an operator $\mathcal{A} : \mathrm{m} \mapsto \{\mathrm{S}, \mathrm{R}\}$ that maps a model m to a subset of sources and receivers $\{\mathrm{S}, \mathrm{R}\}$. This operator is traditionally implemented as a sequential algorithm ($\mathcal{A}_O$) selecting the source/receiver maximising an OED criterion $\mathcal{C}$ from a comprehensive survey.

FWI can be express by an operator $\mathcal{F}^{-1} : \{\mathrm{S}, \mathrm{R}\} \mapsto \mathrm{m}$ that maps a set of source and receiver to a model. Traditionally, this operator is implemented as an optimization algorithm employing gradient descent to converge to an acceptable model.

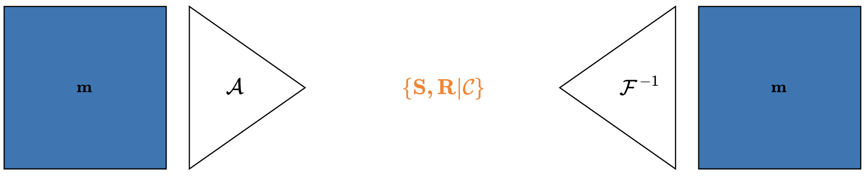


Figure 1: Schematic of the OED and FWI operators.

3 Tasks

3.1 Define a test model

To evaluate various $\mathcal{A}$, $\mathcal{C}$, and $\mathcal{F}^{-1}$ operators, it is essential to define a test case consisting of a velocity model and an specific amount of sources and receivers. Specifically, the test case should exhibit the following characteristics:

- Posses a interpretable solution (sources and receiver layout). The problem should be low dimensions such that a loss landscape can be computed. This would facilitate greatly the tuning of the hyper-parameters of the optimisation strategy.
- Balance between: Many receivers (well constraint, i.e. crosshole setup) v.s. few receivers (less constraints, many more possible solution).
- Small enough to by inverted quickly: $<\approx 60$ sec per gradient descent iteration.
- The model should be extendable to 3D and still be computationally manageable (fit in memory and ideally on a single GPU (8GB)).
- Geologically meaningful.

The initial suggestion is to use the same models as those presented in the Wavelet Criterion paper, which include a model featuring block intrusions of low velocity and a dipping high-velocity reflector, as well as a subsample of the Marmousi model. However, the Helmholtz solver employed in this project (j-Wave) does not implement a free-surface condition. Additionally, the Perfectly Matched Layer (PML) used to enforce absorbing boundaries in the j-Wave solver is included as part of the simulated domain. To address these limitations, a similar velocity model structure with adjusted size can be used to ensure that the PML does not affect the OED results.

After identifying a test model, FWI should be conducted using the comprehensive survey to validate the solvability of the inverse problem. The FWI implementation should leverage j-Wave and be designed to achieve results within a reasonable computational time frame.

Update 1: Choice of the comprehensive survey

Here I decided to go for a 2D grid. Main reasons are that the inversion converges very quickly (for different velocity model) and there is more flexibility in the placement of src/rec allowing for more difference between a poor and an optimal layout. We could also gain insight on the combination of a cross-hole and a surface acquisition. The default is that the loss landscape cannot be calculated and there is no interpretable solution.
Instead of having a single test model, multiple instances of the OPENFWI dataset are used has test model. Here is the results of an inversion with step_lenght=4.0, $\alpha = 0.9, \beta = 0.9$, single source gradient, with smoothing and L2 misfit.

