Geophysical Prospecting

EAGE EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS

Geophysical Prospecting, 2015, **63**, 495–507 doi: 10.1111/1365-2478.12200

A parallel, scalable and memory efficient inversion code for very large-scale airborne electromagnetics surveys

Casper Kirkegaard* and Esben Auken

Department of Geoscience, Aarhus University, C.F. Møllers Allé 4, DK-8000 Aarhus C, Denmark

Received February 2014, revision accepted May 2014

ABSTRACT

Over the past decade the typical size of airborne electromagnetic data sets has been growing rapidly, along with an emerging need for highly accurate modelling. One-dimensional approximate inversions or data transform techniques have previously been employed for very large-scale studies of quasi-layered settings but these techniques fail to provide the consistent accuracy needed by many modern applications such as aquifer and geological mapping, uranium exploration, oil sands and integrated modelling. In these cases the use of more time-consuming 1D forward and inverse modelling provide the only acceptable solution that is also computationally feasible.

When target structures are known to be quasi layered and spatially coherent it is beneficial to incorporate this assumption directly into the inversion. This implies inverting multiple soundings at a time in larger constrained problems, which allows for resolving geological layers that are undetectable using simple independent inversions. Ideally, entire surveys should be inverted at a time in huge constrained problems but poor scaling properties of the underlying algorithms typically make this challenging.

Here, we document how we optimized an inversion code for very large-scale constrained airborne electromagnetic problems. Most importantly, we describe how we solve linear systems using an iterative method that scales linearly with the size of the data set in terms of both solution time and memory consumption. We also describe how we parallelized the core region of the code, in order to obtain almost ideal strong parallel scaling on current 4-socket shared memory computers. We further show how model parameter uncertainty estimates can be efficiently obtained in linear time and we demonstrate the capabilities of the full implementation by inverting a 3327 line km SkyTEM survey overnight. Performance and scaling properties are discussed based on the timings of the field example and we describe the criteria that must be fulfilled in order to adapt our methodology for similar type problems.

Key words: Modelling, Inversion code.

INTRODUCTION

In this paper we describe how we significantly optimized our versatile and flexible inversion code AarhusInv (Auken *et al.* 2014). The code is mostly based on 1D forward formulations and provides support for inversion of a large array of data types, e.g., DC/IP, MRS and time-/frequency domain electromagnetics. Here, we describe how the code has undergone significant optimization to accommodate the challenges of working with very large data sets collected by airborne time- or frequency domain electromagnetic instruments.

*Corresponding author: casper.kirkegaard@geo.au.dk

Constrained inversion is a proven regularization concept in the field of airborne electromagnetics (AEM), which utilizes the assumption that earth structures are spatially coherent. The approach allows for improved resolution of geological features that can be poorly resolved by individual soundings but revealed when coherence constraints are applied to an inversion of a larger group of soundings (Auken *et al*. 2008). Obviously, the methodology works best if there are no practical limits to the size of the data set that can be inverted for. Under this condition, information can migrate freely from model to model and over any distance spanned by the data set, which also proves a benefit when utilizing prior information. In fact, the ability to invert very large problems at a time can even be regarded a prerequisite for many coupled/joint type inversion problems. Often, data density and spatial distribution vary significantly between data types, requiring the inversion of large problems in order to simultaneously cover the different methods vastly different footprints. Ideally the computation time and memory consumption of an inversion code made for such large-scale problems should scale linearly with the size of the data set, however, this is difficult to obtain in practice.

During the last decade the typical size of AEM data sets has been growing at a very rapid rate and surveys of today are often in the order of tens of thousands of line kilometres (SkyTEM, personal communication; Geotech, personal communication; Costelloe *et al*. 2007; Kalvig 2008; Lawrie *et al*. 2010; Podgorski *et al*. 2010). These expensive data sets should ideally be inverted using highly accurate forward modelling (Christiansen, Auken and Viezzoli 2011) but this can represent a true computational challenge. Recently, 3D inversion of airborne EM data has become an important topic in the literature (Cox, Wilson and Zhdanov 2010; Yang and Oldenburg 2012), however, the computational requirements are still much too great for routine application to the very large-scale problems considered here. Examples of 2D inversion of AEM data sets can also be found in the literature, utilizing either approximate forward solutions (Wolfgram, Sattel and Christensen 2003; Guillemoteau, Sailhac and Behaegel 2012) or full solutions (Wilson, Raiche and Sugeng 2006). These approaches either approximate the 3D transmitter geometry in 2D or pay a significant computational penalty to perform the modelling in 2.5D (2D model with 3D transmitter). Recently, a promising low cost numerical strategy for full 2.5D modelling was published with application to CSEM systems (Streich, Becken and Ritter 2011). This type of approach could potentially provide the right balance of model accuracy and computational cost for inverting very large AEM surveys. However, we are unaware of any AEM implementations or applications in the literature. As such, large-scale airborne TEM data sets pose a challenge to an industry where code development is mostly driven by public research and limited by tight funding. Many innovative inversion codes are being developed but focus is typically on implementing novel methodologies rather than to reach mature and stable large-scale production quality. Since optimizing for scalability can be a lengthy process in itself, the inherent data volume issue is often solved by considering each sounding independently. This includes relatively simple data transform techniques (Macnae *et al*. 1998; Reid and Fullagar 1998; Sattel 2005), inversions utilizing approximate forward models (Farquharson, Oldenburg and Li 1999; Christensen 2002; Christensen, Reid and Halkjaer 2009) and also full forward models (Chen and Raiche 1998; Farquharson, Oldenburg and Routh 2003). Constrained codes capable of utilizing the spatial coherence of a layered earth also exist (Santos 2004; Tartaras and Beamish 2005; Vallée and Smith 2009) but scalability issues severely limit the size of the problem that can be inverted at a time.

To our knowledge, the only code capable of handling problems on a scale that even start to resemble modern full size AEM surveys is the holistic inversion by Brodie and Sambridge (2006). This code utilizes sparse data structures and distributed parallel algorithms to invert several thousand line kilometres of frequency domain data at a time on a 64 node cluster (Brodie 2010). These authors further showed how inverting very large data sets at a time allows for the inversion of additional model parameters such as system drift, parameters that can only be resolved if a full survey is inverted as one large problem. Despite being able to handle very large problems their code still faces a fundamental scaling issue, as their parallel linear solver implementation does not scale linearly with the size of the data set. The lack of linear scaling slows down calculations for large problems, puts an upper limit on the size of the problem that can effectively be solved and makes for an iterative algorithm that becomes more prone to numerical instability as the problem size increases. We address these shortcomings by utilizing more specialized linear algebra algorithms, in order to obtain linear scaling in terms of both computation time and memory consumption. We also chose to parallelize our code using OpenMP for stand-alone shared memory computers, rather than the distributed MPI approach of Brodie and Sambridge (2006). This is done to avoid unnecessary complexity, with an end result capable of

inverting even very large airborne time domain surveys for constrained quasi-3D models on commodity computer hardware overnight.

