Chapter 6

Problems with many right hand sides

In tis chapter we discuss a particular set of difficulties that arise when the constraint has the form

$$c(m, U) = A(m)U - B = 0$$

where $U = [u_1, \ldots, u_{N_s}]$ are the solutions for a discretization of a linear PDE with multiple right hand sides.

Multiple right hand sides represent an abandon of sources. Using many sources is a common way to obtain more data and hence, enhance the resolving power of many imaging techniques.

Assuming the constraints are eliminated we can rewrite the optimization problem as

$$\min_m \ \mathcal{J}(m) = \frac{1}{2}\|QA(m)^{-1}B - D\|_F^2 + \alpha R(m) \tag{6.1}$$

Here $D = [d_1, \ldots, d_{N_s}]$ is the data obtained from different sources and $B = [b_1, \ldots, b_{N_s}]$ is a discretization of the sources. We have assumed here that all sources share the same receivers, that is, the observation matrix Q is identical for all sources.

6.1 Difficulties for problems with many right hand sides

Using the standard solution techniques we can compute the gradient that reads

$$\nabla\mathcal{J}(m) = \sum_j J_i(m)^\top (QA(m)^{-1}b_i - d_i) + \alpha\nabla_m R(m) \tag{6.2}$$

where $J_i = \nabla_m(QA(m)^{-1}b_i)$ is the sensitivity matrix for each source and can be written as

$$J_i(m) = -QA(m)^{-1}G(m, u_i)$$

with

$$G(m, u_i) = \nabla_m(A(m)u_i).$$

The Gauss-Newton step reads

$$\left(\sum_i J_i(m)^\top J_i(m) + \alpha \nabla_m^2 R\right) s = -\nabla \mathcal{J}(m) \tag{6.3}$$

Inspecting the above equations one can detect the main difficulties. First, for every step we need to solve N_s forward problems and N_s adjoint problems. Second, if we desire to use the Gauss-Newton then we require to store the fields $u_i, i = 1, \ldots, N_s$. This can be cumbersome even for small model problems! We now discuss a number of approaches to deal with such problems.

6.2 Brute-Force solutions

Maybe the most simple algorithmically (but certainly the least efficient computationally) is the brute force approach. Here we simply compute the function, the gradient, and, if computer memory is not an issue, we also store the fields and use the Gauss-Newton iteration.

This approach can be truly expensive. For each step of the Gauss-Newton method we require to solve $2N_s(n_{CG} + 1)$ where n_{CG} linear systems, where n_{CG} is the number of conjugate gradient iterations needed to solve the Gauss-Newton system. Assume for example, a moderate number of sources, say, 1000 and a small number of CG steps, say 10, then, each Gauss-Newton iteration requires the solution of $22,000$ linear systems! This can be a daunting calculation. At this point using a direct solver and storing the matrix decomposition is highly recommended. This implies that for large scale problems one cannot hope to solve the inverse problem using a straight forward method unless an adequate computational platform is available.

It is possible to work with much more modest computational hardware if first order methods are used. In this case, the gradient can be computed without storing all the fields using the following approach to compute the forward problem and the gradients

- Set $\mathsf{misfit} = 0 \quad \nabla\mathsf{misfit} = 0$
- For $j = 1, \ldots, N_s$
  - Solve $A(m)u_j = b_j$ (the forward problem)
  - $r_j = Qu_j - d_j$
  - $\mathsf{misfit} \leftarrow \mathsf{misfit} + r_j^\top r_j$
  - Solve $A^\top \lambda_j = Q^\top r_j$ (the adjoint problem)
  - $\nabla\mathsf{misfit} \leftarrow \nabla\mathsf{misfit} - G_j^\top \lambda_j$

Note that we write over each field and Lagrange multiplier and therefore do not require the storage of all the fields. This enables the use of steepest descent and the LBFGS methods even for a low memory architecture.

6.3 Subset iteration methods

The previous approach required the solution of at least $2N_s$ linear systems at every iterations in order to obtain the gradient and more iterations to obtain the Gauss-Newton step. An approach to reduce the computation is the subset method. The idea is rather simple. Rather than working with the whole data set divide the data into ℓ groups, where each group contains N_s/ℓ sources. At each iteration, we work with the ℓ's subset to perform a single Gauss-Newton iteration. We then replace the the subset and continue to cycle. This is a nonlinear Kaczmaraz iteration [18] that is commonly used is ray tomography The idea was used by [10] for electromagnetic inversion. While it can be proved that the approach works for linear problems convergence for nonlinear problems, and particularly to non-convex problems is not guarantied. However, even without convergence proof, the method seem to perform well for a large range of problems.