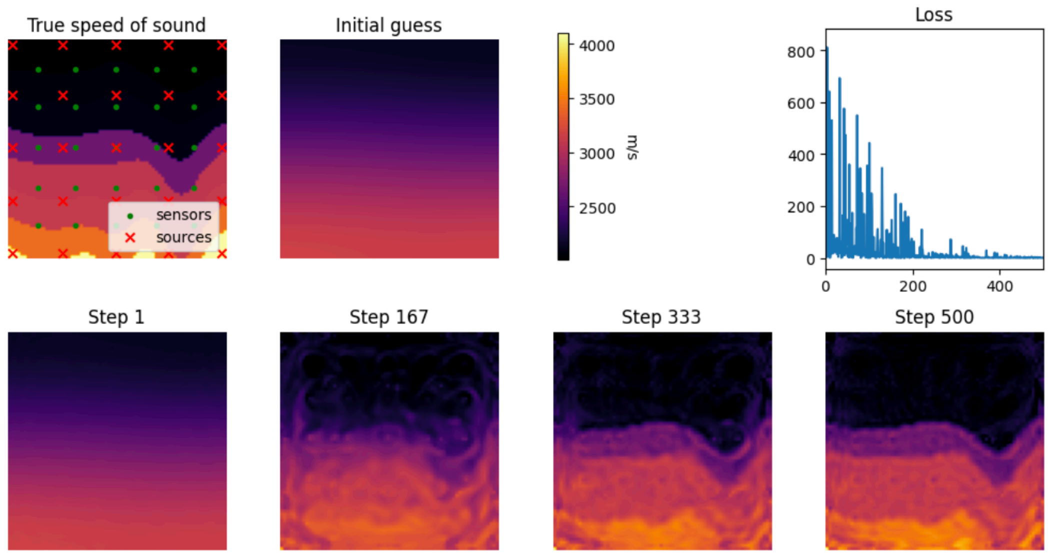


Figure 2: Comprehensive survey, model 7 of curvel

3.2 Definition of the reference $\mathcal{C}$ criterion

An important aspect of this project is the comparison of hybrid methods with well-established OED approaches. In this context, it is crucial to adopt a reliable and robust criterion for evaluating the performance of different $\mathcal{A}$, $\mathcal{C}$, and $\mathcal{F}^{-1}$ operators. The following are key requirements that the reference criterion should meet:

- Differentiable and stable.
- Based on the eigenvalues of $J^T J$.

The determinant criterion (Krampe et al. 2021) is simple to implement, differentiable but highly non stable given the low eigenvalues of $J^T J$. On the other hand, the nRER (Maurer et al. 2017) is more stable but not differentiable in the actual form. I suggest to bring 2 modifications to the classic nRER criterion:

1. In test case B detailed below, a comprehensive survey is not needed. To be able to use the exact same criterion for all test cases, I suggest to count directly the number of eigenvalues above a certain threshold. This modification does not alter the behaviour of the criterion but simplifies its application.
2. Introduce a weighting function instead of a hard threshold to make the criterion differentiable. Namely, I suggest the sigmoid function like depicted into Figure 3.

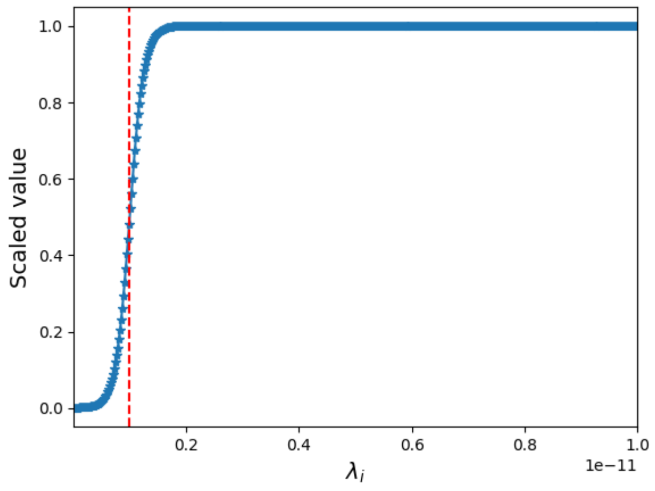


Figure 3: Sigmoid function used to weight the eigenvalues for the nRER criterion. The dashed red line is the hard threshold.

See preliminary test in Figure 4 in appendix.

3.3 Test case using hybrid methods

Here is a summary of the basic test cases that are planned in this project. Once every basic test case is successfully applied, multiple combination will also be implemented in order to investigate the contribution of each operator into a successful OED method.

	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{F}^{-1}$	Comments
Case A	$\mathcal{A}_O$	nRER	-	Traditional OED, sequential algorithm.
Case B	$\mathcal{A}_p : \mathrm{m} \mapsto \{\mathrm{x_s}, \mathrm{y_s}\}$	nRER	-	Continuous optimisation of the position of the sources
Case C	$\mathcal{A}_{NN} = NN(\theta, \mathrm{J})$	nRER	-	Replacing sequential algorithm by a NN that weights the rows of the Jacobians J.
Case D	$\mathcal{A}_O$	$\|\|\mathrm{m_t} - \mathrm{m_i}\|\|_2^2$	Adam	Using an OED metric based on the model differences. Sequential algorithm.
Case BD	$\mathcal{A}_p : \mathrm{m} \mapsto \{\mathrm{x_s}, \mathrm{y_s}\}$	$\|\|\mathrm{m_t} - \mathrm{m_i}\|\|_2^2$	Adam	Outer OED loop and inner FWI loop

Table 1: Summary of the basic test cases.