METHODOLOGY

Data set geometry and spatial constraints

One of the most characteristic features of the AarhusInv approach to inversion is the flexible use of regularizing coherence constraints to couple local 1D models in a larger problem. This is a strategy very well suited for modelling of settings with a laterally coherent geology and can be implemented in numerous ways. In our case the constraints are typically given in the form of lateral constraints (LCI, Auken *et al*. 2005) or spatial constraints (SCI, Viezzoli *et al*. 2008). Here, we focus on the SCI scheme since LCI is just a special case of this more general approach. The fundamental concept of SCI is illustrated in Figure 1, where a survey of profile oriented data is Delaunay triangulated to obtain connections from each sounding/model datum to its nearest neighbours. Spatial smoothness constraints are then applied along these connections to the models of the nearest neighbours. If the size of the entire data set exceeds the practical capability of the inversion code a survey is divided into smaller areas called zones. These zones are of a size that can be efficiently inverted independently and in parallel. An example of such a division is seen in the bottom of Figure 1, where the individual zones are marked by different background colours. Having divided a problem into a number of zones, all zones can be inverted independently followed by a second inversion run for continuity across zone borders. This methodology works well but introduces intrinsic inefficiency. Models lying on zone boundaries are included and inverted in multiple zones and a complete second run of inversions of all zones is needed for continuity. For full details of the former implementation we refer to Viezzoli *et al*. (2008). The most important thing to note for later is how coherence constraints are obtained from triangulation and how the problem has the same characteristic geometry regardless of size. We consider surveys of densely sampled soundings collected along almost parallel lines, interconnected in an almost regular pattern and we want to be able to handle this type of geometry on a very large scale.

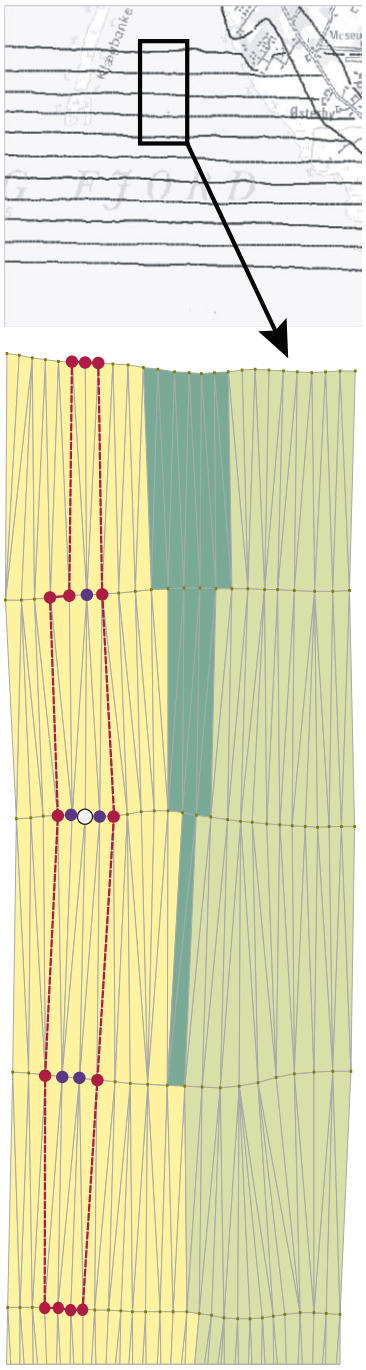

Figure 1 Characteristic SCI geometry. In the top of the figure the flight lines and 1D model positions of a sample airborne survey are shown (line spacing is 300 m). The bottom of the figure illustrates spatial constraints between models and how an SCI inversion is partitioned into smaller zones. The connecting lines indicate spatial smoothness constraints between neighbouring models and each independent zone is marked by a separate background colour. Models marked by large bullets indicate the models used for an approximate local analysis for the point marked in white.

Mathematical solution

From a mathematical point of view our methodology follows the established practice presented by Menke (1989) to solve

a non-linear inverse problem in a linearized iterative manner. The approach is briefly outlined in the following and we refer to Viezzoli *et al.* (2008) and Auken *et al.* (2005) for a detailed description of the SCI/LCI inversion methodology.

We want to minimize the misfit between observed data $\mathbf{d}_{obs}$, having associated errors given by $\mathbf{e}_{obs}$ and the forward response function $\mathbf{g}$. In order to solve this problem we employ a first-order approximation for the non-linear function $\mathbf{g}$ mapping from model space vectors $\mathbf{m}$ into data space:

$$\mathbf{d}_{obs} + \mathbf{e}_{obs} \cong \mathbf{G}(\mathbf{m}_{true} - \mathbf{m}_{ref}) + \mathbf{g}(\mathbf{m}_{ref}). \quad (1)$$

Here, $\mathbf{m}_{true}$ is the true model vector, $\mathbf{m}_{ref}$ is some reference vector and $\mathbf{G}$ is the Jacobian matrix. The equation can be further rewritten in terms of successive iterative model updates $\delta\mathbf{m}_{true}$:

$$\mathbf{G}\delta\mathbf{m}_{true} = \delta\mathbf{d}_{obs} + \mathbf{e}_{obs}. \quad (2)$$

The full-inversion scheme obviously also includes the regularizing SCI constraints and support for *a priori* information, which is easily included by adding more equations of the type seen in equation (2) to the linear system. This full system of equations can be solved in a least squares sense by minimizing the L_2 misfit using the iterative Gauss-Newton minimization scheme with a Marquardt modification. Using this approach we obtain a system of linear equations ($\mathbf{Ax} = \mathbf{b}$) to solve for iterative model updates:

$$\begin{aligned}&(\mathbf{G}^T\mathbf{C}_{obs}^{-1}\mathbf{G}+\mathbf{R}^T\mathbf{C}_c^{-1}\mathbf{R}+\mathbf{C}_{prior}^{-1}+\lambda\mathbf{I})\delta\mathbf{m}=\\ &\mathbf{G}^T\mathbf{C}_{obs}^{-1}\left(\mathbf{d}_{obs}-\mathrm{g}(\mathbf{m}_n)\right)+\mathbf{R}^T\mathbf{C}_c^{-1}\left(-\mathbf{R}\mathbf{m}_n\right)+\mathbf{C}_{prior}^{-1}\left(\mathbf{m}_{prior}-\mathbf{m}_n\right).\end{aligned} \quad (3)$$

Here, $\mathbf{m}_n$ is the model vector for the n-th iterative step, $\delta\mathbf{m}$ the model update for the next iteration, $\mathbf{m}_{prior}$ a vector holding the prior model and $\mathbf{C}_{obs}$, $\mathbf{C}_{prior}$ and $\mathbf{C}_c$ are diagonal covariance matrices describing the uncertainty on the observed data, prior model and constraints, respectively. The matrix $\mathbf{R}$ is a roughness matrix specifying the geometry of the constraints and λ is a Marquart damping parameter (Marquart 1963). During each iteration of the inversion a line search is performed, solving the system for different values of λ until a model update of suitable magnitude is found. To conclude the mathematical details we note that the linearized covariance matrix $\mathbf{C}_{est}$, providing an estimate of the uncertainty of the model result, can be calculated from the following expression (Tarantola and Valette 1982):

$$\mathbf{C}_{est} = (\mathbf{G}^T\mathbf{C}_{obs}^{-1}\mathbf{G} + \mathbf{R}^T\mathbf{C}_c^{-1}\mathbf{R} + \mathbf{C}_{prior}^{-1})^{-1}. \quad (4)$$

One can conveniently view the standard deviation derived from this formula as a relative measure of uncertainty, since the inversion is performed in logarithmic model space. This implies that absolute analysis values from logarithmic space translate into a standard deviation factor in linear space, such that 1.0 is equal to perfect resolution and 1.1 is equal to a standard deviation of approximately 10%. Since these values are further obtained from error estimates in a linear approximation they should be regarded as guidelines.

Table 1 Pseudo code for the inversion scheme. In the right-hand side of the table is the time complexity of the main steps prior to optimization. N is the number of models/data sets.