Clearly, the method depends on the particular choice of sources and the division to subsets. While theoretically all possible divisions have similar convergence properties, it is observed that on practice, convergence is highly influenced by the subset selection. While exact analysis does not exist, choosing subsets such that each subset is mostly independent seem to do better than subsets that are dependent. One way to achieve such subspaces is to choose indices in random and combine random sources into groups.

6.4 Stochastic programming approach

In the above approach we used a subspace method for the solution of the problem. In this section we generalize the approach and show that it is possible to obtain a solution technique that requires considering only a single! PDE at each iteration.

To do that we first define the residual

$$S(m) = QA(m)^{-1}B - D$$

and use a statistical identity and rewrite the misfit as

$$\begin{aligned} \mathsf{misfit} &= \frac{1}{2}\|QA(m)^{-1}B - D\|_F^2 = \frac{1}{2}\mathsf{trace}(S(m)^\top S(m)) \\ &= \frac{1}{2}\mathbf{E}_w\, w^\top S(m)^\top S(m) w \end{aligned} \tag{6.4}$$

where $\mathbf{E}$ is an expected value on a random vector w that has the following properties

- $\mathbf{E}(w) = 0$, that is w has a 0 mean
- $\mathsf{Cov}(w) = I$

We can thus replace the original optimization problem with the stochastic programming problem

$$\min_m\ \mathcal{J}(m) = \frac{1}{2}\mathbf{E}_w\ w^\top (QA(m)^{-1}B - D)^\top (QA(m)^{-1}B - D)w + \alpha R(m) \tag{6.5}$$

Stochastic programming problems are problems with the form

$$\min_m \mathbf{E}_w f(m, w).$$

Such problems are often solved in operation research as well as machine learning and has received both theoretical and algorithmic treatment, se for example, [28, 5, 22, 17]. Some of the solution techniques have one remarkable feature, that is, they can work with a very small sample size, w and this has a dramatic effect on our problem. We now review two such methods.

6.4.1 Stochastic Average Approximation

Stochastic Average Approximation (SAA) uses Monte-Carlo in order to discretize the expectation. The idea is to choose ℓ samples $w_1, \ldots, w_\ell$ and solve the average approximation problem

$$\min_m \ \widehat{\mathcal{J}}(m) = \frac{1}{2\ell} \sum_{j=1}^{\ell} \ w_j^\top (QA(m)^{-1}B - D)^\top (QA(m)^{-1}B - D) w_j + \alpha R(m)$$

If we choose w_j as the unit basis function then we arrive to the original problem. However, other, better, choices can be made. In particular, it is possible to choose that a random w chosen with random entries of ± 1 is the optimal choice [15]. Now, consider the case of computing the misfit for a single sample. We have that

$$(QA(m)^{-1}B - D)w_j = QA(m)^{-1}Bw_j - Dw_j = QA(m)^{-1}(Bw_j) - (Dw_j)$$

Define, $b_{w_j} = Bw_j$, we see that we can compute the above quantity using a single! matrix inversion. Now, if the number of samples is close to the original number of sources then no gain is made by the method. However, if the number of samples is small then the number of systems needed at each iteration is reduced dramatically. We have found that in practice a moderate number of w's is needed. We have successfully used 5-20 samples for problems with more than 1000 sources. The open question that require special attention is, how to choose the number of samples.

It is important to note that SAA is not an algorithm. After discretization, any optimization method can be used in order to solve the deterministic optimization problem. This opens the door to methods such as Gauss-Newton and full space approaches even for problems with multiple sources.

6.4.2 Stochastic Approximation

A second method to solve the problem is by using stochastic approximation (SA). In this method we start at some point m_1 and at each iteration use the steepest descent method but with a different sample. The iteration contains two steps:

- $\widehat{m}_{k+1} = m_k - \mu \nabla \mathcal{J}(m_k, w_k)$
- $m_{k+1} = \frac{1}{k+1} \left(\sum_{1=1}^{k} m_k + \widehat{m}_{k+1} \right)$

Here the parameter μ is fixed and should be chosen judicially (and this may not be easy to do in practice).

The amazing property of this algorithm, applied to our problem is that it requires a single right hand side at each iteration! This is rather remarkable and surprisingly works well in practice [21].

Although no proof of convergence exists, we have used a similar algorithm but with a Gauss-Newton step rather than a steepest descent step. While convergence with high accuracy is difficult if not impossible, it seems that convergence to low accuracy solutions can be quickly obtained.