3.3.1 Case A

Case A is nothing more than the traditional sequential OED algorithm (See algorithm 1). The challenge here is to implement this algorithm in such a way that it can be differentiable. Given the compiled nature of JAX, many array must be static which requires using masking and padding to access specific rows of a matrix. As of the 2nd of December 2024 a preliminary implementation is still under testing. For a given dumb Jacobian matrix (only 5 non-zero rows), the implementation correctly identifies the non-zero row as best candidates to be added to the layout. A summary of the differentiable JAX implementation is detailed in section B. Further test are needed to validate this implementation and perform a complete OED workflow. Test case A takes has input a complete Jacobian matrix. Generating the Jacobian matrix is too long: 25 sources, 25 receivers, 5 frequencies takes about 45 minutes for a 70x70 a curvel model (on local machine and parallelised).

Algorithm 1 Generate Optimal Experiment Design by Selecting Sources (J_i) Using the nRER Criterion.

```
1:  Compute and store the comprehensive survey matrix $J$
2:  Initialize $J_{sub}$ as an empty matrix to store selected sources
3:  while the desired number of sources is not reached in $J_{sub}$ do
4:      Initialize $\mathcal{C}_{max} \leftarrow -\infty$ and source$_{max} \leftarrow$ None
5:      for each potential source$_{pot}$ with rows $J_i$ do
6:          Create a temporary candidate matrix $J_{pot} \leftarrow J_{sub} \cup J_i$
7:          Calculate the approximated Hessian: $H_{pot} = J_{pot}^T \cdot J_{pot}$
8:          Evaluate the criterion $\mathcal{C}_{pot} = \text{Count}\left(\lambda_i > \tau, \forall \lambda_i \in \text{eigvals}\left(H_{pot}\right)\right)$.
9:          if $\mathcal{C}_{pot} > \mathcal{C}_{max}$ then
10:             Update $\mathcal{C}_{max} \leftarrow \mathcal{C}_{pot}$ and source$_{max} \leftarrow$ source$_{pot}$
11:         end if
12:     end for
13:     Add the best source source$_{max}$ to $J_{sub}$
14: end while
15: Return the optimized subset of sources $J_{sub}$
```

Update 2: Implementation of sequential algorithm

The implementation is fully in JAX. It uses mask to track which rows of the Jacobian are already choosen. Optimise for single triplets of src/rec/freq or for a src including all the rec. The algorithm is benchmarked against the classic implementation (MODJO package) for single row implementation of a truncated Jacobians (1000x32768). There is an integration test that has to be ran locally. There is no implementation for a src/rec and all freq like in the wavelet paper.

3.3.2 Case B

This test case continuously optimise the position of the sources and receivers. This method is already implemented and results have already been showed. It leverages the auto-differentiation capability of JAX to obtain the gradient of a specific criterion with respect to the source and receivers position. There is no sequential algorithm needed for this test case. Although this test case is interesting, it requires some hyper-parameter tuning (learning rate). From preliminary study (loss landscape computation for toy problem), the criterion chosen does not always produce smooth and optimisable loss landscape. This greatly impacts the capability of converging to a proper minimum.

Update 3: Position optimisation

The position optimisation has been update to include many src/rec and a larger domain. Using GPU, a single optimisation iteration takes about 35 seconds. There is only a single frequency implementation. The loss function is based on the variance of the eigenvalues but it should be changed for a differentiable version of the nRER. Since the test model is composed of many src/rec and a complex velocity model, there is no simple interpretable solution.

3.3.3 Case C

Test Case C introduces the first learnable component. The goal here is to relax the sequential nature of the original algorithm and explore the potential for discovering better masks (source selections). To achieve this, a simple fully connected network

(FCN) is used to select the k best sources from the comprehensive Jacobian matrix. Instead of adding one source after the other, the network focuses on finding the best k directly. In essence, the for-loop at line 5 in Algorithm 1 is replaced by an FCN that updates the mask M. The input of the FCN is the criterion evaluation ($\mathcal{C}$, e.g., the nRER) for each row of the Jacobian, and the output is a mask M weighting the rows of the comprehensive Jacobian matrix (J). The final survey is then defined as:

$$\mathrm{J}_{NN,k} = \mathrm{J}^T \mathrm{M} \tag{1}$$

Where $\mathrm{J} \in \mathbb{C}^{n \times m}$ and $\mathrm{M} \in \mathbb{R}^n$ with n being the number of data-points and m the number of model parameter in the velocity model. M is a mask that select or deselect rows of J.

