GEOMAR Helmholtz Centre for Ocean Research Kiel

Doctoral Thesis

The application of Krylov subspace methods for the calculation of forward solutions and model sensitivities of 3D time domain marine controlled source electromagnetic problems

Author:
Malte Sommer

Supervisor:
Prof. Dr. Christian Berndt

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in

August 2017

Gutachter:
Prof. Dr. Christian Berndt
Prof. Dr. Lars Rüpke
Prof. Dr. Andreas Hördt

Tag der mündlichen Prüfung: 26.10.2017

Declaration of Authorship

I, Malte Sommer, declare that this thesis titled, 'The application of Krylov subspace methods for the calculation of forward solutions and model sensitivities of 3D time domain marine controlled source electromagnetic problems' and the work presented in it are my own. I confirm:

- that apart from the supervisor's guidance the content and design of the thesis is all my own work,
- whether the thesis has already been submitted either partially or wholly as part of a doctoral degree to another examining body and whether it has been published or submitted for publication;
- that the thesis has been prepared subject to the Rules of Good Scientific Practice of the German Research Foundation;

Signed: _____

Date: _____

Zusammenfassung

Zur Reduzierung der Laufzeiten von 3D Modellierungs- und Inversions-Software für die Interpretation von mariner, aktiver Elektromagnetik (engl. CSEM, controlled source electro magnetics) im Zeitbereich, werden effiziente Algorithmen und Implementierungen auf massiv-paralleler Hardware vorgestellt. Zwei Implementierungen zur Berechnung der Vorwärts Modellierung, sowie eine Implementierung zur Berechnung der Sensitivitäten werden dargestellt. Die zweite Implementierung der Vorwärts Modellierung war ein notwendiger Zwischenschritt, um die Sensitivitäten effizient berechnen zu können. Bei dem ersten Vorwärts Code handelt es sich um eine Implementierung der Spektralen Lanczos Zerlegung (engl. SLDM, Spectral Lanczos Decomposition Method) auf dem Prozessor von Graphik Karten (engl. GPU, Graphics Processing Unit). Hierbei wird die Partielle Differentialgleichung durch Diskretisierung in eine Ordinäre überführt, die mit einer Ansatz Funktion gelöst wird. Der Ansatz kann spektral zerlegt werden, so dass die Lösung mit Eigenwerten und Eigenvektoren bestimmt werden kann, die mit dem Lanzos Algorithmus gefunden werden. Um den Algorithmus massiv parallelisierbar zu machen, wurde eine geeignete Randwertbedingung für den Übergang zum Lufthalbraum gewählt. Die sehr Rechenzeit aufwendige Aufspannung von Krylov Vektoren durch Multiplikationen dünn besetzter Matrizen mit Vektoren (engl. SpMV, Sparse Matrix times Vector) können auf GPUs deutlich schneller berechnet werden als auf CPUs. Die Anwendbarkeit des Codes wird für ein CSEM System demonstriert, wie es am GEOMAR im Einsatz ist. Bei dem Zweiten Vorwärts Code wird die SLDM durch das effektivere Rationale Krylov Unterraum Verfahren (engl. RKSM, Rational Krylov Subspace Method) ersetzt. Dieses reduziert die Dimension und Rechenzeit des Problems deutlich. Die Krylov Vektoren werden hier nicht durch SpMV Operationen, sondern durch Verschieben der System Matrix um eine Polstelle und anschließendes Lösen eines Linearen Gleichungssystems bestimmt. Dies wird mittels einer GPU Implementierung des CG-Verfahrens (vom engl. Conjugate Gradients) durchgeführt. Neben der höheren Effizienz ermöglicht es auch den hier gewählten Ansatz zur Berechnung der Sensitivitäten. Die Genauigkeit des Codes wird für verschiedene Modelle und Kontraste des elektrischen Leitwertes untersucht. Ein Laufzeitvergleich von SLDM und RKSM wird gegeben. Die Sensitivitäten werden mit dem MOR-Verfahren (engl. Model Order Reduction) berechnet. Dabei wird die Ansatz Funktion des Vorwärts Problems nach den Modellparametern abgeleitet, so dass die Sensitivität aus den Eigenpaaren und ihren Ableitungen bestimmt werden kann. Die Ableitung der Eigenpaare wird mittels Perturbationstheorie gefunden. Das RKSM ist hier zum Block-RKSM erweitert. Es wird gezeigt, dass die Methode funktioniert und seine Anwendbarkeit auf einen echten Datensatz demonstriert.

Summary

To reduce the run-times of 3D modeling and inversion software for the interpretation of marine controlled source electromagnetics (CSEM) in time domain, the implementation of efficient algorithms on massive parallel hardware is presented. Two forward modeling implementations as well as an implementation for sensitivity calculation are illustrated. The implementation of the second forward code is necessary to be able to calculate the sensitivities in an efficient way. The first forward code is an implementation of the spectral Lanczos decomposition method on a graphics processing unit (GPU). The partial differential equation is converted to an ordinary by discretization of the spatial operators such that it can be solved by and Ansatz function. The Ansatz is spectrally decomposed such that the solution can be found by eigenvalues and eigenvectors, found by the Lanczos algorithm. To make the algorithm parallelizable, a proper boundary condition for the water-air interface is chosen. The very time consuming spanning of Krylov vectors by multiplications of sparse matrices with vectors (SpMV) can be computed on GPUs much faster than on serial architectures. The applicability of the code for a CSEM system, how it is used at GEOMAR, is demonstrated. In the second forward code, the SLDM is replaced by the more efficient Rational Krylov Subspace Method (RKSM). This reduces the dimension and run-time of the problem drastically. The Krylov vectors are not spanned by SpMV operations, by shifting the system matrix and solving a linear system. This is done by a GPU implementation of the CG- method (conjugate gradients). Additionally to a higher efficiency, this approach make the sensitivity calculation, in the way how it is implemented in this thesis, possible. The accuracy of the code is investigated for different models and conductivity contrasts. The run-times of SLDM and RKSM are compared on different architectures. The sensitivities are computed with the MOR-method (Model Order Reduction). It differentiates the Ansatz function of the forward problem with respect to the model parameters, such that the sensitivities can be found by eigenpairs and its differentiations. The differentiations of the eigenpairs with respect to the model parameters is found by perturbation theory. The RKSM is here extended to its block analog. It is shown that the method works and the applicability to a real data set is shown.

Acknowledgements

Diese Arbeit wäre ohne die Untertstützung und wohl grenzenlosen Liebe meiner Familie nicht zustande gekommen. Sie ermöglichten mir das Leben, welches ich leben möchte. Mein Dank gilt meinen Großeltern Konrad & Ingrid Gerlach sowie meinen Eltern Karin Paulsen & Gerald Sommer.

Bedanken möchte ich mich auch besonders bei meiner Betreuerin Marion Jegen für ihre unendliche Fürsorge und die vielen fachlichen Freiheiten, die sie mir ermöglicht hat. Für die meisten Menschen mit Eigeninitiative wäre es ein Traum eine solche Vorgesetzte zu haben.

Wann immer ich fachliche Hilfe brauchte, konnte ich mich auf Sebastian Hölz verlassen, sofern es sein Fach betraf. Von ihm habe ich viel über marine Elektromagnetik, aber auch ein wenig über mich selbst gelehrnt.

Eine besondere Hilfe war auch Michael Zaslavsky von Schlumberger. Ohne ihn hätte ich einige Probleme mit der Sensitivitätsberechnung nicht gelöst bekommen.

Zu guter Letzt möchte ich mich auch bei dem deutschen Steuerzahler bedanken, sofern es möglich ist sich für eine erzwungene Hilfe zu bedanken. Denn meine Arbeit wurde durch das SUGAR-Projektes des BMBF finanziert.

Introduction

1 Rationale

To understand the context and motivation of this thesis, and how it integrates into current research, this chapter gives an introduction to the technical and theoretical fundamentals. Marine Controlled Source Electromagnetics will be explained in chapter 1.1. The sub-chapter 1.1.1 introduces the acquisition methods. The fundamentals of electrodynamics and how they relate to electromagnetic methods will be explained in chapter 1.1.2. As an important tool for interpretation of 3D data and focus of this thesis, the 3D modeling and inversion will be introduced in chapter 1.2 and 1.3, respectively.

1.1 Marine CSEM

Marine Controlled Source Electromagnetics (CSEM) is a method for the investigation of the sub seafloor distribution of electric conductivity (or its reciprocal, the electrical resistivity) by interpreting the diffusion of artificial electric fields ([11], [6]). This can practically be done by placing electric dipoles for transmitting and receiving onto the seafloor or by towing the transmitter and possibly the receiver(s) through the water column at an altitude of at most a few hundreds of meters above the seafloor 1.2. For the interpretation of acquired CSEM data, two main classes are common. Firstly, the interpretation may be performed in the frequency domain (FD), where the time series measured at the receiver is transformed into the frequency domain and is then, after normalization with the source spectrum, interpreted at a certain predefined set of frequencies of the source normalized spectrum. The change in amplitude and phase at receiver's position gives conclusion about the sub seafloor conductivity distribution. In the interpretation, the inclusion of several frequencies is benifitial, because the depth of sensitivity is frequency dependent. Secondly, measured data may directly be interpreted in time domain (TD), which usually requires a suitable processing of data to obtain the transient response. Both methods have their benefits and drawbacks that will be

explained below. But, since the physics behind both methods is equivalent, both methods should yield similar results, if all information contained within the measured data is considered within the interpretation.
The typical way of interpreting data is to utilize imaging ([40]), forward and inverse modeling techniques, where a conductivity model is systematically modified until synthetic data explain the measured data. For inversions the interpretation by means of 1D models can be considered to be basic standard. The interpretation in terms of 2D or 3D models is more challenging and is a current topic of scientific research. 3D modeling techniques will be explained in chapter 1.2.

During the last 20 years, marine CSEM is firmly established in academia science and industrial exploration for two reasons ([9]): First, it reveals electrical resistivity which is an important property of hydrocarbon reservoirs ([18]). Second, despite the fact that it has a lower resolution than seismics, it has a much better resolution than other non-seismic methods like potential field methods. Besides of industrial hydrocarbon exploration (EMGS: [21], ExxonMobile: [38], Statoil: [15]), the method is also well established in academia for the investigation of other resistive targets like salt diapirs, free gas and gas hydrates ([34]), because seismics fails to determine the vertical extend of the hydrates above a BSR (bottom simulating reflector). Also conductive targets like sulphide deposits ([39]) are recently investigated with CSEM.

1.1.1 Acquisition methods of marine CSEM data

Although the frequency and time domain methods base on equivalent physical principles, the frequency domain horizontal electric dipole-dipole configuration, as illustrated in Fig. 1.2 below, is most common in the marine environment. One reason is that the frequency domain method allows to concentrate all energy at certain frequencies, increasing signal to noise ratio and allowing higher penetration depths. In a time domain signal that is normally a step on/off response, the energy is not equally distributed to all frequencies. This can be understood by the Fourier transform of a typically used step on signal in Fig. 1.1.

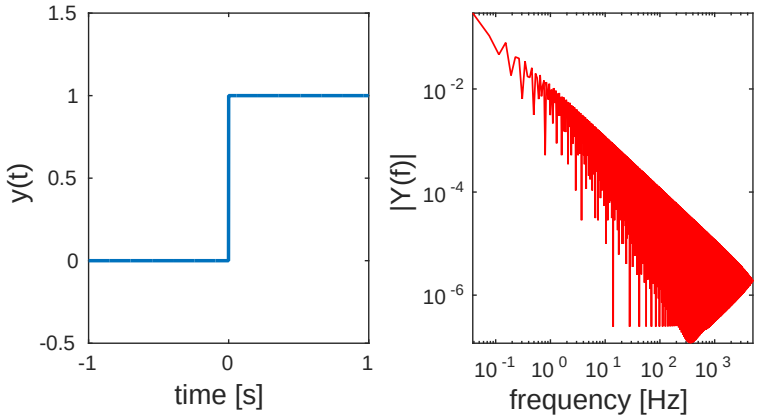

FIGURE 1.1: Time domain (Heaviside) signal (left) and its Fourier transform (right).

Furthermore, receivers only have to measure the amplitude of the electric field, whereas in time domain, a complete transient has to be recorded at high sampling rates. As a consequence, the development of electronic hardware for time domain acquisition is more challenging. A higher dynamic range has to be acquired because more energy is emitted at low than at high frequencies. As a consequence, the signal to noise ratio at high frequencies is poor. Additionally, the sample frequency has to be high. But, the advantage of using time domain is the higher resolution. One transmission contains all possible information. Low frequencies penetrate deeper and require more energy than shallow structures. This is naturally given by the signal shape (Fig. 1.1).
Another question is the type of source, electric- or magnetic-dipole, like a coil. Horizontal dipoles excite both, vertical and horizontal electric currents, increasing the resolution. A vertical magnetic dipole would only generate horizontal currents in 1D [5].

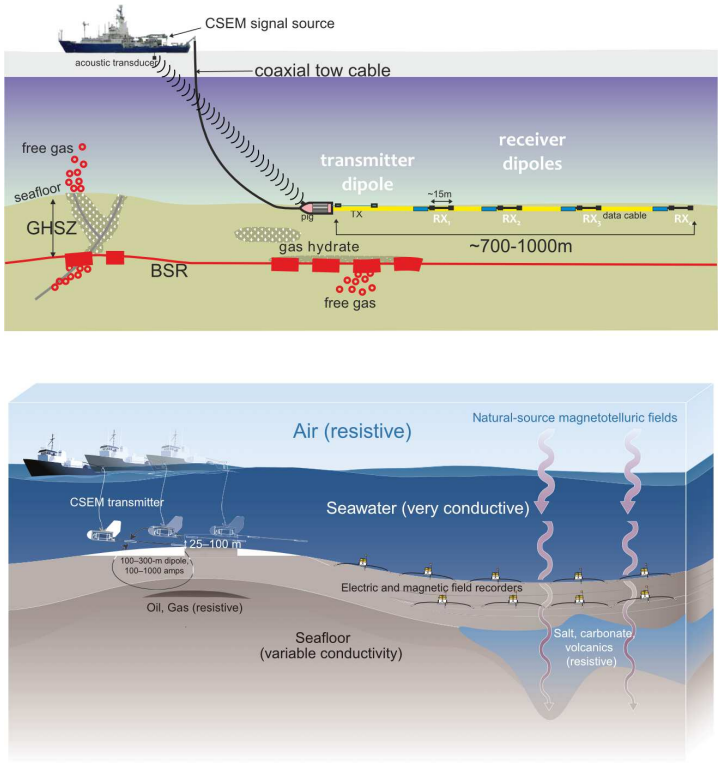

FIGURE 1.2: Examples of CSEM acquisition systems with dipole-dipole configurations. Top: cable based system with fixed geometry, which is moved along the seafloor ([34]), Bottom: system with "flying" source and stationary receiver nodes on the seafloor ([9]). Both systems are mainly used to measure the inline response.

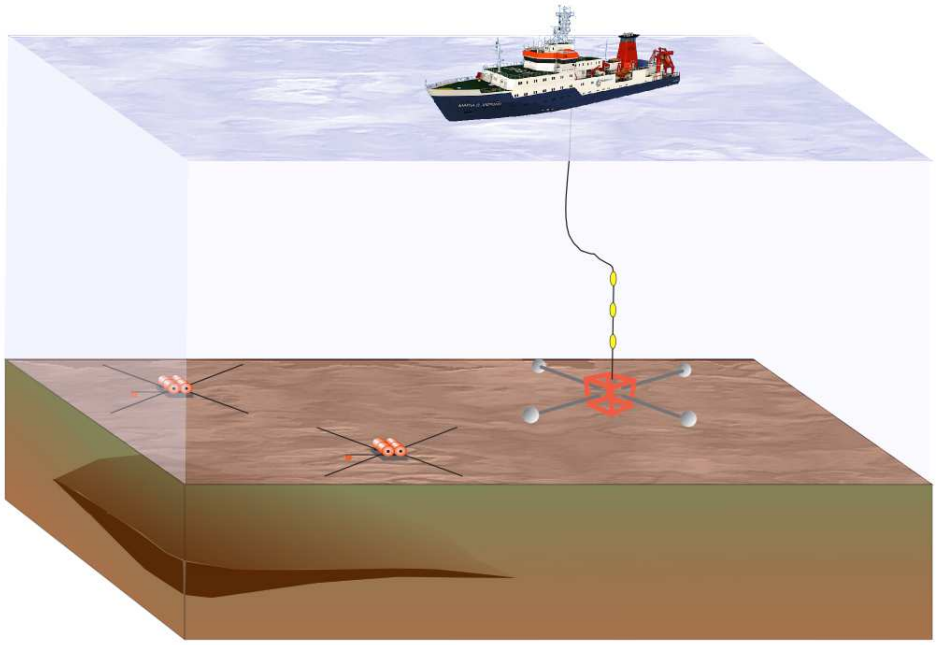

FIGURE 1.3: Time domain system with two horizontal dipole transmitters fixed on a frame.

A time domain dipole-dipole system, as shown in Fig. 1.3, is used at GEOMAR Helmholtz Center for Ocean Research Kiel [19]. Here, two dipole transmitters are mounted on a frame in both horizontal directions that can be placed on the seafloor. Emitting and recording signals in both polarizations results in a higher resolution. The scale of the target is smaller, therefore accurate acoustic navigation systems are required. The recorded data set has to be interpreted by 3D tomography software, rather than 2D profiling techniques, common in frequency domain systems.
Current developments of industrial systems ([27]) improved the signal to noise ratio by emitting higher energies, such that penetration depth of 4500 meters can be achieved.

1.1.2 Fundamental Theory of CSEM

We will use this chapter to introduce the fundamentals of CSEM theory.
All EM methods are based on Maxwell's equations. The equations are named after *James Clerk Maxwell*, who introduced them in 1864. They are four partial differential equations (PDE), which describe the interaction of electric and magnetic fields and electric charges and currents for given boundary conditions. They can be written in differential or integral formalism, former is used throughout this thesis.
The **Gauss's law**:

$$\vec{\nabla} \cdot \vec{E} = \frac{\tilde{\rho}}{\epsilon_0} \tag{1.1}$$

with $\vec{E}$ being the electric field, $\tilde{\rho}$ the electric charge density (we call it $\tilde{\rho}$ to avoid confusion with the resistivity ρ) and ϵ_0 being the electric permittivity of free space.
The **Gauss's law for magnetism**:

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{1.2}$$

with $\vec{B}$ being the magnetic flux density.
Faraday's Law:

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{1.3}$$

and **Ampère's circuital law**:

$$\vec{\nabla} \times \vec{B} = \mu_0 \vec{j} + \mu_0 \epsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{1.4}$$

with μ_0 being the magnetic permeability of free space and $\vec{j}$ being the electric current density. The electric current density in a medium depends on the specific electric resistivity (ρ) and electric field described by **Ohm's law**: $\vec{j} = \frac{\vec{E}}{\rho}$. The reciprocal resistivity $\rho^{-1} = \sigma$ is called electric conductivity.
For CSEM it is essential to be able to describe the electric field for a given environment including geology and sea water, and transmitter signal originating within this environment, on basis of those equations. A way used by Kane Yee in 1966 [45] is to let the Maxwell's equations unchanged and solve both, E- and B-field consecutively in a leap frog way. The more common approach is to combine the equations in a way, such that the whole physics are described by one of the fields alone and a partial differential equation is gained that can be solved with common methods.

Using Ohm's law and differentiating Eq. (1.4) with respect to time, one obtains:

$$\vec{\nabla} \times \frac{\partial \vec{B}}{\partial t} = \mu_0 \sigma \frac{\partial \vec{E}}{\partial t} + \mu_0 \epsilon_0 \frac{\partial \vec{E}^2}{\partial t^2}. \tag{1.5}$$

Substituting Faraday's law yields the Helmholtz equation:

$$\vec{\nabla} \times \vec{\nabla} \times \vec{E} = -\mu_0 \sigma \frac{\partial \vec{E}}{\partial t} - \mu_0 \epsilon_0 \frac{\partial \vec{E}^2}{\partial t^2}. \tag{1.6}$$

Because of the low frequencies used in CSEM and the fact that σ is about 10^9 times larger than ϵ_0 in the marine environment, the wave term can be omitted such that the **Diffusion equation** is obtained:

$$\vec{\nabla} \times \vec{\nabla} \times \sigma^{-1} \mu_0^{-1} \vec{E} = -\frac{\partial \vec{E}}{\partial t} \tag{1.7}$$

It is a parabolic partial differential equation. The initial field of the transmitter can be used as a boundary condition $\vec{E}(t = 0) = \vec{E}_0$. For 1D problems it is normally transformed to frequency domain in cylindrical coordinates by the Hankel transform, to solve an ordinary, instead of an partial differential equation. The solution is back-transformed by an inverse Hankel transform. Numerical solution strategies will be introduced in the next chapter.
The diffusion equation in frequency domain can be found by derivation of the Fourier transformation $\frac{\partial}{\partial t}$ which results in a multiplicator with $i\omega$ where $\omega = 2\pi f$ is the angular frequency and f the frequency. With this, Eq. (1.7) becomes

$$\vec{\nabla} \times \vec{\nabla} \times \sigma^{-1} \mu_0^{-1} \vec{E} = -i\omega \vec{E} \tag{1.8}$$

A typical way to solve the problem numerically in frequency domain is to discretize the spatial operators $\vec{\nabla} \times \vec{\nabla} \times \sigma^{-1} \mu_0^{-1} \rightarrow \mathbf{A}$ (system matrix) and shift $\mathbf{A}$ by frequency $\mathbf{K} = (\mathbf{A} + i\omega \mathbf{I})$ such that a linear system has to be solved

$$\vec{E} = \mathbf{K}^{-1} \vec{E}_0 \tag{1.9}$$

for every frequency. The system has to be solved for several frequencies, because the skin depth of the method is frequency dependent, according to:

$$z_s = \frac{500 meters}{\sqrt{\sigma f}} \tag{1.10}$$

such that low frequencies penetrate deeper.

1.2 3D CSEM modeling

Since reliable and accurate field instruments for CSEM are developed by now, the method is mainly limited by missing interpretation tools. In most applications of CSEM, resistive structures of arbitrary shape and size are encountered that cannot be described by analytical solutions. Most common for 3D time domain are finite difference (FD) and finite element (FE) modeling techniques to discretize Eq. (1.7) and solve the partial differential equation explicitly by time stepping ([44]), implicitly by an Ansatz function ([12]) or Fourier transform the frequency domain solution (Eq. (1.8)) into time domain ([16]).
The most common way to utilize finite difference methods is to discretize the electric field on a staggered yee grid as illustrated in Fig. 1.4. It was introduced by [45] and is the most stable grid for electrodynamic problems [41].

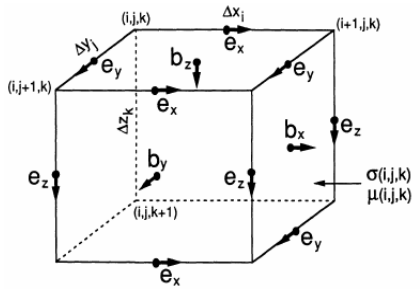

FIGURE 1.4: Yee discretization. The electric fields are defined on the edges of the cell and parallel to them. The magnetic field is defined on the faces of the cell perpendicular to them.

The concept is to discretize the electric field parallel onto the edges and the magnetic field perpendicular to the faces of the cell. The idea was that electric and magnetic field can easily be converted into each other by using the curl equations (Eq. (1.3) and Eq. (1.4)) in every time step such that the solution is found in a leap frog way. Also the concept of setting the electric fields on the edges is suitable for pure electric field computation as well.
A finite difference time stepping code was introduced to CSEM by [44], where the time differential is discretized as well, such that the electric field can be computed at a time t_i if the e-field at t_{i-1} is known such that a solution for all times can be found for a given source field at t_0. The difference $\Delta t = t_i - t_{i-1}$ has to be chosen according to the Courant-criteria ([10]) to give an exact solution. This would lead to around 10^3 time steps for a complete marine transient and every time step requires a matrix-vector multiplication such that computer run times become very high for a complex data set. In marine CSEM, transient are longer since diffusion speed is slower in a conductive

environment and thus require more time steps than land CSEM. Derivation of an electrical resistivity model from data by inversion requires many forward calculations (see chapter 1.3). Optimization and developmend of fast algorithms is therefore a major issue in CSEM. The time stepping approach was further improved, [8] introduced a parallel version of [44]. The back and forward substitution method [43] is a more recent example, of a faster time stepping code.

A completely different strategy has been introduced by [12]. Instead of discretizing the time, the authors solved the differential equation Eq. (1.7) with a spectrally decomposed exponential Ansatz function. The spectra has been approximated by a Krylov subspace projection. Once the spectra is found, the solution can be calculated at any time without knowing previous time steps, such that the Courant-criteria becomes obsolete. This reduces the run times drastically, especially for marine data sets, with its long transients it is of great interest. [44] mentioned 5.5 hours run time, [12] mentioned 15 minutes to 4 hours. However, a one to one comparison is difficult since they were not benchmarked on the same model. Both codes were developed almost at the same time. The Krylov methods were further improved by replacing the polynomial Krylov subspace by the rational Krylov subspace that reduces its dimension drastically [22], [13], [47], [14], [36]). Authors of [3] implemented a rational Krylov algorithm on finite elements.

Nowadays, the research of forward modeling is focused on reducing run time by improving algorithms and implementing codes on new hardware([37], [31]).

1.3 3D CSEM inversion

The purpose of inversion theory is to find, for a given data set, a geological model that explains the data.
The physics of the method are described by a forward modeling method (chapter 1.2). From the pure theoretical point of view, a solution exists by simply trying all geological reasonable models until synthetic and measured data fit. But from the practical point of view, this would be very time consuming (more than 10k years on a desktop). This makes clear that inversion theory is not only about "How to find a solution?", but rather about "How to find a solution **quickly**?".
In the scope of this thesis, we can only give a very brief introduction into this broad topic, more interested readers are refered to [29] and [42]. It is common in inversion theory, to define a model space $\vec{m} \in \Omega_{model}$ and a data space $\vec{d} \in \Omega_{data}$, such that the forward problem is a mapping $f : \Omega_{model} \rightarrow \Omega_{data}$, and $\vec{d}_{model} = f(\vec{m})$. In the beginning of every inversion method, a criteria has to be defined that measures the misfit between modeled and observed data. It is called *objective function* and has for non-linear problems the general form:

$$\Phi(\vec{m}) = \|\mathbf{W}_d \left(\vec{f}(\vec{m}) - \vec{d}_{observed} \right) \|_l + \lambda \Phi_{model}(\vec{m}, \mathbf{W}_m) \tag{1.11}$$

For the l-norm $\|\cdot\|_l$, l is normally set $l = 2$. $\mathbf{W}_d$ and $\mathbf{W}_m$ are data and model weighting matrices. In most geophysical methods, several models would explain the data. This number can be reduced by introducing as additional constraint the *model objective function* Φ_{model}, weighted by a *Legendre parameter* λ. There are several ways how to define Φ_{model}, like the *smallest model*: $\|\mathbf{W}_m \vec{m}\|_2$, the *flattest model*: $\|\vec{\nabla} \vec{m}\|_2$ or the *smoothest model*: $\|\vec{\nabla}^2 \vec{m}\|_2$, for example.
With the objective function, the inversion problem can be described as a minimization problem:

$$\vec{m}_{best} = \underset{\vec{m} \in \Omega_{model}}{\operatorname{argmin}} \ \Phi(\vec{m}) \tag{1.12}$$

The class of gradient based methods is very popular and easy interpretable. For a starting model $\vec{m}_0$, the gradient of the objective function $\vec{\nabla}\Phi(\vec{m}_0)$ is computed and the model is changed in gradient direction by a step length α, such that the iteration $\vec{m}_{i+1} = \vec{m}_i - \alpha \vec{\nabla}\Phi(\vec{m}_i)$ converge to the minimum of Φ.

