Phase-encoded inversion and randomized sampling

Applying linearized inversion to many thousands of shots can quickly become unfeasible; it is possible to intelligently reduce the data size by using two classes of techniques - phase encoding and randomized sampling. It is also possible to augment these two concepts to further increase inversion feasibility. This chapter will address this data size problem, and analyze these solutions.

PHASE ENCODING

Typically, in seismic imaging the size of the data space is much larger than the model space, thus data movement can become an impediment to an efficient inversion scheme. Phase encoding is a method of artificially combining (blending) data that will result in a algorithm that favours additional computation, per iteration, over reduced data movement. In some cases, the overall computation can be reduced. The aim is to macroscopically improve convergence as a function of overall cost.

In phase encoding seismic shots are scaled, shifted and summed together to create a series of super-shots (**?**; **?**). In the most extreme case all shots can be combined to create a single supershot. For Ocean Bottom Node (OBN) or Ocean Bottom Cable (OBC) type surveys this can be viable, since often the receivers are fixed. How many super-shots one should create is a question of survey geometry and source sampling (**?**).

As discussed in the previous and final chapters, data movement can be a significant source of cost during conventional migration and modeling (**?**). Every shot must be read and written, from disk, during every iteration. This can create a bottleneck both within and across iterations. Furthermore, as discussed in the final chapter, a GPU based propagator is used. This greatly accelerates the speed of wave propagation, exaggerating the penalties of I/O and disk access. Phase encoding reduces the data-space dimensionality within each iteration, favouring additional computation over disk access. Under many conditions this can give a significant reduction of cost for a given level of sub-surface property convergence. Again, this discussion will focus on scattering potential recovery.

Designing the encoding matrix

The first decision is how to combine these data into subsets. Consider $\mathbf{d}_e = \alpha\mathbf{d}$, where $\mathbf{d}$ is the full data vector, $\mathbf{d}_e$ is a supershot (or series of supershots) and α is the encoding matrix. This encoding matrix contains all information about how to shift, scale and sum the shots together.

Designing α is both a question of convergence properties, efficiency, and data geometry. Phase encoding, for the purposes of seismic imaging, was first postulated in **?**, in the context of combining shots for prestack migration. Herein it is concluded

that using a random series of time shifts provides the best results. Furthermore it is claimed that no more then ten shots should be combined at a time. However, if post-imaging amplitude analysis is desirable such a technique can have drawbacks (**?**).

For pre-stack imaging a variety of different encoding functions, α, have been analysed, in addition to simple random shifts (**?**). For example, plane wave encoding, **?**, can perform very well for dense source sampling, since crosstalk artifacts can be mitigated. In the context of phase encoded inversion, as opposed to migration, many more shots can be combined since the solver searches for common model properties between iterations. **?** used a single bit encoding function (1s and -1s) to scale the data, performing no time shifts. This results in the most efficient convergence, since the time records are not being prolongated and the cost of each iteration is massively smaller. Many iterations are required to reduce artifacts, thus relative usefulness is determined as a function of cost and convergence.

Phase encoded inversion

Linearized inversion can be extended to incorporate phase encoding. The most natural extension will be termed 'static' phase encoding, whereby the encoding function, α, is constant across iterations. Under this condition the main loop of the algorithm remains unchanged, but in initialisation the encoding is applied to the full dataset to create these reduced data, $\mathbf{d}_e$. Each loop is significantly faster to implement since only super-shots are being looped over, rather than each individual shot. Solver issues aside, this speed up will be directly proportional to the ratio of the total number of shots to the number of super-shots, $n_{shots}/n_{supershots}$. This is largely similar to linearized inversion and is summarized in algorithm 1.

It is immediately apparent that such a scheme will induce many artifacts; moreover, these artifacts will be consistent between iterations. Consequently, many iterations will be required to reach a given level of convergence. Given this, a comparison between phase encoded inversion and linearized inversion will be done under the context of data-residual reduction as a function of computational cost. This cost will be normalized to one full iteration of linearized inversion and convergence will be a normalized measure of the data space residual.

Reducing the cross-talk artifacts in this way can be a slow process. These artifacts are reduced significantly more quickly if the encoding function is changed between iterations. This will be termed 'dynamic' phase encoding (**?**). This creates a huge advantage for cross-talk reduction: these artifacts will now be incoherent between iterations, causing them to stack-out rapidly. The solver will search for common model components between iterations and thus residual reduction, especially for early iterations, will be accelerated.

The caveat with dynamic phase encoding, however, is that by changing the encoding between iterations the 'observed data,' that each loop is implicitly considering,

Algorithm 1 Static phase encoding

```
Create α
d_e = αd
while iter < n_iter; iter++ do
  Create gradient of d_e
  Create conjugate gradient (size of d_e)
  Model update
end while
Output model
```

are changed. Thus the solver cannot update the data-space residual, which is needed to calculate the subsequent gradient. This process is now non-linear, and before the gradient calculation an additional forward modeling step is needed to estimate the updated data-space residual. The consequence of these attributes is that a non-linear solver must be used, and the cost (per iteration) is roughly 1.5x that of the static algorithm. The dynamic sequence is summarized in algorithm 2.

Algorithm 2 Dynamic phase encoding

```
while iter < n_iter; iter++ do
  Create α
  d_e = αd
  Forward model to estimate d_e residual
  Create gradient of d_e
  Create conjugate gradient (size of d_e)
  Model update
end while
Output model
```

The relative advantage of phase encoded inversion is tightly tied to survey type and geometry. Under certain circumstances, the advantage over linearized inversion can be prodigious, under other conditions it can be detrimental. These will be considered for a variety of marine conditions.