Since we want to relax the sequential approach and let the method discover better masks, we avoid basing the training of the FCN on the results of the sequential method. Additionally, we aim to eliminate the need for a large labelled dataset, which would require performing optimal experimental design (OED) for every training data point. To address these goals, I propose a self-supervised approach where the network directly minimize the loss function. The loss function is related to the nRER criterion, which maximize the number of eigenvalues of

$$\mathrm{H}_{NN,k} = \left(\mathrm{J}^T \mathrm{M}\right)^T \left(\mathrm{J}^T \mathrm{M}\right) \tag{2}$$

above a certain threshold τ for the masked (optimized) Jacobian. The loss function $\mathcal{L}$ is defined by

$$\mathcal{L} = -N_\tau \tag{3}$$

with

$$N_\tau = \mathrm{Count}\left(\lambda_i > \tau, \forall \lambda_i \in \mathrm{eigvals}\left(\mathrm{H}_{NN,k}\right)\right). \tag{4}$$

This approach allows the network to learn directly from the underlying criterion without bias from the sequential method. The testing of the trained network involves comparing the nRER values produced by the network with those obtained from the sequential method.

In summary the training and testing dataset will be composed of:

- Training:
  - Velocity models, Jacobian matrices, and $\mathcal{C}$ values for each row.
  - Subset of OpenFWI (Deng et al. 2021) dataset. To start, the training dataset will be constructed with 200 velocity models from each of following category: CurveVel, CurveFault-A, CurveFault-B, FlatFault-B and Style-A. By default, these velocity models are of size 70×70.
- Testing:
  - Velocity models, Jacobian matrices, $\mathcal{C}$ values for each row.
  - Masks from the sequential method (for comparison).

In terms of the FCN architecture, a good starting point would be between 3 to 5 layers that decreases in dimension. For example, given 8 possible sources, 8 possible receivers and 4 frequencies ($n = 256$):

$$n \rightarrow 2048 \rightarrow 1024 \rightarrow 512 \rightarrow n$$

The last layer having a sigmoid activation function to generate the wanted mask.

In a subsequent step, more flexibility is given to the learnable component. Instead of feeding the criterion evaluation ($\mathcal{C}$) for each row, a convolutional neural network (CNN) could be used to extract spatial information from the rows and predict the mask M. This implementation is more complex and requires careful consideration of the best way to feed the Jacobian matrix into the CNN. Potential inputs for the CNN include the Jacobian matrix itself, its combination with $\mathcal{C}$, or the singular values of the Jacobian. The same loss function would be use as in Equation 3.

In a first test the input for this CNN could be a tensor of shape $n \times 70 \times 70$ representing the Jacobian matrix itself. A starting point for the CNN architecture could then to use Conv2D layers (2-3) to extract features from every row of the Jacobians. Following these convolution layers, the output could be flatten to obtain a 2D matrix of $n \times f$ where f is the number of features. To capture inter-row information, Conv1D layer could be used on the columns of this matrix. The last layers would be a fully-connected layer with sigmoid activation function to produces the wanted mask $\mathrm{M} \in \mathbb{R}^n$.

Update 4: FCN implementation

The first implementation of a Fully Connected Network to predict the best sources from a comprehensive survey. The inputs are the evaluation of the nRER criterion for each sources. This leads to an input vector of the size (n-sources x 1). This approach is independant of the number of model parameters. The network is then compromised of 4 layers (25x56, 56x56, 56x56, 56x25). The output is then a mask with the same dimension as the input that can be applied to select rows of the Jacobian matrix. The activation functions are tanh for the first 3 layers and then sigmoid for the last to enforce a probability output. The loss function is a differentiable version of the nRER criterion. Then, a sigmoid function with a specific sharpness is applied to the ouptut vector leading to a mask with either very high values (≈ 1) or very low values (≈ 0). This mask is then repeated to span the size of the full Jacobian and multiplied by the full Jacobian. Some regularisation is applied on the output mask in order to obtain a relevant output. Namely a L1 norm loss of the output mask and a top k regularisation.

3.3.4 Case D

Test case D does not have any learning components and will use the sequential algorithm ($\mathcal{A}_O$). However, it leverage the auto-differentiation capabilities of JAX. Test case D aims at relaxing the use of an approximation criterion to evaluate the layouts. The suggestion here is to implement a criterion based on the results of a few FWI iterations. The criterion $\mathcal{C}$ becomes the norm between the true model ($\mathrm{m}_{\mathrm{true}}$) and the reconstructed model (m_k) with only k sources.

$$\mathcal{C} = ||\mathrm{m}_{\mathrm{true}} - \mathrm{m}_k||_2^2 \tag{5}$$

This process will require numerous FWI iterations to achieve the optimal layout, emphasizing the need for a small and simple test velocity model.

If test cases B and C are proven successful, their respective operators, $\mathcal{A}_p$ and $\mathcal{A}_{NN}$, will be prioritized. These operators eliminate the need for a sequential algorithm, thereby reducing the number of FWI iterations required to construct an optimal survey design. If $\mathcal{A}_{NN}$ is used to select the rows, the training would be the same as suggested in test case C (self-supervised learning) with the loss function being the criterion $\mathcal{C}$ in Equation 5.