1. Read input and initialize data structures	
2. for $n = 0,1,\ldots,$ until Convergence do	
3. Calculate forward response $\mathrm{g}(\mathbf{m}_n)$	$O(N)$
4. Calculate derivatives for Jacobian $\mathbf{G}$	$O(N)$
5. Solve for model update $\delta\mathbf{m}$	$O(N^2)$
6. Set $\mathbf{m}_{n+1} = \mathbf{m}_n+\delta\mathbf{m}$	
7. Test convergence criteria	
8. end do	
9. Solve for model analysis	$O(N^3)$
10. Write output	

Solution algorithm

Given the mathematical formulation and problem geometry we can now write the pseudo code for the process to optimize, as seen in Table 1. From the lines in this table, practically all the time is spent in lines 3, 4, 5 and 9. Lines 3 and 4 are forward- and forward-based derivative calculations that pose an embarrassingly parallel problem in a 1D formulation. In line 5 the linear system of equation (3) is successively solved for the next iterative model update $\delta\mathbf{m}$. Here, linear algebra operations are needed to form the left-hand matrix $\mathbf{A}$ and right-hand vector $\mathbf{b}$ but the most crucial step is the actual solution of the linear system of equations. Standard dense solver algorithms, such as the ones we were replacing, are slow with complexity $O(N^2)$ (Press *et al.* 2007), so in order to make the code scalable we employ sparse matrix data structures and efficient sparse solvers. The final critical step in line 9 of the inversion algorithm is the calculation of a model parameter analysis by the expression given in equation (4). Note how this equation involves taking the full inverse of a matrix that is of $O(N^3)$ time complexity using dense methods (Press *et al.* 2007). In order to make this step scalable serious algorithmic changes are needed, which will be obtained by an approximation in the formulation. In the following we describe the methodology behind each optimization in detail.

OPTIMIZATIONS

Parallelization

Several different frameworks for parallelizing scientific codes are available, the most popular being OpenMP (Chapman, Gabriele and van der Pas 2007) and MPI (Gropp, Lusk and Skjellum 1999). MPI is utilized in the implementation of Brodie and Sambridge (2006) for distributing data and computations over a cluster of multiple computers, whereas OpenMP is for parallel computation on a single multi CPU computer with a common local memory space for all threads. The ability to distribute a workload over a whole cluster of compute nodes is a definite advantage of MPI, however, it also requires the code to be written with distributed data structures in mind. This is not easily obtained for a large existing sequential codebase, so an actual need for more cores than provided by a single machine should be present in order to justify a significant rewrite. Given the amount of parallelism found in modern inexpensive computer hardware we argue and demonstrate in the results, that even in the case of very large data sets distributed computing is no longer needed for 1D AEM inversion. We thus chose our target platform as shared memory machines of large core count and OpenMP parallelized the loops iterating over forward-/derivative calculations of each 1D model.

Sparse matrices and solvers

Conventional methods for operating on general matrices in a 2D array structure provide poor scaling, having a memory and time complexity of $O(N^2)$ for the example of solving a linear system. This problem can be overcome, however, when the matrices involved consist mostly of 0 (zero) entries, i.e., the matrices are sparse. Several popular large libraries providing this type of functionality are available (PetSc, Balay *et al.* 1997; TAUCS, Toledo, Chen and Rothkin 2001), often providing MPI parallelized routines. For the optimization presented here we implemented a custom Fortran 90 sparse linear algebra library, using core routines from SPARSKIT (Saad 1990). In the following we outline the concepts behind the sparse iterative solver scheme and note that the time required for all other sparse linear algebra operations is safely negligible.

Choosing an optimal method for solving large sparse linear systems of the type $\mathbf{Ax} = \mathbf{b}$ can be challenging. A wealth of different algorithms is available and often there are options available that exploit specific properties of the system but with no guarantee that it will be the optimal solution. Even though matrices can be categorized in subclasses, these classes are very broad and the performance and stability of even a specialized solver will depend heavily on the actual sparsity pattern of the matrix. In fact, it has been shown that no single solver algorithm is consistently the best (Nachtigal, Reddy and Trefethen 1992; Ern *et al.* 1994). When choosing a sparse solver the standard approach is to look for performance results for comparable matrices and perform an actual test of the most promising algorithms. Review papers do exist (Gould, Scott and Hu 2007), however, these are basically just benchmarks of different solvers tested on a wide range of different matrices. Recently, suggestions for accurate automated solver recommendation systems were proposed by Bhowmick, Toth and Raghavan (2009) and George, Gupta and Sarin (2008) but for the time being the decision process is still manual.

Sparse linear solvers can be divided into two distinct categories: direct and iterative. Direct solvers use factorization and back-substitution to solve the system, whereas iterative solvers rely on an iterative improvement of an initial starting guess until convergence to the solution is reached. Modern parallel direct sparse solvers such as Pardiso (Schenk and Gärtner 2004), SuperLU (Li and Demmel 2003) and MUMPS (Amestoy *et al.* 2002) are sophisticated research projects on their own and provide robust 'black box' solvers, where little to no customization is necessary. Their robustness and ease of use comes at the price of storing a full factorization of the original matrix, resulting in unpredictable memory consumption that can easily exceed 10–20 times the memory requirements of the matrix **A** (Rücker, Günther and Spitzer 2006). Iterative solvers, on the other hand, are often simple algorithms that need to be customized for a specific problem in terms of preconditioning with the reward of low and potentially predictable memory consumption. Such an algorithm was chosen for the optimization of AarhusInv, since this is the option that is able to handle the largest possible problems.

The iterative conjugate gradient algorithm (CG, Hestenes and Stiefel 1952) is widely used within the geophysical community (Wu 2003; Brodie and Sambridge 2006; Rücker *et al.* 2006; Mueller-Petke and Yaramanci 2010), however, we employ the preconditioned bi-conjugate gradient stabilized algorithm (BICGSTAB, Van der Vorst 1992). Theoretically, CG should be the most suitable algorithm as our linear system is symmetric positive definite but we chose BICGSTAB due to its empirically proven stability in solving our particular problem. Convergence cannot always be guaranteed for iterative solvers, so algorithm stability was an important parameter in the decision process. Apart from CG and BICGSTAB, we included several other popular iterative solver algorithms in

our tests, including the bi-conjugate gradient (BICG, Fletcher 1976) and the generalized minimum residual (GMRES, Saad and Schultz 1986).