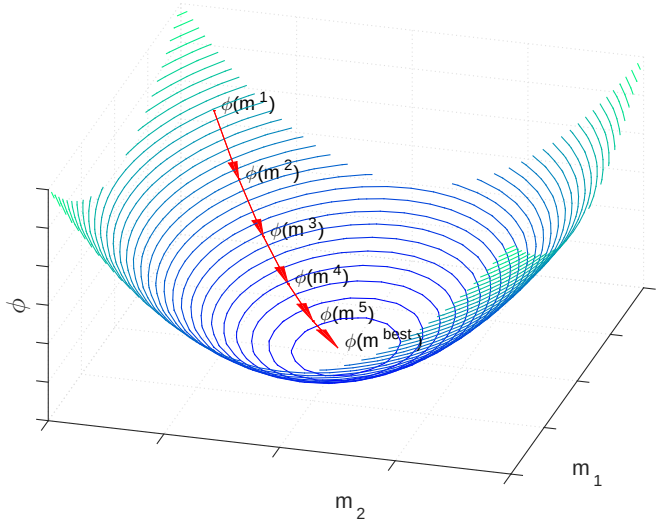

FIGURE 1.5: Illustration of the *steepest descent* method for two model parameters. The method iterates along the the gradient of the objective function Φ, until its minimum is found.

This method is called *steepest descent* and is graphically demonstrated in Fig.1.5. Convergence is very dependent on an optimal choice of the step length α, chosen by a *Line search* algorithm ([29]).

A much better convergence rate can be achieved by the Nonlinear Conjugate Gradient (NLCG) method. The main difference to the steepest gradient method is a choice of the search direction that is much better than the gradient by avoiding a search in one direction multiple times. [Shewchuk] gives a very clear geometrical interpretation of the conjugate gradient method. NLCG is broadly used in EM inversion, because it avoids the storage of large matrices ([7], [23], [33], [30]).

Gradient based methods (steepest descent & NLCG) have the disadvantage that they might converge to local minimas. To find the global minimum, it would be useful to know the complete objective function, what is impossible. But it can locally be approximated by a Taylor series expansion:

$$\Phi(\vec{m}+\delta\vec{m}) \approx \Phi(m) + \vec{\nabla}\Phi(\vec{m})\delta\vec{m} + \frac{1}{2}\vec{\nabla}^2\Phi(\vec{m})\delta\vec{m}^2 \tag{1.13}$$

with $\mathbf{H} = \vec{\nabla}^2\Phi(\vec{m})$ being the Hessian matrix and $\vec{g} = \vec{\nabla}\Phi(\vec{m})$ being the gradient. The global minimum can now be found by setting Eq. (1.13) equal zero, such that:

$$\mathbf{H}\delta\vec{m} = -\vec{g} \tag{1.14}$$

setting $\delta\vec{m} = \vec{m}_{i+1} - \vec{m}_i$ gives the iteration:

$$\vec{m}_{i+1} = \vec{m} - \mathbf{H}_i^{-1}\vec{g}_i. \tag{1.15}$$

This approach is called the *Newton method* (CSEM implementation by [2]). But the computation of the Hessian matrix is very expensive. Therefore, many strategies exist how to approximate it. The Broyden Fletcher Goldfarb Shanno (BFGS) ([4]) algorithm approximates the Hessian by a combination of several gradients. It is also used for marine CSEM ([46]). But authors of [28] show that the *Gauss-Newton* (GN) method leads to much more accurate results. The GN method, approximates the Hessian (**H**) by a product of the Jacobian matrix (**J**):

$$\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \tag{1.16}$$

There are different strategies how to compute the Jacobian. The *Brute Force* or *Finite Difference* method is the most simple way:

$$\mathbf{J} = \frac{\partial \vec{f}(\vec{m})}{\partial \vec{m}} \approx \frac{\vec{f}(\vec{m} + \Delta\vec{m}) - \vec{f}(\vec{m})}{\Delta\vec{m}} \tag{1.17}$$

But it requires a forward solution for every (perturbed) model cell, resulting in enormous run times. Furthermore, the model cells have to be large enough to give an accurate forward solution.

Most common is the *Adjoint Green's Functions approach* (for frequency domain: [25], for time domain: [20]). The time domain version can be deduced from the frequency domain version by convolution. It makes use of the *Born approximation* of the electric field:

$$\vec{E}(\vec{r}, t) = \vec{E}_p(\vec{r}, t) + \int_V \int_0^t \mathcal{G}(\vec{r}\,', t \mid \vec{r}, \tau)\vec{J}_a(\vec{r}\,', \tau)\, d\tau\, d\vec{r}\,' \tag{1.18}$$

that can be understood by Fig. 1.6.

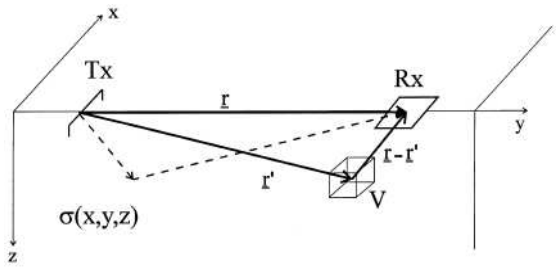

FIGURE 1.6: Illustration of the adjoint principle according to [20].

For a pair of Tx and Rx with offset $\vec{r}$ on a conductive half space $\sigma(x, y, z)$, the Born approximation gives the E-field at Rx-position ($\vec{r}$), by adding on a primary field $\vec{E}_p(\vec{r}, t)$ the effect of a current density $J_a(\vec{r}\,', t)$ within a volume V with anomalous conductivity σ_a at position $\vec{r}\,'$ on the position of Rx, done by an integration with the *Green's function*

$\mathcal{G}(\vec{r}\,', t \mid \vec{r}, \tau)$. It describes the effect of an anomaly at position $\vec{r}\,'$ on $\vec{r}$. Because of reciprocity $\mathcal{G}(\vec{r}, t \mid \vec{r}\,', \tau) = \mathcal{G}(\vec{r}\,', t \mid \vec{r}, \tau)$. The Born approximation neglects the effect of structures surrounding V. With Ohm's Law $\vec{E}_a = \frac{\vec{J}_a}{\sigma_a}$ and some calculus, Eq. (1.18) can be written:

$$\frac{\partial \vec{E}(\vec{r}, t)}{\partial \sigma} \approx \frac{\vec{E}(\vec{r}, t) - \vec{E}_p(\vec{r}, t)}{\sigma_a} = \int_V \int_0^t \mathcal{G}(\vec{r}, t \mid \vec{r}\,', \tau) \vec{E}_a(\vec{r}\,', t')\, d\tau\, d\vec{r}\,' \tag{1.19}$$

what is the Jacobian, per definition. The Green's function can be found by differentiating the full space solution for Rx as source position in every volume V with respect to time. The benefit of the adjoint method is that instead of computing a forward solution for every model cell, only a full space solution for every Tx and Rx is required. The adjoint method is used for frequency domain inversions by [1] and in time domain by [24]
Other ways are to differentiate the solution of the forward problem with respect to the model parameter. For frequency domain, this was done by [26] by derivative calculation of Eq. (1.9) such that the Jacobian is:

$$\mathbf{J} = \left[\frac{\partial \vec{E}}{\partial m_1}, \ldots, \frac{\partial \vec{E}}{\partial m_{N_{mod}}} \right] = \mathbf{K}^{-1}\mathbf{G} \tag{1.20}$$

with $\mathbf{K}$ as defined for Eq. (1.9) and

$$\mathbf{G} = [(\partial \vec{E}_0/\partial m_1 - \partial \mathbf{K}/\partial m_1 \vec{E}), (\partial \vec{E}_0/\partial m_2 - \partial \mathbf{K}/\partial m_2 \vec{E}), \ldots, \tag{1.21}$$

$$(\partial \vec{E}_0/\partial m_{N_{mod}} - \partial \mathbf{K}/\partial m_{N_{mod}} \vec{E})] \tag{1.22}$$

because $\mathbf{K^{-1}} = \mathbf{K}^T$. This method is also used in [17], one of the few existing academic 3D inversion codes that are applied to real data sets ([32]).
Derivation of the forward problem solution in time domain is much more complicated due to the more elaborate nature of the solution strategies. One way is described in [47], where the Jacobian is gained by differentiating the exponential Ansatz function of [12] by the product rule with respect to the model parameter. The eigenpairs are differentiated with respect to the model parameters by perturbation theory. It also leads to a compression of the Jacobian in the Rational Krylov subspace.

2 Hypothesis

For modelling 3D time domain marine CSEM data, Krylov subspace methods are very efficient. Krylov subspace methods are parallelizable on GPU, resulting in a speedup of run time. Rational Krylov subspace methods are more efficient than polynomial ones. Sensitivities can be computed efficiently and accurately by the model order reduction framework that compresses the Jacobian by a rational Krylov subspace.

3 Objectives

The main goal of this thesis is to provide software tools for the interpretation of 3D marine CSEM datasets in time domain, how they are gathered at GEOMAR Helmholtz Centre for Ocean Research Kiel. Because of the nature of those datasets, with many source receiver combinations, transmitting and receiving in both horizontal directions and long transients, the reduction of computation runtime is an essential challenge. The thesis aims 1. to implement a forward modeling code on Graphics Processing Units (GPUs) that might have a much higher computational power than CPUs, and 2. implement a sensitivity calculation algorithm on GPU, because it is the most time consuming step in solving the inverse problem.

For step 1. the suitability of Krylov subspace methods for implementation on GPUs as well as for this pyhsical problem have to be investigated. The polynomial as well as the rational Krylov subspace method are of interest, latter one is essential for the sensitivity calculation. Both Krylov subspace methods have to be compared with each other. Runtimes have to be validated.

For the sensitivity calculation, the implementation of the Model Order Reduction Method (MOR), based on Rational Krylov subspaces, is aimed. There is not much experience gather about the MOR method yet. Therefore, the method's operationability has to be proven by comparison with other methods. Runtime has to be validated as well.

4 Thesis structure

The thesis consists of three parts.
The first part is the publication [37], that describes a GPU parallelization of a 3D forward modeling CSEM code in time domain. The work is similar to the way presented in the Diploma thesis *Adaption and GPU based parallelization of the code TEMDDD for marine CSEM*, but has some significant differences. In the diploma thesis, a GPU parallelization of a *Householder tridiagonalization* and an eigensolver was described. The parallelization of the Krylov vector computation was not successful, because the boundary condition turned out to be problematic. With the new boundary condition as presented in [37], the Krylov vector computation could be parallelized. This made the Householder tridiagonalization obsolete. The eigensolver was replaced as well, such that the code presented in [37] is completely new.
The second part is the submitted publication presented in [36], that replaces the polynomial by a rational Krylov algorithm. This reduces the run time and is also essential for the implementation of the model order reduction method to compute the Jacobian matrix. This is explained in the third part of this thesis as a chapter.

Bibliography

[1] Abubakar, A., Habashy, T. M., Li, M., and Liu, J. (2009). Inversion algorithms for large-scale geophysical electromagnetic measurements. Inverse Problems, 25:1–30.

[2] Amaya, M., Hansen, K. R., and Morten, J. P. (2016). 3D CSEM data inversion using Newton and Halley class methods. Inverse Problems, 32:1–27.

[3] Börner, R.-U., Ernst, O. G., and Güttel, S. (2015). Three-dimensional transient electromagnetic modelling using Rational Krylov methods. Geophysical Journal International, 202(3):2025.

[4] Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms. Journal of the Institute of Mathematics and Its Applications, 6:76?90.

[5] Chave, A., Constable, S., and Edwards, R. (1991). Electromagnetic Methods in Applied Geophysics - Volume 2, Applications, chapter Electrical Exploration Methods for the Seafloor, pages 931–966. SEG.

[6] Chave, A. and Cox, C. (1982). Controlled Electromagnetic Sources for Measuring Electrical Conductivity Beneath the Oceans: 1. Forward Problem and Model Study. Journal of Geophysical Research, 87(B7):5327–5338.

[7] Commer, M. (2003). Three dimensional inversion of transient electromagnetic data: A comparative study. PhD thesis, Universität Köln.

[8] Commer, M. and Newman, G. (2004). A parallel finite-difference approach for 3D transient electromanetic modeling with galvanic sources. Geophysics, 69 (5):1192–1202.

[9] Constable, S. (2010). Ten years of marine CSEM for hydrocarbon exploration. Geophysics, 75 (5):75A67–75A81.

[10] Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen Differenzengleichungen der mathematischen Physik. Mathematische Annalen, 100:32–74.

[11] Cox, C. S. (1980). Electromagnetic induction in the oceans and inferences on the constitution of the Earth. Geophysical Surveys, 4:137–156.

[12] Druskin, V. and Knizhnerman, L. (1994). Spectral approach to solving three-dimensional Maxwell's equations in the time and frequency domain. Radio Science, 29:937–953.

[13] Druskin, V., Knizhnerman, L., and Zaslavsky, M. (2009). Solution of large scale evolutionary problems using rational Krylov subspaces for with optimized shifts. SIAM Journal, 31:3760–3780.

[14] Druskin, V. and Simoncini, V. (2011). Adaptive Rational Krylov Spaces for Large-Scale Dynamical Systems. Systems & control Letters, 32:2485–2496.

[15] Ellingsrud, Eidesmo, Johansen, Sinha, MacGregor, and Constable (2002). Remote sensing of hydrocarbon layers by seabed logging SBL: Results from a cruise offshore Angola. The Leading Edge, 21:972–982.

[16] Everett, M. E. and Edwards, N. (1992). Transient marine electromagnetics: the 2.5-d forward problem. Geophysical Journal International, 113:545–561.

[17] Grayver, A. V., Streich, R., and Ritter, O. (2013). Three-dimensional parallel distributed inversion of CSEM data using a direct forward solver. Geophysical Journal International, 193:1432–1446.

[18] Hesthammer, J., Stefatos, A., Boulaenko, M., Vereshagin, A., Gelting, P., Wedberg, T., and Maxwell, G. (2010). CSEM technology as a value driver for hydrocarbon exploration. Marine and Petroleum Geology, 27:1872–1884.

[19] Hölz, S. and Jegen, M. (2010). The resistivity structure of the North Alex Mud Volcano as derived from the interpretation of CSEM data. In EGU General Assembly Conference Abstracts, volume 12 of EGU General Assembly Conference Abstracts, page 9841.

[20] Hördt, A. (1998). Calculation of electromagnetic sensitivities in the time domain. Geophysical Journal International, 133:713–720.

[21] Johansen, S., Brauti, K., Fanavol, S., Amundsen, H., Wicklund, T. A., Danielsen, J., Gabrielsen, P. T., Lorentz, L., Frenkel, M., Dubois, B., Christensen, O., Elshaug, K., and Karlsen, S. A. (2008). How EM survey analysis validates current technology, processing and interpretation methodology. First Break, 26, No. 6:83–88.

[22] Knizhnerman, L. A., Druskin, V. L., and Zaslavsky, M. (2009). On optimal convergence rate of the rational krylov subspace reduction for electromagnetic problems in unbounded domains. SIAM Journal, 47:953–971.

[23] Mackie, R. L. and Madden, T. R. (1993). Three-dimensional magnetotelluric inversion using conjugate gradients. Geophysical Journal International, 115:215–229.

[24] Martin, R. (2009). Development and application of 2D and 3D transient electromagnetic inverse solutions based on adjoint Green functions: A feasibility study for the spatial reconstruction of conductivity distributions by means of sensitivity. PhD thesis, Universität zu Köln.

[25] McGillivray, P. R., Oldenburg, D. W., Ellis, R. G., and Habashy, T. M. (1994). Calculation of sensitivities for the frequency-domain electromagnetic problem. Geophysical Journal International, 116:1–4.

[26] Newman, G. A. and Hoversten, G. M. (2000). Solution strategies for two- and three-dimensional electromagnetic inverse problems. Inverse Problems, 16:1357–1375.

[27] Nguyen, A. K., Hanssen, P., Mittet, R., Jensen, H. R., Fogelin, L. T. T., Skarø, M., Rosenquist, M., and van der Sman, P. (2017). The Next Generation Electromagnetic Acquisition System. EAGE Conference & Exhibition.

[28] Nguyen, A. K., Nordskag, J. I., Wiik, T., Bjørke, A. K., Boman, L., Pedersen, O. M., Ribauso, J., and Mittet, R. (2016). Comparing large-scale 3D Gauss-Newton and BFGS CSEM inversion. SEG International Exposition and 86th Annual Meeting, pages 872–877.

[29] Nocedal, J. and Wright, S. (2006). Numerical Optimization. Springer, second edition edition.

[30] Plessix, R.-E. (2008). A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. Inverse Probl., 24:1–22.

[31] Puzyrev, V., Koric, S., and Wilkin, S. (2016). Evaluation of parallel direct sparse linear solvers in electromagnetic geophysical problems. Computers & Geosciences, 89:79–87.

[32] Ritter, O., Tietze, K., Patze, C., and Veeken, P. (2017). Onshore 3d csem inversion in practice. 6th International Symposium on Three-Dimnesional Electromagnetics.

[33] Rodi, W. and Mackie, R. L. (2001). Nonlinear conjugate gradients algorithm for 2-D magnetotelluric inversion. Geophysics, 66:174–187.

[34] Schwalenberg, K., Haeckel, M., Poort, J., and Jegen, M. (2010). Evaluation of gas hydrate deposits in an active seep area using marine controlled source electromagnetics: Results from Opouawe Bank, Hikurangi Margin, New Zealand. Marine Geology, 272:79–88.

[Shewchuk] Shewchuk, J. R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report.

[36] Sommer, M., Hölz, S., Avdeeva, A., and Jegen, M. (2017). Comparison of Polynomial- and Rational Krylov Subspace methods for marine CSEM on GPU. Journal of Computational Physics, submitted.

[37] Sommer, M., Hölz, S., Moorkamp, M., Swidinsky, A., Heincke, B., Scholl, C., and Jegen, M. (2013). GPU parallelization of a three dimensional marine CSEM code. Computers & Geosciences, (58):91–99.

[38] Srnka, L. (1986). Method and apparatus for offshore electromagnetic sounding utilizing wavelength effects to determine optimum source and detector positions. US Patent 4,617,518.

[39] Swidinsky, A., Hölz, S., and Jegen, M. (2012). On mapping seafloor mineral deposits with centeral loop transient electromagnetics. Geophysics, 77(3):E171–E184.

[40] Swidinsky, A., Hölz, S., and Jegen, M. (2015). Rapid resistivity imaging for marine controlled-source electromagnetic surveys with two transmitter polarizations: An application to the North Alex mud volcano, West Nile Delta. Geophysics, 80(2):E97–E110.

[41] Taflove, A. and Hagness, S. C. (2005). Computational Electrodynamics: The Finite-Difference Time-Domain Method. Artech House Antennas and Propagation Library, Philadelphia, PA.

[42] Tarantola, A. (1987). Inverse Problem Theory: methods for data fitting and model parameter estimation. Elsevier Science Publishing Co.

[43] Um, E. S., Harris, J. M., and Alumbaugh, D. L. (2010). 3d time-domain simulation of electromagnetic diffusion phenomena: a finite-element electric-field approach. Geophysics, 75:112–126.

[44] Wang, T. and Hohmann, G. (1993). A finite-difference, time-domain solution for three-dimensional electromagnetic modelling. Geophysics, 58:797–809.

[45] Yee, K. (1966). Numerical Solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Transactions on Antennas and Propagation, AP-14:302–307.

[46] Zach, J. J., Bjørke, A. K., Støren, T., and Maaø, F. (2008). 3D inversion of marine CSEM data using a fast finite-difference time-domain forward code and approximate Hessian-based optimization. Technical report, SEG.

[47] Zaslavsky, M., Druskin, V., and Knizhnerman, L. (2011). Solution of 3D time-domain electromagnetic problems using optimal subspace projection. 76.

Computers & Geosciences 58 (2013) 91–99

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

GPU parallelization of a three dimensional marine CSEM code

M. Sommer [a,*], S. Hölz [a], M. Moorkamp [b], A. Swidinsky [a], B. Heincke [a], C. Scholl [c], M. Jegen [a]

[a] GEOMAR, Helmholtz Centre for Ocean Research Kiel, Wischhofstr. 1-3, 24148 Kiel, Germany
[b] University of Leicester, Department of Geology, University Road, Leicester LE1 7RH, UK
[c] Fugro EM Italy - a CGG Company, Via Cardinale Mezzofanti 34, 20133, Italy

ARTICLE INFO

Article history:
Received 4 October 2012
Received in revised form
25 February 2013
Accepted 9 April 2013
Available online 20 May 2013

Keywords:
Finite difference modeling technique
Marine geophysics

ABSTRACT

One of the main problem of 3D time domain controlled source electromagnetic (CSEM) inversion is the high runtimes of forward modeling codes. We reduced the runtime of the 3D time domain finite difference CSEM code TEMDDD by the GPU-parallelization of expensive algorithms. The code solves the electromagnetic diffusion equation by discretization of spatial operators and subsequent calculation of eigenpairs. These eigenpairs are found by approximation of the eigenspace in a Krylov subspace using the spectral Lanczos decomposition Method. This algorithm was in its original form not parallelizable due to implementation of the upper boundary condition at the air–water interface. We show for the marine case that replacing the original boundary condition at the air–water surface by a discretized air layer allows GPU parallelization of every time consuming algorithm of the code in the marine case. Speedups between 20 and 60 have been achieved compared to the original code for a larger 3-D model. In this model the bathymetry from a survey area offshore Egypt is used as an example demonstrating that the parallelized version of the code is applicable to real survey scenarios.

1. Introduction

Marine controlled source electromagnetic (MCSEM) methods are used to image the spatial resistivity distribution of the sub-seafloor. Typical applications for which MCSEM measurements are used to include the exploration of resistive hydrocarbons such as oil, gas and gas hydrates (e.g. Hesthammer et al., 2010; Schwalenberg et al., 2010) and the detection and characterization of highly conductive ore deposits like marine massive sulphide bodies (Swidinsky et al., 2012). Generally, a MCSEM survey consists of an artificial electromagnetic source and multiple electromagnetic field receivers. In most experimental setups the sources are either towed behind a vessel (e.g. Constable and Srnka, 2007; Schwalenberg et al., 2010) or mounted on a remotely operated vehicle (Hölz and Jegen, 2009). The receivers are usually either arranged in an array on the seafloor and remain stationary during the measurements (Constable and Srnka, 2007) or are fixed at a streamer that is moved in a purely inline configuration along the seafloor (Yuan and Edwards, 2000). The interpretation of MCSEM surveys typically relies on modeling and inversion of electric and/or magnetic field data either at one or several source normalized discrete frequencies (frequency domain; see e.g. Gribenko and Zhdanov, 2007) or at a number of samples representing the full transient waveform (time domain; see e.g. Commer et al., 2006). A common approach for EM modeling is the use of finite differences (FD) in time- and frequency-domain, because the formulation is relatively easy and modeling is computationally effective when sparse matrix approaches are used. However, depending on the implementation the full solution requires either many time steps (Oristaglio and Hohmann, 1984) or elaborate implicit solvers (Druskin and Knizhnerman, 1994), which generally makes calculations computationally more expensive than in the frequency domain.

In the scope of this paper we investigate the potential for parallelization of the time domain code TEMDDD (Árnason, 1999) that was originally designed for modeling of land based EM-surveys. For average sized 3-D models this code has runtimes of 20 min ($50 \times 50 \times 40$ grid cells) up to 1 h ($70 \times 70 \times 50$ grid cells) on one standard CPU kernel for a single forward calculation. One way to speedup a code is to use graphics processing units (GPUs). GPUs, originally developed for graphic rendering, have a massively parallel structure and have a high computational power that can also be used for scientific computing. Moorkamp et al. (2010) used GPUs to speedup a gravity forward code by a factor of 30, Komatisch et al. (2010) improved the performance of a seismic forward code by a factor of 20 and da Silva et al. (2012) parallelized a frequency domain CSEM code on GPUs.

In this paper we investigate possibilities to speedup the most time consuming algorithms of TEMDDD. We show that for the marine case, the two most time consuming algorithms can be skipped because the seawater layer above the source and receivers

* Corresponding author. Tel.: +49 431 600 2556.
E-mail address: msommer@geomar.de (M. Sommer).

0098-3004/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cageo.2013.04.004

allows a simplification of the water–air boundary condition. Another advantage of such a modified boundary is that the remaining algorithms can be sped up by implementing GPU parallelization, which was not possible in its original form. By means of synthetic models we demonstrate that the runtime for typical model sizes can be reduced drastically by these modifications. The synthetic models are related to a CSEM data set that is collected on a mud volcano in the West Nile Delta.

2. The FD time domain code

Before introducing our modifications and subsequent parallelization, we first describe the main characteristics of the TEMDDD code. The original FD time domain code is developed by Árnason (1999) and solves the electromagnetic diffusion equation

$$\nabla \times \nabla \times (\sigma\mu)^{-1}\vec{E} = -\frac{\partial \vec{E}}{\partial t} \tag{1}$$

with the initial value $\vec{E}(t=0)=\vec{E}_0$. Here, σ is the spatially varying electric conductivity, μ is the magnetic permeability, which is set to the permeability of free space, and $\vec{E}$ is the electric field within the modeling domain. The source field is described by current densities that are placed on the edges of the FD grid cells. Electric field components are discretized in space using central finite differences on a Yee grid (Yee, 1966), where electric field vectors are defined in the middle of the cell edges and the conductivities are kept constant within each cell (see Fig. 1). Outside of the lateral and lower bounds of the modeling domain, the grid is embedded in a perfect conductor such that the electric field vanishes here and Dirichlet boundary conditions are enforced. At the water–air interface the $\vec{E}$ –field components immediately above the interface (see Fig. 1) are located in the air, where $\sigma \to 0$. Therefore, Eq. (1) cannot be applied and Laplace equation $\nabla^2\vec{E}=0$ is used instead. The electric field values in the air can then be calculated using a convolution integral of field values at the water air–interface by upward continuation (e.g. Oristaglio and Hohmann, 1984; Weidelt, 2000). This means that for this setup the air layer is not a part of the modeling grid in the original code and therefore two separate problems have to be to solved: (i) the solution of the electric fields inside the volume and (ii) the solution at the water–air boundary.