3.4 Software to develop

Here are a few important software implementation that is needed in order to implement the 4 test cases and their combination. This is a non-exhaustive list and will likely change during the first stage of the implementation.

- Forward solver wrapper
- Sequential OED algorithm.
- Criterion helper function that can be called inside the sequential algorithm or in a continuous optimisation.
- Learnable component module using `Equinox` package. Full waveform inversion in time domain based on the j-Wave example for CT.

3.5 Time estimates

Arnaud's PhD project includes other significant components, such as external collaborations. Given the possibility of downtime during these collaborations, a flexible work schedule is recommended. On average, it is proposed that Arnaud dedicates 40% of his time to developing hybrid OED methods during January and February 2025. During this period, and barring any unforeseen issues, tests A, B, and C (FCN, not CNN) should be implemented. Following February, the objectives and timeline can be reassessed.

For supervision and support, it is suggested to arrange meetings with Ben Moseley as needed, either bi-weekly or monthly, to review progress and provide feedback on algorithm implementation and tuning. Progress updates and detailed development can be tracked via the GitHub repository: Hybrid-OED.

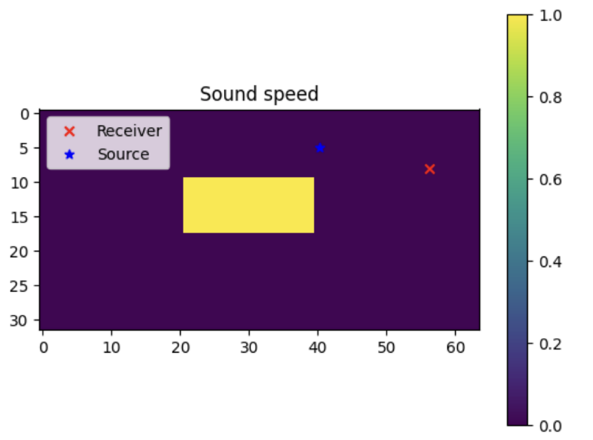
References

Deng, Chengyuan, Shihang Feng, Hanchen Wang, Xitong Zhang, Peng Jin, Yinan Feng, Qili Zeng, Yinpeng Chen, and Youzuo Lin (Nov. 4, 2021). "OpenFWI: Large-scale multi-structural benchmark datasets for seismic full waveform inversion". In: *arXiv [cs.LG]*. arXiv: 2111.02926 [cs.LG].

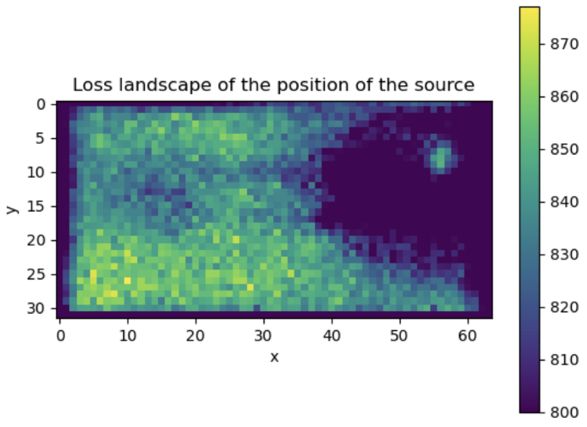
Krampe, Valérie, Pascal Edme, and Hansruedi Maurer (Jan. 2021). "Optimized experimental design for seismic full waveform inversion: A computationally efficient method including a flexible implementation of acquisition costs". en. In: *Geophysical Prospecting* 69 (1), pp. 152–166. ISSN: 0016-8025,1365-2478. DOI: 10.1111/1365-2478.13040.

Maurer, Hansruedi, André Nuber, Naiara Korta Martiartu, Fabienne Reiser, Christian Boehm, Edgar Manukyan, Cédric Schmelzbach, and Andreas Fichtner (2017). "Optimized experimental design in the context of seismic full waveform inversion and seismic waveform imaging". In: *Advances in Geophysics*. Vol. 58. Advances in geophysics. Elsevier, pp. 1–45. ISBN: 9780128124130. DOI: 10.1016/bs.agph.2017.10.001.