For a streamer type geometry, phase-encoding causes two issues. Firstly, the aperture of each supershot is increased as a function of the number of combined shots. The receivers are constantly moving, and each must be considered when forward modeling. As a result, for streamer surveys, the choice of how many shots to combine, or indeed whether phase encoding is advantageous at all, can be hard to determine. Secondly, since the receivers are not static, some shots will not have the full complement of recorded data in order to calculate residuals. This can thought of as: when combining shots together, not each shot is present in each receiver.

For Ocean Bottom Cable (OBC) or Ocean Bottom Node (OBN) type surveys the receivers are spatially fixed (at least for a group of shots). Subsequently, for each

combined shot there may be no aperture increase. For inversion, under the most extreme condition, all shots can be summed into one supershot. Also, each receiver has the full complement of shots. The cost of migrating this single supershot will be the same cost as migrating any of the individual shots.

Initially, this latter type of survey will be analyzed. A section of the SEAM model will be used again, with fixed receivers and source positions. Using synthetic modeling 120 shots were simulated over the same model section in the previous chapter and the same linearized results from that chapter will be used for comparison. To push the algorithm, and to reduce data movement as much as possible, all shots will be scaled by a randomly selected single bit function and combined into a single super shot, with no time shifts. Dynamic phase-encoding will be the method used, and this will create a scheme that maximally favors computation over data movement.

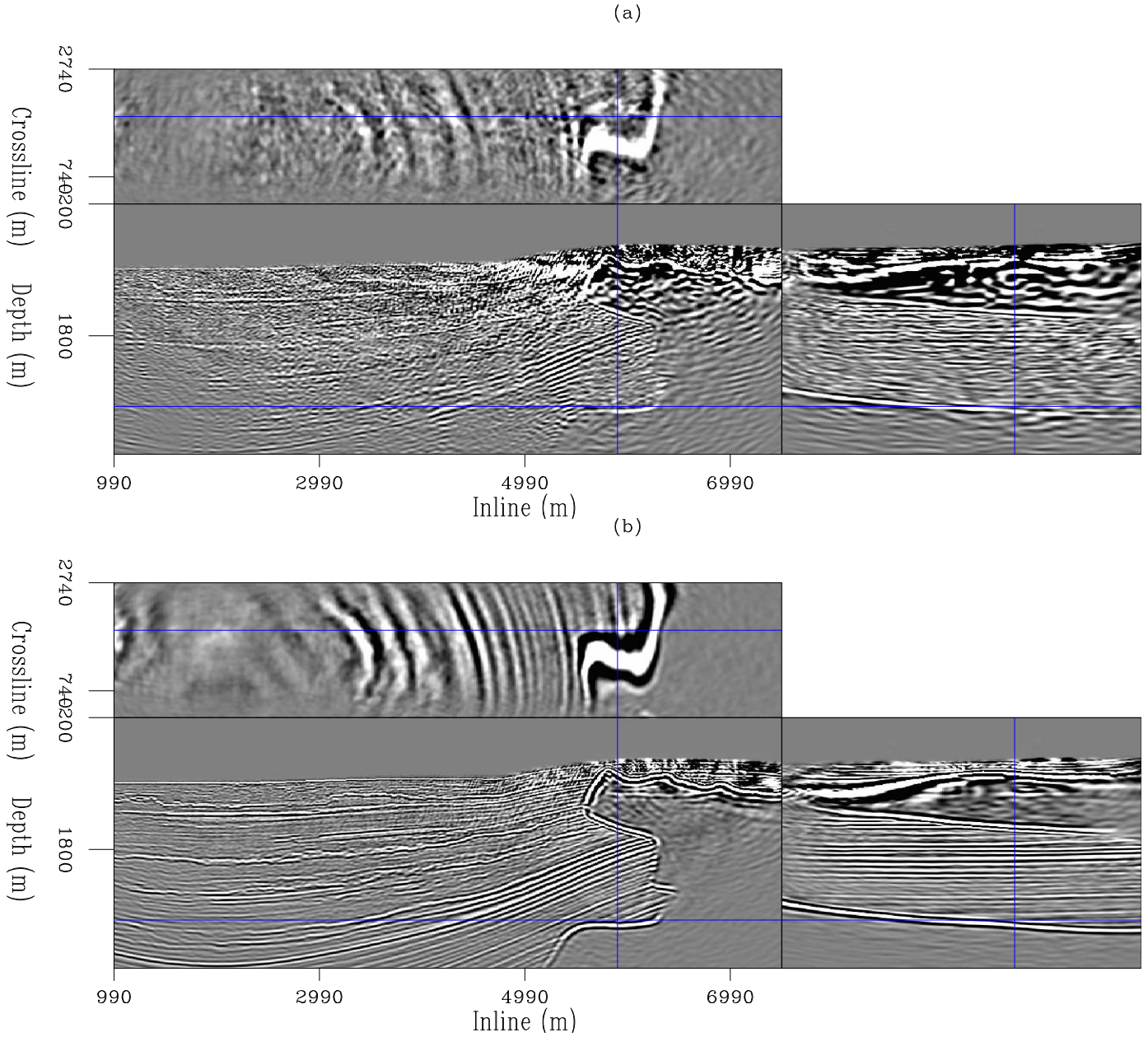

Figure 1: The result after one iteration of phase encoded inversion, (a), and after 50 iterations, (b). Here 120 shots were combined into one supershot. [**CR**]

Unsurprisingly, early images for this choice of data combination are noisy, although macroscopic structure is discernible, and many keys reflectors can be mapped. A comparison between the first iteration image and the 50_{th} can be seen in Figure 1. Figure 2 then shows the first iteration image from LSRTM compared to the equivalent cost image from phase encoding. It is immediately apparent that the image from phase encoding has improved amplitude balancing, features higher wavenumber content, a reduced acquisition footprint, and an increase in high wavenumber artifacts. Phase encoding preserves all dips present in the data, however it noticeable (particularly in the top panel) that resolution is lost towards the edge of the model, where data redundancy is reduced. This is fairly intuitive, since the phase encoding scheme is more reliant on stacking for coherency.