2.1. Solving the volume problem

Discretization of the spatial operators and the electric conductivities on the left side of Eq. (1) yields a banded operator matrix **A** with 13 (see Fig. 1) entries per row acting on the associated electric field component

$$\mathbf{A}\vec{E} = -\frac{\partial \vec{E}}{\partial t}. \tag{2}$$

This is a partial differential equation of first order which can be solved by an exponential ansatz

$$\vec{E} = \vec{E}_0 \exp(-t\mathbf{A}) = \sum_{i=1}^{n} E_{0i} \exp(-t\lambda_i)\vec{z}_i. \tag{3}$$

The integer n is the number of rows of **A**, E_{0i} is the ith components of $\vec{E}_0$, λ_i is the ith eigenvalue of **A** and $\vec{z}_i$ is the corresponding eigenvector. The validity of the eigendecomposition can be shown by Taylor series expansion (Druskin and Knizhnerman, 1994). For typical 3D models in MCSEM, where the size of the system matrix **A** is on the order of hundreds of thousands, determining the eigenpairs of **A** using a full eigensolver is not feasible. Therefore, a subspace approximation of **A** is required. This is done by the spectral Lanczos decomposition method (SLDM) (Parlett, 1998), which approximates **A** by a projection to a Krylov subspace $\mathbf{K}^k = span\{\vec{E}_0, \mathbf{A}\vec{E}_0, \mathbf{A}^2\vec{E}_0, \ldots, \mathbf{A}^{k-1}\vec{E}_0\}$, where the vectors of the spanning set are called Krylov vectors. An orthonormal basis $\vec{q}_0, \ldots, \vec{q}_{k-1}$ can be generated by Gram–Schmidt orthogonalization of the Krylov vectors by using the

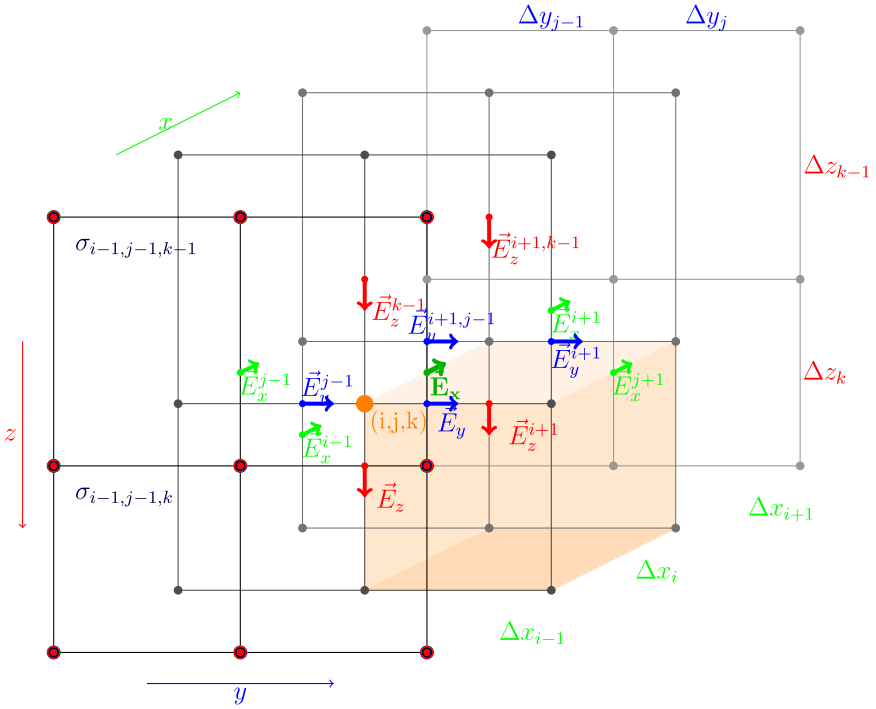

Fig. 1. Sketch showing 13 field components that are associated with node (i,j,k), indicated by orange circle. Its corresponding cell with conductivity $\sigma_{i,j,k}$ is orange colored. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

recursion formula

$$\hat{\vec{q}}_{i+1} = \mathbf{A}\vec{q}_i - (\vec{q}_i^T \mathbf{A}\vec{q}_i)\vec{q}_i - (\vec{q}_{i-1}^T \mathbf{A}\vec{q}_i)\vec{q}_{i-1} \quad (4)$$

and $\vec{q}_0 = \vec{E}_0/\|\vec{E}_0\|$. By normalizing $\vec{q}_{i+1} = \hat{\vec{q}}_{i+1}/\|\hat{\vec{q}}_{i+1}\|$, defining $\alpha_i = \vec{q}_i^T \mathbf{A}\vec{q}_i$, and $\beta_i = \vec{q}_i^T \mathbf{A}\vec{q}_{i+1} = \|\hat{\vec{q}}_{i+1}\|$, Eq. (4) becomes $\mathbf{A}\vec{q}_i = \beta_{i-1}\vec{q}_{i-1} + \alpha_i\vec{q}_i + \beta_i\vec{q}_{i+1}$. This linear system can then be written as the matrix equation $\mathbf{AQ} \approx \mathbf{QH} \Rightarrow \mathbf{A} \approx \mathbf{QHQ}^{\mathbf{T}}$, with $\mathbf{H} \in \mathbb{R}^{\mathbf{k}} \times \mathbb{R}^{\mathbf{k}}$ being the Ritz approximation

$$\mathbf{H} = \begin{pmatrix} \alpha_1 & \beta_1 & \dots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \beta_{k-1} & \alpha_k \end{pmatrix}. \quad (5)$$

The eigenvalues of $\mathbf{H}$ are the k largest eigenvalues of $\mathbf{A}$ and the eigenvectors of $\mathbf{A}$ can be found by the back transformation $\vec{z}_i = \mathbf{Q}^T\vec{v}_i$, with $\vec{v}_i$ being the ith eigenvector of $\mathbf{H}$. Instead of solving $\mathbf{A}$ for all n eigenvalues, just k eigenvalues, usually less than a few thousands, are required. Further details are given in Druskin and Knizhnerman (1994) and van der Vorst (1987).

2.2. Solving the water–air boundary problem

The electric field vectors in the air above the water–air boundary are computed by upward continuation of the horizontal field vectors at the sea surface. To solve the Laplace equation, a Householder tridiagonalization, an eigensolver and a convolution in the SLDM are required. For further details see Árnason (1999) and Weidelt (2000).

The Householder tridiagonalization is used to transform the dense surface matrix (nearly all of its elements are nonzero) into a sparse matrix that can be solved by more efficient eigensolvers. As an arbitrarily dense matrix $\mathbf{A}_d$ can be expressed as $\mathbf{A}_d = \mathbf{QTQ}^T$ and the Householder method transforms it by orthogonal projections $\mathbf{Q}$ to a tridiagonal matrix $\mathbf{T}$. Such orthogonal projections do not change the eigenvalues of a matrix, but rotate the eigenvectors. This means that the eigenvalues of $\mathbf{T}$ and $\mathbf{A}$ are the same and its eigenvectors can be found by backtransformation of the eigenvectors of $\mathbf{T}$

$$\vec{v}_{\mathbf{A}_d} = \mathbf{Q}^T\vec{v}_{\mathbf{T}} \quad (6)$$

because $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ (see Householder, 1958). The eigensolver used here is the QR-algorithm (Watkins, 1982), that is a standard algorithm for eigenvalue computation (Golub and Van Loan, 1996).

3. Gridding and runtime profiling

In the original, non-parallelized code, three algorithms are computationally expensive: (1) the Householder tridiagonalization (see Section 2) used to solve the discretized Laplace operator, (2) the SLDM and (3) the tridiagonal symmetric full eigensolver, which is invoked twice, once for the surface matrix and once for the Ritz approximation equation (5).

Fig. 2 shows runtimes of these three algorithms. Most time consuming are clearly the algorithms related to the surface matrix, namely the Householder tridiagonalization and its eigensolver. For typical model sizes their combined runtime exceeds 1 h. Algorithms computing the eigenpairs of a sparse system matrix as well as the SLDM typically need less than 10 min each.

To skip algorithms used for the surface approximation, we replaced the originally implemented upper boundary condition by a perfect conductor Hördt and Müller, 2000. This yields Dirichlet boundary conditions at the upper boundary like it is already the case for the lateral and lower bounds of the modeling domain. This now requires that the air is explicitly included as a resistor into the modeling domain. In Section 3.2 we will show that this approach is valid with numerically accurate results for a wide range of relevant geometries. Section 3.3 will then show, how this approach improves the performance of the code and how it compares to non-parallelized codes.

3.1. Technical details

GPU calculations were performed on a regular graphics card (Nvidia GeForce GTX 275, GT200b processor). The parallel code was developed using CUDA (v4.2) with the according Nvidia C compiler (nvcc).

They are compared with the serial, non-parallelized code, which was run on a single core of an Intel Core i7 CPU 860 2.80 GHz. The CPU code was compiled with the GNU C compiler (gcc) as well as with the commercial intel C compiler (icc). Maximum optimization (-O3 flag) was used in both cases. No significant performance differences between these compilers was evident. Since the original TEMDDD code is not optimized, we also performed additional benchmark tests with the code SLDMEM (Druskin and Knizhnerman, 1994), which is a highly optimized, serial CPU code. Both codes are based on the same numerical algorithms.

3.2. Gridding

The model discretization of a rectilinear grid strongly governs the accuracy and the efficiency of the TEMDDD code. Choosing cell sizes which are too large yields inaccurate results, while very small cells result in a large grid increasing the runtime. In our case an adequate discretization has to be found for the test model we use for illustration throughout this paper. The test model was chosen with respect to the CSEM experiment carried out during investigations at a mud volcano in the West Nile Delta (Hölz and Jegen, 2009). The mud volcano is situated at a water depth of 500 m and contains a gas chimney, which supposedly is a 3D resistive structure due to the presence of free gas that displaces conductive pore water. In the modeling grid sources and receivers are located within the conductive medium, i.e. the sea water. The required cell sizes depend on the field curvature and field strength at cell locations. The most decisive parameter for the discretization is the spacing in x/y-direction between source and receiver. A regular grid of small cell sizes would work well, but requires a huge amount of cells for large distances. Fortunately, in diffusive processes the field generally decreases with the third power over space, which makes it a suitable choice to increase cell sizes with increasing distances from the source like $\Delta x_n = c^n \Delta x_0 | n = 1, 2, \dots$. Beyond the source–receiver distance cell sizes may be increased further and a larger factor c can be used. In total, cell sizes in x/y-direction range from 5 m around the source up to several hundreds of meters at the side boundaries. Cell sizes in the z-direction also increase below and above the source and are locally refined at high resistivity contrasts like the air–water interface. A comparison of 3D calculations on the 1D test model described in Fig. 3 with (quasi) analytic 1D solutions shows that grids with sizes of $65 \times 65 \times 35$ up to $80 \times 80 \times 50$ elements and Krylov dimensions of 1500–3000 are sufficient to achieve numerically accurate results.

To skip algorithms used at the water–air interface, we replaced the originally implemented upper boundary condition by a perfect conductor. This yields Dirichlet boundary conditions at the upper boundary like it is already the case for the lateral and lower bounds of the modeling domain and the air has to be included explicitly as a resistor into the modeling domain. By doing this, the distance between the transmitter/receiver and the newly inserted perfect conductor at the upper boundary can be increased to

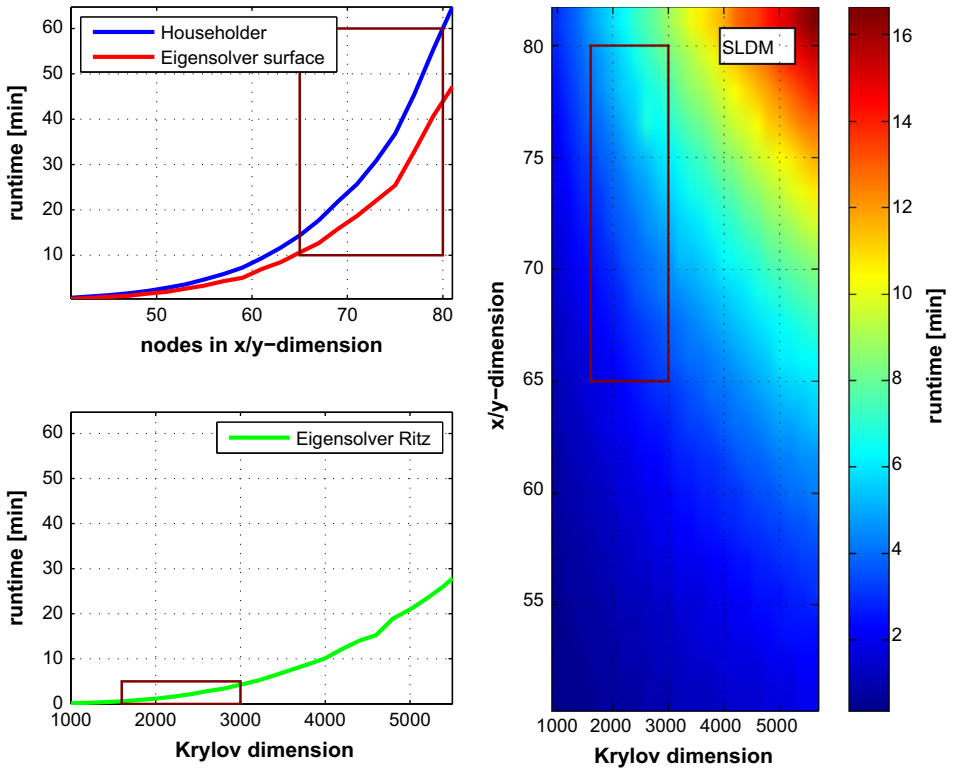

Fig. 2. Runtimes of most time consuming algorithms in original, non-parallelized TEMDDD code from Árnason (1999). Runtimes of algorithms computing surface approximation, namely the Householder tridiagonalization and its eigensolver, only depend on number of nodes in the x/y-direction (top left). Runtime of eigensolver of Ritz approximation only depends on Krylov dimension (bottom left). SLDM's runtime depends on dimension in the x/y-direction as well as on the Krylov dimension (right). Parameter bounds considered during this study are marked with brown frames. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

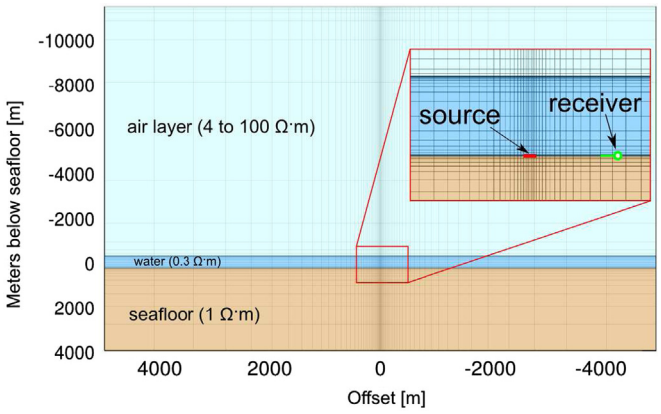

Fig. 3. Setup and grid discretization used for our test model. An air layer (cyan) is located above a 500 m thick layer of seawater (blue) and a half space representing marine sediments (beige). Air- and sea-layers are separated by a very thin gradient layer (3 cells, 20 m total thickness) in order to arrive at a smoother resistivity contrast. Source is placed in middle of grid on seafloor, where cell sizes are smallest. In our test, air layer resistivities range from 4 to 100 Ω m. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

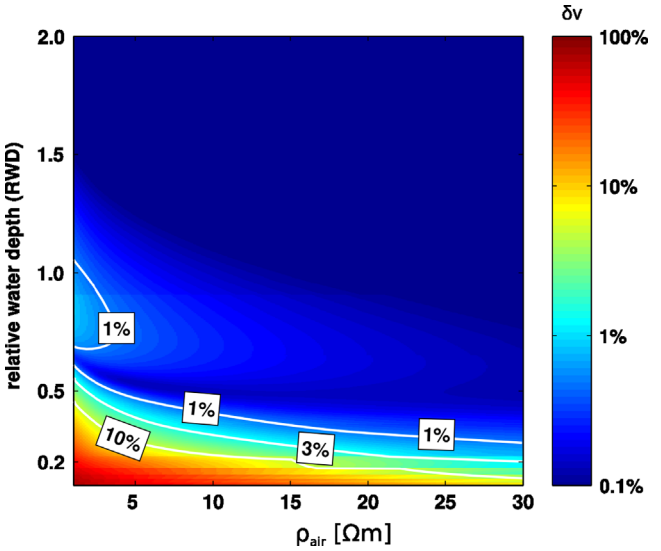

Fig. 4. Based on 1D models similar to the one used for Fig. 3, relative deviations (δv) between the 1D-analytic solutions for $\rho_{air}=0$ (reference transient) and the analytic solutions for $\rho_{air}>0$ are calculated. The relative water depth is defined by the absolute water depth normalized by the source–receiver offset. Note that for the calculation of the relative difference δv we have only used values, which fall within 95% of the DC value of the reference transient.

a point, where the effect of the boundary becomes negligible. Remember, that in our experiment, sources and receivers are placed on the seafloor (Fig. 3). Using a real resistivity of air (which is about $10^{14}\,\Omega\,\mathrm{m}$) would yield an extremely high resistivity contrast resulting in a badly conditioned system matrix, thus, requiring extremely high Krylov dimensions. To reduce the Krylov dimension and accordingly the runtime we investigate, how much the resistivity of the air can be reduced without generating significant errors. In addition, the influence of the thickness of the air layer as well as the Krylov dimension is compared between the 3D modeling results and the according analytic 1D-solutions.

Before we tried to find proper grids and numerical parameters to stabilize the 3D problem with a discretized air layer, we first

checked the validity of this approach by comparing the 1D analytic solutions of a model with $\rho_{air} \to \infty$ to solutions for models with finite ρ_{air}. Fig. 4 shows how the relative deviation depends on the air resistivity ρ_{air} and the relative water depth (RWD), which is the ratio of the water depth to the source–receiver offset. For RWD≥0.5, the deviation is below 1% for ρ_{air}≥5 Ω m. RWDs of down to 0.2 with deviations of ≤3% are possible, if the resistivity of the air layer is further increased (see Fig. 4). This shows that the approach is – at least in terms of 1D analytical calculations – valid for wide range of RWDs, if the resistivity of the air layer is chosen sufficiently high. However, it may be problematic for TX-RX geometries in shallow water (RWD≤0.2).

We then tested the approach with calculations using TEMDDD (Fig. 5). An air layer thickness of 10 km with 19 air layers turned out to be sufficient for all RWDs (not shown).

To arrive at meaningful results, we calculated deviations as χ^2-misfits, in which the differences between 1D and 3D transients are weighted with an appropriate error curve. This is done to avoid an over-weighting of small amplitudes in the transients' early times, which usually have larger error bars in real measurements. We chose an error curve proportional to $t^{-1/2}$, which is a typical characteristic of stacked transient data (Munkholm and Auken, 1996). The error reaches a source normalized level of 10^{-11} V/Am^2 at 1 s and is chosen according to the CSEM acquisitions during the WND project (comp. Fig. 10). Numerical errors can be considered small with respect to the realistic measurement errors, if the computed χ^2-misfits is much smaller than 1. In the following, we will assume results to be numerically accurate, if $\chi^2 < 0.1$.

For RWD = 1.5, the air-layer is far away from the TX-RX geometry and has little influence on calculations. Small resistivities of 1–2 Ω m and a Krylov dimension of 1500–2000 will yield accurate results. Moderate increases of the air-resistivity (4 Ω m) and the Krylov dimension (3000) are necessary for moderately shallow water applications (RWD = 0.5). For shallow water applications (RWD = 0.2) low resistivities (≤10 Ω m) do not approximate the insulating air half space well anymore. Reasonable results may be obtained by using a higher resistivity of about 20 Ω m in combination with an increased Krylov dimension of about 5000. The resulting transients (Fig. 5, bottom right) show that in this case errors are restricted to the late times of the transient, i.e. for amplitudes close to the DC-level.

A comparison with Fig. 4 shows that deviations are not necessarily due to numerical inaccuracies of TEMDDD, but are rather shortcomings of the basic assumption of a non-insulating air layer. Deviations are only problematic for shallow water applications, i.e. small RWDs, where the χ^2-misfits increases due to deviations close to the DC level at late times of transients. To circumvent this problem, one may simply exclude the later times of a transient (e.g. all amplitudes ≥80% of the DC level). This approach is feasible, if the relevant information about the subsurface structure is expected to be in the early times of the transient, but may fail for deep targets (see Constable, 2010, Fig. 13).

3.3. Speeding up the code

Avoiding the computationally expensive Householder tridiagonalization and the eigensolver involved in the calculation of the

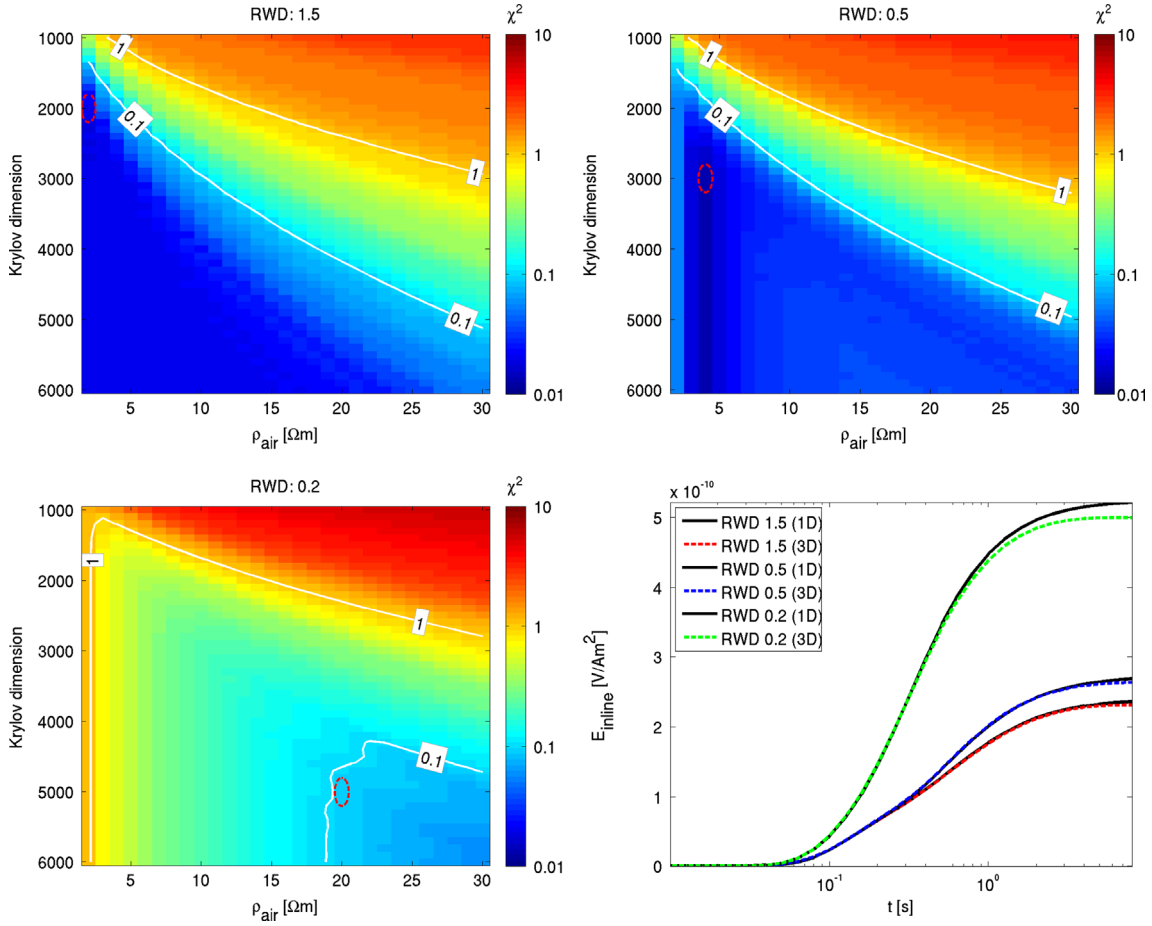

Fig. 5. χ^2 misfit between analytic 1D and 3D responses for various air layer resistivities, Krylov dimensions and relative water depths (RWD) of 1.5, 0.5, and 0.2. Example transients (bottom, right) for parameters marked with red ellipses show that deviations are usually small. They become significant only for shallow water depth (RWD=0.2) close to the DC level (green curve). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

surface matrix already leads to a significant speedup of the code. The remaining computationally expensive algorithms, namely the SLDM and the QR algorithm that is used as an eigensolver, involve many matrix–vector and matrix–matrix operations. Therefore, they are well suited for GPU parallelization.

The computationally expensive part of the SLDM is to build up the Krylov space (Eq. (4)), which comprises sparse matrix times vector multiplications and calculations of scalar products for the orthogonalization. These two operations are well parallelizable on a GPU. The dimension of **A**, which is usually on the order of several hundred thousands to a million, makes it necessary to store **A** using a sparse matrix format, such as the ELL format (see Commer et al., 2012), where its components are matched to the ones of the vector $\vec{E}_0$ (see Eq. (4)) by a matrix of indices. This has the disadvantage that the coalesced memory structure is destroyed, thus, limiting the maximum achievable performance gain by parallelization. It also obstructs the possibility to use the GPU's low latency shared memory. Instead, the much slower global GPU memory has to be used. Still, the achievable performance gain is significant, because every thread – approximately 15.000 parallel virtual threads are possible on the GPU – computes one matrix row times vector multiplication. One positive aspect of the reduced size of the sparse system matrix is that the influence of the bottleneck of data transfer between CPU and GPU is limited. Since the system matrix has to be copied just once to the graphics device, the computational cost of the data transfer is negligible. The parallelization of scalar products is well described in nearly every CUDA-programming guide, here the CUBLAS library (a BLAS library for CUDA) was used.

Generally, the runtime of the eigensolver increases by $\mathcal{O}(k^2)$ for tridiagonal matrices (Golub and Van Loan, 1996) where k is the dimension of the Ritz approximation **H** (see Eq. (5)), which is equal to the Krylov dimension. In the new version of the code the standard eigensolver is replaced by the GPU optimized QR-algorithm from the CULA (CUDA LAPACK) library (Humphrey et al., 2010).

While the total runtime of the SLDM algorithm (Fig. 2, right) depends on the Krylov dimension as well as the grid discretization in x/y-direction, the speedup (Fig. 6, left) only shows a dependency on the grid discretization. This is due to the fact that the SLDM algorithm is recursive and parallelizable operations are part of each Krylov iteration. Therefore, the number of Krylov dimension has no impact on the speedup of the SLDM. For typical grid dimensions, the speedup increases linearly up to a factor of 10. In contrast, the speedup of the QR-algorithm (Fig. 6, right) only depends on the Krylov dimension and falls in a range between 10 and 40 for Krylov dimensions between 2000 and 5000, respectively. As additional reference, we have also included a comparison to a parallelized LAPACK QR-algorithm (Fig. 6, right), which was executed in parallel on 4 CPU cores. It shows a much better performance when compared to the TEMDDD eigensolver, but is still significantly slower than the GPU implementation.

While the Krylov dimension has only a minor impact on the original code's runtime for a large number of cells in the horizontal directions, it is increasingly important for the GPU code runtime (Fig. 7, left). For models sizes of $65 \times 65 \times 50$ up to $80 \times 80 \times 50$ cells, runtimes of the original code are on the order of 2000–7000 s for Krylov dimensions of 1000 and 3000, respectively. The same calculations performed with the SLDMEM code show the superior performance of this code with runtimes of 60–180 s, respectively. For numerically accurate results of the parallelized GPU code, higher Krylov dimensions of 3000 (see Fig. 5, top) up to 5000 are necessary for shallow water applications (see Fig. 5) or 2000 for deep water. The according runtimes are 20–40 s (Krylov dimension 2000), 50–70 s (Krylov dimension 3000) and 70–130 s (Krylov dimension 5000).