In configuring a robust iterative solver scheme, proper preconditioning is of uttermost importance (Saad and Van der Vorst 2000). The convergence rate of conjugate gradient type methods depends on the spectral condition number of the matrix, i.e., the ratio between the maximum and minimum eigenvalue. Preconditioning is essentially the process of finding a linear transformation $\mathbf{M}$ that approximates $\mathbf{A}^{-1}$, such that $\mathbf{MA} \approx \mathbf{I}$ is fulfilled to a reasonable degree. Solving the transformed system where $\mathbf{M}$ is multiplied on both sides of the equality $\mathbf{Ax} = \mathbf{b}$ greatly improves the stability and convergence rate, since the left-hand side matrix is now close to being the identity matrix for which all eigenvalues are unity. One way of applying preconditioning to the BICGSTAB algorithm, is by approximating a LU factorization of the $\mathbf{A}$ matrix with the benefit that an approximation to the direct inverse does not need to be computed (Van der Vorst 1992). We apply this preconditioning methodology using an incomplete LU factorization with a dual dropping strategy (ILUT, Saad 1994). This is essentially a standard sparse LU factorization where small non-zero elements are set to zero based on dropping criteria. In a given row i, the maximum of the $M_{\max}$ largest elements are kept under the additional criteria that all elements smaller than εA_{ii} are also dropped. Using these dropping rules make the maximum memory consumption of the solver controllable by $M_{\max}$, obviously requiring that the factorization can be reasonably well approximated in this way. In order for the strategy to work well, while also keeping $M_{\max}$ low, the number of elements needed per row of the factorization should be relatively constant. In Figure 2(a,b) we show the sparsity pattern of a characteristic SCI matrix and its full LU factorization, respectively. From this figure it is clear how the number of non-zero elements in each row of the factorization can vary from very few, to filling in almost the entire width of the matrix. In this case ILUT provides a very poor approximation to the sparsity pattern of the LU factorization as seen in Figure 2(c). In order for the approximation to be good for low values of $M_{\max}$ the sparsity pattern of the factorization must be made more evenly distributed across the rows. This can be obtained using reordering algorithms. For our type of problem we chose to reorder the numbering of the triangulated datums of the data set, using the reverse Cuthill-Mckee algorithm (RCM, Cuthill and Mckee 1969) to minimize the bandwidth of the $\mathbf{A}$ matrix as seen in Figure 2(d). This type of reordering applied to our particular geometry produces matrices whose factorization is very evenly distributed across rows as seen in Figure 2(e), hence making ILUT a good approximation as seen in Figure 2(f). Also note how the RCM reordering produces an LU factorization with much fewer non-zero elements, an effect that becomes more pronounced with increasing matrix size. Using the RCM reordering we find that $M_{\max} = 25$ and $\varepsilon = 10^{-5}$ works very well regardless of the size of matrix, as will be shown later. For stopping criteria of the iterative solver we find a relative residual of 10^{-6} to work well, which is also the value used by Rücker *et al.* (2006) for their conjugate gradient solver. Using these settings we typically reach convergence within 5–30 iterations regardless of the size of the problem and with an accuracy on par with direct solvers.

Approximate sensitivity analysis

For the final optimization step we have to turn to an approximation. We want to calculate the model parameter sensitivity analysis for the model result by equation (4) but this equation involves calculating the full inverse of a matrix. Calculating the inverse of an (NxN) matrix corresponds to solving the linear system N times for each column vector of the identity matrix, making the process $O(N^3)$ if the linear solver is of complexity $O(N^2)$. Even if an $O(N)$ scaling solver can be provided the scaling of the matrix inversion will be $O(N^2)$, rendering this calculation unfeasible even for moderately sized problems. To overcome this issue we bring down the size of the matrix to be inverted by solving an approximate local analysis problem around each model position. This idea is illustrated in Figure 1 for the calculation of the parameter sensitivity analysis for the model marked in white. Starting at this point we first expand the problem to include all neighbouring models connected by constraints, e.g., the points marked by purple bullets. Another expansion to tier 2 neighbours can then be performed, including all models constrained to the members of the initial tier 1 expansion (red bullets). A local analysis can then be carried out using equation (4) but including only the models contained in the local problem. In the case of 19 layer models including an altitude model parameter this implies inverting a 38×38, 228×228 or 760×760 matrix for the respective cases of single site analysis, tier 1 and tier 2 expansions. Using this type of local approximation we reformulated the problem from inverting one very large matrix into inverting a significantly smaller matrix for each model position. The number of models in each local analysis problem can vary over the positions of a survey but the characteristic number of neighbours is independent of the size of the data set. This implies that the average size of the matrix to be inverted in each local problem is independent of the size of the

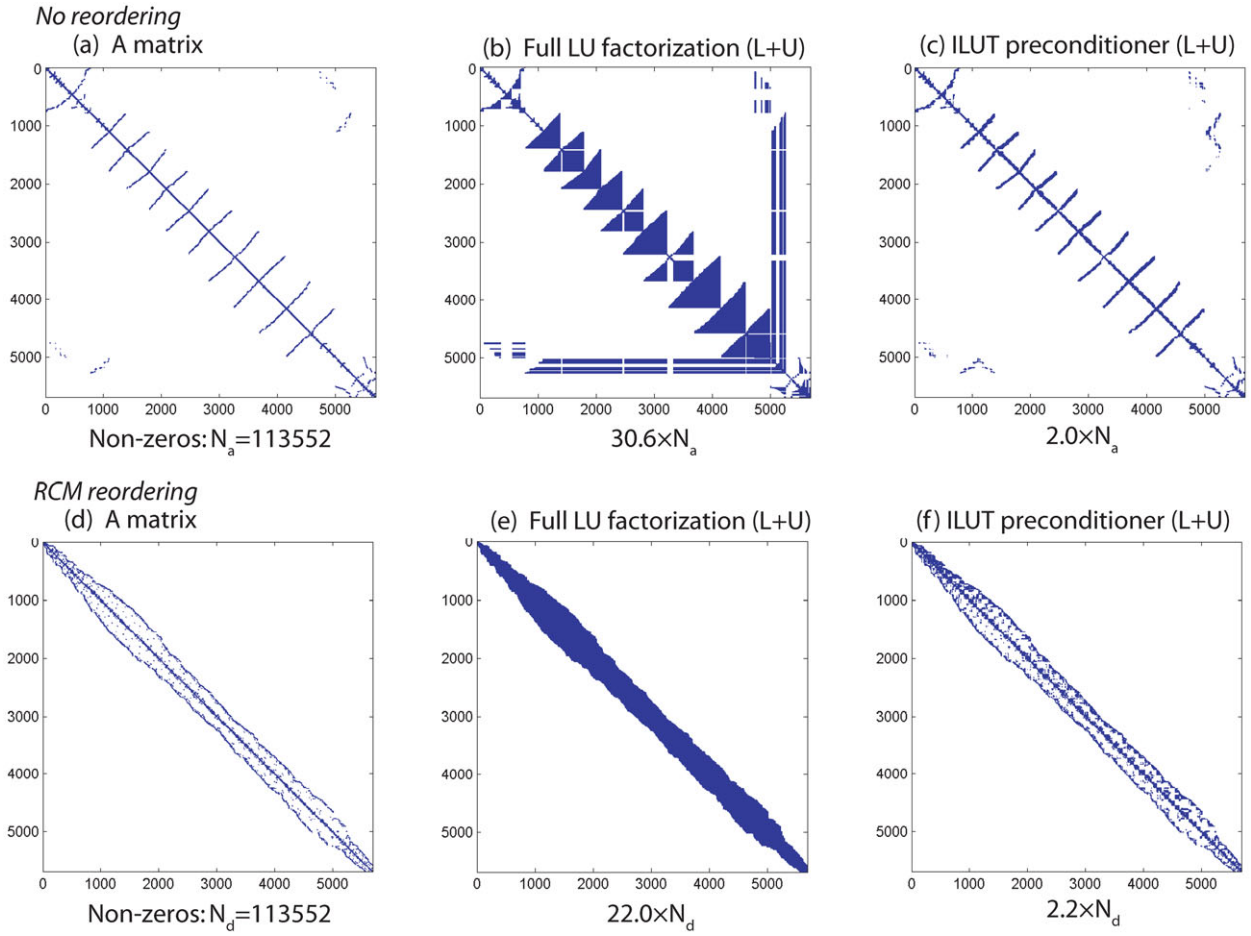

Figure 2 Characteristic matrix sparsity patterns. In the top row are results for unordered model nodes and in the bottom using RCM reordering. (a) and (d) show the left-hand side matrix in equation (3) and (b) and (e) the sum of L and U from an LU factorization. (c) and (f) show L+U for an ILUT preconditioner using the previously described settings $m_{max} = 25$ and $\varepsilon = 10{-}5$.

data set, making the modified algorithm scale linearly. For solution of the small linear systems in the local analysis we use a simple sparse direct solver algorithm (sparse Gauss elimination, Akin 1982) and solve the local problems in parallel using OpenMP.