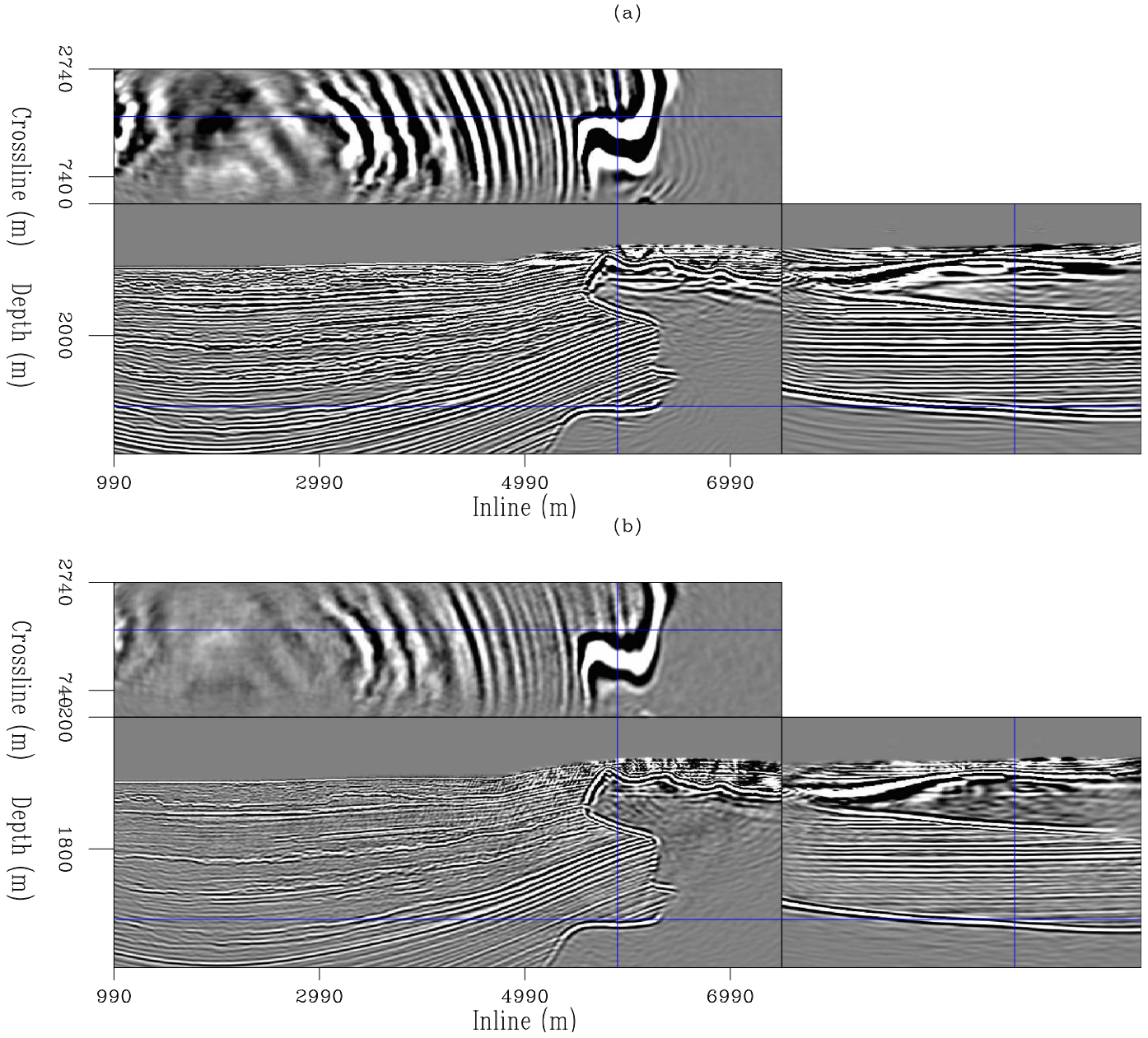

Figure 2: The result after one iteration of LSRTM, (a), and after 50 iterations of phase encoded inversion, (b). [**CR**]