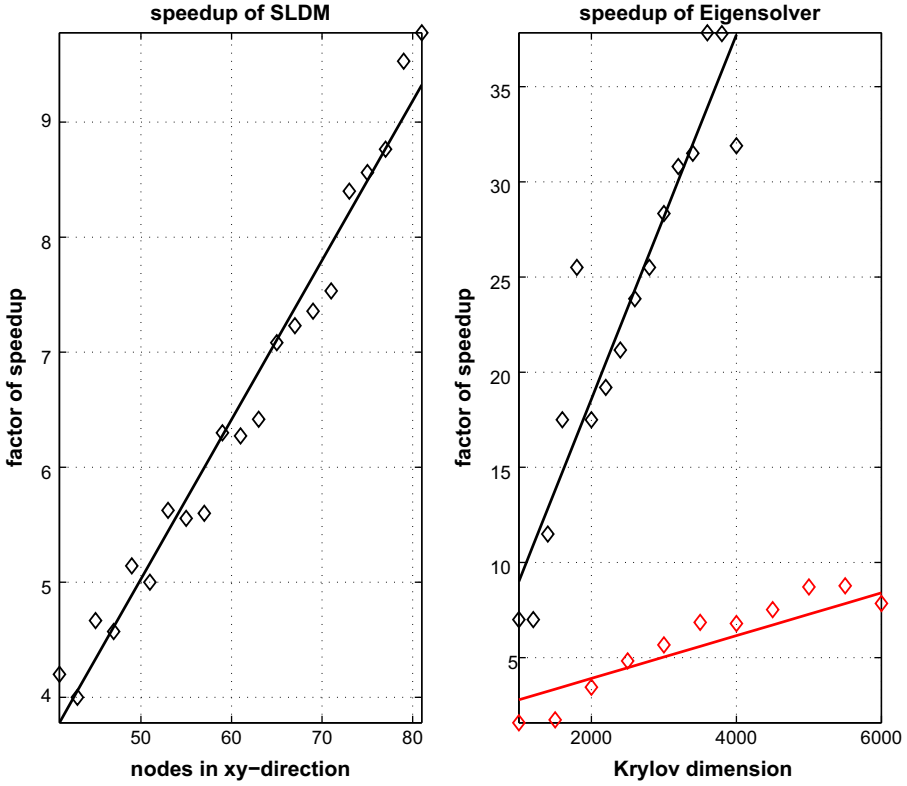

Fig. 6. Left: Speedups of SLDM calculations of GPU-version relative to original CPU-version. Right: Speedup of CULA-QR-Eigensolver with respect to original eigensolver (black) and relative to LAPACK QR-algorithm on four cores (red). A linear trend is evident for both algorithms. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

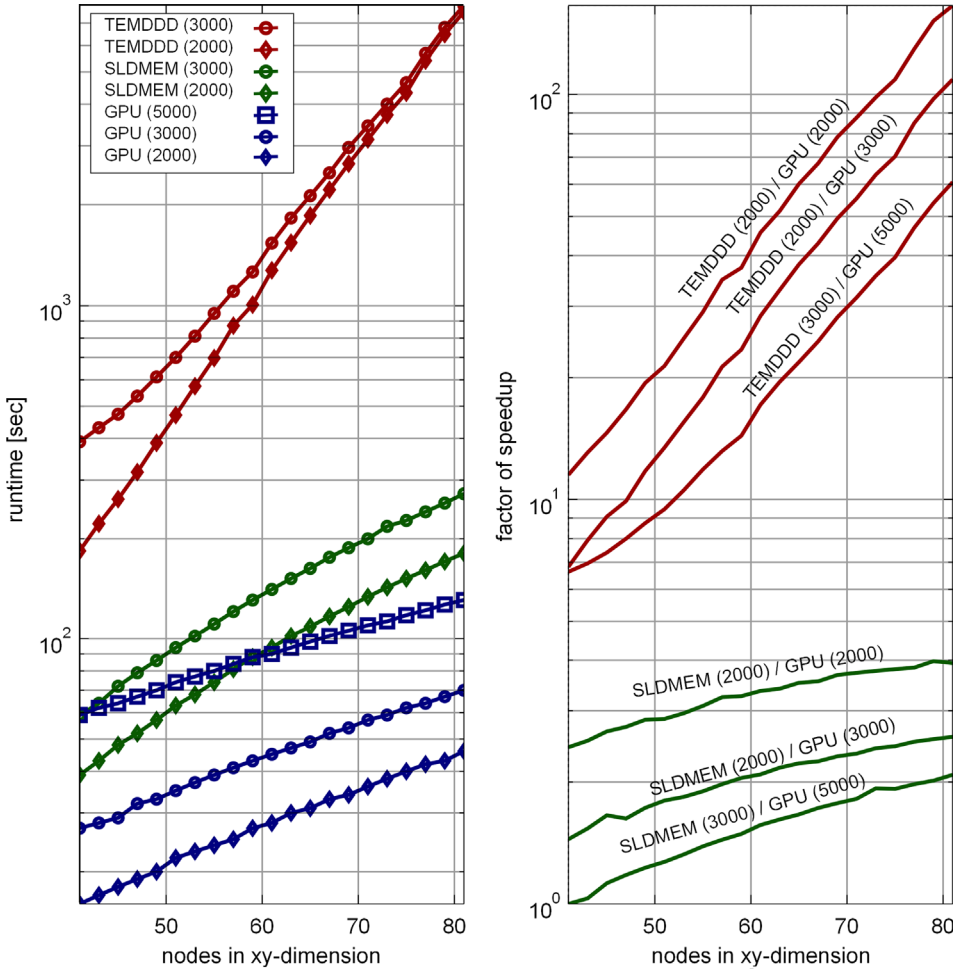

Fig. 7. Left: Total runtimes of original code (red), the modified GPU-parallelized code (blue) and the SLDMEM code (green) for different Krylov dimensions. Models for the parallelized code include additional grid cells in z-direction for the air layer (comp. Fig. 3). Right: Speedup of GPU relative to CPU codes. For calculations of speedups we have used higher Krylov dimension for the GPU-parallelized code, which is necessary only for shallow water applications (see Fig. 5). Note that all CPU calculations were carried out on a single CPU core. Therefore, additional performance gains are possible by a coarse parallelization (see Discussion). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

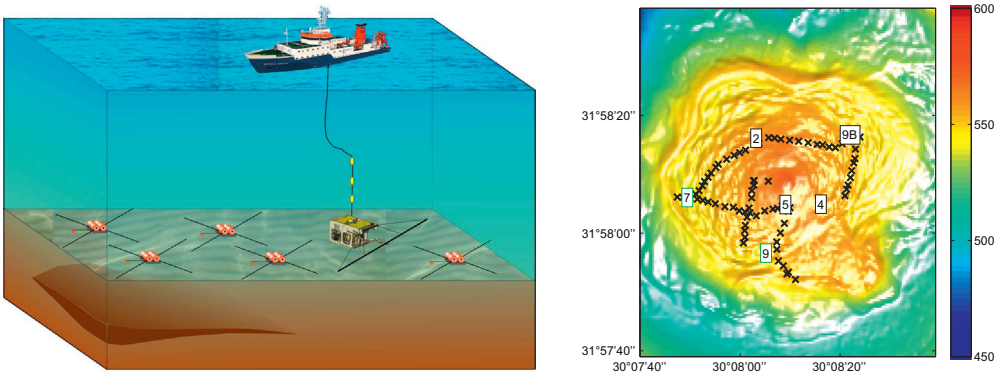

Fig. 8. Left: Design of marine CSEM-system from GEOMAR. A source field is emitted from a dipole antenna, which is mounted on an ROV. The induced secondary fields depend on the spatial conductivity distribution of the subsurface. Horizontal field components are measured at a number of receivers that are deployed at the seafloor. Right: Station map of CSEM-experiment at mud volcano offshore Egypt (Hölz and Jegen, 2009). Crosses and boxes show transmitters and receivers positions, respectively.

4. Application example

We use the GPU parallelized code to investigate the effects of bathymetry and non-plain source–receivers geometries on transient responses with respect to an MCSEM experiment conducted on a mud volcano located in the West Nile Delta (Hölz and Jegen, 2009). At this mud volcano the bathymetry varies by about 25 m within the station grid (see Fig. 8). The effect of the non-flat bathymetry is investigated for all TX-RX pairs. For each pair we generated a 3D model with a discretized bathymetry holding TX and RX at exact *xyz*-locations on the seafloor (Fig. 9) and calculated the according transient inline response. We then used the same grid with a flat bathymetry, i.e. practically a 1D model, and recomputed the inline response, this time with the TX and RX located on the same depth level. Grid cells in

the water column and the seafloor have resistivities of 0.2 Ω m and 1 Ω m, respectively. Resistivities of cells that are intersected by the seafloor are averaged. The air resistivity is set to 4 Ω m and the air thickness to 10 000 m. The model size is $63 \times 63 \times 52$. For models for which the runtime of the original code is 1440 s (Krylov dimension of 2000), the parallel code needs 59 s (Krylov dimension of 3500). Hence, computation of the models for all transmitter–receiver pairs (80 transmitters and 15 receivers) take 19 h instead of 19 days, in case of using the original code. Newer GPUs are even faster (up to a factor of 10) because they support double precision.

To quantify the effect of bathymetry and non-plain TX-RX geometry, we calculate the χ^2-misfit between the computed transients from the 1D and 3D bathymetry. Misfits are weighted by the time dependent errors of the true measurements (Fig. 10). A clear dependency between misfit and offset is apparent. Generally, for TX-RX pairs with a χ^2-misfit above 1, the differences of the 1D and 3D curves are in total larger than the error bars. For these pairs the 3D effects due to bathymetry and non-plain RX-TX geometries are significant. This in turns also means that an interpretation of these data using a 1D code will intrinsically be biased due to these effects. For the given data set, 3D effects may no longer be neglected for TX-RX pairs with offsets smaller than approximately 125 m.

5. Discussion and conclusion

The attempt to significantly reduce the runtime of the 3D time domain CSEM code TEMDDD was achieved by porting the code to a GPU. For relevant model sizes of $65 \times 65 \times 50$ up to $80 \times 80 \times 50$ cells the speedup falls in a range between 21 and 150x. The explicit discretization of the air turned out to be a proper method to avoid computationally expensive surface algorithms. The SLDM was parallelized with speedups up to a factor of 10. For the eigensolver the CULA-library has speedups up to a factor of 40. Even when using a CPU-parallelized eigensolver (CPU, 4 cores), the CULA library still outperforms it by up to a factor of 10. A comparison to the SLDMEM code shows that a great deal of the speedup can be attributed to the non-optimized structure of the original TEMDDD code. Still, the GPU code outperforms SLDMEM by a factor of 1.8–4x for relevant model sizes.

To put this into the right perspective we again stress the fact that in the results presented so far, the non-parallelized SLDMEM and TEMDDD codes were run on a single core of the CPU. We have also performed a test, where several instances of these codes were started in parallel on all available CPU cores. In this test of a coarse parallelization the best performance was achieved, if one thread was started on each core leading to a 3x speedup on four cores.

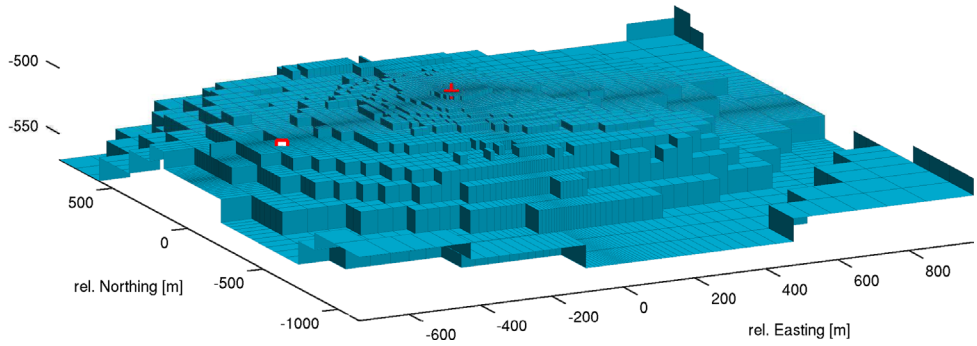

Fig. 9. 3D grid with discretized bathymetry and transmitter (red cross) located in center of grid and a receiver (red square) (comp. Fig. 8). Resistivity of water is set to 0.2 Ω m and resistivity of sediments is set to 1 Ω m. Note strong vertical exaggeration. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

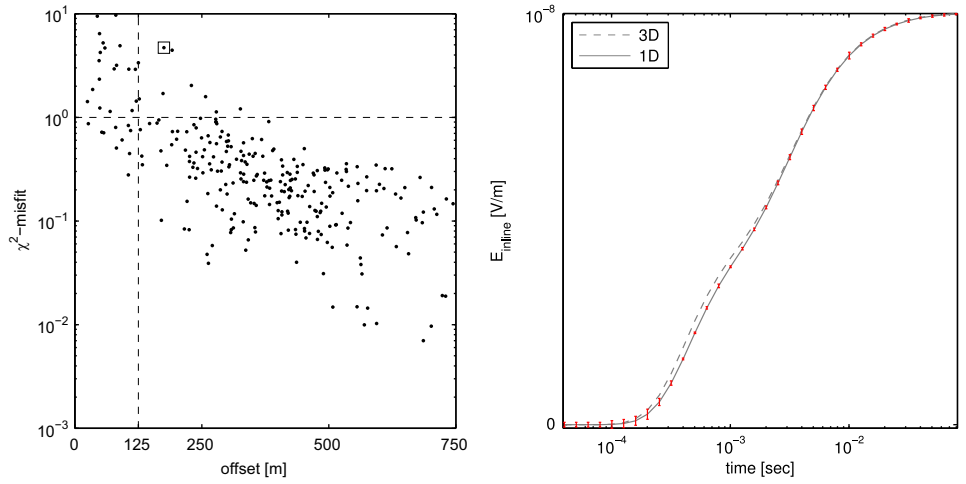

Fig. 10. χ^2-misfit versus offset for all TX-RX pairs (left). For pairs with $\chi^2 > 1$, the according 1D and 3D curves are distinct in terms of measurement errors and their measurements are, thus, significantly affected by the bathymetry. For the given data set and offsets < 125 m this is the case for the majority of station-pairs. In the right figure the transients for the pair with the highest misfit above 125 m are shown (highlighted in the left figure with a rectangle). Error bars are highlighted with red colors. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Therefore, all reported speedups should be divided by this number, which still yields massive speedups when comparing the original and GPU optized TEMDDD code (7–50x) and shows similar performances of the GPU version and a coarsely parallelized version of SLDMEM (0.6–1.3x).

Our results represent the status-quo of the hardware used during development and testing of our code. Further speedups could be obtained by using the most up to date GPU boards dedicated to scientific computing (e.g. Tesla K20 with GK110 processor). According to Nvidia, these GPU boards are about 15x faster as the currently used (GTX275 with GT200b processor) in double precision calculations, since they natively implemented double precision calculations according to the IEEE 574 standard. Furthermore, these new GPU processors also made progress in data unification between the different multiprocessors by providing efficient, high speed data sharing across the GPU, which especially should benefit sparse matrix multiplications (SpMV) (NVIDIA, 2012). Profiling of the code shows that currently 60–80% of the runtime are deduced to the SpMV operation where the memory is badly coalesced as well as recalling data from the low latency global memory. Of course the development of CPUs is also an ongoing process with up to date processors currently offering 12 (e.g. Intel i7-3970X) to 16 cores (e.g. AMD Opteron 6380), which also promises a significant potential for a coarse parallelization of the problem. Ultimately, the question which architecture offers the best performance remains an open question, which cannot be answered within the scope of this paper.

Acknowledgment

We like to thank Knútur Árnason, for allowing us to use his source-code as basis for the parallelized code. Benchmark results for the SLDMEM code were provided courtesy of the University of Toronto. We also kindly thank RWE Dea AG (Hamburg, Germany) for campaign funding during the WND project, and RWE Dea's Egypt Branch as well as their concession partner BP (operator) for good collaboration, assistance in permitting and organizational matters. Our work was supported by the SUGAR-project.

References

Árnason, K., 1999. Consistent discretization of electromagnetic fields and transient modeling. In: Oristaglio, M., Spies, B., Cooper, M. (Eds.), Three-Dimensional Electromagnetics, pp. 103–118, SEG.

Commer, M., Helwig, S., Hördt, A., Scholl, C., Tezkan, B., 2006. New results on the resistivity structure of Merapi Volcano (Indonesia), derived from three-dimensional restricted inversion of long-offset transient electromagnetic data. Geophysical Journal International 167 (3), 1172–1187.

Commer, M., Maia, F., Newman, G., 2012. Iterative Krylov solution methods for geophysical electromagnetic simulations on throughput-oriented processing units. International Journal of High Performance Computing Applications 26, 378–385, http://dx.doi.org/10.1177/1094342011428145.

Constable, S., 2010. Ten years of marine CSEM for hydrocarbon exploration. Geophysics 75 (5) 75A67–75A81.

Constable, S., Srnka, L.J., 2007. An introduction to marine controlled-source electromagnetic methods for hydrocarbon exploration. Geophysics 72 (March–April (2)), WA3–WA12, http://dx.doi.org/10.1190/1.2432483, ISSN 0016-8033.

da Silva, N.V., Morgan, J.V., MacGregor, L., Warner, M., 2012. A finite element multifrontal method for 3D CSEm modeling in the frequency domain. Geophysics 77.

Druskin, V., Knizhnerman, L., 1994. Spectral approach to solving three-dimensional Maxwell's equations in the time and frequency domain. Radio Science 29, 937–953.

Golub, G.H., Van Loan, C.F., 1996. Matrix Computations, 3rd edition John Hopkins University Press, Baltimore.

Gribenko, A., Zhdanov, M., 2007. Rigorous 3D inversion of marine CSEm data based on the integral equation method. Geophysics 72, 73–84.

Hesthammer, J., Stefatos, A., Boulaenko, M., Vereshagin, A., Gelting, P., Wedberg, T., Maxwell, G., 2010. CSEM technology as a value driver for hydrocarbon exploration. Marine and Petroleum Geology 27 (Oct.), 1872–1884.

Hölz, S., Jegen, M., 2009. Development of a CSEM system for the electromagnetic study of the North Alex Mud Volcano. In: EGU General Assembly Conference Abstracts, EGU General Assembly Conference Abstracts, vol. 11.

Hördt, A., Müller, M., 2000. Understanding LOTEM data from mountainous terrain. Geophysics 65, 1113–1123.

Householder, A.S., 1958. Unitary tridiagonalization of a nonsymmetric matrix. Journal of ACM.

Humphrey, J., Price, D., Spagnoli, K., Paolini, A., Kelmelis, E., 2010. CULA: hybrid GPU accelerated linear algebra routines. In: SPIE Defense and Security Symposium (DSS).

Komatisch, D., Erlebacher, D.G., Michèa, D., 2010. High-order finite-element seismicwave propagation modeling with MPI on a large GPU cluster. Journal of Computational Physics 229, 7692–7714.

Moorkamp, M., Jegen, M., Roberts, A.W., Hobbs, R., 2010. Massively parallel forward modelling of scalar and tensor gravimetry data. Computers & Geosciences 36 (5), 680–686.

Munkholm, M., Auken, E., 1996. Electromagnetic noise contamination on transient electromagnetic soundings in culturally disturbed environments. JEEG 1 (2), 119–127.

NVIDIA, 2012. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110. Technical Report, NVIDIA Corporation.

Oristaglio, M., Hohmann, G., 1984. Diffusion of electromagnetic fields into a two-dimensional earth: a finite-difference approach. Geophysics 49, 870–894.

Parlett, B.N., 1998. The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs, NJ. (republished by SIAM, Philadelphia).

Schwalenberg, K., Haeckel, M., Poort, J., Jegen, M., 2010. Evaluation of gas hydrate deposits in an active seep area using marine controlled source electromagnetics: results from Opouawe Bank, Hikurangi Margin, New Zealand. Marine Geology 272, 79–88.

Swidinsky, A., Hölz, H., Jegen, M., 2012. On mapping seafloor mineral deposits with central loop transient electromagnetics. Geophysics 77 (3), E171–E184.

van der Vorst, H.A., 1987. An iterative solution method for solving f(A)x=b, using Krylov subspace information obtained for the symmetric positive definite matrix A. Journal of Computational and Applied Mathematics 18.

Watkins, D.S., 1982. Understanding the QR-algorithm. SIAM 24.

Weidelt, P., 2000. Numerical modeling of transient-electromagnetic fields in three-dimensional conductors: a comparative study. Elektromagnetische Tiefenforschung, Kolloquiumsband zur Tagung in Altenberg, 216–230.

Yee, K., 1966. Numerical Solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Transactions on Antennas and Propagation AP-14, 302–307.

Yuan, J., Edwards, R., 2000. The assessment of marine gas hydrates through electronic remote sounding: hydrate without a BSR?. GRL 27 (16), 2397–2400.

Comparison of Polynomial- and Rational Krylov Subspace methods for marine CSEM on GPU

M. Sommer[a], S. Hölz[a], A. Avdeeva[b], M. Jegen[a]

[a] *GEOMAR, Helmholtz Centre for Ocean Research Kiel, Wischhofstr. 1-3, 24148 Kiel, Germany*
[b] *University of Leicester, Department of Geology, University Road, Leicester, LE1 7RH, UK*

Abstract

High run-times of 3D time domain controlled source electromagnetics (CSEM) modeling codes are one of the main problems for inversion. Run-time depends on the choice of algorithms and hardware. We solve the electromagnetic diffusion equation implicitly with an exponential Ansatz function, that is spectrally decomposed. The spectra of the finite difference discretization of the spatial operators are approximated by Krylov subspace methods. We have implemented a rational Krylov subspace (RKS) algorithm on Graphics Processing Units (GPU) and compared it with the implementation of a polynomial Krylov subspace method on GPU. We demonstrate the benefits of RKS-algorithms over polynomial ones. We show the benefits of GPUs over CPUs and compare older GPU generations with modern ones to reveal the progress in hardware. A speedup of 20 has been achieved, compared to the polynomial method on an old GPU. We demonstrate the applicability of the code for different conductivity contrasts and for a 3D model.

Keywords: Rational Krylov Subspace, GPU, marine CSEM

1. Introduction

For the investigation of the spatial distribution of sub-seafloor resistivity, marine controlled source electromagnetic (CSEM) methods are used. Typical exploration targets of marine CSEM are resistive hydrocarbons [1], gas hydrates [2, 3], complex salt diapirs [4] and conductive targets like massive sulphide deposits [5]. In general, a marine CSEM survey consists of a source generating an electric field and several antennas receiving the perturbed electromagnetic field. A common setup is to tow the sources behind a vessel [6] or mount them on a remotely operated vehicle [7] or submersibles [8]. The receivers can either be deployed on the seafloor [6] or be part of a streamer moving along the seafloor [9].
By the perturbation of the received electric field, the sub-seafloor resistivity structure can be reconstructed. This is typically done by modeling or inversion. Considerable research is spent on the problem of how to numerically implement

3D CSEM forward and inversion codes in such a way that the computational costs are reduced. In this process, two parallel developments can be observed: changes in mathematical algorithms and changes in hardware, both are always dependent on each other.
Even for state-of-the-art time-stepping codes, such as [10], run-time is high. Since the introduction of Lanczos-algorithms to CSEM [11], computationally expensive time stepping algorithms introduced in [12] became more and more unpopular in CSEM. Unlike in time-stepping, solutions do not depend on solutions of previous times. Because the results are always represented on a logarithmic time axis, this gives a computational benefit. While the original formulation of Lanczos-algorithms is based on polynomial Krylov Subspace, recent developments have been focused on the implementation of more efficient rational Krylov Subspace formulations, see [13, 14, 15, 16, 17, 18], among others.
Along with a shift in algorithms, hardware has also changed over the years. While increases in processor frequencies have practically come to a halt, the speed of floating point operations is boosted through parallelization. One common way to achieve this parallelization is to use graphics processing units (GPUs) on graphics cards, which offer a massively parallel architecture. However, this requires the adaptation of existing or development of new algorithms and codes, which make proper use of the GPU hardware.
In this work, we present an implementation of the rational-Krylov-subspaces-method (RKSM) on a GPU for 3D marine time domain CSEM calculations. We show that the rational method improves performance in comparison to the polynomial method, which we have previously presented in [19]. Furthermore, we investigate to what extent these performance gains can be attributed to the algorithm and to what extent they are driven by the improvements of the utilized hardware. This hardware comparison will use an older graphics card (Nvidia GTX275), which only emulates support for double precision, and a more recent graphics card (Nvidia Titan Black), which offers native hardware support for double precision arithmetics.

2. Theory

Starting from the physical problem (Section 2.1), we describe a solution strategy (Section 2.2), which is similar for both, the polynomial- and rational algorithm. It differs only in the way how the approximating subspace is spanned. The solution strategy is then applied to the polynomial and to the rational implementations (Section 2.3 and 2.4, respectively).

Conventions:
The letters j, k, n are used for integers, scalars are denoted by small letters like $t, \alpha, \ldots,$ vectors are indicated by an arrow above: $\vec{q}, \vec{E}$ and matrices are all denoted in bold, capital letter like: $\mathbf{A}, \mathbf{Q}$. The hat $\hat{\cdot}$ denotes vectors, that are not orthogonalized and the dash $\cdot'$ denotes non-normalized vectors. The letter i stands for the imaginary unit $\sqrt{-1}$.

2.1. Formulation of the physical problem

The physics of the marine CSEM problem is described by the electromagnetic diffusion equation in the quasi-static approximation

$$\vec{\nabla} \times \vec{\nabla} \times (\sigma\mu)^{-1}\vec{E}(\vec{r},t) = -\frac{\partial \vec{E}(\vec{r},t)}{\partial t} \tag{1}$$

with the initial values $\vec{E}(\vec{r}, t = 0) = \vec{E}_0$, the isotropic electric conductivity σ, the electrical permittivity μ, which is set to the permeability of free space and $\vec{E}$ the electric field within the modeling domain. We will write $\vec{E} = \vec{E}(\vec{r}, t)$ in the following. The source field is described by a Heaviside function, switched on at $t = 0$, such that $\vec{E}_0$ has a constant value at the source location and zero in the remaining modeling domain. The modeling domain is embedded in a perfect conductor, such that the electric field vanishes at the boundaries and Dirichlet boundary conditions are enforced. At the water air-interface, $\vec{E}$-field components immediately above the water are located within the air where $\sigma \to 0$ such that eq. (1) can not be applied. It is therefore common to solve the Laplace problem $\vec{\nabla}^2\vec{E} = 0$ in the air instead. However, in the marine case it is also possible to estimate the air layer by a finite layer of elevated resistivity, which can be beneficial in terms of computational speed as pointed out by [19].

2.2. Numerical solution strategy

To solve eq. (1) numerically, the spatial operators are discretized by central finite differences (FD) on a rectilinear, staggered grid [20] and σ is averaged, such that $\vec{\nabla} \times \vec{\nabla} \times (\sigma\mu)^{-1} \to \mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{A}$ is a sparse, symmetric matrix. Then eq. (1) becomes

$$\mathbf{A}\vec{E}^{FD} = -\frac{\partial \vec{E}^{FD}}{\partial t}, \tag{2}$$

where $\vec{E}^{FD} \in \mathbb{R}^n$ is the vector storing the electric field components on the edges of the grid and n is the number of edges. For more details see [21].

The system of partial differential equations is now approximated by a system of ordinary differential equations, which can be solved either explicitly by also discretizing the time variable and using a time stepping scheme, or implicitly by an Ansatz function. In the time stepping approach a full space solution is required at every time step. The steps need to be small enough to suffice the Courant criteria [22]. In diffusive problems, only small changes occur at late times, therefore transients are commonly represented in a logarithmic time space. But time steps do not increase logarithmically such that the transients become oversampled for late times. Implicit methods, where the solution does not depend on the full space solution of any previous time, avoid this problem. The exponential Ansatz function was chosen as described in [11] and [21]:

$$\vec{E}^{FD}_{Ansatz} = exp(-t\mathbf{A})\vec{E}^{FD}_0 = \sum_{j=1}^{n} \vec{z}_j exp(-t\lambda_j)\vec{z}_j^T\vec{E}^{FD}_0. \tag{3}$$

In eq. 3, $\lambda_j \in \mathbb{R}$ are the eigenvalues, and $\vec{z}_j \in \mathbb{R}^n$ are the eigenvectors of $\mathbf{A}$. Since $\mathbf{A}$ is a large matrix, computation of the eigenpairs is very expensive and memory consuming. Therefore, $\mathbf{A}$ is approximated by the Ritz matrix $\mathbf{H} \in \mathbb{R}^{k\times k}$:

$$\mathbf{H} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}. \tag{4}$$

where $\mathbf{Q} = [\vec{q}_0, \ldots, \vec{q}_{k-1}] \in \mathbb{R}^{n\times k}$ is a subspace projection of dimension k. Ritz-values and Ritz-vectors of matrix $\mathbf{H}$ are denoted by $\theta_j \in \mathbb{R}$ and $\vec{\psi}_j \in \mathbb{R}^k$, $(j = 0, \ldots, k-1)$, respectively, and satisfy $\mathbf{H}\vec{\psi}_j = \theta_j\vec{\psi}_j$. The Ritz-values θ_j approximate the eigenvalues of $\mathbf{A}$ and the vectors $\mathbf{Q}\vec{\psi}_j$ in turn approximate corresponding eigenvectors. Then eq. (3) becomes the Galerkin approximation [23], in other words the solution is approximated by a superposition of weighted base functions:

$$exp(-t\mathbf{A})\vec{E}_0^{FD} \approx \sum_{j=0}^{k-1} \mathbf{Q}\vec{\psi}_j exp(-t\theta_j)\vec{\psi}_j^T\mathbf{Q}^T\vec{E}_0^{FD} = \vec{E}_{Galerkin}^{FD} \tag{5}$$

Since $k \ll n$ the problem can be solved with much lower computational cost. Two classes of algorithms, namely the Polynomial and Rational Krylov subspace methods, are commonly used to define the subspaces related to $\mathbf{Q}$.