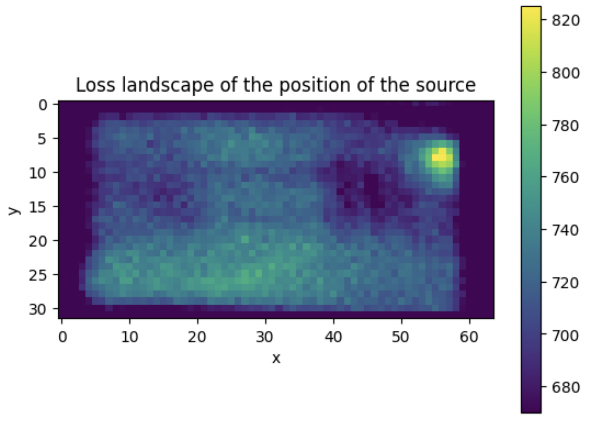
A Preliminary test with nRER criterion



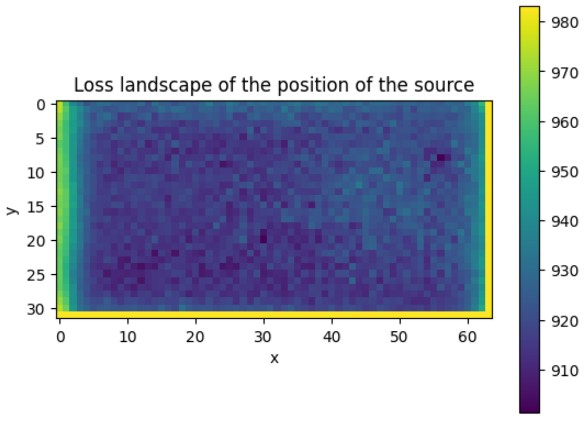
(a) Velocity model with source and receiver. The receiver is kept fixed while the all possible source positions is computed. background velocity is 1540 m/s and the intrusion is at 2000 m/s.



(b) Loss landscape for the nRER criterion for a threshold of 1e-12.



(c) Loss landscape for the nRER criterion for a threshold of 1e-14.



(d) Loss landscape for the nRER criterion for a threshold of 1e-14 and weighted by a sigmoid function to enable differentiation.

Figure 4: Loss landscape for different nRER criterion formulation. There is no comprehensive survey, the λ_i are counted until the threshold. Here the optimal value is the highest value.

B Differentiable implementation of nRER criterion

Here I detail the process of transforming a discrete row selection algorithm into a fully differentiable implementation using JAX. The task involves iteratively selecting rows from a matrix J_c, based on an eigenvalue-based criterion, to form a new matrix J_o. The implementation must ensure differentiability to leverage JAX's automatic differentiation capabilities while adhering to constraints such as preventing reselection of rows.

Given a matrix $J_c \in \mathbb{R}^{n\times m}$, the objective is to construct a submatrix $J_o \in \mathbb{R}^{k\times m}$ by iteratively selecting k rows from J_c. The selection is based on a criterion that measures the quality of J_o in terms of the eigenvalues of $J_o^T J_o$. Specifically:

- The eigenvalue-based criterion prioritizes rows that maximize the number of eigenvalues exceeding a given threshold.
- Selected rows must not be reselected in subsequent iterations.

Eigenvalue-Based Criterion

The eigenvalue-based criterion is defined as:

$$\text{criterion}(J_o) = \sum_{i=1}^{m} \sigma(\lambda_i - \text{threshold}),$$

where:

- λ_i are the eigenvalues of $J_o^T J_o$.
- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, ensuring differentiability.

This criterion promotes rows that contribute to increasing the eigenvalues of $J_o^T J_o$ beyond the threshold.

Techniques for Differentiable Implementation

To adapt the discrete row selection process for JAX, the following techniques were employed:

1. Iterative Row Selection

The submatrix J_o is constructed iteratively. At each iteration:

1. A row from J_c is tentatively added to J_o.
2. The eigenvalue-based criterion is evaluated for the updated J_o.
3. The row that maximizes the criterion improvement is selected.

2. Preventing Reselection

To ensure rows are not reselected:

- A *selection mask* selection_mask $\in \mathbb{R}^n$ is maintained. Initially, all entries are set to 1.
- After a row is selected, its corresponding entry in selection_mask is set to 0.
- Scores for rows with selection_mask = 0 are set to $-\infty$ to exclude them from future selection.

3. Differentiability via Soft Operations

To maintain differentiability:

- Hard row selection (e.g., using argmax) is performed only after all scores are computed, ensuring differentiability of the scoring process.
- The scoring function uses $jax.vmap$ for vectorized computation of criterion improvements for all rows in J_c.

4. Efficient Iterative Implementation with JAX `scan`

The iterative process is implemented using JAX's `lax.scan` to:

- Efficiently loop through the iterations while preserving static shapes.
- Log the criterion values and selection history for debugging and analysis.