SCALING AND PERFORMANCE RESULTS

In Figure 3 we show the performance and scaling results of the presented optimizations for the case of SCI inversion of SkyTEM data for models of 19 layers in a fixed vertical discretization. The benchmarks were run on a server purchased in 2011 with 64 GB RAM and 4 physical AMD Opteron 6168 processors, for a total of 48 CPU cores running at 1.9 GHz. As of early 2014 a newer version having 64 cores and a significantly higher clock frequency can be purchased for well below €10 000. Our benchmark computer is thus representative of the type of computational power that is easily obtainable within the budget of any large AEM survey.

For the benchmark system we find that the parallel scaling of the important calculation of forward responses is virtually ideal, as seen in Figure 3(a). A slight deviation from the ideal scaling is introduced when the last few cores are put to use and we therefore generally reserve a core for running the operating system and background processes. The key to obtaining good parallel scaling for this type of compute bound embarrassingly parallel work load, is to obtain good load balancing across all processors throughout almost all iterations of the parallel loop. Using a dynamic load balancing strategy this is easily obtained when the number of 1D models is much greater than the number of processors, as in the case of large AEM data sets.

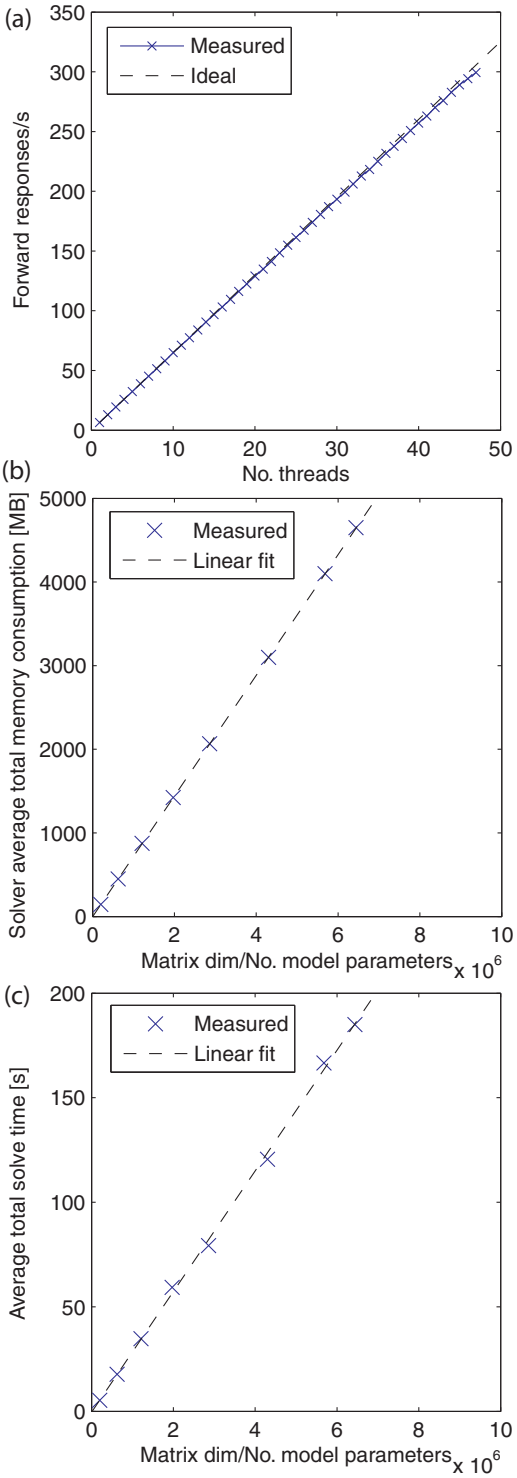

Figure 3 Performance results. (a) The parallel scaling of forward/derivative calculations, (b) the total peak memory requirements for storing and solving a linear system as a function of its size and (c) average total time for solving a linear system as a function of size.

For the sparse matrices of the minimization algorithm in combination with the preconditioned BICGSTAB solver, we show the results for memory consumption and solve time in Figure 3(b,c), respectively. Figure 3(b) shows the total memory requirements for solving the system, i.e., storing both the left-hand matrix itself and the preconditioner and in Figure 3(c) we show the sum of the time used for generating the preconditioner and actually solving the system iteratively. Both figures include a linear fit clearly showing the linear scaling with the size of the problem. As the number of required solver iterations varies along with the utilization of memory allowance of the ILUT algorithm, we show average numbers over full-inversion jobs in Figure 3(c) and peak memory consumption in Figure 3(b).

For the local approximate analysis we use a sparse direct solver to solve small systems of linear equations in parallel after the actual inversion is done. The algorithm is embarrassingly parallel and the calculation time too insignificant to justify systematic benchmarking. In the case of a fully local analysis and tier 1 expansion the sensitivity analysis consistently adds no more than around 1% to the total inversion time. For a tier 2 analysis the calculation adds time comparable to performing an additional iteration in the inversion. A complete inversion run typically consists in 10–15 Gauss-newton type iterations, making the time required for a tier 2 analysis in the order of an additional 7–10%. We will consider the accuracy of different degrees of approximation later and merely note for now that the local algorithm is fast, parallel and scales linearly with the size of the problem.

INVERSION OF A LARGE AIRBORNE ELECTROMAGNETICS DATA SET

Having shown benchmarks for individual optimizations we now demonstrate the codes full capabilities in terms of conducting a single entity spatially constrained inversion of a large data set. The results shown in Figure 4 are extracted from a 19 layer SCI inversion of a 3327 line km groundwater mapping survey conducted over a 730 km^2 area by the Danish/German border. The survey was flown during 2008–2009 using the SkyTEM system (Sørensen and Auken 2004) and consists of around 100 000 model positions of corresponding dual moment soundings. Full details of the survey are given by Jørgensen *et al.* 2012, who interpreted an inversion of the data set obtained with the previous version of our code. The flight lines of the survey are outlined in Figure 4(a), whereas Figure 4(b-c) shows the resistivity and corresponding