2.3. The Polynomial Krylov Subspace Method

Since $\mathbf{A}$ is symmetric, $\mathbf{Q}$ (eq. 4) can be evaluated by the Spectral Lanczos Decomposition Method (SLDM) [24, 25].
For the approximating subspace one may choose a polynomial Krylov subspace [26], which is defined as:

$$\mathcal{K}_{poly}^k(\mathbf{A}, \vec{q}_0) = span(\mathbf{A}^0\vec{q}_0, \mathbf{A}^1\vec{q}_0, \mathbf{A}^2\vec{q}_0, \ldots, \mathbf{A}^{k-1}\vec{q}_0) \tag{6}$$

where the vectors of the spanning set are called Krylov vectors and the matrix composed of those vectors as columns is called the Krylov matrix. The vector $\vec{q}_0 \in \mathbb{R}^n$ can be chosen as an arbitrary vector and we set $\vec{q}_0 = \vec{E}_0^{FD}/\|\vec{E}_0^{FD}\|$. An elegant and clear explanation, why Krylov spaces are so powerful for approximating eigenspaces is given in [27].
A Gram-Schmidt-orthogonalization of eq.6 can be found by the recursion formula

$$\vec{q}_{j+1}^{'} = \mathbf{A}\vec{q}_j - (\vec{q}_j^T\mathbf{A}\vec{q}_j)\vec{q}_j - (\vec{q}_{j-1}^T\mathbf{A}\vec{q}_j)\vec{q}_{j-1} \tag{7}$$

which can be understood by Fig. 1.

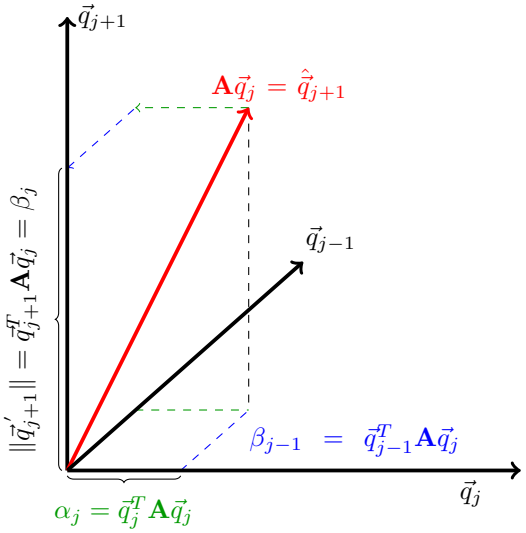

Figure 1: Graphical visualization of Gram-Schmidt-orthogonalization step in SLDM. New Krylov-vector $\mathbf{A}\vec{q}_j$ gets orthonormalized to previous, orthonormalized Krylov-vectors $\vec{q}_0, \ldots, \vec{q}_j$.

Spanning a new Krylov-vector $\hat{\vec{q}}_{j+1} := \mathbf{A}\vec{q}_j$ and (scalar) multiplying it by $\vec{q}_j$ gives the projection of $\hat{\vec{q}}_{j+1}$ on $\vec{q}_j$. We define it as $\alpha_j = \vec{q}^T_j\mathbf{A}\vec{q}_j$, such that $\mathbf{A}\vec{q}_j - \alpha_j\vec{q}_j$ becomes orthogonal to $\vec{q}_j$, because $\|\vec{q}_j\| = 1$. In the same way, $\hat{\vec{q}}_{j+1}$ can be orthogonalized to $\vec{q}_{i-1}$ with $\beta_{j-1} = \vec{q}^T_{j-1}\mathbf{A}\vec{q}_j$ resulting in eq. (7). Normalizing gives $\vec{q}_{j+1} = \vec{q}^{'}_{j+1}/\|\vec{q}^{'}_{j+1}\|$. The norm $\|\vec{q}^{'}_{j+1}\|$ is the projection of $\hat{\vec{q}}_{j+1}$ on $\vec{q}_{j+1}$ such that $\|\vec{q}^{'}_{j+1}\| = \vec{q}^T_{j+1}\mathbf{A}\vec{q}_j = \beta_j$ (it can be shown that $\vec{q}^T_{j+1}\mathbf{A}\vec{q}_j = \vec{q}^T_j\mathbf{A}\vec{q}_{j+1}$).
Equation (7) can be written in matrix notation as $\mathbf{AQ} \approx \mathbf{QH}$, where $\mathbf{H}$ is a tridiagonal matrix with $(\alpha_j, \quad j = 0, \ldots, k-1)$ on the main diagonal, and $(\beta_j, \quad j = 0, \ldots, k-2)$ on the sub- and super-diagonal. We set $\beta_0 = \sqrt{\|\mathbf{A}\vec{q}_0\|^2 - \alpha_0^2}$, hence this guaranties orthonormality of $\vec{q}_0$ and $\vec{q}_1$. The Ritz approximation of $\mathbf{A}$ is then given by $\mathbf{QHQ}^T$, and $\mathbf{Q} = [\vec{q}_0, \ldots, \vec{q}_{k-1}]$ is called Lanczos base.

2.4. The Rational Krylov Subspace Method

In [27] Ruhe showed, that the optimal subspace projection to approximate eigenspaces is the rational Krylov subspace (RKS)

$$\mathcal{K}^k_{rat}(\mathbf{A}, \vec{q}_0, \vec{s}) = span\left\{(\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_0, \ldots, \prod_{j=1}^{k}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_0\right\}. \qquad (8)$$

The corresponding RKS-algorithm is similar to the Polynomial Lanczos Algorithm. However, instead of spanning the Krylov vectors by matrix vector multiplications, they are found by computing the rational coefficient $(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_0$

with a linear solver, for a given set of poles $\vec{s} \in \mathbb{R}^k$.
Most RKS-algorithms differ in the way of computing these poles. An overview for different pole selection strategies is given in [28]. The necessary dimension k strongly depends on an optimal choice of these strategies.
In [29] it is stated that the error of the approximation in eq. (5) satisfies the inequality

$$\sqrt{\int_0^\infty \|\vec{E}^{FD}_{Ansatz} - \vec{E}^{FD}_{Galerkin}\|^2 dt} \leq \sqrt{\frac{2}{\pi}} c \min_{\lambda_1,\ldots,\lambda_k} \frac{\max_{\lambda\in[\lambda_{min},\lambda_{max}]} \mid r_k(\lambda) \mid}{\min_{s\in i\mathbb{R}\cup\infty} \mid r_k(s) \mid} \tag{9}$$

with $\vec{E}^{FD}_{Ansatz}$ and $\vec{E}^{FD}_{Galerkin}$ from eq. (3) and eq. (5), respectively. The variables c and r_j are:

$$c = \sqrt{\int_{-i\infty}^{+i\infty} \left| \frac{1}{\lambda_{\min} + z} \right|^2 \cdot \mid dz \mid}, \qquad r_j(z) = \prod_{l=1}^{j} \frac{z - \lambda_l}{z + s_l} \tag{10}$$

and j being the current Krylov-iteration ($j = 2, \ldots, k-1$), s_1 and s_2 are computed by eq.(12). Here, $\lambda_0, \ldots, \lambda_j$ are the eigenvalues of the Ritz approximation of the current iteration ($\mathbf{H}_j = \mathbf{Q}_j \mathbf{A} \mathbf{Q}_j^T$ with $\mathbf{Q}_j = [\vec{q_0}, \ldots, \vec{q_j}]$). The approximation error of eq. (5), which depends on the choice of poles $\vec{s}$ becomes small for a choice of poles, that minimizes the right hand side of eq. (9). It can be shown ([16]), that the poles become real when $\mathbf{A}$ is symmetric. Then eq. (9) simplifies, such that the poles can recursively be found by

$$s_{j+1} = \arg\max_{s\in[s_j,s_2]} \frac{1}{\mid r_j(s) \mid} \tag{11}$$

As the initial search interval we choose $[\lambda_{min}, \lambda_{max}]$, since all poles lie within this interval. We find this interval using the Gershgorin circle theorem [30], such that

$$s_1 = \lambda_{min} = \frac{\pi^2}{h^2_{max} \cdot \sigma_{max}}, \qquad s_2 = \lambda_{max} = \|\mathbf{A}\|_\infty \tag{12}$$

with $\|\cdot\|_\infty$ being the infinity norm, h_{max} being the maximum size of the model domain and σ_{max} being the maximum conductivity. The Gershgorin cycle is dependent on the choice of a diagonal element. For s_2, the row of $\mathbf{A}$ with the largest diagonal element has been chosen, and for s_1, the smallest possible diagonal element.

3. Implementation

We describe the implementation of the SLDM (Section 3.1) and RKSM (Section 3.2) separately, to elucidate the differences. The implementation of the Conjugate Gradient (CG) algorithm, required for RKSM, and its difficulties in

preconditioning will be explained in section 3.2.1, because it constituted the highest run-time in RKSM.
To obtain the system matrix $\mathbf{A}$, the operator $\nabla \times \nabla \times (\mu\sigma)^{-1}$ is discretized. This discretization is described in [19] and will not be explained here.

3.1. Implementation of SLDM on GPU

We give a short account of the SLDM implementation with details to be found in [19].
Corresponding to eq. (7), the SLDM can be written as pseudo code. The matrix $\mathbf{Q}$ changes size such that $\mathbf{Q}_j \in \mathbb{R}^{n\times j}$.

Algorithm SLDM Given: $\mathbf{A}$, $\vec{E}_0^{FD}$, k

1. $\vec{q}_0 = \vec{E}_0^{FD}/\|\vec{E}_0^{FD}\|$, $\mathbf{Q}_1 = \vec{q}_0$
2.a $\hat{\vec{q}}_1 = \mathbf{A}\vec{q}_0$
2.b $\alpha_0 = \vec{q}_0^T \cdot \hat{\vec{q}}_1, \quad \beta_0 = \sqrt{\hat{\vec{q}}_1^T \cdot \hat{\vec{q}}_1 - \alpha_0^2}$
2.c $\vec{q}_1 = (\hat{\vec{q}}_1 - \alpha_0\vec{q}_0)/\beta_0$
2.d $\mathbf{Q}_2 = [\mathbf{Q}_1 \quad \vec{q}_1]$
3. **for** $j = 1, \ldots, k-2$
3.a $\hat{\vec{q}}_{j+1} = \mathbf{A}\vec{q}_j$
3.b $\alpha_j = \vec{q}_j^T \cdot \hat{\vec{q}}_{j+1}$
3.c $\beta_j = \sqrt{\|\hat{\vec{q}}_{j+1}\|^2 - \alpha_j^2 - \beta_{j-1}^2}$ (Pythagoras in Fig. 1)
3.d $\vec{q}_{j+1} = (\hat{\vec{q}}_{j+1} - \alpha_j\vec{q}_j - \beta_{j-1}\vec{q}_{j-1})/\beta_j$
3.e $\mathbf{Q}_{j+2} = [\mathbf{Q}_{j+1} \quad \vec{q}_{j+1}]$
end

In step 1., the normalized initial E-field $\vec{E}_0^{FD}$ is used as initial vector $\vec{q}_0$ of the Krylov subspace. The second step is needed to initialize the following iteration, which is implemented as a loop in step 3. In 3.a the next Krylov vector is spanned by a sparse matrix-vector multiplication. This step constitutes the computationally most expensive part of the whole SLDM. Steps 3.b to 3.d require two vector-vector products as well as several scalar-vector products and additions, which are all computationally cheap. Step 3.d performs the Gram-Schmidt orthonormalization, 3.e appends the orthonormal Krylov vectors onto the columns of $\mathbf{Q}$.
The handling of sparse matrix-vector mulitplications (SpMV) on GPUs, like in steps 2a and 3a ($\mathbf{A}\vec{q}_j$), is a current research topic by itself (e.g. [31]). The computational run-times depend on the access to computer memory, that in turn is strongly affected by the way sparse matrices are stored. We use the ELL format (named after ELLPACK [32]). It will be explained by the following example:

$$\underbrace{\begin{pmatrix} 1 & 0 & 4 & 0 \\ 0 & 0 & 7 & 3 \\ 6 & 4 & 0 & 1 \end{pmatrix}}_{sparse\ matrix} \Rightarrow \underbrace{\begin{pmatrix} 1 & 3 & * \\ 3 & 4 & * \\ 1 & 2 & 4 \end{pmatrix}}_{column\ indices}, \underbrace{\begin{pmatrix} 1 & 4 & * \\ 7 & 3 & * \\ 6 & 4 & 1 \end{pmatrix}}_{values} \tag{13}$$

A sparse matrix is decomposed into a matrix storing integers of column indices in the same row as in the original matrix and a matrix of floating point numbers, storing the actual values of the sparse matrix. So the first row of the index matrix contains 1 and 3, because the non-zero values of the sparse matrix, namely 1 and 4, are in the first and third column. The non-zero elements are stored in the matrix of floating point values. The ELL format is appropriate, since every thread in the SpMV operation has only to deal with one row and can sum up the elements of one row. However, the vector with which the matrix is multiplied is problematic. The column index in every thread equals the index of vector elements. Every thread, of every streaming-processor, has access to those elements. Therefore, this vector has to remain in global memory, which typically has a higher latency.

3.2. Implementation of RKSM on GPU

In principle, the RKS-algorithm is similar to the Polynomial Lanczos algorithm, but instead of spanning the Krylov vectors by matrix-vector multiplications, they are spanned by shifting and inverting $\mathbf{A}$ (see eq. (8)). The matrix $\mathbf{Q}$ is successively populated with column vectors $\vec{q}_j$ such that $\mathbf{Q}_j \in \mathbb{R}^{n\times j} \mid j = 1,\ldots,k$ and $\mathbf{Q}_j = [\mathbf{Q}_{j-1} \quad \vec{q}_j]$.

Algorithm RKS Given: $\mathbf{A}$, $\vec{E}_0^{FD}$, k

1. compute initial poles $s_1, s_2 \in \mathbb{R}$ (see. eq. (12))

2.a $\vec{q}_0 = \vec{E}_0^{FD}/\|\vec{E}_0^{FD}\|$

2.b $\vec{q}_1^{\,'} = (\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_0$, $\vec{q}_1 = \vec{q}_1^{\,'}/\|\vec{q}_1^{\,'}\|$, $\mathbf{Q}_1 = \vec{q}_1$

3. **for** $j = 1,\ldots,k-1$

3.a compute next pole s_{j+1} by eq. (11) (exept s_2, because of step 1.)

3.b $\hat{\vec{q}}_{j+1} = (\mathbf{A} - s_{j+1}\mathbf{I})^{-1}\vec{q}_j$

3.c **for** $l = 1,\ldots,j$ (orthogonalization $\hat{\vec{q}}_{j+1} \to \vec{q}_{j+1}^{\,'}$)

$\hat{\vec{q}}_{j+1} = \hat{\vec{q}}_{j+1} - \vec{q}_l(\hat{\vec{q}}_{j+1}^{T}\vec{q}_l)$

end

$\vec{q}_{j+1}^{\,'} = \hat{\vec{q}}_{j+1}$

3.d $\vec{q}_{j+1} = \vec{q}_{j+1}^{\,'}/\|\vec{q}_{j+1}^{\,'}\|$, $\mathbf{Q}_{j+1} = [\mathbf{Q}_j \quad \vec{q}_{j+1}]$

end

The initial poles in step 1. are computed according to eq. (12). The first Krylov vector is computed by shifting and inverting $\mathbf{A}$ with the first pole s_1 and multiplying this inverse matrix with $\vec{E}_0^{FD}$ (step 2.b). This can be achieved with a linear solver. We choose a linear solver based on the Conjugate Gradient (CG) algorithm, which we have implemented on the GPU (see below). Step 3. loops over the remaining $k-1$ dimensions of the Krylov subspace. In every iteration, a new pole is first computed with eq. (11). To achieve this, the current interval $[s_j, s_2]$ is partitioned in a finite sequence $s_j = \xi_1 < \xi_2 < \cdots < \xi_{1000} = s_2$ with 1000 test-poles, for which eq. (11) is evaluated. The test pole ξ_l, which maximizes eq. (11) is chosen as the next pole s_{j+1}. For unknown eigenvalues, those of the current Ritz matrix, which is calculated from $\mathbf{H}_j = \mathbf{Q}_j\mathbf{A}\mathbf{Q}_j^T$ with $\mathbf{Q}_j = [\vec{q}_o,\ldots,\vec{q}_j]$, are used for eq. (11). For further details see [15], [16].
Generally, it is advisable to reduce memory transfer between the graphics card

and CPU domain and, thus, to implement all steps of an algorithm on a GPU. Data are copied to the GPU before initializing the RKS algorithm. However, in our case we had to apply the serial eigensolver to $\mathbf{H}_j$, which is copied onto the CPU memory every iteration, because a full eigensolver is difficult to implement on GPU. Therefore, further improvements of our GPU implementation are still possible.
In step 3.b, similarly to initial step 2.b, the next Krylov vector is spanned by using a CG algorithm. Step 3.b is the computationally most expensive part of the RKS-algorithm. The orthogonalization in step 3.c differs significantly from SLDM. Since the RKS-algorithm does not provide a tridiagonal Ritz approximation, every new Krylov vector has to be orthogonalized to the complete previous basis. However, the Krylov dimension of the RKS-algorithm is much smaller than for SLDM. Therefore, the application of a serial, dense eigensolver on the Ritz matrix does not cause a significant performance loss. In 3.d, the current Krylov vector is normalized and concatenated with $\mathbf{Q}$.

3.2.1. Implementation of Conjugate Gradient Method on GPU

The Conjugate Gradient (CG) method is commonly used as a linear solver for large, sparse, symmetric, positive-definite matrices. However, since it is the computationally most expensive part of our RKS implementation (steps 2.b and 3.b of Alg. RKS), the CG algorithm and its parallelization is described in this publication, mainly focusing on the performance aspects.
Our implementation follows the description in Section 10.2.6 of [33]. We have implemented it such that data on the graphics card are passed to the function without memory transfer. Vector additions and scalar products were implemented with the CUBLAS library, the blas-library offered by Nvidia CUDA. The computationally most expensive part is the SpMV operation. It was implemented with the CUSPARSE library, offered by Nvidia as well.
Since the condition number of matrix $\mathbf{A} - \mathbf{s_j I}$ can be large (up to 10^{13}) it is important to use the preconditioned CG method. However, no appropriate preconditioner is readily available on GPU. Implementation of state-of-the-art preconditioners, such as ones based on Algebraic Multigrid (AMG) methods [34], on GPU is a difficult task, and we leave such work out of scope of this paper. Finding the best preconditioner for our problem and implementing it on GPU will be a subject of our future research. For this paper we therefore resorted to invert our matrices without a preconditioner, obtaining good results as shown in the next section.

4. Numerical tests of accuracy and run-time

We demonstrate the applicability and improvements of the code by a simple 1D model. First, we investigate the accuracy of the code in dependence of conductivity contrasts, Krylov dimension and CG adjustment (Section 4.1). For proper parameters found in this way, run-time is benchmarked (Section 4.3) and compared with run-times of the SLDM based code and the *sldmem* code on and old and new graphics card.

4.1. Numerical Accuracy

As discussed in the previous section, we omit preconditioning. We now show, that low run-times and accurate solutions can be achieved anyway by an appropriate choice of parameters.
Higher Krylov-dimensions (N_{Krylov}) and a high value for the maximum number of iterations for CG (N_{CG}^{max}) increase the accuracy, yet also the run-time. The required N_{CG}^{max} is dependent on the condition number and therefore on the conductivity contrasts within the model.

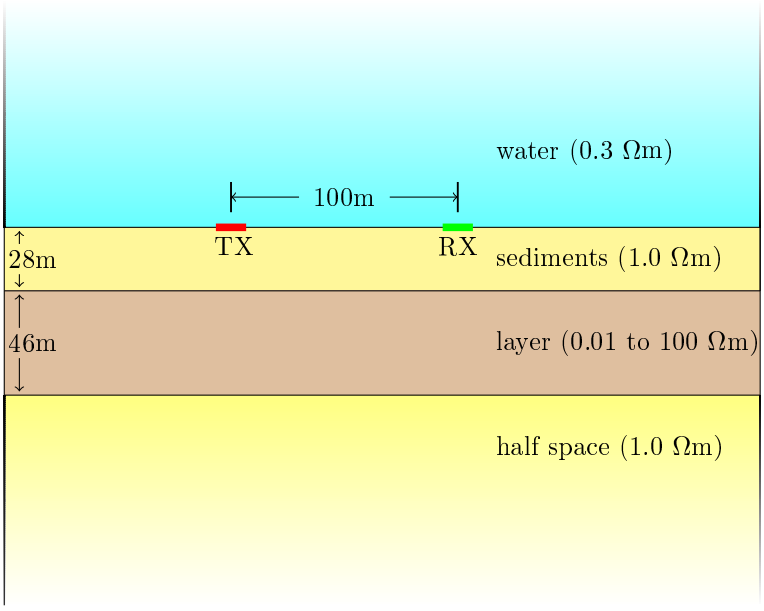

Figure 2: Model used for benchmarks. Two inline, horizontal dipoles (TX: transmitter, RX: receiver) are placed on a homogeneous seafloor (1Ωm), in which an anomalous layer of varying resistivity ($0.01 - 100\Omega$m) and a thickness of 46m is embedded at a depth of 28m.

We begin with an example 1D-model which includes a layer of varying resistivity from 0.01 to 100 Ωm (Fig. 2). For the marine case, those values are realistic. The water depth was set to a large value (1000 m), such that problems

with modeling the conductivity contrast at the air-water interface, as described in [19], can be neglected. The source-receiver offset was set to 100m. An anomalous layer is hosted in an 1Ωm half space (typical for salt water saturated sediments) at a depth from 28m to 74m (values chosen according to cell spacing). This model is discretized on a rectilinear grid with a polynomial cell size increase. Its size is $87 \times 87 \times 41$.
We chose a 1D model, such that the response can easily be verified by the quasi analytic solution of a 1D code. To compare the results, we computed the χ^2-misfit, where the differences between 1D- and 3D-results are weighted with an appropriate error. Over-weighting of small amplitudes at early times was avoided by choosing an error curve proportional to $t^{-\frac{1}{2}}$ [35]. A source normalized error level of 10^{-11}V/Am^2 at 1s was chosen according to the CSEM acquisition during the WND project [7].

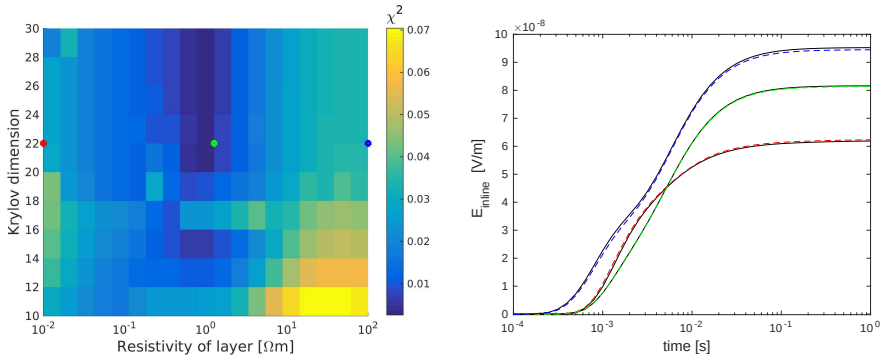

Figure 3: Left: χ^2 misfit between 3D and 1D responses for an anomalous layer with resistivities between 0.01 to 100Ωm (compare Fig. 2) and a Krylov dimension from 10 to 28. The maximum value of CG-iterations was set to 5000. Right: Corresponding transients for the 1D model with anomalous layer resistivities of 0.01, 1.0 and 100Ωm, plotted with the same color as the corresponding dot in the left plot. Black lines show the associated responses calculated with a 1D code.

First, we check the influence of N_{Krylov} on the numerical accuracy for different resistivity contrasts between the layer and halfspace as depicted in Fig. 3. To focus on the influence of N_{Krylov} alone we fixed the value for N_{CG}^{max} to a high value from 5000 down to 100 in this test. It can easily be seen, that the accuracy does not increase anymore for the N_{Krylov} above 22 for all models. For low resistivity contrasts, between 0.1 to 10 Ωm, a dimension of even 14 is sufficient. For all following tests, the N_{Krylov} was set to a constant value 22.

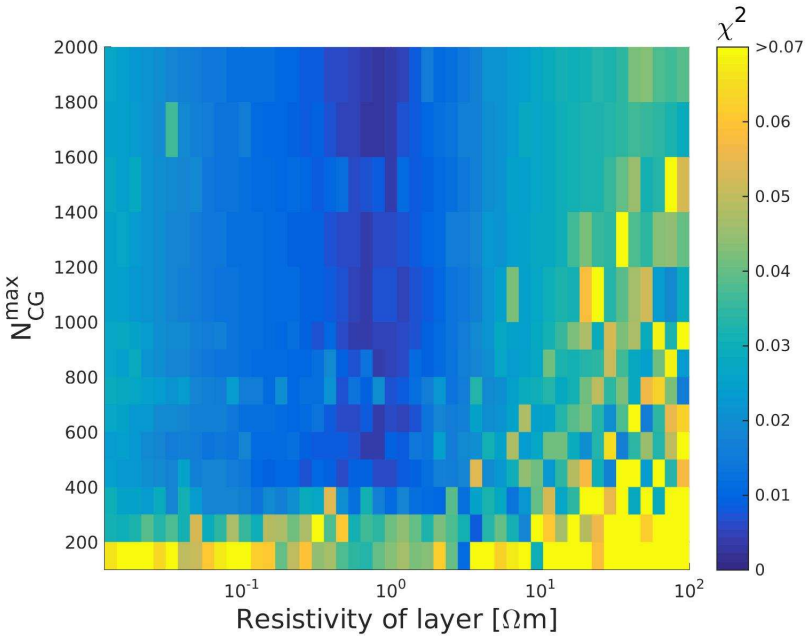

Figure 4: χ^2-misfit for a maximum number of CG iterations (N_{CG}^{max}) from 100 to 2000 and a resistive layer from 0.01 to 100Ωm. Krylov dimension was set constant to 22. A $\chi^2 > 0.07$ is depicted in yellow to achieve a colorscale identical to Fig.3.