Visualization of Results

The following visualizations are generated to analyze the selection process:

- **Criterion Improvement Heatmap**: Shows the criterion values for all rows across iterations, highlighting the quality of each row's contribution.
- **Selection History Heatmap**: Annotates the iteration number at which each row was selected, providing an intuitive understanding of the selection sequence.
- **Final Selection Vector**: A bar plot of the vector O, indicating the rows selected for J_o.

Key Code Components

Eigenvalue Criterion Function

```
def eigenvalue_criterion(J_o, threshold=0.5):
    JTJ = J_o.T @ J_o
    eigenvalues = jnp.linalg.eigh(JTJ)[0]
    return jnp.sum(jax.nn.sigmoid(eigenvalues - threshold))
```

Iterative Selection with Prevention of Reselection

```
def iterative_selection_no_reselection(J_c, num_rows, threshold=0.5):
    ...
    def step_fn(carry, _):
        ...
        scores = jax.vmap(compute_score)(J_c)
        scores = jnp.where(selection_mask == 1, scores, -jnp.inf)
        ...
        selection_mask = selection_mask.at[best_row_idx].set(0)
        ...
```

Algorithm 2 Differentiable Row Selection with Eigenvalue-Based Criterion

```
Require: $J_c \in \mathbb{R}^{n\times m}$ (input matrix), $k$ (number of rows to select), threshold (eigenvalue threshold)
Ensure: $J_o \in \mathbb{R}^{k\times m}$ (selected rows)
 1: Initialize $J_o \leftarrow \mathbf{0}_{k\times m}$ (empty matrix for selected rows)
 2: Initialize $O \leftarrow \mathbf{0}_n$ (selection vector)
 3: Initialize mask $\leftarrow \mathbf{0}_k$ (active row mask for $J_o$)
 4: Initialize selection_mask $\leftarrow \mathbf{1}_n$ (eligible rows in $J_c$)
 5: for $t = 1,\ldots,k$ do
 6:     $J_o^{\text{active}} \leftarrow J_o \cdot \text{mask}[:, \text{None}]$          ▷ Apply mask to $J_o$
 7:     Compute current_criterion ← eigenvalue_criterion($J_o^{\text{active}}$, threshold)
 8:     Define function compute_score(row):
 9:         Compute $J_o^{\text{temp}} \leftarrow J_o^{\text{active}}[\text{idx}, \text{row}]$          ▷ Tentatively add row at the first non-zero row
10:         Return eigenvalue_criterion($J_o^{\text{temp}}$, threshold) − current_criterion
11:     Compute scores ← vectorize(compute_score)($J_c$)          ▷ Apply to all rows
12:     Apply selection mask: scores ← where(selection_mask = 1, scores, $-\infty$)
13:     Select best row: best_row_idx ← argmax(scores)
14:     Add selected row to $J_o$:
15:         Find next available row index: idx ← argmax(mask = 0)
16:         Update $J_o$[idx] ← $J_c$[best_row_idx]
17:     Update mask: mask[idx] ← 1
18:     Update selection vector: $O$[best_row_idx] ← 1
19:     Exclude selected row: selection_mask[best_row_idx] ← 0
20: end for
21: return $J_o$, $O$
```

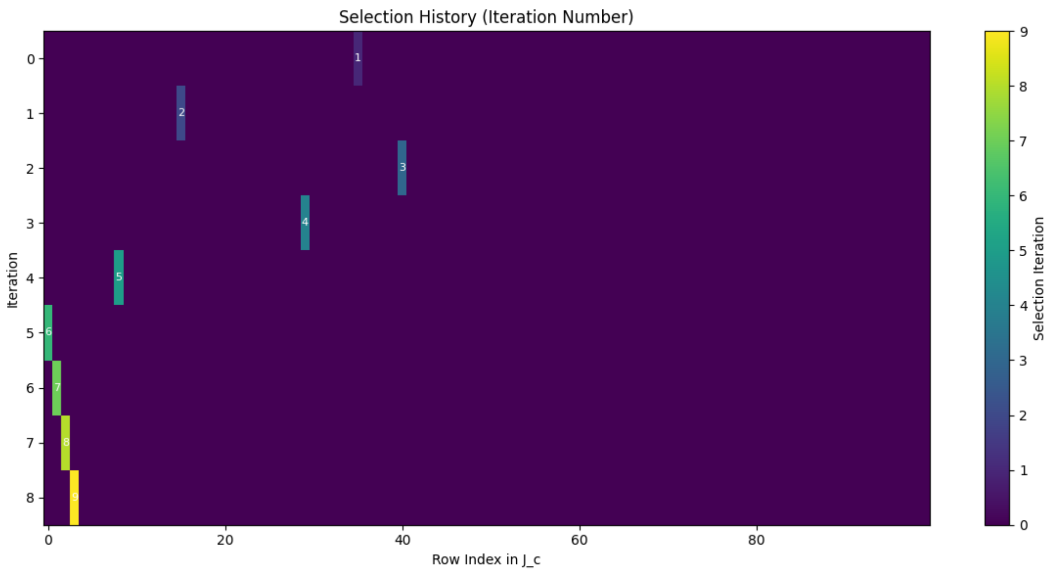


Figure 5: Representation of the order of row selection. Here 5 rows of J_c are non-zero. We can see that these row are selected first and then the following 4 rows are from 0 to 3.