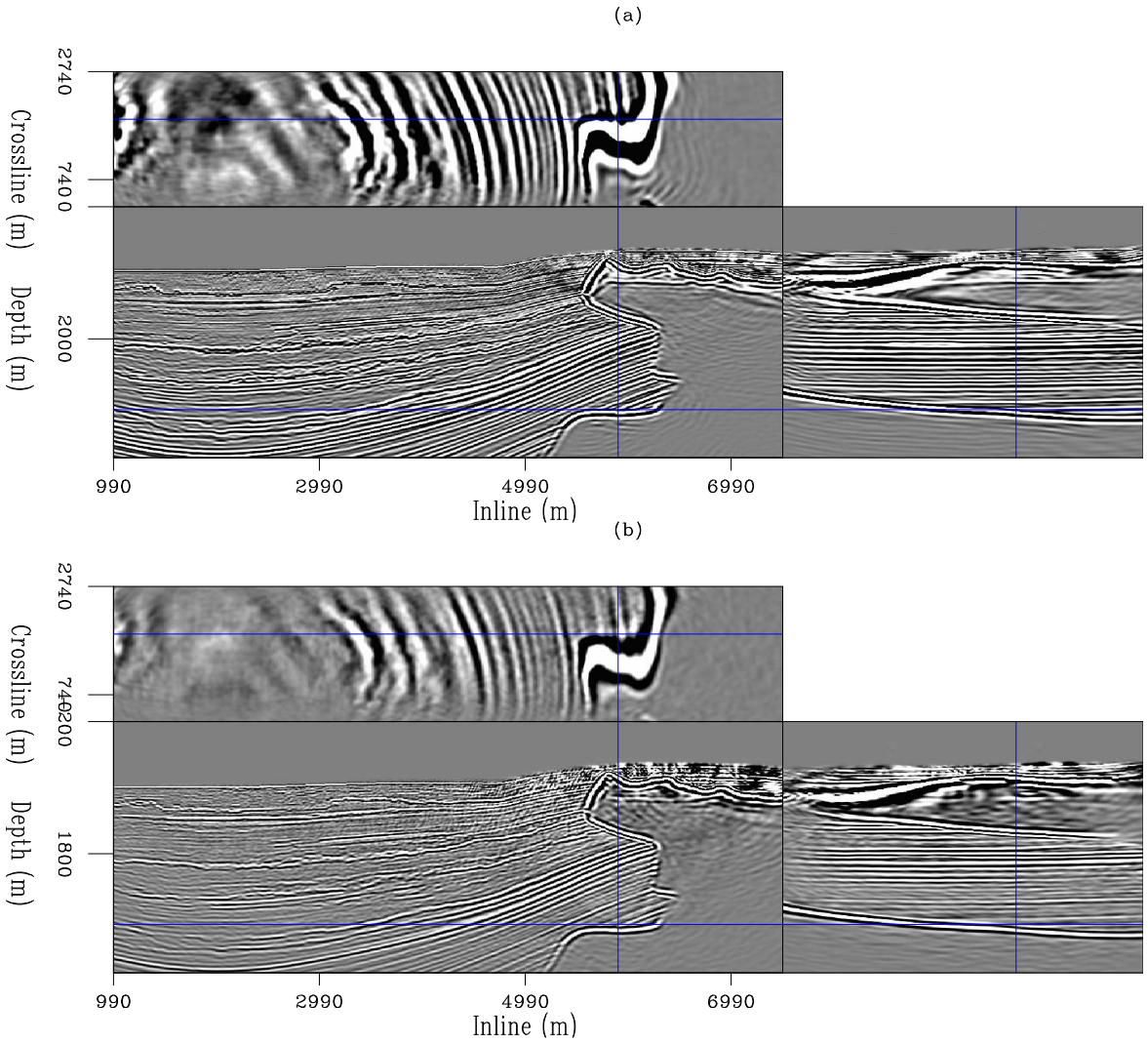

Figure 3: The result after ten iterations of LSRTM, (a), and after 50 iterations of phase encoded inversion, (b). [**CR**]

Next, Figure 3 shows a comparison of ten iterations of LSRTM against the equivalent cost image of phase encoded inversion. At this stage the amplitude and wavenumber fidelity of the LSRTM image is much improved, and the relative differences between the images to harder to ascertain. This makes sense by comparing the relative convergence properties as a function of cost, Figure 4 and Figure 5. This shows that phase encoding falls to a lower residual quickly, but can then plateau. As a function of iteration number, phase encoding is less favourable (in terms of data space residual reduction.) However, when this is posed as a cost comparison, phase encoding can exhibit significant residual reduction before LSRTM has completed a single iteration. A viable strategy would be to use phase encoding to estimate an initial model, and then perform several iterations of LSRTM to reduce noise and represent the amplitude character more representatively.

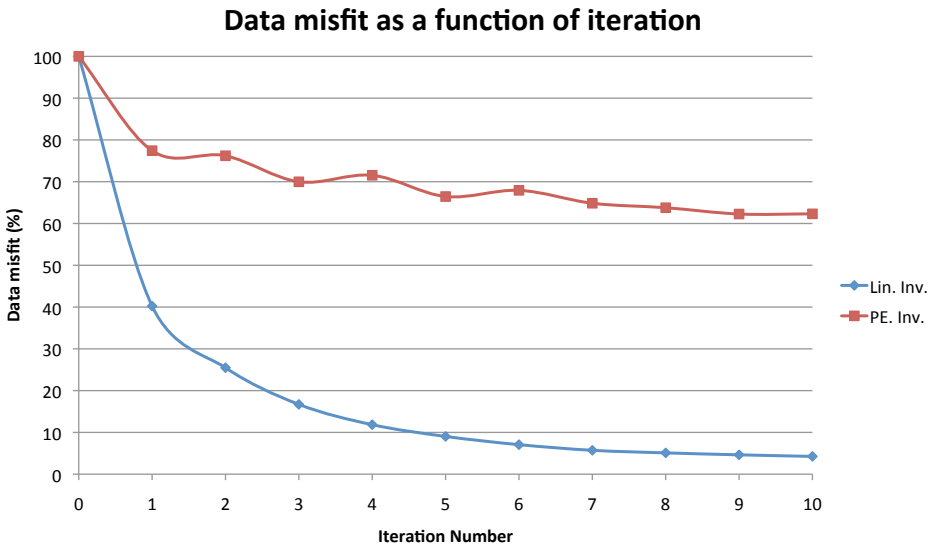

Figure 4: How the data fit improves as a function of iteration number, for linearized inversion and for phase encoded inversion. [**NR**]

Parallel data encoding

Although within each iteration data movement is massively reduced, however, considering the cost of the encoding can be important. When combining (reading, summing and writing) 120 shots from disk, which are then imaged and modeled, the cost of the encoding is almost as expensive as the rest of the procedure. This is because disk access can be very slow, especially when compared to an optimized GPU based propagator. To illustrate, a typical hard-disk used for this study could transfer to the CPU at around 200 Mb/s, then the CPU-GPU transfer speed saturates at 2 Gb/s, and the GPU (Fermi M2090) can operate at a bandwidth of 177 Gb/s.