In our next experiment, we test the influence of N_{CG}^{max} on the numerical accuracy. The resulting χ^2-misfit is depicted in Fig. 4. Its sprinkled pattern is created by a non-linear convergence of the two transients on each other. Misfits larger than 0.07 were plotted in yellow as well, to make the misfit comparable with Fig.3. Therefore, the misfits for very high $N_{CG}^{max} = 2000$ look identical to the misfits in Fig.3 for high Krylov numbers and are the lowest errors achievable by adjusting those two parameters. Reducing N_{CG}^{max} first causes increasing mistakes for high resistors, such that N_{CG}^{max} should not be less than 1000. Low conductivity contrasts, like resistors around $1\Omega m$, are numerically the most well conditioned such that $N_{CG}^{max} \approx 500$ gives accurate results. High conducters seem to behave much more tolerant numerically than high resistors, for both, N_{CG}^{max} as well as N_{Krylov}.

4.2. 3D comparison

To show that the code also works in 3D, we compared the solution for a 3D model with the solution of the code *sldmem* [11].
The 3D model, depicted in Fig.5 left, is just replaced by a body with a horizontal extension $of 100m \times 100m$ with a resistivity of $20\Omega m$.

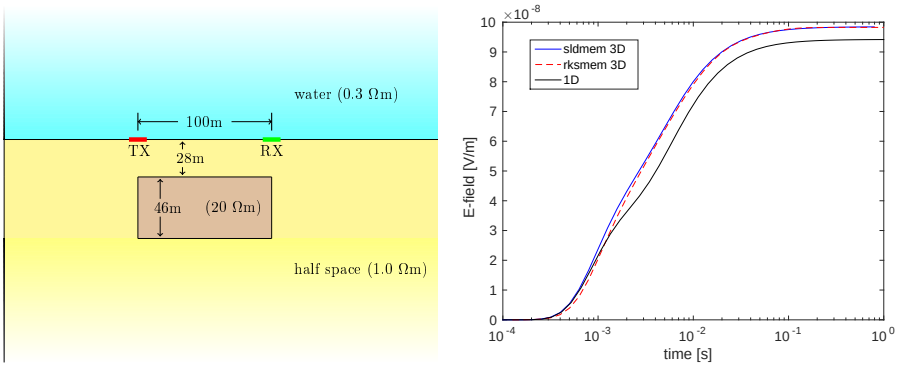

Figure 5: Left: 3D model used for comparison. It is differs from the previous 1D model, only by replacing the resistive layer by a body with a horizontal extension of 100m x 100m with a resistivity of 20Ohmm. Right: Comparison of our code *rksmem* (red, dashed) with *sldmem* (blue, solid) for the 3D model. The analytic transient for the 1D case with a $20\Omega m$ resistive layer (black, solid) reveals the 3D effect.

The grid used for the calculations with our code is identical to the one used in the previous chapter. The grid for *sldmem* has been chosen according to its own requirements to give stable results. Fig.5 right, depicts the transients for both codes. In general, they look very similar. Except for early times, results fit on each other. Changing the resistivity of the body still gives overlapping transients, we just showed one case as a representative example.

4.3. Total run-times

Finally, the most crucial parameters determining run-time are the size of the system matrix $\mathbf{A}$ and the maximum number of CG iterations (N_{CG}^{max}). Run-times are benchmarked on an Nvidia GeForce GTX TITAN Black, CUDA version 7.0, and are depicted in Fig. 6.

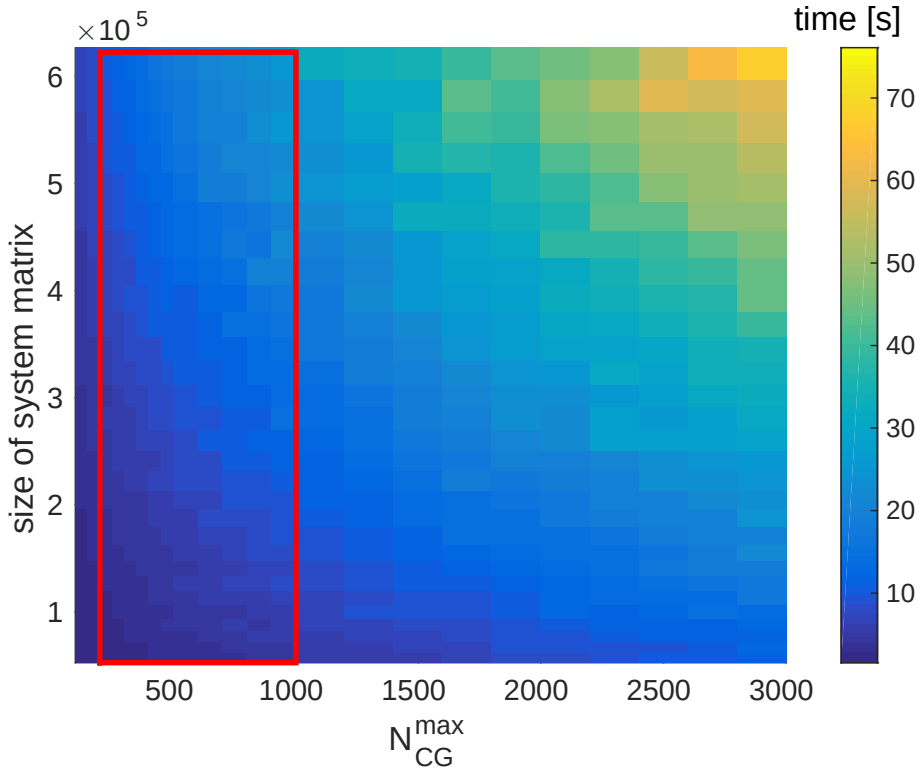

Figure 6: Run-times for a ill-conditioned model (worst case, highest run-times) for different sizes of system matrix **A** and maxima of CG-iterations.

Here, run-times are shown for the worst case scenario with a layer resistivity of 100Ωm. The sufficient range of N_{CG}^{max}, found in section 4.1 is marked by a red frame. Run-times for realistic model sizes are between 2 and 10 seconds or even less for less resistive layers.
To evaluate the enhancements of the *rksmem* code described herein, run-times for four different codes have been compared:
1. *rksmem* on a new *GeForce GTX Titan Black* graphics card,
2. *rksmem* on an old *GeForce GTX275* graphics card,
3. the GPU parallelized code *temddd GPU* based on SLDM ([19]) and
4. the CPU-optimized code *sldmem* ([11])

In this section transients for one source and one receiver are computed. However, we show in the next section, that the code *rksmem* offers great benefits when computations of fields at many grid locations are required. The code *sldmem* is described in [11] and its run-times are measured on one core of an *Intel Core i7* CPU 860 with 2.80GHz. Its runtime was optimized for CPUs and compiled with a fast intel compiler, but it is out-of-date. We used it, because we have no access to other expensive state of the art codes from industry and also because

it serves as a benchmark to our previous study. We ran it on one core. The code *temddd GPU* only runs on old graphics cards, because it uses the commercial CULA library that is no longer supported on new GPU-architectures.
For calculations in this experiment, the number of cells in x/y-direction is varied, whereas all other parameters are kept constant. For SLDM-based codes, a Krylov dimension of 2000 (reasonable values were validated in [19]) and for RKS-based codes a Krylov dimension of 20 and N_{CG}^{max} is used. The model is an 1Ωm half space with a 720m thick water layer above. We use an inline configuration with an offset of 100m. This model is chosen for the purpose to compare the code's run-times and therefore is kept as simple as possible.

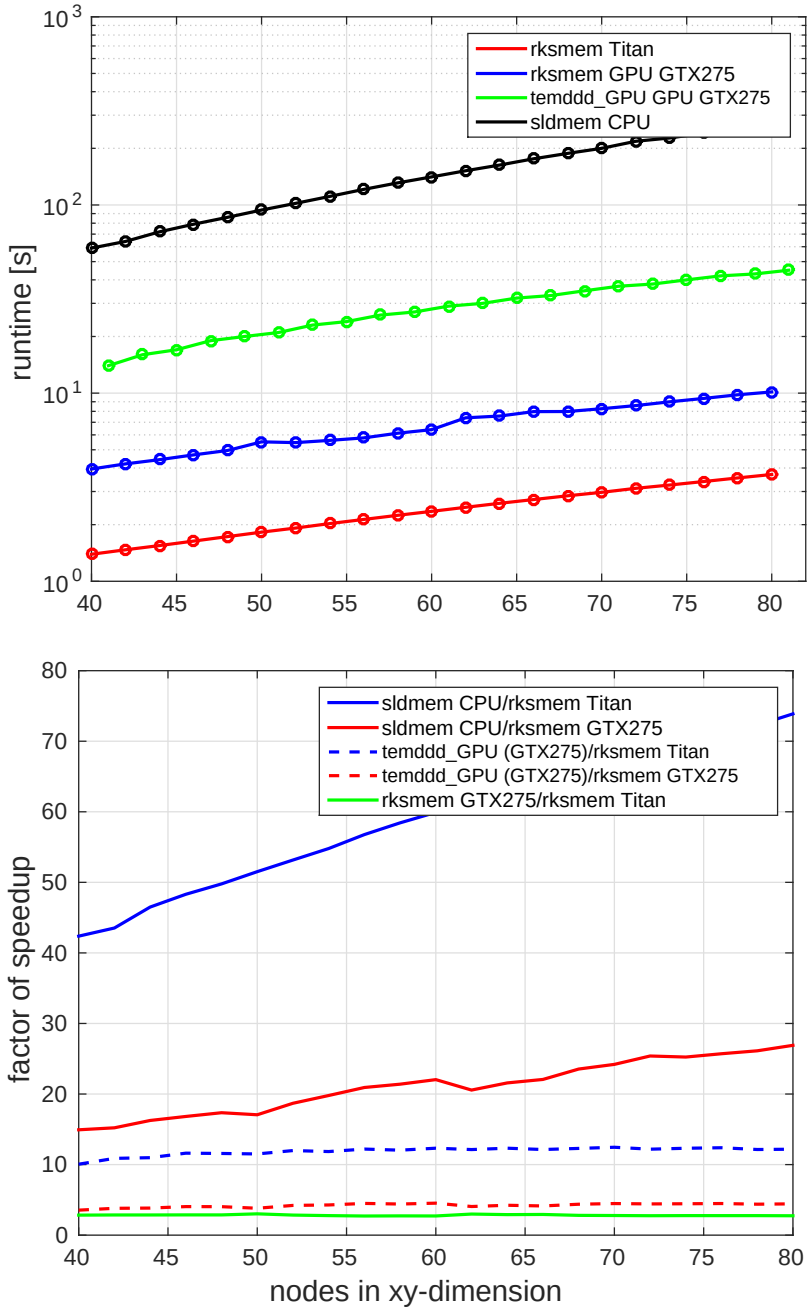

Figure 7: Top: Total run-times of *sldmem* (black), *temddd_GPU* on a GPU *GTX275* (green), *rksmem* on *GTX275* (blue) and *rksmem* on Titan (red) in dependence of model sizes in x/y-direction. Krylov dimension was set to 2000 for SLDM based codes and to 20 for RKSM based codes. Bottom: Speedups of the codes with respect to each other, to reveal the impact of different algorithms and architectures on run-time.

The results are shown in Fig. 7. The run-time of the serial code *sldmem* (top Fig., black, solid) ranges from 60 to 300 seconds and is the highest. A GPU parallelization of the same algorithm was realized with *temddd GPU* (top Fig., green, solid). Its run-times vary from 27 to 70 seconds, which is about four times faster. Using the RKSM instead of SLDM reduces the run-times to 4 - 10 seconds (top Fig., blue, solid) on the same graphics card (GTX 275). This reveals the speedup by algorithm alone (Bottom Fig., red, dotted). Running it on a new *GeForce Titan Black* card gives a run-time of 1.4 to 3.7 seconds (Top Fig., red, solid). The speedup of *rksmem* on a *GeForce Titan Black*, compared to a *GeForce GTX 275* is constant at a factor of about 3 (Bottom Fig., green, solid). It reveals the improvement of the graphics cards alone. It is faster, yet lower, than a factor of 10 speedup for GPUs designed for scientific computations, as promised by *Nvidia*. A possible explanation for this discrepancy would be overload. The total speedup of the new code on a new card over the old GPU parallelized code is constant around 12 (Bottom Fig., blue dotted). It is likely the product of the speedups of hardware and algorithm. Compared to the CPU code, the speedup is between 40 and 70 (Bottom Fig., blue, solid).
Please note that a $1\Omega m$ half space is used. Therefore, the low conductivity contrasts cause a good condition, resulting in a better performance of the RKSM. Run-times of the worst case scenario in Fig. 6 are higher, but would also lead to higher run-times for polynomial codes.

4.3.1. Run-times for fullspace solutions and applicability for inversion

One main application of forward codes is its integration as part of inversion codes. Gauss-Newton type schemes are very common, but require the computation of the Jacobian matrix. This is mostly done by the adjoint method [36], which requires the fullspace solution, meaning the computation of all electric field components for all cells of the grid. This number can reach up to several hundreds of thousands, such that the back projection of the Ritz vectors and the Galerkin Ansatz of eq. (5) become very expensive. Using a rational Lanczos base reduces those costs of eq. (5) drastically, because $k \approx 22$ is much smaller for the RKSM as compared to $k = 2000$ for polynomial methods.

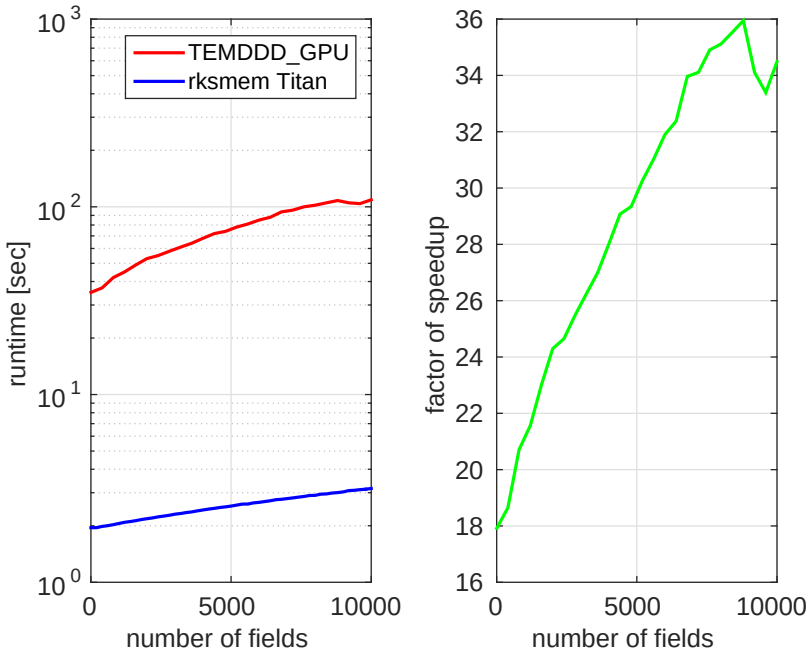

Figure 8: Left: run-time of *temddd_GPU* on a *GTX275* (red) and *rksmem* on a *Titan* (blue) for up to 10000 receivers. Right: Factor of speedup for up to 10000 receivers.

Fig. 8 left shows the run-times for both, the SLDM on *GeForce GTX275* and RKSM (on GeForce Titan Black) based codes. For both codes, run-times increase with increasing number of computed electric field components. The speedup (right) of *rksmem* over *temddd_GPU* increases with the number of receivers and reaches a value of around 36 for a $1\Omega m$ half space model of size of $54 \times 54 \times 50$ cells, $N_{kry} = 22$, $N_{CG}^{max} = 500$ and 10000 receivers. The run-time does not depend on the receivers location, such that they were chosen arbitrarily. The increasing speedup reveals that RKSM methods are more suitable for the adjoint method, rather than SLDM. The kink in the speedup at 9500 is caused by a decrease of runtime in the code *TEMDDD_GPU* and seems to be related to internal factors within the CULA library, which are not accessible to us.

5. Discussion and Conclusion

We have shown that by both implementing a new algorithm (rational- instead of polynomial Krylov method), as well as using new graphics cards reduces the run-time significantly.
The computationally expensive operations for both algorithms, are the sparse matrix times vector multiplications. In the polynomial method, Krylov vectors are spanned, as defined, by a matrix times vector operation. In the rational, every Krylov vector is computed by solving a linear system with Conjugate Gradients, based on matrix vector multiplications as well. A necessary N_{CG}^{max} of up to 1000 was mentioned, but the required number of matrix/vector operations in the rational method is always lower than the polynomial method. Only one (the second) Krylov vector requires so many iterations, all other vectors require less than 10 iterations, to achieve the required tolerance. This is the reason, why preconditioning is not so crucial.
A second reason that makes the RKSM faster is the much smaller eigenproblem to solve, to determine the Ritz-values from the Ritz matrix. Where a 2000×2000 matrix has to be solved for SLDM, eigenpairs of a 22×22 matrix have to be found for the RKSM. As seen in Fig. 7, the code becomes a factor of 7 faster by using RKSM instead of SLDM.
We also investigated the speedup related to changes in hardware architecture, by benchmarking the code on old and new graphics cards. The old one, a *GeForce GTX275* was launched in 2009 and is based on a *tesla* processor-architecture, designed for gaming. This means, that scientific computations on this device are only supported by slow emulation of double precision (IEEE754). The new *GeForce TITAN Black* on the other hand, was launched in 2014 and is based on the *kepler* processor-architecture, supporting scientific computations. Until now, it is the last *GeForce* card supporting double precision, but *tesla* cards, designed for clusters, still support double precision.
In this paper we showed that the code was running faster by a factor of 3 on the *GeForce TITAN Black* than on the *GeForce GTX275* card. *Nvidia* advertised a speedup of 10, but without overload. Unlike CPU implementations, we can expect that hardware for our implementation will become faster in future. CPU frequencies did not increase during the last 10 years, but computational power of GPUs is still growing.
We demonstrated the applicability of the code for different conductivity contrasts. High conductivity contrasts lead to poor conditioning of the system matrix, but fast and accurate results could be computed anyway for realistic models. Preconditioning could be an improvement and will be part of future work. We compred the results of our code with results of an external 3D code for a 3D model and achieved similar results.
Future works will focus on the application of rational Krylov methods for inversion. We showed that the computation of the full space solution, required for an adjoint Jacobian construction, can be computed much more efficiently with the RKS method, rather than SLDM.

6. Acknowledgment

This research was done in the framework of the SUGAR project. Benchmark results for the *sldmem* code were provided by courtesy of the University of Toronto. We thank Mikhail Zaslavsky for very kind and immediate response to our questions regarding Rational Krylov subspaces.

References

[1] J. Hesthammer, A. Stefatos, M. Boulaenko, A. Vereshagin, P. Gelting, T. Wedberg, G. Maxwell, CSEM technology as a value driver for hydrocarbon exploration, Marine and Petroleum Geology 27 (2010) 1872–1884.

[2] K. Schwalenberg, M. Haeckel, J. Poort, M. Jegen, Evaluation of gas hydrate deposits in an active seep area using marine controlled source electromagnetics: Results from Opouawe Bank, Hikurangi Margin, New Zealand, Marine Geology 272 (2010) 79–88.

[3] S. Hölz, A. Swidinsky, M. Sommer, M. Jegen, J. Bialas, The use of rotational invariants for the interpretation of marine csem data with a case study from the north alex mud volcano, west nile delta, Geophysical Journal International 201 (1) (2015) 224. doi:10.1093/gji/ggv015.

[4] A. Zerilli, T. Labruzzo, M. Zanzi, M. P. B. Buonora, J. L. Crepaldi, P. d. T. L. Menezes, Broadband marine CSEM: New benefits for subsalt and around salt exploration, in: SEG Technical Program Expanded Abstracts 2014, 2014, pp. 750–754. arXiv:http://library.seg.org/doi/pdf/10.1190/segam2014-1201.1, doi:10.1190/segam2014-1201.1.
URL http://library.seg.org/doi/abs/10.1190/segam2014-1201.1

[5] A. Swidinsky, H. Hölz, M. Jegen, On mapping seafloor mineral deposits with centeral loop transient electromagnetics, Geophysics 77 (3) (2012) E171–E184.

[6] S. Constable, L. J. Srnka, An introduction to marine controlled-source electromagnetic methods for hydrocarbon exploration, Geophysics 72 (2) (2007) WA3–WA12. doi:{10.1190/1.2432483}.

[7] S. Hölz, M. Jegen, Development of a CSEM system for the electromagnetic study of the North Alex Mud Volcano, in: EGU General Assembly Conference Abstracts, Vol. 11 of EGU General Assembly Conference Abstracts, 2009.

[8] P. A. Wolfgram, R. N. Edwards, L. K. Law, M. N. Bone, Polymetallic sulfide exploration on the deep sea floor: The feasibility of the MINI-MOSES experiment, Geophysics 51 (9) (1986) 1808–1818. doi:10.1190/1.1442227.

[9] J. Yuan, R. N. Edwards, The assessment of marine gas hydrates through electronic remote sounding: Hydrate without a BSR?, Geophysical Research Letters 27 (16) (2000) 2397–2400.

[10] C. Yin, Y. Qi, Y. Liu, J. Cai, 3D time-domain airborme EM forward modeling with topography, Journal of Applied Geophysics 134 (2016) 11–22.

[11] V. Druskin, L. Knizhnerman, Spectral approach to solving three-dimensional Maxwell's equations in the time and frequency domain, Radio Science 29 (1994) 937–953.

[12] T. Wang, G. Hohmann, A finite-difference, time-domain solution for three-dimensional electromagnetic modelling, Geophysics 58 (1993) 797–809.

[13] R. U.-. Börner, O. G. Ernst, K. Spitzer, Fast 3d simulation of transient electromagnetic fields by model reduction in the frequency domain using krylov subspace projection, Geophysics 173 (2008) 766–788.

[14] L. A. Knizhnerman, V. L. Druskin, M. Zaslavsky, On optimal convergence rate of the rational krylov subspace reduction for electromagnetic problems in unbounded domains, SIAM Journal 47 (2009) 953–971.

[15] V. Druskin, C. Lieberman, M. Zaslavsky, On Adaptive Choice of Shifts in Rational Krylov Subspace Reduction of Evolutionary Problems, SIAM Journal 32 (2010) 2485–2496.

[16] V. Druskin, V. Simoncini, Adaptive Rational Krylov Spaces for Large-Scale Dynamical Systems, Systems & control Letters 32 (2011) 2485–2496.

[17] M. Zaslavsky, V. Druskin, L. Knizhnerman, Solution of 3D time-domain electromagnetic problems using optimal subspace projection, Geophysics 76.

[18] R.-U. Börner, O. G. Ernst, S. Güttel, Three-dimensional transient electromagnetic modelling using Rational Krylov methods, Geophysical Journal International 202 (3) (2015) 2025. doi:10.1093/gji/ggv224.

[19] M. Sommer, S. Hölz, M. Moorkamp, A. Swidinsky, B. Heincke, C. Scholl, M. Jegen, GPU parallelization of a three dimensional marine CSEM code, Computers & Geosciences (58) (2013) 91–99.

[20] K. Yee, Numerical Solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Transactions on Antennas and Propagation AP-14 (1966) 302–307.

[21] K. Árnason, Consistent discretization of electromagnetic fields and transient modeling, in: M. Oristaglio, B. Spies, M. R. Cooper (Eds.), Three-Dimensional Electromagnetics, SEG, 1999, pp. 103–118.

[22] R. Courant, K. Friedrichs, H. Lewy, Über die partiellen Differenzengleichungen der mathematischen Physik, Mathematische Annalen 100 (1928) 32–74.

[23] B. G. Galerkin, Beams and plates, series for some problems of elastic equilibrium of beams and plates, Wjestnik Ingenerow 10 (1915) 897–908, (in Russian).

[24] C. Lanczos, A iterative method for the solution of the eigenvalue problem of linear differential and integral operators, Journal of Research of the National Bureau of Standarts 45 (1950) 255–282.

[25] B. N. Parlett, The symmetric eigenvalue problem, Prentice-Hall, Englewood Cliffs, NJ, (republished by SIAM, Philadelphia), 1998.

[26] A. N. Krylov, On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined, Izv. Akad. Nauk SSSR 7:4 (1931) 491–539, (in Russian).

[27] A. Ruhe, Rational Krylov sequence methods for eigenvalue computation, Linear Algebra and its Applications 58 (1984) 391–405.

[28] S. Güttel, Rational krylov methods for operator functions, Ph.D. thesis, TU Bergakademie Freiberg (2010).

[29] V. Druskin, L. Knizhnerman, M. Zaslavsky, Solution of large scale evolutionary problems using rational Krylov subspaces for with optimized shifts, SIAM Journal 31 (2009) 3760–3780.

[30] S. Geršgorin, Über die Abgrenzung der Eigenwerte einer Matrix, Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na 6 (1931) 749–754.

[31] S. Yan, C. Li, Y. Zhang, H. Zhou, yaSpMV: yet another SpMV framework on GPUs, ACM SIGPLAN Notices 49 (8) (2014) 107–118. doi:10.1145/2692916.2555255.
URL http://doi.acm.org/10.1145/2692916.2555255

[32] J. Rice and R. Boisvert, Solving Elliptic Problems Using ELLPACK, Springer Series in Computational Mathematics, 1985.

[33] G. H. Golub, C. F. Van Loan, Matrix Computations, 3rd Edition, John Hopkins University Press, Baltimore, 1996.

[34] W. Briggs, V. Henson, S. McCormick, Back Matter, 2000, pp. 189–193. arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9780898719505.bm, doi:10.1137/1.9780898719505.bm.
URL http://epubs.siam.org/doi/abs/10.1137/1.9780898719505.bm

[35] M. Munkholm, E. Auken, Electromagnetic Noise Contamination on Transient Electromagnetic Soundings in Culturally Disturbed Environments, JEEG 1 (2) (1996) 119–127.

[36] P. R. McGillivray, D. W. Oldenburg, R. G. Ellis, T. M. Habashy, Calculation of sensitivities for the frequency-domain electromagnetic problem, Geophysical Journal International 116 (1994) 1–4.

Sensitivity calculation for marine controlled source electromagnetics in time domain by model order reduction in the rational Krylov subspace

1. Introduction

Marine Controlled Source Electromagnetics (CSEM) is an exploration method that allows reconstruction of the spatial conductivity distribution in the sub seafloor. Typical exploration targets that are characterized by a resistivity anomaly are hydrocarbons like gas and oil [1], gas hydrates [2], complex salt diapirs [3] and massive sulphide deposits [4]. Most common are surveys, where the source is towed behind a ship and the receivers are placed on the seafloor [5] or being mounted in a streamer moving along the seafloor [6]. In our system, dipole sources (Tx) and receivers (Rx) are placed on the seafloor [7] and transmit a Heaviside function in time-domain. Because survey design, as well as the nature of the geological targets (gas hydrates, sulphide deposits), we have to deal with 3D data sets.

Interpretation of the data is normally done by perturbing a geologic model and 3D forward modeling, until the synthetic and observed data fit, or by inversion. Because of high run times of the 3D modeling codes, global or statistical approaches are not appropriate. Gauss Newton ([8]) based 3D inversion of CSEM are usually preferred since they are less susceptible to potentially rough objective functions. However, it is challenging to compute the required Jacobian numerically stable and fast. In general, there exist three approaches: brute force, adjoint method and the model order reduction method. The brute force method calculates the Jacobian by differencing forward calculations for every model parameter. This takes enormous computation time and it is also not very accurate. The adjoint method is most common. It requires one full space solution for every source/receiver pair and is therefore much faster than the brute force method. However, it also has problems with accuracy, because of a strong approximation within the method. The approach we have chosen is the model order reduction (MOR) of [9], that compress the Jacobian into rational Krylov subspaces. The method is quite new and not much experience has been gained yet about benefits and drawbacks of this approach.