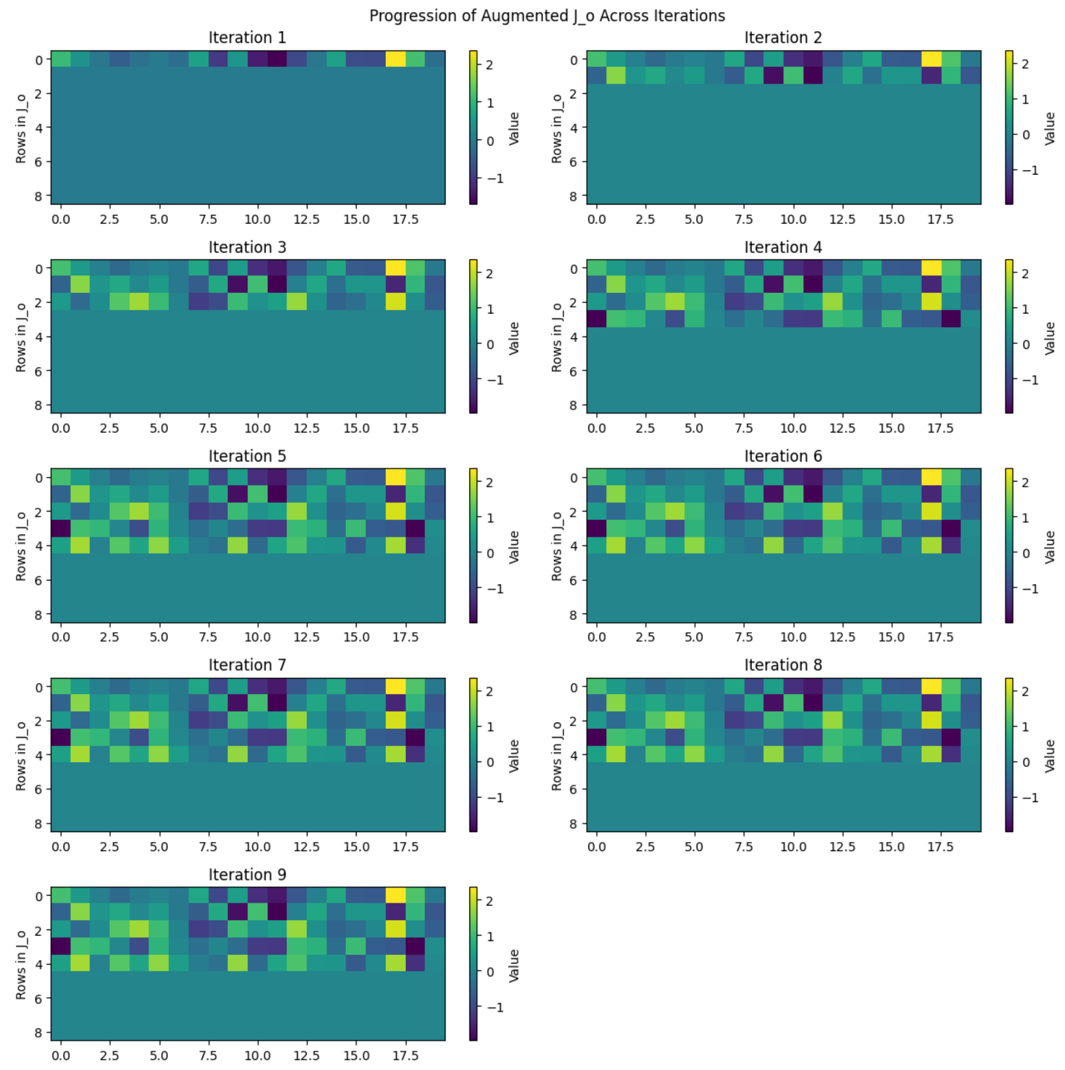


Figure 6: Progression of the augmented J_o at each iteration. We can see that the first 5 non-zero rows are appended at the first non-zero row and then the following are zeroed rows.