Fortunately, there is a viable solution, which takes advantage of the multi-threading

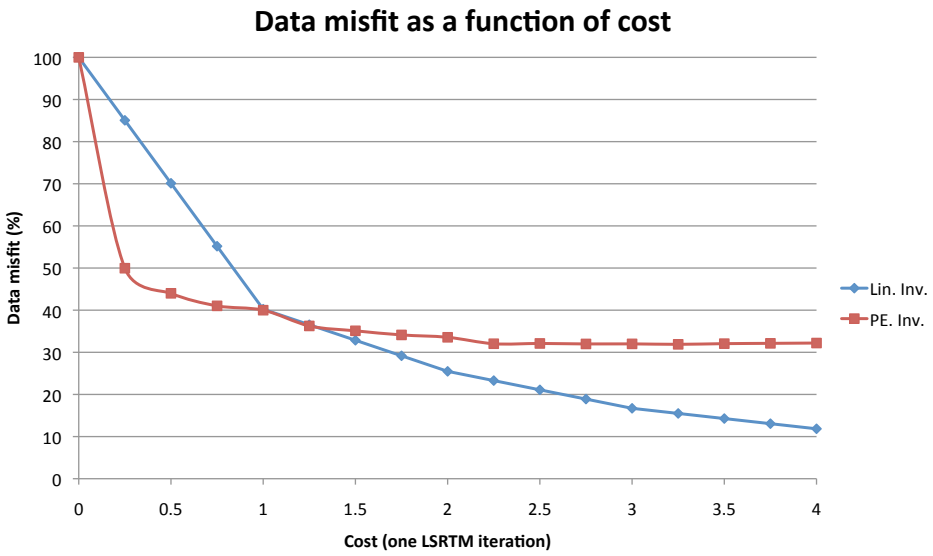

Figure 5: The same plot as Figure 4, but as a function of cost. The cost is normalized to that of one iteration of LSRTM, with results for LSRTM linearly interpolated for the sake of representation. [**NR**]

capability of CPUs. The encoding matrix α changes between each iteration, but each subsequent α is independent of previous choices, and of previous iterations. During GPU propagation, a separate compute thread can be used to prepare the encoded data for the next iteration. Assuming encoding time is faster than the cost of the iteration, this will be hidden for all subsequent loops. For the first iteration waiting for encoding to complete is unavoidable.

RANDOMIZED SAMPLING

A different class of data-space reduction known as randomized sampling (**?**). This is often also referred to as stochastic dimensionality reduction, or as stochastic gradient methods (**?**; **?**). Conceptually, randomized sampling is very simple; between iterations a different subsection of the data is ignored. How much data is ignored, and how many iterations are required, will again be a function of survey geometry and shot density.

This concept has been analysed for both LSRTM (**?**; **?**) and Full Waveform Inversion (FWI) (**?**; **?**). This discussion will briefly look at recreating similar results to these, and to then extend the concept to incorporate phase encoding.

How much data do we need?

Selecting the subset of the data is simply done by designing random function of zeroes and ones, with the number of ones corresponding to the percent of the input data to be used for the given iteration. This can be thought of as a separate encoding sequence of zeroes and ones. The same SEAM 120 shot 3D dataset was used, validating equivalent comparisons to the LSRTM and phase encoding results. Stochastic inversion was run using 75%, 50% and 25% of the data over a range of iterations and each was then normalized to the cost of one iteration using 100% of the data.