In this publication, we describe the implementation of the MOR-Jacobi of [9]. It is based on the rational Krylov subspace forward code described in [10], parallelized on Graphics processing units (GPU). First we explain the theory and subsequently the implementation on GPUs and the grid design. In the end,

we show some numerical tests and validate the approach by comparison with brute force Jacobians. Finally, we show an application example by computing the sensitivities of a realistic resistivity model using the Tx,Rx geometry of a real data set.

2. Theory

To motivate the Jacobian computation, a basic introduction to inversion theory is given in 2.1. The theory of the here described sensitivity calculation approach is based on the theory of the forward problem. Therefore, it is shortly described 2.2. The Jacobian computation by the model reduction approach is explained in 2.3.

Conventions:
The letters j, k, n are used for integers, scalars are denoted by small letters like $t, \alpha, \ldots,$ vectors are indicated by an arrow above: $\vec{q}, \vec{E}$ and matrices are all denoted in bold, capital letter like: $\mathbf{A}, \mathbf{Q}$.

2.1. Fundamentals of inversion

The fundamental idea of inversion theory is to minimize an objective function, for example:

$$\Phi = \|\vec{d}_{obs} - \vec{d}_{mod}(\vec{m})\| + \lambda\Phi_{mod} \tag{1}$$

with $\vec{d}_{obs}$ being the observed data, $\vec{d}_{mod}$ are modeled by a forward code for a given model $\vec{m} \in \Omega$ where Ω is a Hilbert space representing all geologically reasonable models. Here, it will be a model of conductivity values $\vec{m} = \vec{\sigma} = [\sigma_1, \ldots, \sigma_{N_{mod}}]$. Because less data than model parameters are normally available in geophysics, additional constraints have to be included by using a model objective function Φ_{mod}, weighted by a Lagrange parameter λ. The given model $\vec{m}$ has to be modified, until Φ reaches its global minimum and the model explains best the data. It is impossible to know Φ for all $\vec{m} \in \Omega$, but Φ can be locally approximated by a Taylor series expansion:

$$\Phi(\vec{m} + \delta\vec{m}) \approx \Phi(m) + \nabla\Phi(\vec{m})\delta\vec{m} + \frac{1}{2}\nabla^2\Phi(\vec{m})\delta\vec{m}^2 \tag{2}$$

with $\mathbf{H} = \nabla^2\Phi(\vec{m})$ being the Hessian matrix and $\vec{g} = \nabla\Phi(\vec{m})$ being the gradient. In order to find the global minimum of Eq. (1), we set $\nabla\Phi(\vec{m} + \delta\vec{m}) = 0$ such that

$$\mathbf{H}\delta\vec{m} = -\vec{g} \tag{3}$$

with $\delta\vec{m} = \vec{m}_{i+1} - \vec{m}_i$. The global minimum can be found iteratively:

$$\vec{m}_{i+1} = \vec{m}_i - \mathbf{H}_i^{-1}\vec{g}_i. \tag{4}$$

This approach is called the *Newton's Method*. An approximation of $\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$ defines the *Gauss-Newton Method*, where $\mathbf{J}_{ij} = \frac{\partial \vec{d}_i(\vec{m})}{\partial m_j}$.
Calculation of the Jacobian, also called sensitivity, is computationally most expensive step in inversion. The most simple way to compute a Jacobian is the *brute force* or *finite difference* Method:

$$\mathbf{J} = \frac{\partial \vec{d}}{\partial \vec{m}} \approx \frac{\vec{d}(\vec{m} + \Delta\vec{m}) - \vec{d}(\vec{m})}{\Delta\vec{m}} \tag{5}$$

It is problematic for two reasons. A small perturbation in one model parameter changes the modeled data only slightly and is therefore numerically not robust. Furthermore, a forward solution has to be computed for every model parameter. This can be hundred thousands, resulting in enormous computer run times.
Most common is the adjoint Green's Functions approach [11]. The Greens functions are obtained by a full space solution of the forward solver which is then convolved with a background field. It reduces the number of forward calculations from the model size down to the number of transmitters. But it also has problems with accuracy.
We have chosen the model reduction method of [9]. This will be explained and demonstrated in the following.

2.2. The forward problem

The forward problem is not the focus of this publication. Therefore, we give only a very brief explanation, more details are found in [10].
The physics of marine CSEM is described by the electromagnetic diffusion equation:

$$\vec{\nabla} \times \vec{\nabla} \times (\sigma\mu)^{-1}\vec{E}(\vec{r}, t) = -\frac{\partial \vec{E}(\vec{r}, t)}{\partial t} \tag{6}$$

with $\vec{E}$ being the electric field, σ the isotropic, electrical conductivity and μ the magnetic permeability. The spatial operators are discretized by a finite difference (FD) Yee-scheme [12] as described in [13], such that $\vec{\nabla} \times \vec{\nabla} \times (\sigma\mu)^{-1} \rightarrow \mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{A}$ being symmetric [14]. This reduces the partial differential equation (PDE) Eq. (6) to an ordinary differential equation (ODE):

$$\mathbf{A}\vec{E}^{FD} = -\frac{\partial \vec{E}^{FD}}{\partial t}, \tag{7}$$

Instead of discretizing time and solving the differential equation by time stepping, the ODE is solved by an exponential Ansatz function as described in [15]:

$$\vec{E}^{FD}_{Ansatz} = exp(-t\mathbf{A})\vec{E}^{FD}_0 = \sum_{j=1}^{n} \vec{z}_j exp(-t\lambda_j)\vec{z}_j^T\vec{E}^{FD}_0. \tag{8}$$

It is spectrally decomposed, such that λ_j and $\vec{z}_j$ are the eigenvalues and eigenvectors of $\mathbf{A}$. Because of the size n of $\mathbf{A}$, it is compressed in a subspace:

$$\mathbf{H} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}. \tag{9}$$

with $\mathbf{H} \in \mathbb{R}^{N_k \times N_k}$ being the Ritz matrix and $\mathbf{Q} = [\vec{q}^{\,0}, \ldots, \vec{q}^{\,N_k-1}] \in \mathbb{R}^{n \times N_k}$ being the orthonormal subspace projection matrix. The eigenpairs θ_i and $\vec{\psi}_k$ of $\mathbf{H}$, for which $\mathbf{H}\vec{\psi}_i = \theta\vec{\psi}_i$ for $i = 1, \ldots, N_k$, are called Ritz-values and Ritz-vectors. Ritz-values approximate the eigenvalues of $\mathbf{A}$ and the back projection of the Ritz vectors $\mathbf{Q}\vec{\psi}_i$ approximate the eigenvectors of $\mathbf{A}$. Inserting the Ritz-pairs into Eq. (8) gives the Galerkin approximation:

$$exp(-t\mathbf{A})\vec{E}_0^{FD} \approx \sum_{i=0}^{N_k-1} \mathbf{Q}\vec{\psi}_i exp(-t\theta_i)\vec{\psi}_i^T \mathbf{Q}^T \vec{E}_0^{FD} \tag{10}$$

We use as a compressing subspace, the Rational Krylov [16] subspace:

$$\mathcal{K}_{rat}^{N_k}(\mathbf{A}, \vec{q}^{\,0}, \vec{s}) = span\left\{(\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}^{\,0}, \ldots, \prod_{i=1}^{N_k}(\mathbf{A} - s_i\mathbf{I})^{-1}\vec{q}^{\,0}\right\}. \tag{11}$$

It is spanned by the system matrix $\mathbf{A}$, an initial vector $\vec{q}^{\,0}$ and a set of poles $\vec{s} = [s_0, \ldots, s_{N_k}]$. Orthonormalization of the spanning set gives the column vectors of $\mathbf{Q}$. A pole selection strategy is explained in [17]. They can be found by the recursion:

$$s_{j+1} = \underset{s \in [s_j, s_2]}{\arg\max} \frac{1}{\mid r_j(s) \mid}, \qquad r_j(z) = \prod_{l=1}^{j} \frac{z - \lambda_l}{z + s_l}, \quad j = 2, \ldots, N_k \tag{12}$$

Initial poles s_1 and s_2 are found by Gershgorin cycles [10] that give an upper and lower bound of a matrix spectra. For λ_l, the Ritz values corresponding to the $\vec{q}$-vectors computed so far, are used.

2.3. Jacobian computation by model order reduction

The model order reduction is based on the idea that the Jacobian is, per definition, the derivative of the forward solution for the given model parameters. The Jacobian can be computed by deriving the Ansatz function Eq. (8) for conductivity:

$$\mathbf{J}_{jl} = \frac{\partial \vec{E}_{jl}}{\partial \sigma_{1,\ldots,N_{mod}}} = \frac{\partial}{\partial \sigma_{1,\ldots,N_{mod}}} \vec{p}_j exp(-t\mathbf{A})\vec{E}_{0,l}^{FD} \tag{13}$$

where j is the index of the receiver, l of the source, $\vec{p}_j$ and $\vec{p}_l$ are the canonical vectors representing the receiver and source positions. Eq. (8) would give the full space solution, if it is not multiplied by $\vec{p}_j$. We will set $\vec{p}_l = \vec{E}_{0,l}$ in the following.
Let us define the Lanczos vectors $\vec{\xi}_i = \mathbf{Q}\vec{\psi}_i, \quad i = 1, \ldots, N_k$, such that Eq. (13), with the Galerkin approximation Eq. (10) becomes, by applying the product and chain rules:

$$\mathbf{J}_{jl}(t) = \sum_{i=1}^{N_k} \left[\vec{p}_j^{\,T} \frac{\partial \vec{\xi}_i}{\partial \sigma} \vec{\xi}_i^{\,T} \vec{p}_l + p_j^{\,T} \vec{\xi}_i^{\,T} \frac{\partial \vec{\xi}_i^{\,T}}{\partial \sigma} \right] e^{-\theta_i t} - \sum_{i=1}^{N_k} \vec{p}_j^{\,T} \vec{\xi}_i \vec{\xi}_i^{\,T} \vec{p}_l t e^{-\theta_i t_k} \frac{\partial \theta_i}{\partial \sigma} \tag{14}$$

The canonical vectors $\vec{p}_j$ and $\vec{p}_l$ represent the receivers and sources locations, respectively. This equation can be represented as a matrix-matrix multiplication:

$$\mathbf{J}_{jl} = \mathbf{E}_{jl}^T \mathbf{B}_{jl} \tag{15}$$

with $\mathbf{E}_{jl} \in \mathbb{R}^{2 \cdot N_k \times N_{t_j}}$ being the *time decomposition* or *Laplace transform* of the Jacobian

$$\mathbf{E}_{jl}^T = \begin{pmatrix} e^{-\theta_1 t_1} & \dots & e^{-\theta_{N_k} t_1} & t_1 e^{-t_1 \theta_{N_k}} & \dots & t_1 e^{-t_1 \theta_{N_k}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e^{-\theta_1 t_{N_{t_j}}} & \dots & e^{-\theta_{N_k} t_{N_{t_j}}} & t_{N_{t_j}} e^{-t_{N_{t_j}} \theta_{N_k}} & \dots & t_{N_{t_j}} e^{-t_{N_{t_j}} \theta_{N_k}} \end{pmatrix} \tag{16}$$

and the *spatial decomposition* $\mathbf{B}_{jl} \in \mathbb{R}^{2 \cdot N_k \times N_{mod}}$

$$\mathbf{B}_{jl} = \begin{pmatrix} \beta_{11}^{jl} & \dots & \beta_{1,N_k}^{jl} & -\epsilon_{11}^{jl} & \dots & -\epsilon_{1,N_k}^{jl} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \beta_{N_m,1}^{jl} & \dots & \beta_{N_m,N_k}^{jl} & -\epsilon_{N_m,1}^{jl} & \dots & -\epsilon_{N_m,N_k}^{jl} \end{pmatrix} \tag{17}$$

we introduced $\beta_{im}^{jl} = \vec{p}_j^T \left(\frac{\partial \vec{\xi}_m}{\partial \sigma_i} \vec{\xi}_m^T + \vec{\xi}_m \frac{\partial \vec{\xi}_m^T}{\partial \sigma_i} \right) \vec{p}_l^{jl}$ and $\epsilon_{im}^{jl} = \vec{p}_j^T \vec{\xi}_m \vec{\xi}_m^T \vec{p}_l \frac{\partial \theta_m}{\partial \sigma_i}$ only to make the matrix easier to read. An interesting feature of this approach is, that the size of the Jacobian is compressed from size $N_{mod} \times N_{data}$ to two matrices of sizes $2 \cdot N_k \times N_{data}$ and $2 \cdot N_k \times N_{mod}$.

2.3.1. Deriving Ritz values

Ritz values and Lanczos vectors are given by the Lanczos decomposition mentioned in Section 2.2, yet we still need to determine their derivative. This can be done with the help of the *Rayleigh Quotient* [18]:

$$r(\vec{v}, \mathbf{A}) = \frac{\vec{v}^T \mathbf{A} \vec{v}}{\vec{v}^T \vec{v}} \tag{18}$$

gives, for $\vec{v}$ being an eigenvector of $\mathbf{A}$, the corresponding eigenvalue. For $\|\vec{v}\| = 1$, the derivative of the *Rayleigh Quotient* is given by (after applying the product rule twice):

$$r' = (\vec{v}^T)' \mathbf{A} \vec{v} + \vec{v}^T \mathbf{A}' \vec{v} + \vec{v}^T \mathbf{A} \vec{v}' \tag{19}$$

Neglecting the first and the last terms, which is a strong assumption, inserting Ritz values and Lanczos vectors gives

$$\frac{\partial \theta_m}{\partial \sigma_i} \approx \vec{\xi}_m^T \frac{\partial \mathbf{A}}{\partial \sigma_i} \vec{\xi}_m \tag{20}$$

The derivative of the eigenvectors is explained by perturbation theory in [19]. Applying theirs approach to Ritz values and Lanczos vectors gives:

$$\vec{p_l}^T \frac{\partial \vec{\xi}_m}{\partial \sigma_i} = \sum_{j=1, j\neq m}^{N_k} \frac{\vec{\xi_j}^T \frac{\partial \mathbf{A}}{\partial \sigma_i} \vec{\xi}_m}{\theta_m - \theta_j} \vec{p_l}^T \vec{\xi_j} \tag{21}$$

The derivation of the system matrix $\mathbf{A}$ can now be found by central finite differences and perturbation of the inversion model

$$\frac{\partial \mathbf{A}}{\partial \sigma_i} \approx \frac{\mathbf{A}([\sigma_1, \ldots, \sigma_i + \Delta\sigma, \ldots, \sigma_{N_{mod}}] - \mathbf{A}([\sigma_1, \ldots, \sigma_i - \Delta\sigma, \ldots, \sigma_{N_{mod}}])}{2\Delta\sigma} \tag{22}$$

Setting Eq. (22) into Eq. (21) and Eq. (20), and setting Eq. (21) and Eq. (20) into Eq. (16) and Eq. (17) gives the decomposed Jacobian.

2.3.2. Block Krylov Spaces

For problems with multiple source terms, the Block Krylov subspace is computationally more efficient:

$$\mathcal{K}_{block}^{N_k} = span\{(\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_1^0, \ldots, (\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_{N_b}^0, \tag{23}$$

$$\prod_{j=1}^{2}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_1^0, \ldots, \prod_{j=1}^{2}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_{N_b}^0 \tag{24}$$

$$\vdots \tag{25}$$

$$\prod_{j=1}^{N_k}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_1^0, \ldots, \prod_{j=1}^{N_k}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_{N_b}^0\} \tag{26}$$

It allows to compute the E-field and Jacobian for multiple sources and receivers with only one Krylov space. Replacing the Ansatz function by

$$\begin{pmatrix} E_{11}(t) & \ldots & E_{1,N_b}(t) \\ \vdots & \ddots & \vdots \\ E_{N_b,1}(t) & \ldots & E_{N_b,N_b}(t) \end{pmatrix} = \sum_{j=1}^{n} \mathbf{P}^T \vec{z}_j exp(-t\lambda_j) \vec{z}_j^T \mathbf{P} \tag{27}$$

where the columns of $\mathbf{P} = [\vec{p}_{TX}^1 \ldots \vec{p}_{TX}^{N_{tx}} \quad \vec{p}_{RX}^1 \ldots \vec{p}_{RX}^{N_{rx}}]$ store the canonical vectors of source and receiver locations, gives the Block Forward solution. Thus, the elements of the matrix on the left side of Eq. (27) represent the solutions for different source/receiver pairs. The diagonal contains the solutions of coincident positions. N_b is the sum of receivers and transmitters $N_b = N_{rx} + N_{tx}$. The projection matrix $\mathbf{Q}$ of Eq. (9) will consist of block matrices $\mathbf{Q}_{block} = [\bar{\mathbf{Q}}_1, \ldots, \bar{\mathbf{Q}}_{N_k}]$. The blocks are filled with Krylov vectors orthonormalized to all previous vectors of the base such that $\prod_{j=1}^{i}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_1^0, \ldots, \prod_{j=1}^{i}(\mathbf{A} - s_j\mathbf{I})^{-1}\vec{q}_{N_b}^0$ yield the vectors $\vec{q}_1^i, \ldots, \vec{q}_{N_b}^i$. The block matrices are filled such that $\bar{\mathbf{Q}}_i = [\vec{q}_1^i, \ldots, \vec{q}_{N_b}^i]$. Repeating the derivation of the Jacobian by using Eq. (27) instead of Eq. (8) gives the Block Jacobian, where every block represents a Jacobian of a source/receiver pair.

3. Implementation

The code is described by the following items:

1. Grid generation (ch. 3.1)
2. Computation Block Rational Krylov Subspace (ch. 3.2)
3. Derivation of system matrix $\frac{\partial \mathbf{A}}{\partial \sigma}$ (ch. 3.2)
4. Computation derived Ritz-pairs (ch. 3.3)
5. fill $\mathbf{E}$ and $\mathbf{B}$ of Eq. (16) and Eq. (17) with Ritz-values/vectors and their derivatives

A sensitivity calculation always includes a solver of the forward problem. We describe the implementation of the Block Rational Krylov Subspace Method (RKSM) in ch. 3.2. It is a modified version of the RKSM described in [10], such that it computes the Block RKSM analog version. We have to deal with two grids, one of the model to invert and a second grid to solve the forward problem numerically accurate. To avoid confusion, we will call them *inversion grid* and *forward grid*, former will henceforth contain the model parameters $\sigma_i,\ i = 1, \ldots, N_{mod}$ of the previous chapters. We will still call them *model parameters*, they will always refer to σ_i of an inversion grid cell, never to a forward grid cell in the following. Gridding and computation of $\frac{\partial \mathbf{A}}{\partial \sigma}$, from which the Jacobian is derived in Eq. (20) and Eq. (21), are described in 3.1. Deriving the Ritz pairs is the computationally most expensive operation and is described in 3.3. Finally, all previously described steps are depicted in a flow chart in 3.4, to understand how they interact for the Jacobian computation. Within all parts of the implementation, we make vast use of GPU. The code was written in the programming languages *C* and *CUDA* (Compute Unified Device Architecture) of *Nvidia*.

3.1. Grid generation and derivation of $\frac{\partial \mathbf{A}}{\partial \sigma}$

A discretization of the inversion grid has to fulfill two requirements, the cell's sizes should depict the resolution and the number of model parameters should be as low as possible for the sake of low run-times and computational complexity. Diffusion problems in geophysics have a decreasing resolution with depth, such that cell's sizes can be increased with depth. For the inversion grid we implement a quad-tree meshes as illustrated in Fig.1.

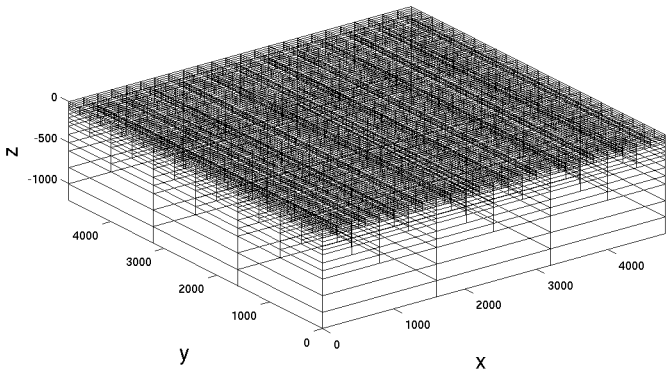

Figure 1: The inversion grid is discretized by a quad-tree grid that allow increasing cell's sizes with depth to reduce number of model parameters.

At certain depths, four cells get merged together in horizontal direction and reduce the number of model parameters to reduce run time.

This grid would not be stable enough to solve the forward problem, such that a second grid has to be introduced. This grid is characterized by smaller cells in the vicinity of Tx and Rx to achieve required numerical accuracies. It is a rectilinear grid as described in [13]. The inversion grid has to be merged on the forward grid by weighting the conductivities by the volumes of the intersecting cells:

$$\sigma^m_{fwd} = \sum_{i=1}^{N_{inv}} \frac{V^m_{fwd} \cup V^i_{inv}}{V^m_{fwd}} \sigma^i_{inv}, \qquad m = 1, \ldots, N_{fwd} \tag{28}$$

with V^m_{fwd} and V^i_{inv} being the volumes of the m-th forward grid and i-th inversion grid cell's, respectively (here $N_{inv} = N_{mod}$).

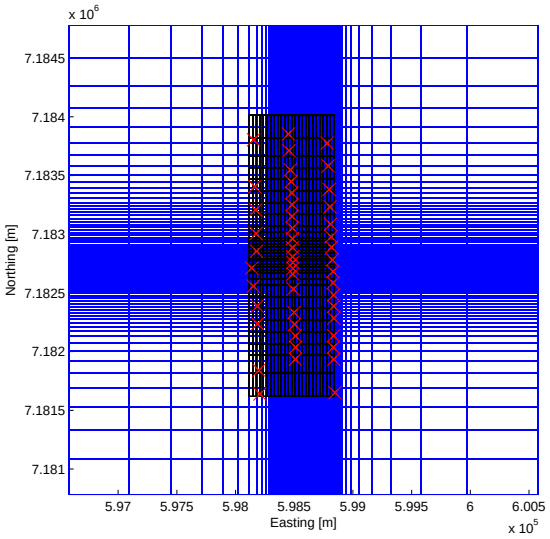

Figure 2: Top view of the rectilinear mesh for the forward problem (blue) and quad-tree-inversion mesh (black), applied to the geometry of a real data set. Coordinates are in UTM, red crosses indicate receivers-stations. Tx and Rx are not depicted.

The system matrix $\mathbf{A}$ is derived according to Eq. (22). Therefore, the projection Eq. (28) is applied to fill two meshes, one perturbed and one non-perturbed mesh. For both, system matrices are generated and used for Eq. (22). The resulting $\frac{\partial \mathbf{A}}{\partial \sigma}$ is very sparse, which has a favorable impact on run time.

The system matrix $\mathbf{A}$ is built on the forward grid, yet the derivative of $\mathbf{A}$ with respect to an inversion grid cell is needed Eq. (22). This is achieved by applying the projection Eq. (28) to obtain $\mathbf{A}$ for a perturbed and unperturbed inversion grid cell.

3.2. Implementation of the Block Krylov Method on GPU

The Block-Krylov subspace is computed by modification of the Rational Krylov subspace method as described in [10]. The latter spans a block Krylov space as defined in Eq. (26) and orthonormalises it such that a Ritz approximation can be computed as described in Eq. (9).

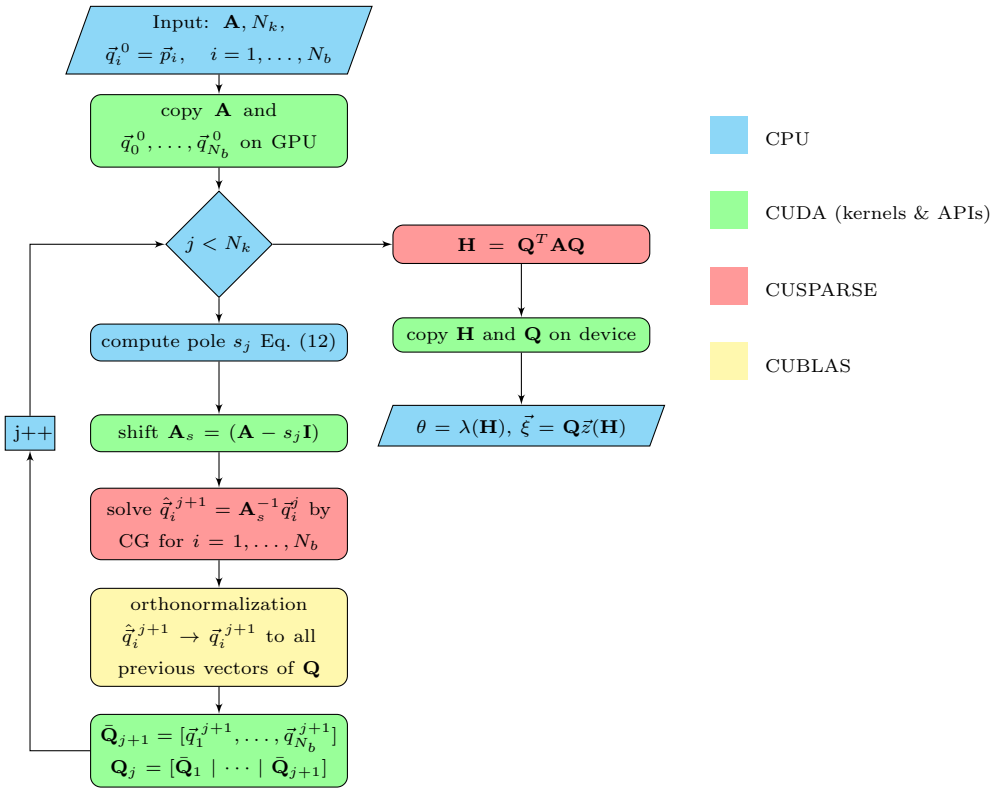

Figure 3: Flowchart of the Block-Rational-Krylov algorithm implementation. The colors depict which architecture (CPU or GPU) and which GPU libraries have been utilized.

Fig.3 illustrates a flowchart of the Block-Rational-Krylov algorithm implementation. Parts, running on the CPU are blue, CUDA kernels and APIs (application programming interface) are in green. Two GPU libraries have been utilized, the CUSPARSE and CUBLAS library. They perform BLAS (Basic Linear Algebra) operations for sparse (CUSPARSE) or dense (CUBLAS) operations on GPU.
We set $N_b = N_{tx}+N_{rx}$ the sum of sources and receivers, respectively. $\vec{p}_1, \ldots, \vec{p}_{N_b}$ are the canonical vectors of source and receiver indices. They were used as initial vectors for spanning the Block-Krylov vectors. The system matrix $\mathbf{A}$ as well as the initial vectors were copied on the GPU before switching into the loop. Poles are computed in the CPU domain with Eq. (12). As eigenvalues in Eq. (12), the Ritz values of the Ritz matrix of the Block Krylov vectors in the current iteration $\mathbf{H}_j = \mathbf{Q}_j^T\mathbf{A}\mathbf{Q}_j$ have been used. The pole is copied on the GPU to shift the matrix. This can easily be done in parallel, such that every thread access one diagonal element. For the conjugate gradient (CG) solver, the CUSPARSE library has been used to perform matrix vectors multiplications. Its implementation is described in [10]. CG is applied to every vector $\vec{q}_i^{\,j+1}$ (see Eq. (9)). The scalar products of the orthonormalization were implemented with the CUBLAS

library for dense operations. The orthonormalized vectors are written onto $\mathbf{Q_j}$ and in the last iteration to $\mathbf{Q}$. The Ritz matrix is computed with CUSPARSE for $\mathbf{AQ} = \mathbf{R}$ and with CUBLAS for $\mathbf{Q}^T\mathbf{R}$, according to Eq. (9). The Ritz matrix and Krylov base is copied on the CPU domain again, where a serial eigensolver (QR-algorithm) gives the Ritz-pairs and Lanczos vectors.