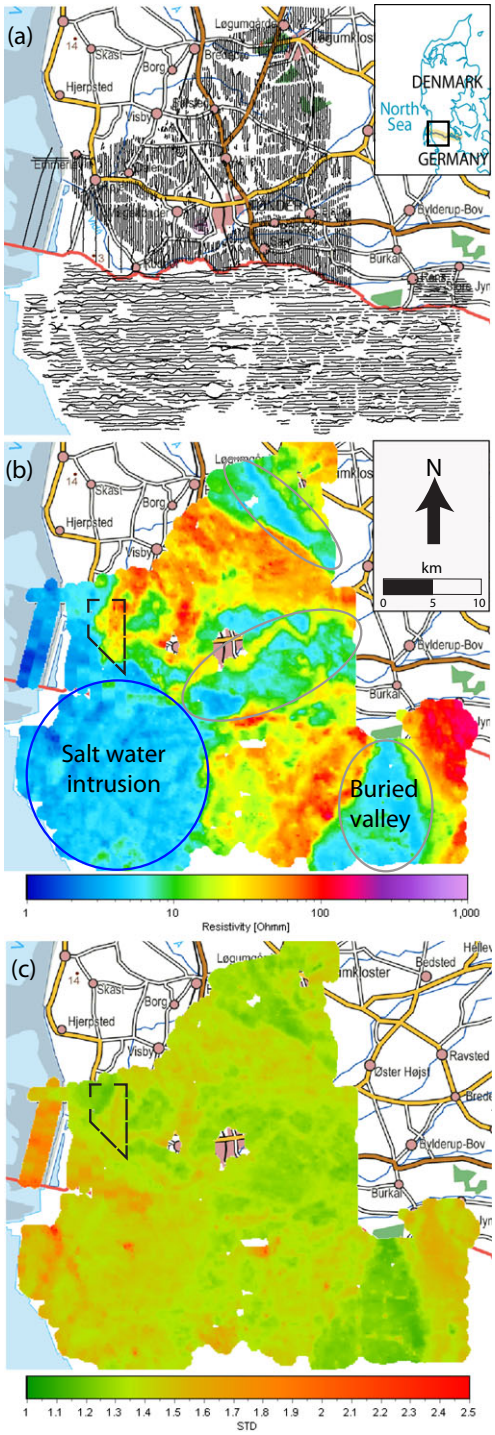

Figure 4 Result of 19 layer SCI inversion of a large SkyTEM data set. (a) Outline of the 3327 km of flight lines, (b) resistivity of the layer at 74–88 m depth and (c) STD factor of the resistivity in (b) obtained from a tier 1 local sensitivity analysis. The black dotted line in (b) and (c) mark the sub-area used for Figure 5 and the blue/grey lines in (b) mark an area of salt water intrusion and position of buried valleys, respectively.

tier 1 approximated STD factor for the model layer situated at 74–88 m depth.

The results in Figure 4 reveal a wealth of features of the subsurface, most notably how salt water from the North Sea intrudes very far inland and how several buried valleys incise the area. A detailed interpretation of these features is outside the scope of this paper and we merely note that our full survey inversion provides a virtual reproduction of the original results of Jørgensen *et al.* 2012. The important thing to note is that we now arrive at the same result in a fraction of the time, while using an approach that is significantly easier to implement, easier to integrate with other modelling types of a large footprint and without conceptual problems in integration of prior information on partition borders. The specifics of the timing and its relation to parallel scaling are discussed in the next section but first we show results comparing different degrees of approximation in the sensitivity analysis as seen in Figure 5. For these results we consider only a small sub-area, marked by a black dotted line in Figure 4, since calculating the full sensitivity analysis for the entire survey area is not computationally feasible. The selected area contains a modest 3660 model positions but it still took a sparse direct solver more than 24 hours to calculate the full sensitivity analysis of Figure 5(d). By comparing this result with the local approximations in Figure 5(b,c), it is clear that results of adequate precision can be produced in a scalable manner that are orders of magnitude faster. On the other hand, Figure 5(a) also clearly shows that a very significant deviation from the full analysis is introduced when too much approximation is introduced. The compromise solutions thus become tier 1 and tier 2 approximations, where we find the tier 1 approximation generally to be accurate enough for most large-scale airborne EM purposes.

TIMING AND DISCUSSION

Utilizing 47 cores of our reference system, the full SCI example problem of the previous section was solved overnight in 14.5 hours, peaking at a memory consumption of 16 GB. Out of this time 90% was spent on parallel forward/derivative calculations, 5.8% on an iterative sequential solution of linear systems, 3.0% on sequential file I/O, sparse matrix algebra and other minor operations, while 1.2% was used for parallel calculation of a tier 1 approximate sensitivity analysis. Assuming ideal parallel scaling of the parallel regions, as justified by Figure 3(a), we calculate the parallel speed-up as 43x for the 47 threads. Using the same assumptions we calculate the parallel speed-up as 142x for a hypothetical machine of 200 cores.

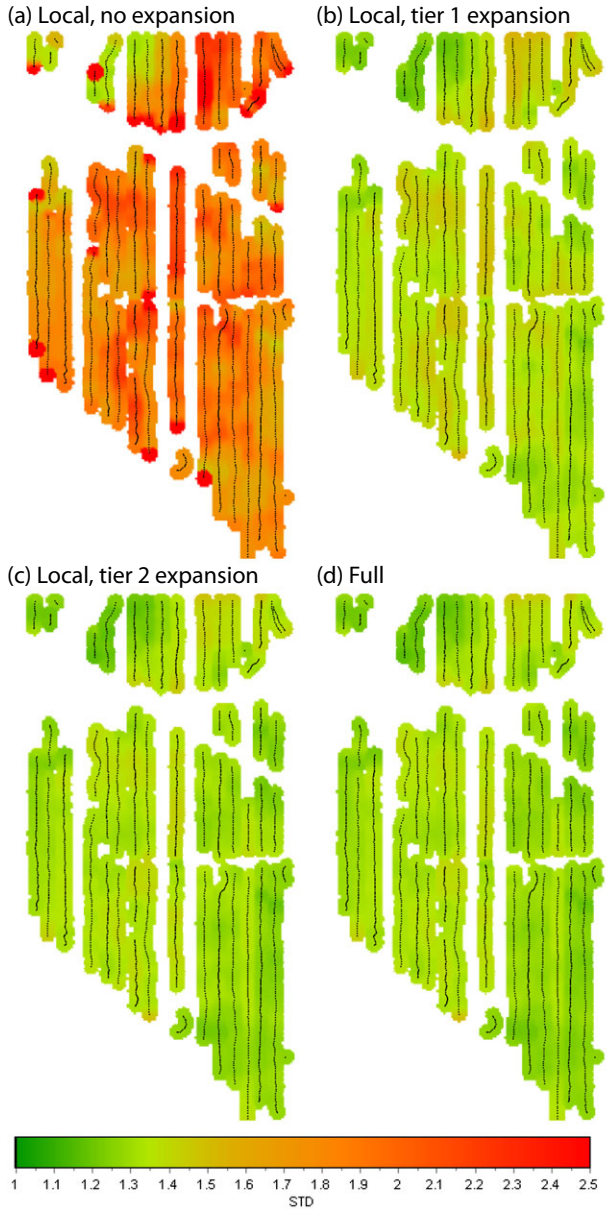

Figure 5 Comparison of sensitivity analysis approximations for a sub-area of the resistivity map shown in Figure 4(b). The area is marked by a black dotted line in Figure 4. (a) Result of local single site sensitivity analysis, (b) local analysis including tier 1 neighbours, (c) local analysis including tier 2 neighbours and (d) full sensitivity analysis. Calculating (a)–(c) is orders of magnitude faster than (d).

At this elevated level of parallelism Amdahls law reduces the parallel efficiency to 71%, as compared to the 91% of the benchmark computer. It is thus clear that the use of a sequential solver does not pose a bottleneck for current generation computer architectures but for significantly higher core counts or a less time-consuming forward response it is bound to become a bottleneck. Typically, 85–90% of the time used in the solver is spent calculating the ILUT preconditioner, with the remainder of the time divided between back-substitution and sparse matrix-vector products. Out of these operations the only readily parallelizable algorithm is in the matrix-vector product. However, given the relatively small amount of time spent on this operation the effect of parallelization is close to negligible. A certain degree of parallelism can be extracted from the process of generating ILU type preconditioners (Saad 2003) but the algorithms are complicated and there are other types of preconditioners that naturally offer a much higher degree of parallelism. We were unable to find a suitable parallel implementation of ILUT and have started to look towards massively parallel preconditioners for future implementation (e.g., approximate inverses and polynomial preconditioners (Saad and Van der Vorst 2000)). However, the need for a parallel solver is far from pressing and the details of these investigations are beyond the scope of this paper.