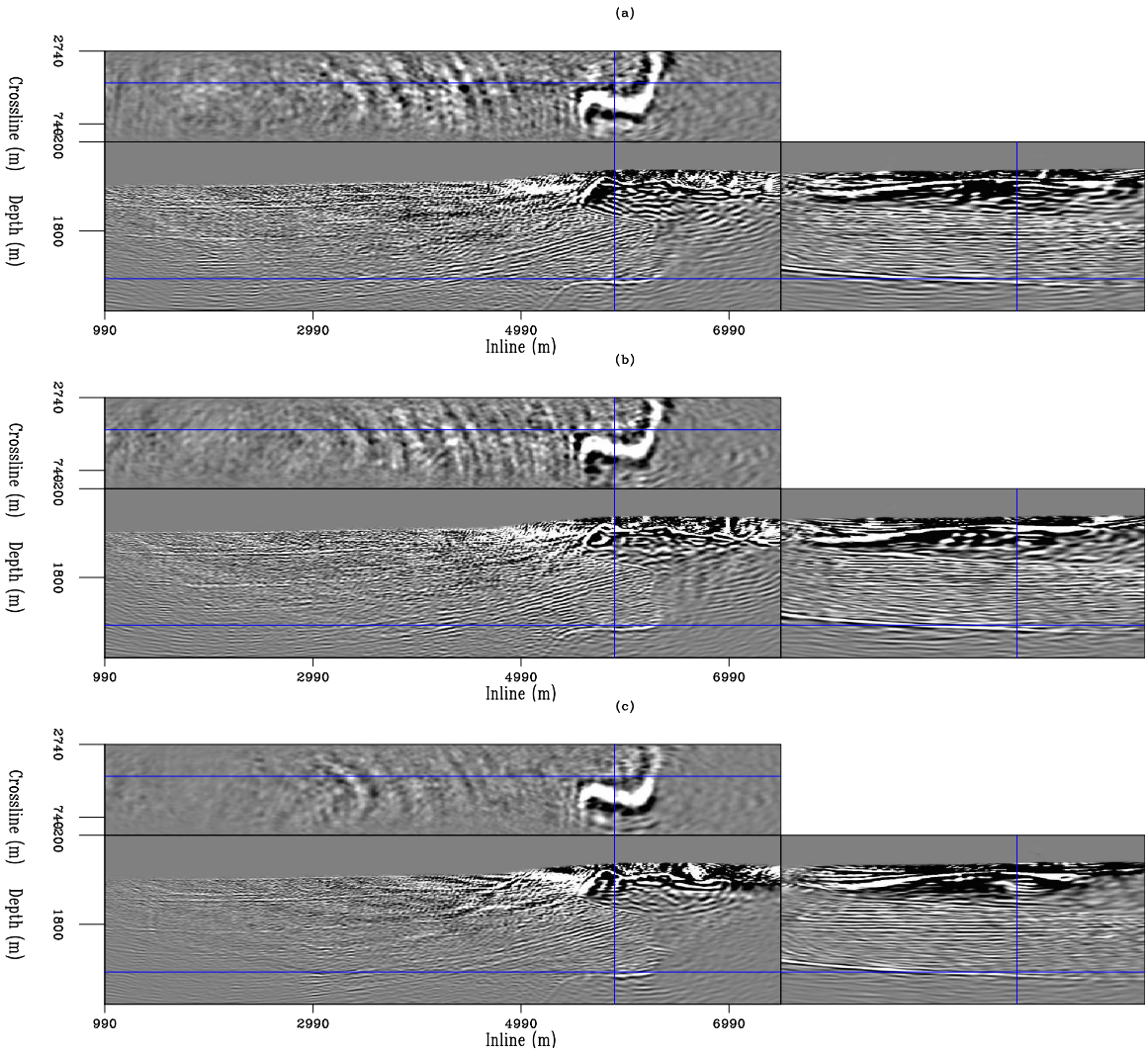

Figure 6: Images obtained after one iteration using 75%, 50% and 25% of the data, respectively. [**CR**]

Figure 6 shows the same SEAM section after a single iteration for each subset. Predictably, using more data provides a better image for a single iteration. This is particularly noticeable in the depth slice. The same section after ten and 50 iterations are shown in Figure 7 and Figure 8. At high iteration counts, the difference between