3.3. Deriving Ritz values and Lanczos vectors

Deriving the Ritz values and Lanczos vectors is the computationally most expensive part in Jacobian computation. Therefore, it has to be implemented with special care. To compute Eq. (21), the sum $\sum_{j=1,j\neq m}^{N_k} \vec{\xi}_j^T \frac{\partial \mathbf{A}}{\partial \sigma_i} \vec{\xi}_m$ can be rearranged to a matrix-matrix multiplication:

$$\mathbf{M}_i = \mathbf{\Xi}^T \frac{\partial \mathbf{A}}{\partial \sigma_i} \mathbf{\Xi} \tag{29}$$

with $\mathbf{\Xi} = \begin{bmatrix} \vec{\xi}_1 & \dots & \vec{\xi}_{N_k} \end{bmatrix}$. The columns of $\mathbf{M}_i$ are the derived Lanczos vectors. The derived Ritz values, according to Eq. (20) are on the diagonal of $\mathbf{M}_i$. This means, that for every model parameter, two matrix-matrix multiplications have to be performed. $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ is very sparsely populated because the perturbation of one inversion grid cell touches only a small number of forward grid cells. This allows to cut sub matrices $\left(\frac{\partial \mathbf{A}}{\partial \sigma_i}\right)_{sub}$ and $\mathbf{\Xi}_{sub}$ of $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ and $\mathbf{\Xi}$, respectively. The sub matrix $\left(\frac{\partial \mathbf{A}}{\partial \sigma_i}\right)_{sub}$ contains all rows and columns of $\left(\frac{\partial \mathbf{A}}{\partial \sigma_i}\right)$ with at least one element being non-zero. The sub matrix $\mathbf{\Xi}_{sub}$ contains the corresponding rows of $\mathbf{\Xi}$ that are multiplied with non-zero rows and columns of $\frac{\partial \mathbf{A}}{\partial \sigma_i}$.

$$\mathbf{M}_i = \left(\begin{array}{c} \vdots \\ \hline \mathbf{\Xi}_{sub} \\ \hline \vdots \end{array}\right)^T \cdot \left(\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & \left(\frac{\partial \mathbf{A}}{\partial \sigma_i}\right)_{sub} & 0 \\ \hline 0 & 0 & 0 \end{array}\right) \cdot \left(\begin{array}{c} \vdots \\ \hline \mathbf{\Xi}_{sub} \\ \hline \vdots \end{array}\right) \tag{30}$$

Thus, the size of $\mathbf{\Xi} \in \mathbb{R}^{N_k \times N_A}$ is drastically reduced. Where $N_A \approx 500000$ *to* 1000000, this size is reduced to less than 1000.
The sparse and dense operations were implemented with CUSPARSE and CUBLAS, respectively. The CUSPARSE library requires the CSR (Compressed Sparse Row) format for storing the sparse matrix. Conversion of the more common COO (COOrdinate Format) to CSR by the CUDA command is very time consuming, if it has to be called N_{mod} times. Therefore, we set up the sparse matrix $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ in CSR, right from the beginning.

3.4. Summary of Jacobi computation

To complete the Jacobian computation, the parts of chapters 3.1, 3.1, 3.2 and 3.3 have to be put together.

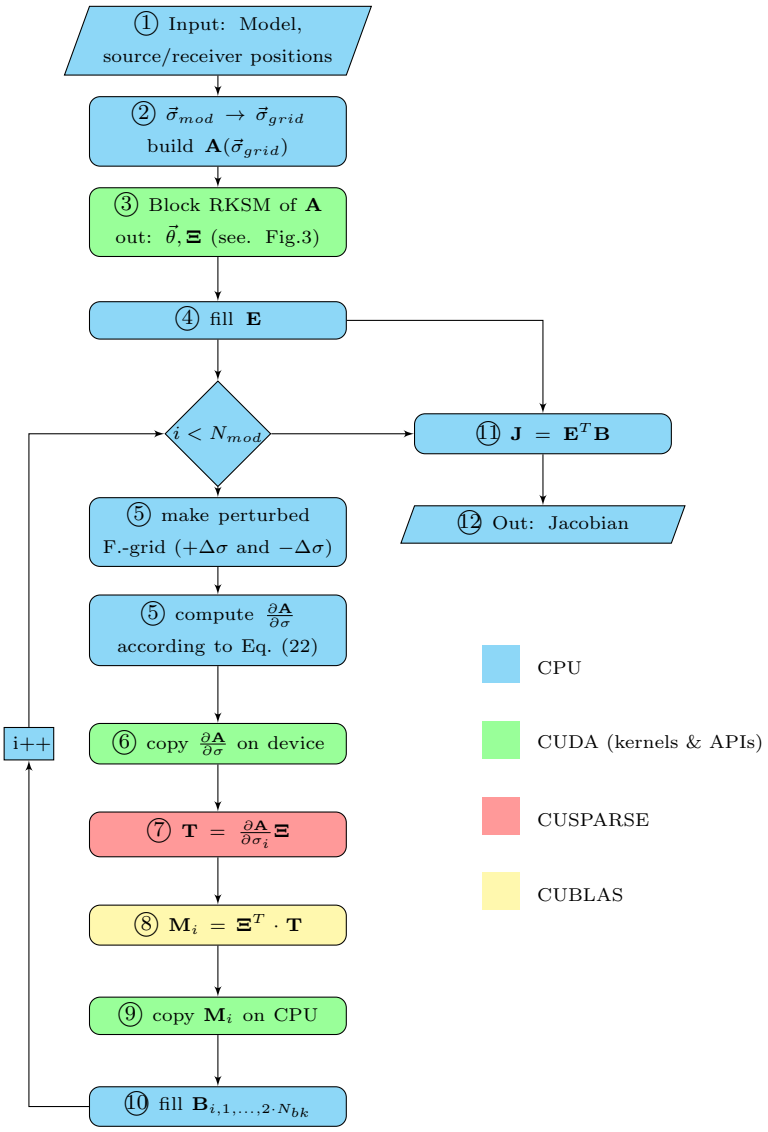

Figure 4: Flowchart of the implementation of the Jacobian computation.

The flowchart in Fig. 4 depicts the general overview of the implementation. For an input inversion grid and a geometry of sources and receivers ①, a forward grid is filled with conductivities of this inversion grid, for which a system matrix $\mathbf{A}$ is generated ②. For this unperturbed system matrix, the Ritz values and Lanczos vectors are computed with the Block Krylov algorithm as explained in ch. 3.2 ③. Because the *Laplace decomposition* is independent of the model parameters, the matrix $\mathbf{E}$ can be filled now ④. To compute $\mathbf{B}$, a loop will iterate through all model parameters. For every parameter, a perturbed system matrix

$\frac{\partial \mathbf{A}}{\partial \sigma_i}$ is computed by central finite differences ⑤ according to Eq. (22). Therefore, the inversion grid cell is perturbed by $+\Delta\sigma$ and $-\Delta\sigma$. For both perturbations, system matrices for Eq. (22) are generated to compute $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ ⑤. The derived system matrix $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ is copied on the graphics card now ⑥. Memory transfer is the main bottleneck in GPU implementations. We will discuss in ch. 4.2 how much a transfer of $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ weakens the code. A calculation of the derived system matrix on a GPU is very complicated to implement. The multiplications of $\frac{\partial \mathbf{A}}{\partial \sigma_i}$ and $\mathbf{\Xi}$ ⑦ are therefore implemented with CUSPARSE and CUBLAS, since they are computationally most expensive. Here, we introduce the temporary matrix $\mathbf{T} = \frac{\partial \mathbf{A}}{\partial \sigma_i}\mathbf{\Xi}$. The derived Ritz values and Lanczos vectors are now copied back on the CPU domain ⑨, such that the i-th row of $\mathbf{B}_{jl}$ can be filled ⑩. Once $\mathbf{E}_{jl}$ and $\mathbf{B}_{jl}$ are found, $\mathbf{J}$ can be computed ⑪.

4. Numerical Test

First, we will show in ch. 4.1 how well the algorithm works in comparison to a Brute force Jacobian. In ch. 4.2, we show profiling results.

4.1. 1Ωm half space example

To test if this new approach of Jacobian computation works, we generated a small inversion grid of $3 \times 6 \times 13$ inversion grid cells of 20 m size (see. Fig.5). The source and receiver are inline and have a 100 m offset.

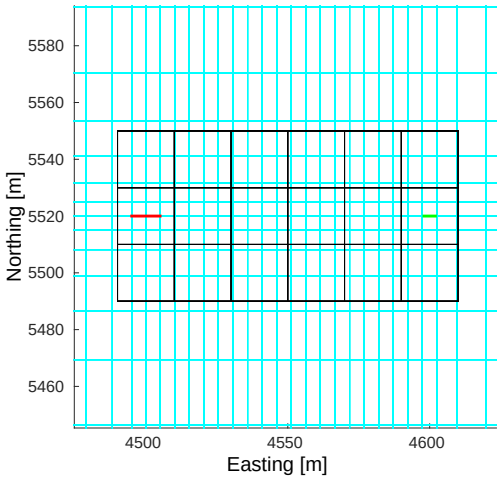

Figure 5: Small inversion grid (black grid) used for test. It consists of $6 \times 3 \times 13$ 20 m cubes. The forward grid (cyan) is regular between source (red) and receiver (green) at a 100m offset.

To compare if the MOR-Jacobian is correct, we computed the brute force Jacobian according to Eq. (5). Both Jacobians are depicted in Fig. 6. The

sensitivities are plotted for 0.0 to 1.1 seconds and all model parameters. The model parameters have a model cell index that can be computed by $idx = N_x \times N_y \times k + N_x \times j + i$, for $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $k = 1, \ldots, N_z$. N_x, N_y and N_z is the size of the inversion grid. For the MOR-Jacobi, a Krylov dimension of 28 has been used.

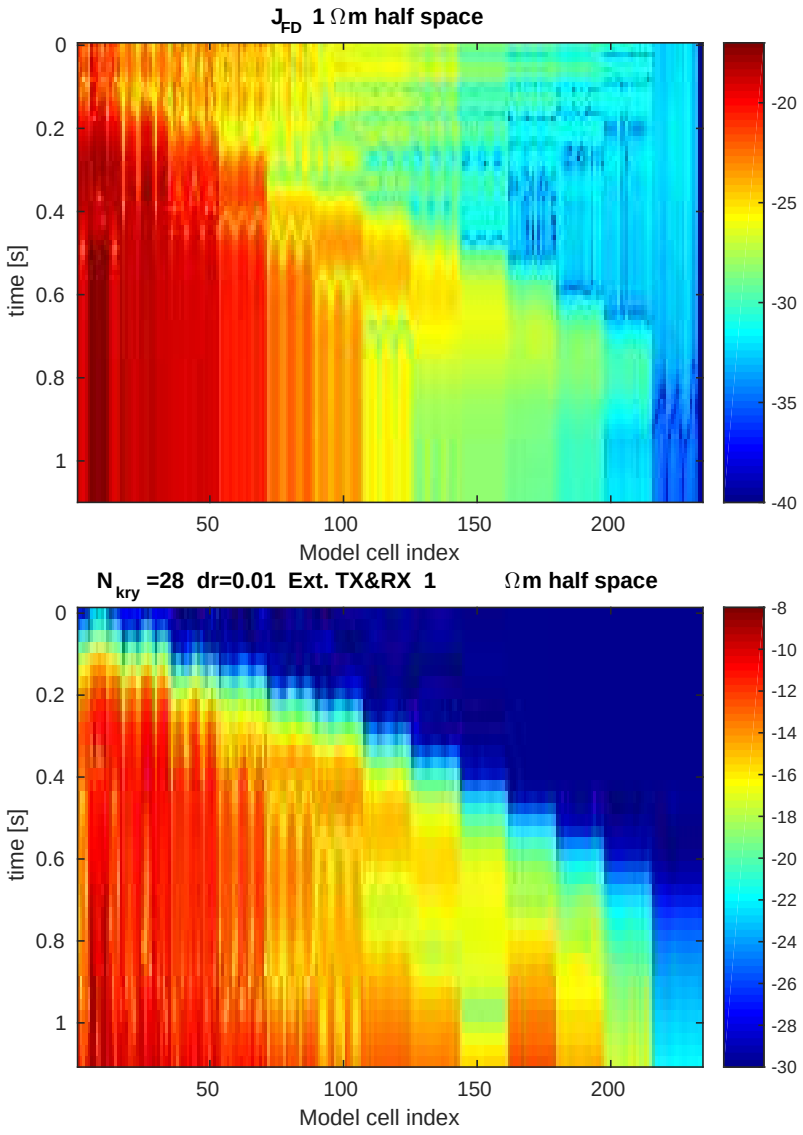

Figure 6: Brute forced Jacobian (top) and MOR-Jacobian (bottom) for the inversion grid depicted in Fig.5.

The first six inversion grid cells correspond to the southern line of black

inversion grid cells in Fig. 5. Index 7 to 12 to the center line and 13 to 18 to the northern line. Indices 18 to 36 reveal the sensitivities of the next deeper level of inversion grid cells. In general, the sensitivities of the middle line are most sensitive, because they are in-between source and receiver. With increasing depth, the sensitivities appear at later times. The upper graphic depicts the sensitivity computed by the brute force method. The brute force method is not very accurate, if the inversion grid cells are small. A sensitivity unequal zero between 0.0 and 0.1 seconds, as seen in the FD-Jacobian, is physically not possible. Therefore, getting identical FD-. and MOR-Jacobians is not a requirement. We should rather ask about similarities. The MOR-Jacobi, on the other hand, does not produce artifacts at early times, but at saturation level, where the electric field becomes constant over time, the sensitivities become very high. This numerical instability is not problematic, because sensitivities at saturation level are not of importance.

4.2. Run time and profiling

An important issue about 3D inversion are run times, especially if it is implemented on GPU. To discuss run times, we profiled the code by running it on a GeForce Titan Black graphics card from Nvidia, compiled with CUDA-7.0, for a representative inversion grid of 2000 model cells. The complete run time was $6\frac{1}{2}$ minutes, for $N_{kry} = 26$.

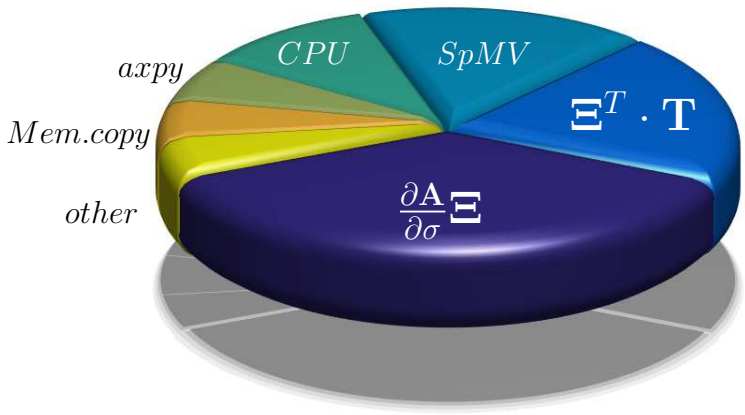

Figure 7: Pie chart showing the portion of run times of the most time consuming functions.

The pie chart in Fig. 7 reveals that about 90% of the code run times correspond to functions running on GPU. A typical bottleneck of GPU computations is the memory transfer between CPU and GPU domain. But, despite of the memory transfer of $\frac{\partial \mathbf{A}}{\partial \sigma}$ constructed in the CPU domain as depicted in Fig.4, memory transfer takes only around 5% of the run time. Computationally most expensive is the multiplication of the sparse and dense matrix $\frac{\partial \mathbf{A}}{\partial \sigma}\Xi$. The

dense matrix times dense matrix multiplication $\mathbf{\Xi}^T \cdot \mathbf{T}$ takes around 20% of run time, such that the two most time consuming algorithms are both required for the derivation of the Ritz-pairs. The sparse matrix times vector multiplications (SpMV) are required in the RKSM algorithm for spanning the Krylov vectors. If N_{mod} is higher (here it was 2000), its portion will decrease. The *axpy* operations are vector additions and require only limited run time.

5. Application to a real data set

We will demonstrate in this chapter, how the sensitivity calculation is applied in practice.
In January 2014, a CSEM data set was collected in the Danube delta fan of the Western Black Sea to identify submarine gas hydrate deposits. Seismics revealed a bottom simulating reflector, that is commonly used as indicator for the presence of gas hydrates. It appears at a depth of 100mbsf (meters below sea floor).

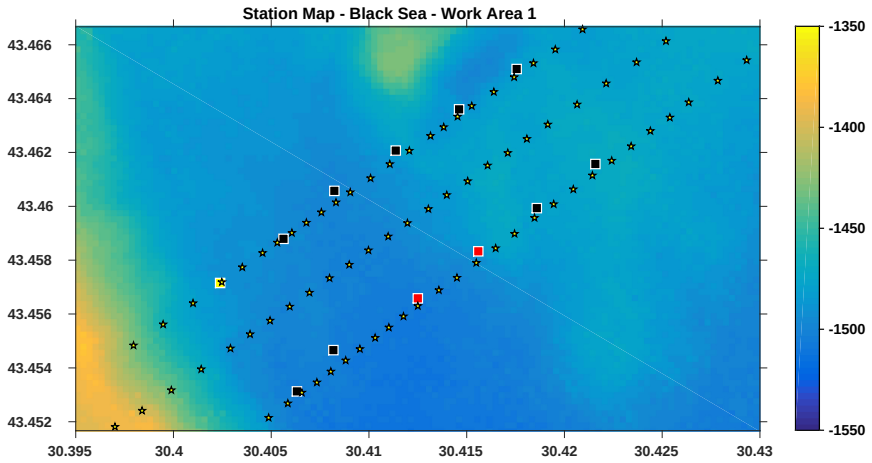

Figure 8: Station map of a real CSEM data set. Squares indicate the locations of receivers, stars the locations of transmissions.

Fig. 8 illustrates the survey layout. Twelve receivers (squares) were placed along two profiles and from 76 transmission stations (stars) on three profile lines of 2700 meters length, signals were emitted in both horizontal directions.
For inversion, an inversion grid of size $74 \times 190 \times 63$ with cell sizes of $20m \times 20m \times 20m$ has been defined and rotated parallel to the profile lines. Because the Jacobian for every Tx/Rx combination is computed separately, only a subset of cells are effected by one Tx/Rx combination, such that the sensitivities are only computed for cells in a certain vicinity of source and receiver.

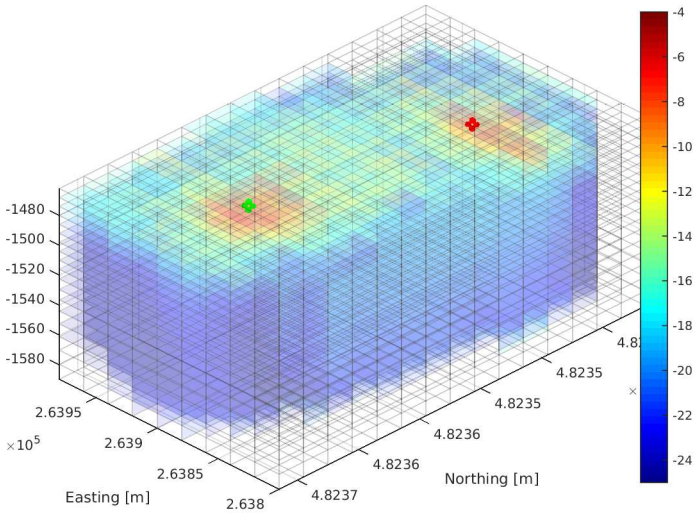

Figure 9: Sensitivity for a model desinged according to a real data set and one Tx/Rx combination for a subset of the inversion grid. Here, no gas hydrates are included in the model.

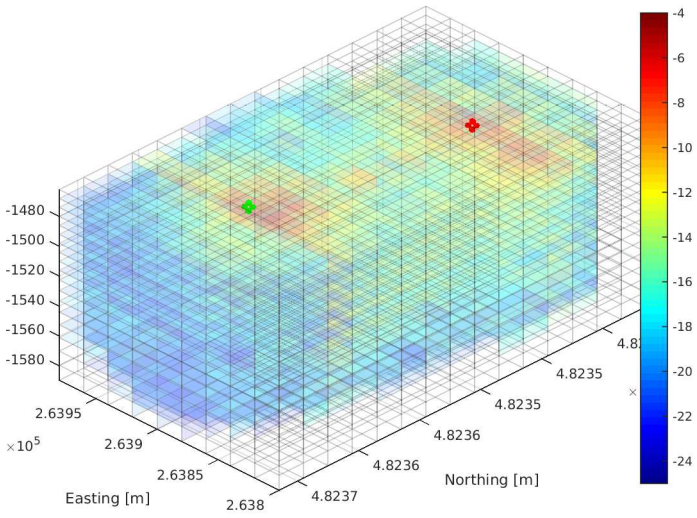

Figure 10: Identical model as illustrated in Fig. 9, but with a resistive ($20\Omega m$) gas hydrate layer in the upper sediments.

This is demonstrated in Fig. 9, where the sensitivities of a subset of size $9 \times 15 \times 18$ were computed. At the borders of this subset, sensitivities are around zero. This reduces computation time drastically. The sensitivities in Fig. 9 were computed for a model with a sub seafloor resistivities of $1\Omega m$. Cells around source and receiver's positions are most sensitive. To demonstrate that the method is sensitive to gas hydrates, the same calculation was repeated for a model with a resistive layer of $10\Omega m$ (Fig. 10). The resistor is guiding the electric field faster than the surrounding conductive structures, such that the higher sensitivities are distributed much broader.

6. Conclusion

7. Acknowledgment

We especially thank Michael Zaslavsky. Without his advise, this work could not have been done.

References

[1] J. Hesthammer, A. Stefatos, M. Boulaenko, A. Vereshagin, P. Gelting, T. Wedberg, G. Maxwell, CSEM technology as a value driver for hydrocarbon exploration, Marine and Petroleum Geology 27 (2010) 1872–1884.

[2] K. Schwalenberg, M. Haeckel, J. Poort, M. Jegen, Evaluation of gas hydrate deposits in an active seep area using marine controlled source electromagnetics: Results from Opouawe Bank, Hikurangi Margin, New Zealand, Marine Geology 272 (2010) 79–88.

[3] A. Zerilli, T. Labruzzo, M. Zanzi, M. P. Buonora, J. L. Crepaldi, P. de Tarso Luiz Menezes, Broadband marine CSEM: New benefits for subsalt and around salt exploration, in: SEG Technical Program Expanded Abstracts 2014, 2014, pp. 750–754. arXiv:http://library.seg.org/doi/pdf/10.1190/segam2014-1201.1, doi:10.1190/segam2014-1201.1.
URL http://library.seg.org/doi/abs/10.1190/segam2014-1201.1

[4] A. Swidinsky, S. Hölz, M. Jegen, On mapping seafloor mineral deposits with centeral loop transient electromagnetics, Geophysics 77 (3) (2012) E171–E184.

[5] S. Constable, L. J. Srnka, An introduction to marine controlled-source electromagnetic methods for hydrocarbon exploration, Geophysics 72 (2) (2007) WA3–WA12. doi:{10.1190/1.2432483}.

[6] J. Yuan, R. N. Edwards, The assessment of marine gas hydrates through electronic remote sounding: Hydrate without a BSR?, Geophysical Research Letters 27 (16) (2000) 2397–2400.

[7] S. Hölz, M. Jegen, Development of a CSEM system for the electromagnetic study of the North Alex Mud Volcano, in: EGU General Assembly Conference Abstracts, Vol. 11 of EGU General Assembly Conference Abstracts, 2009.

[8] A. Tarantola, Inverse Problem Theory: methods for data fitting and model parameter estimation, Elsevier Science Publishing Co., 1987.

[9] M. Zaslavsky, V. D. A. Abubakar, T. Habashy, V. Simoncini, Large-scale gauss-newton inversion of transient controlled-source electromagnetic measurement data using the model reduction framework 78.

[10] M. Sommer, S. Hölz, A. Avdeeva, M. Jegen, Comparison of Polynomial- and Rational Krylov Subspace methods for marine CSEM on GPU, Journal of Computational Physics submitted.

[11] A. Hördt, Calculation of electromagnetic sensitivities in the time domain, Geophysical Journal International 133 (1998) 713–720.

[12] K. Yee, Numerical Solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Transactions on Antennas and Propagation AP-14 (1966) 302–307.

[13] M. Sommer, S. Hölz, M. Moorkamp, A. Swidinsky, B. Heincke, C. Scholl, M. Jegen, GPU parallelization of a three dimensional marine CSEM code, Computers & Geosciences (58) (2013) 91–99.

[14] K. Árnason, Consistent discretization of electromagnetic fields and transient modeling, in: M. Oristaglio, B. Spies, M. R. Cooper (Eds.), Three-Dimensional Electromagnetics, SEG, 1999, pp. 103–118.

[15] V. Druskin, L. Knizhnerman, Spectral approach to solving three-dimensional Maxwell's equations in the time and frequency domain, Radio Science 29 (1994) 937–953.

[16] A. Ruhe, Rational Krylov sequence methods for eigenvalue computation, Linear Algebra and its Applications 58 (1984) 391–405.

[17] V. Druskin, V. Simoncini, Adaptive Rational Krylov Spaces for Large-Scale Dynamical Systems, Systems & control Letters 32 (2011) 2485–2496.

[18] G. H. Golub, C. F. Van Loan, Matrix Computations, 3rd Edition, John Hopkins University Press, Baltimore, 1996.

[19] C. D. Meyer, G. W. Stewart, Derivatives and Perturbations of Eigenvectors, SIAM Journal on Numerical Analysis 25 (1988) 679–691.

Summary & Conclusion

This thesis has two main achievements: 1. the development of fast time domain CSEM forward codes and 2. the implementation of a sensitivity calculation.
For the implementation of the forward code, following features are essential. Based on central finite differences on Yee-cells, the spectra of the solution space was computed with the polynomial Krylov subspace method. The time consuming operations were parallelized on Graphics Processing Units (GPUs). The surface boundary condition was solved by inserting a resistive air layer. This simplified the matrix, such that the Krylov subspace could be computed more efficiently. The accuracy of this boundary condition was carefully investigated. Comparisons of run times with CPU based codes revealed an enormous speed-up due to the use of Krylov methods as well as the computational power of GPUs.
In a next step, the polynomial Krylov subspace method was replaced by the rational Krylov subspace method. This was necessary for two reasons. First, it is faster than the polynomial approach because the dimension of the subspace is drastically reduced. This also eliminates the run time of an eigensolver that take a quite significant part of costs in the polynomial approach. Second, it made the sensitivity calculation, in the way how it is presented in this work, possible. The polynomial and rational method were carefully compared with each other. The rational method turned out to be faster. This work is described in.
The sensitivity calculation was implemented according to the Model Order Reduction (MOR) framework. Thereby, the rational Krylov subspace method was changed to its block version. Other computationally expensive parts, like the derivative calculation of the spectra, were parallelized. The result was compared with a brute force solution of a simple model and simple geometry, to demonstrate that the approach works. Finally, the applicability of the method for a real data set is demonstrated.

Outlook

A major improvement of the Rational Krylov Subspace based forward code would be the development of a proper preconditioner. This could reduce the number of iterations in the CG-solver significantly and reduce computational runtimes. But in general, the code can be applied to interpret marine CSEM data sets in time domain by forward modeling.
The sensitivity calculation method, presented in this thesis, will be tested further for different models to validate the broadness of its applicability. A sensitivity matrix alone can be used to compute model- or data resolution matrices. But the main motivation to compute a Jacobian matrix is its implementation as part of a 3D inversion. Currently, the Jacobian presented here, is used as part of a Joint inversion in cooperation with TERRASYS. The development of a single inversion will be the next big challenge.