The presented combination of reordering, preconditioner and solver algorithm was found to provide excellent scaling properties for the specific problem of spatially constrained inversion. However, it should also be applicable to other comparable types of problems. The most crucial aspect in obtaining stability and linear scalability from our approach is for the problem to have proper geometry. When the data set consists of almost parallel flight lines and an occasional tie line, having their datums connected by a Delaunay triangulation, the RCM algorithm is able to reorder the nodes for a very evenly distributed non-zero matrix pattern with a narrow envelope as seen in Figure 2(d). This pattern is extremely well suited for preconditioning with an ILUT preconditioner with low threshold parameters, which is the key to obtaining the scaling. For a similar type of iterative solver, Rücker *et al.* (2006) used approximate minimum degree reordering (AMD, Amestoy, Davis and Duff 1996), since it produces a full factorization of much fewer elements than RCM. This reordering also produces a full factorization of much fewer elements in the case of SCI but the obtained factorization is very unevenly distributed and not well approximated by an ILUT preconditioner. It is therefore worth noting that part of the key to obtaining linear scaling is the use of a reordering algorithm that seems inferior at first glance. For other problems where the nodes can be reordered for a similar pattern the methodology should be equally successful. An example of this could be grid-like geometries, whereas points lying on a circle connected by a Delaunay triangulation would work very poorly. Given a problem of the right geometry it is also important to take relative timing into account. Obviously, the

parallelizable parts of the problem should be adequately time-consuming, not to make the sequential solver a bottleneck at low core counts. In the case of e.g., frequency domain data our forward implementation is more than an order of magnitude faster than the time-domain response, making the sequential solver a bottleneck at a significantly lower core count. We are also about to publish results on an approximate time-domain response for an order of magnitude faster evaluation of partial derivatives (Kirkegaard *et al*. 2013), which adds to the relative importance of the solver also in the case of time-domain data.

CONCLUSIONS

We showed how an existing inversion code was optimized for scalable constrained inversion of very large AEM or similar type data sets. The algorithms underlying our implementation scale linearly with the size of the data set in terms of both memory consumption and computation time, providing the capabilities for spatially constrained inversion of even the largest AEM problems on inexpensive commodity server hardware. To the best of our knowledge, no other large-scale AEM inversion code provides similar scalability. The ability to efficiently solve very large problems further eliminates the need for problem partitioning, which served as an intrinsic inefficiency in our original algorithm. In the new implementation we not only avoid the significant complexity involved in performing the partitioning itself, we also avoid difficulty in employing prior information on border models and speed-up the total inversion time by at least an order of magnitude.

In terms of parallelism we demonstrated how the core regions of the code were parallelized for shared memory multi processor computers using OpenMP. The implementation provides virtually ideal parallel scaling within the parallel regions and good parallel scaling of the application as a whole. Our current 48 core reference system delivers 91% parallel efficiency for full inversion of a 3327 line km SkyTEM field data example and we project that this value will remain at a reasonably efficient 71% for 200 hypothetical cores. From the performance results of our field example we obtained a total inversion time of 4.5 hours and a peak memory consumption of 5 GB pr. 1000 line km of data. These numbers are directly translatable to other survey sizes and we further note that we will soon publish separate results on the use of approximate partial derivatives in SCI inversion, which can be used for very significant speed-up of these numbers.

The most important part of our results is the iterative solver methodology, since this facilitates the linear scaling of the full algorithm. While the solver algorithm is not readily parallelizable its sequential form is so efficient that it allows for excellent parallel scaling of the full code, well beyond the core count of current 4-socket shared memory architectures. Our specific combination of a reordering algorithm, preconditioner and iterative solver is found to be optimal for a sparsity pattern stemming from Delaunay triangulation of 1D point measurements sampled along a flight line pattern. Other problems of comparable geometry and sparsity should be able to benefit from the same solver methodology, provided that parallel parts of the code are numerically expensive enough to make the sequential solver a bottleneck only at very high core counts.

ACKNOWLEDGEMENTS

We thank Dr. Ross Brodie, Geoscience Australia, for in-depth discussions on large-scale inverse problems and for providing a preliminary copy of his PhD thesis. We also thank HOBE (Center of excellence funded by the Villum foundation) and the Faculty of Science and Technology, Aarhus University, for providing the necessary funding for Casper Kirkegaard. Lastly, we thank Dr. Colin Farquharson and an anonymous reviewer for their helpful comments to the original manuscript.

REFERENCES

Akin J.E. 1982. *Application and Implementation of Finite Element Methods. First Edition.* Academic Press.

Amestoy P.R., Davis T.A. and Duff I.S. 1996. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications* **17**, 886–905.

Amestoy P.R., Duff I.S., L'Excellent J.Y. and Koster J. 2002. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications* **23**, 15–41.

Auken E., Christiansen A.V., Jacobsen B.H., Foged N. and Sørensen K.I. 2005. Piecewise 1D Laterally Constrained Inversion of resistivity data. *Geophysical Prospecting* **53**, 497–506.

Auken E., Christiansen A.V., Jacobsen L. and Sørensen K.I. 2008. A resolution study of buried valleys using laterally constrained inversion of TEM data. *Journal of Applied Geophysics* **65**, 10–20.

Auken E., Christiansen A.V., Kirkegaard C., Fiandaca G., Schamper C., Behroozmand A.A., Binley A., Nielsen E., Effersø F., Christensen N.B., Sørensen K.I., Foged N. and Vignoli G. 2014. An overview of a highly versatile forward and stable inverse algorithm for airborne, ground-based and borehole electromagnetic and electric data. *Exploration Geophysics*, 1–13, DOI: 10.1071/EG13097.

Balay S., Gropp W.D., McInnes L.C. and Smith B.F. 1997. Efficient management of parallelism in object oriented numerical software

libraries. *Modern Software Tools in Scientific Computing*, 163–202.

Bhowmick S., Toth B. and Raghavan P. 2009. Towards low-cost, high-accuracy classifiers for linear solver selection. *Computional Science – ICCS 2009*. pp. 463–472. Springer.

Brodie R. 2010. *Holistic Inversion of Airborne Electromagnetic Data*. PhD thesis, The Australian National University, Canberra, Australia.

Brodie R. and Sambridge M. 2006. A holistic approach to inversion of frequency-domain airborne EM data. *Geophysics* **71**, G301–G312.

Chapman B., Gabriele J. and van der Pas R. 2007. *Using OpenMP: Portable Shared Memory Parallel Programming*. MIT press.

Chen J. and Raiche A. 1998. Inverting AEM data using a damped eigenparameter method. *Exploration Geophysics* **29**, 128–132.

Christensen N.B. 2002. A generic 1D imaging method for transient electromagnetic data. *Geophysics* **67**, 438–447.

Christensen N.B., Reid J.E. and Halkjaer M. 2009. Fast approximate inversion of SkyTem airborne electromagnetic data. 1–7.

Christiansen A.V., Auken E. and Viezzoli A. 2011. Quantification of modeling errors in airborne TEM caused by inaccurate system description. *Geophysics* **76**, F43–F52.

Costelloe M.T., Whitaker A., Brodie R., Fisher A. and Sorensen C. 2007. Paterson airborne electromagnetic survey, onshore energy and minerals, Geoscience Australia. *ASEG Extended Abstracts 2007*, 1.