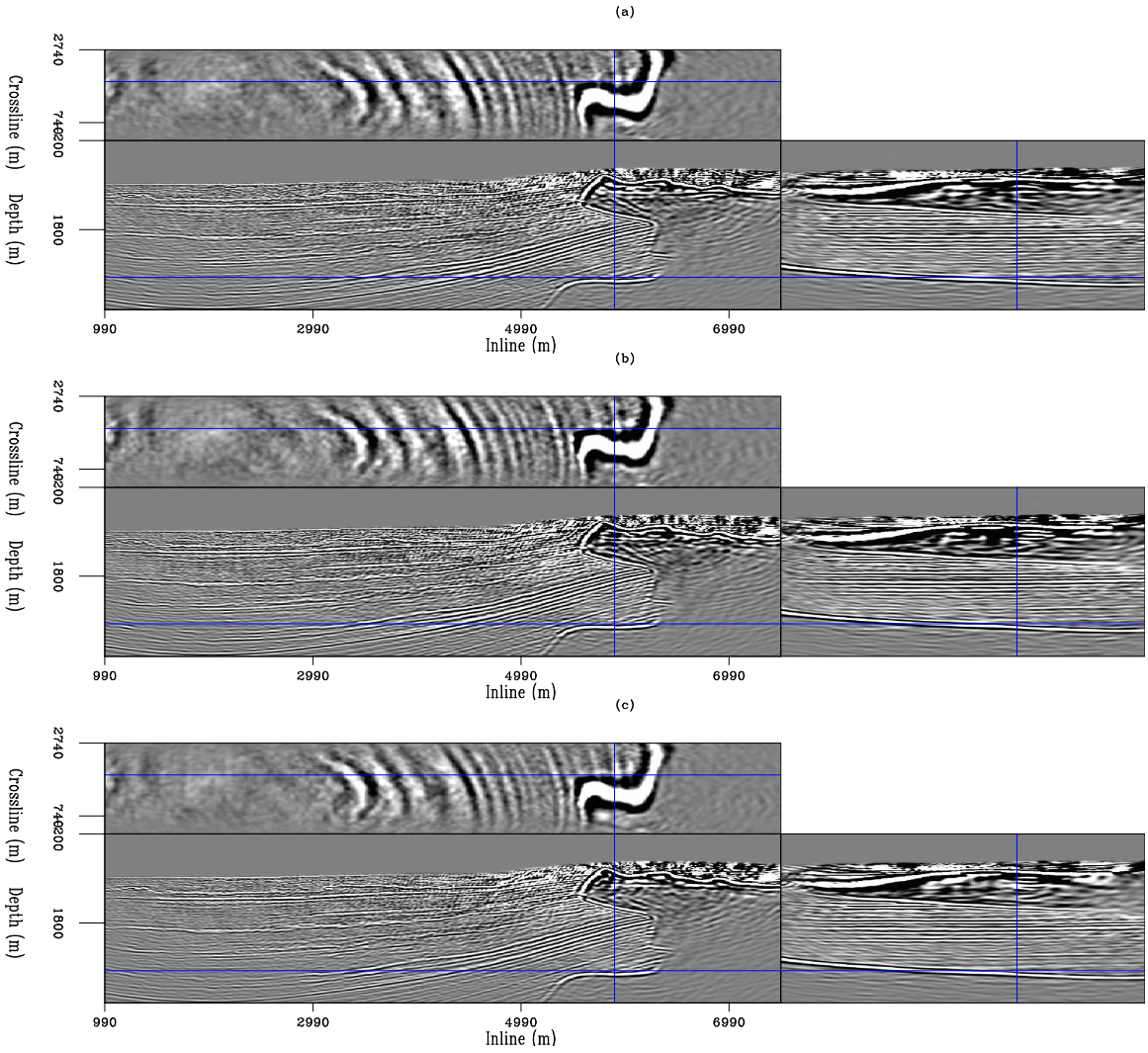

Figure 7: Images obtained after ten iterations using 75%, 50% and 25% of the data, respectively. [**CR**]

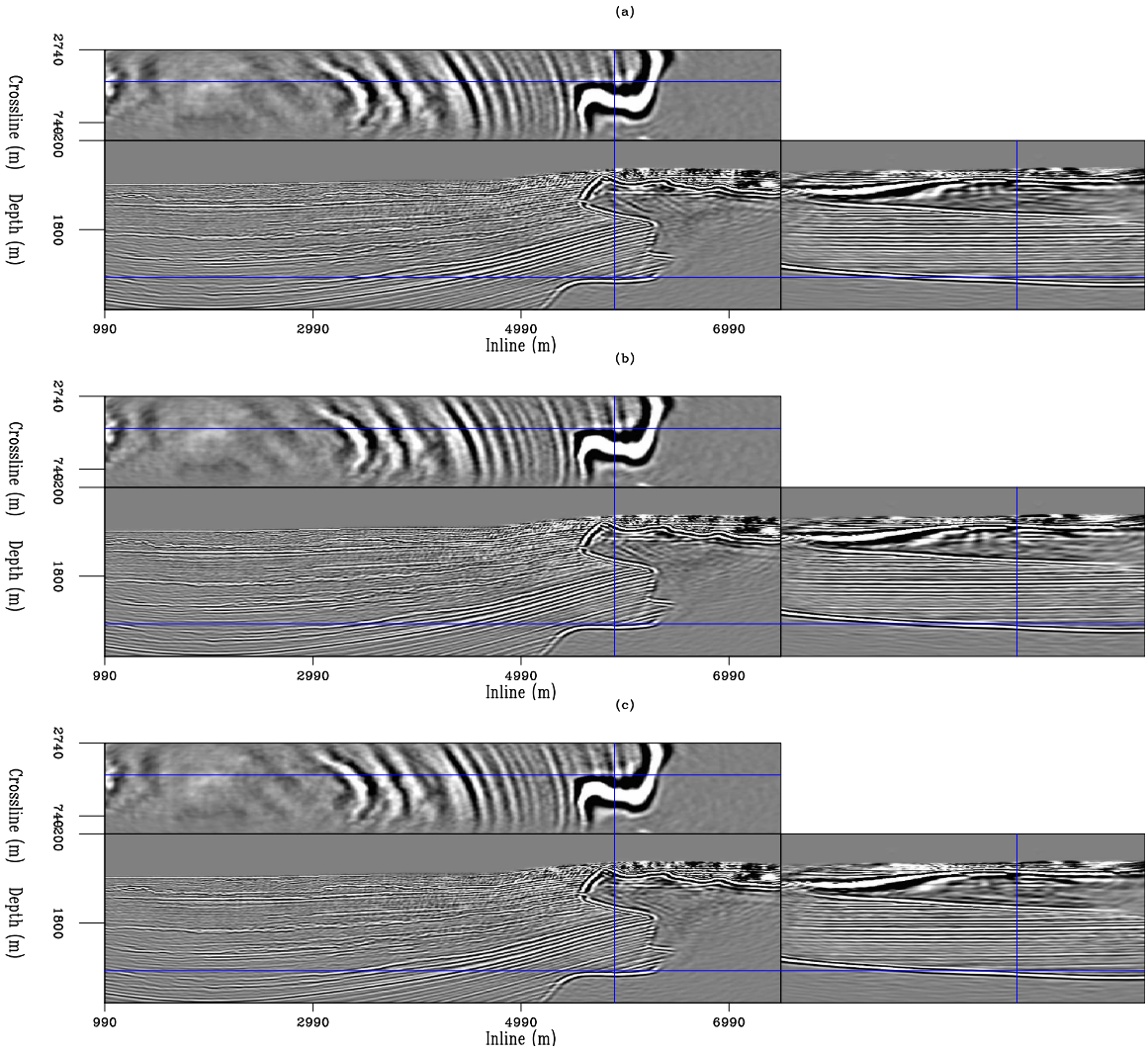

Figure 8: Images obtained after 50 iterations using 75%, 50% and 25% of the data, respectively. [**CR**]

these images becomes marginal, despite the widely favouring amount of data used per iteration. This shows that there will likely be a cost reduction for a given level of convergence.

Figure 9 shows some of these convergence curves, as a function of iteration number and quantity of data used. Similarly to phase-encoding, the 'observed' data change between iterations, creating a non-linear problem. This means that convergence properties are not smooth, and the residual can locally increase. It is noticeable that the trend for all subsets is a residual reduction. Also, if less data is used, then the less predictable the residual properties. This is because, at early iterations, sections of the model may not be well resolved, since only 25% of these data are being used. This unbalanced, noisy image, is then used for forward simulation.