Cox L.H., Wilson G.A. and Zhdanov M.S. 2010. 3D inversion of airborne electromagnetic data using a moving footprint. *Exploration Geophysics* **41**, 250–259.

Cuthill E. and Mckee J. 1969. Reducing the bandwidth of sparse symmetric matrices. *ACM '69 Proceedings of the 1969 24th National Conference*, pp. 157–172.

Ern A., Giovangigli V., Keyes D.E. and Smooke M.D. 1994. Towards polyalgorithmic linear system solvers for nonlinear elliptic problems. *SIAM Journal on Scientific Computing* **15**, 681–703.

Farquharson C.G., Oldenburg D.W. and Li Y. 1999. An approximate inversion algorithm for time-domain electromagnetic surveys. *Journal of Applied Geophysics* **42**, 71–80.

Farquharson C.G., Oldenburg D.W. and Routh P.S. 2003. Simultaneous 1D inversion of loop-loop electromagnetic data for magnetic susceptibility and electrical conductivity. *Geophysics* **68**, 1857–1869.

Fletcher R. 1976. Conjugate gradient methods for indefinite systems. *Lecture Notes in Mathematics*. pp. 73–89. Springer.

George T., Gupta A. and Sarin V. 2008. A recommendation system for preconditioned iterative solvers. *ICDM '08 Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pp. 803–808.

Gould N.I.M., Scott J.A. and Hu Y. 2007. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. *ACM Transactions on Mathematical Software* **33**.

Gropp W.D., Lusk E. and Skjellum A. 1999. *Using MPI: Portable Parallel Programming with the Message-Passing Interface 2nd Edition*. MIT press.

Guillemoteau J., Sailhac P. and Behaegel M. 2012. Fast approximate 2D inversion of airborne TEM data: Born approximation and empirical approach. *Geophysics* **77**, WB89–WB97.

Hestenes M.R. and Stiefel E. 1952. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* **49**, 409–436.

Jørgensen F., Scheer W., Thomsen S., Sonnenborg T.O., Hinsby K., Wiederhold H. et al. 2012. Transboundary geophysical mapping of geological elements and salinity distribution critical for the assessment of future sea water intrusion in response to sea level rise. *Hydrology and Earth System Sciences* **16**, 1845–1962.

Kalvig P. 2008. A conceptual geological model of the Voltaian basin based on GEOTEM data. The Voltaian Basin, Ghana.Workshop and Excursion, March 10–17, 2008, Abstract Volume 25–29.

Kirkegaard C., Schamper C., Christiansen A.V., Vignoli G. and Auken E. 2013. An efficient hybrid inversion scheme combining approximate and full forward solutions for AEM. *SAGA AEM*, South Africa, 1–2.

Lawrie K.C., Tan P., Halas L., Apps H., Cullen K., Brodie C. *et al.* 2010. Using AEM Data as Part of an Integrated Assessment of the Salinity Hazard and Risk to Gunbower State Forest and the River Murray Floodplain in the Gunbower Island-Barr Creek Reach of the Murray River, SE Australia.*ASEG Extended Abstracts 2010*, 1.

Li X.S. and Demmel J.W. 2003. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Transactions on Mathematical Software* **29**, 110–140.

Macnae J., King A., Stolz N., Osmakoff A. and Blaha A. 1998. Fast AEM data processing and inversion. *Exploration Geophysics* **29**, 163–169.

Marquart D. 1963. An Algorithm for Least Squares Estimation of Nonlinear Parameters. *SIAM, Journal of Applied Mathematics* **11**, 431–441.

Menke W. 1989. *Geophysical Data Analysis Discrete Inverse Theory*. Academic Press.

Mueller-Petke M. and Yaramanci U. 2010. QT inversion – Comprehensive use of the complete surface NMR data set. *Geophysics* **75**, WA199–WA209.

Nachtigal N.M., Reddy S.C. and Trefethen L.N. 1992. How fast are nonsymmetric matrix iterations. *SIAM Journal on Matrix Analysis and Applications* **13**, 778–795.

Podgorski J.E., Kgothang L., Ngwisanyi T., Ploug C., Auken E. and Green A.G. 2010. Introducing the Okavango Delta, Botswana, Airborne TEM Survey. EAGE Near Surface 2010.

Press W.H., Teukolsky S.A., Vetterling W.T. and Flannery B.P. 2007. *Numerical Recipes* 3rd Edition. Cambridge University Press.

Reid J.E. and Fullagar P.K. 1998. Conductivity-depth transformation of slingram transient electromagnetic data. *Exploration Geophysics*, 570–576.

Rücker C., Günther T. and Spitzer K. 2006. Three-dimensional modelling and inversion of dc resistivity data incorporating topography – I. Modelling. *Geophysical Journal International* **166**, 495–505.

Saad Y. 1990. SPARSKIT: A basic tool kit for sparse matrix computations. Technical Report RIACS-90–20, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Saad Y. 1994. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications* **1**, 387–402.

Saad Y. 2003. *Iterative Methods for Sparse Linear Systems* (2nd edition). SIAM.

Saad Y. and Schultz M.H. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* **7**, 856–869.

Saad Y. and Van der Vorst H.A. 2000. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics* **123**, 1–33.

Santos F.A.M. 2004. 1D laterally constrained inversion of EM34 profiling data. *Journal of Applied Geophysics* **56**, 123–134.

Sattel D. 2005. Inverting airborne electromagnetic (AEM) data with Zohdy's method. *Geophysics* **70**, G77–G85.

Schenk O. and Gärtner K. 2004. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems* **20**, 475–487.

Sørensen K.I. and Auken E. 2004. SkyTEM – A new high-resolution helicopter transient electromagnetic system. *Exploration Geophysics* **35**, 191–199.

Streich R., Becken M. and Ritter O. 2011. 2.5D controlled-source EM modeling with general 3D source geometries. *Geophysics* **76**, F387–F393.

Tarantola A. and Valette B. 1982. Generalized nonlinear inverse problems solved using a least squares criterion. *Reviews of Geophysics and Space Physics* **20**, 219–232.

Tartaras E. and Beamish D. 2005. Laterally constrained inversion of fixed-wing frequency-domain AEM data. 12th European Meeting of Environmental and Near Surface Geophysics.

Toledo S., Chen D. and Rothkin V. 2001. Taucs – A library of sparse linear solvers: http://www.tau.ac.il/~stoledo/taucs/.

Vallée M.A. and Smith R.S. 2009. Inversion of airborne time-domain electromagnetic data to a 1D structure using lateral constraints. *Near Surface Geophysics* **7**, 63–71.

Van der Vorst H.A. 1992. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* **13**, 631–644.

Viezzoli A., Christiansen A.V., Auken E. and Sørensen K.I. 2008. Quasi-3D modeling of airborne TEM data by Spatially Constrained Inversion. *Geophysics* **73**, F105–F113.

Wilson G.A., Raiche A. and Sugeng F. 2006. 2.5D inversion of airborne electromagnetic data. *Exploration Geophysics*, 363–371.

Wolfgram P., Sattel D. and Christensen N.B. 2003. Approximate 2D inversion of AEM data. *Exploration Geophysics* **34**, 29–33.

Wu X. 2003. A 3D finite-element algorithm for DC resistivity modelling using the shifted incomplete Cholesky conjugate gradient method. *Geophysical Journal International* **154**, 947–956.

Yang D. and Oldenburg D.W. 2012. Three-dimensional inversion of airborne time-domain electromagnetic data with applications to a porphyry deposit. *Geophysics* **77**, B23–B34.