It should also be noted that there is a concern with fair comparison. The raw initial residuals for these four sets of simulations were wildly different, and when plotted on the same scale, can be hard to interpret. For the sake of demonstration, these have all been scaled to percentages of data misfit. However, it should be noted that a given level of misfit for these varying data sizes are not directly comparable.

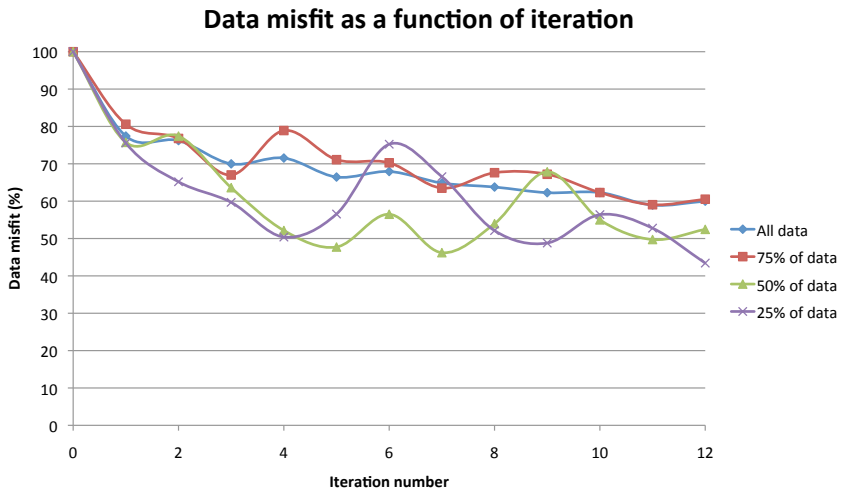

Figure 9: Plot of how the data misfit varies as a function of iteration number. [**NR**]

Residual reduction as a function of relative cost can then be seen in Figure 10. It is clear, once again, that this stochastic dimensionality reduction has induced some local non-linearity, since microscopically the residual can often increase.

Making firm conclusions is difficult, since the problem is so dependent on source sampling. A dense source carpet would lend itself more easily to encouraging to a randomized sampling methodology than a sparse survey. The input dataset was relatively sparse, especially in the crossline, compared to many modern datasets. Since several iterations are being performed then in this case using 50% of the data gives the most favorable, and trustworthy, convergence. This ratio provided the right balance between source-sampling and inversion cost.

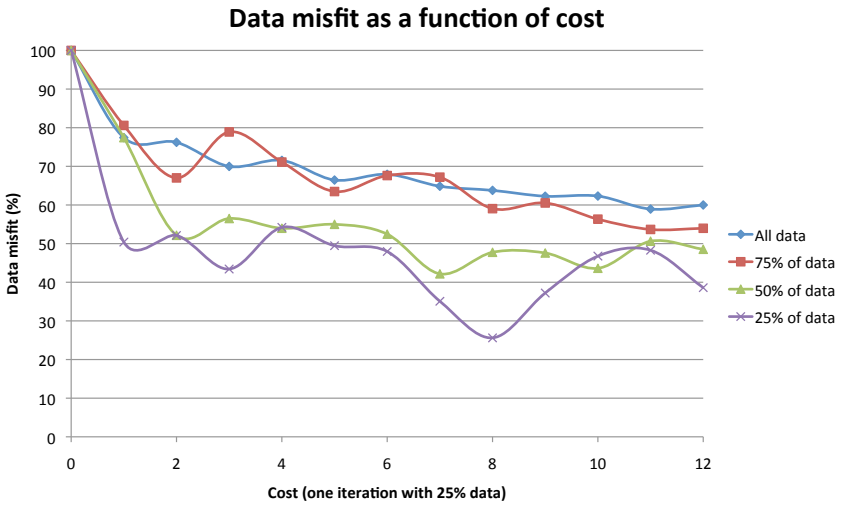

Figure 10: The same plot as in Figure 4, but as a function of cost. [**NR**]

A more pronounced reduction would be seen for a denser survey, so these results show that even for a fairly sparse survey randomized sampling can be a powerful method to improve inversion feasibility.

PHASE ENCODING AND RANDOMIZED SAMPLING

When multiple super-shots are used for phase encoding, these two techniques can be combined. There will now be two loops of encoding, the inner loop being the phase encoding, and the outer loop being the randomized sampling.

Combining phase encoding with randomized sampling may have advantages since artifacts from phase encoding are incoherent between inner loop iterations, and the data reduction from the outer loop should save additional computation time.

CONCLUSIONS

Conventional linearized inversion is a very powerful method, however for large datasets, computational requirements can become unfeasible. By trying to reduce data movement, and the work done within each iteration, this feasibility can be improved.

Phase encoding addresses this by reducing the data-space dimensionality, without losing any information contained within these data. Shots are combined, imaged, and modeled concurrently. Crosstalk artifacts slow convergence as a function of iteration number, but by changing the encoding sequence between iterations, excellent convergence as a function of cost is observed relative to LSRTM (for a fixed receiver geometry.)

Optionally, a rolling subset of these data can be ignored within each iteration (randomized sampling.) This is quite different to the concept of phase encoding - some data is ignored, but no crosstalk artifacts are reduced. For dense source sampling an improvement in convergence as a function of cost is seen.

It is possible to augment these techniques. For certain survey geometries, this combination gives the most favourable data-space convergence, as a function of overall cost.