6th International Symposium on Three-Dimensional Electromagnetics
Berkeley, California, March 28–30, 2017

Implementation of the Rational Krylov subspace method for marine tCSEM forward modeling and sensitivity calculation on GPU

M. Sommer[1], S. Hölz[1] and M. Jegen[1]
[1]GEOMAR Helmholtz Centre for Ocean Research Kiel

Summary

We implemented the Rational-Krylov-Subspace (RKS) Algorithm on graphics cards for marine 3D time domain CSEM forward modeling as well as sensitivity calculation. We present a comparison between the run-time of a) the new code and the older Polynomial-Krylov-Subspace method code and b) with run-times on old and new graphics cards (to quantify the improvements by hardware and by algorithms). We show, that both together improve the performance significantly by a factor of 20 in comparison with an old GPU parallelized code.
For sensitivity computation, we expanded the implementation of the RKS algorithm to block spaces and implemented a model reduction framework, instead of the commonly used adjoint method. Comparisons with brute-forced Jacobians validate the approach.

Keywords: forward modeling, sensitivity, finite difference, marine CSEM, GPU, Rational Krylov Subspace

Introduction

For a marine 3D time domain CSEM system developed at GEOMAR (Hölz, Swidinsky, Sommer, Jegen, & Bialas, 2015) we implement 3D forward codes (Sommer et al., 2013) and are currently developing a 3D inversion. Problematic in 3D problems are high run-times. There are two ways to make codes faster, chose better algorithms or better hardware. We tried both, by implementing the Rational Krylov subspace method on Graphic Processing Units (GPUs) and running it on different cards.
For 3D-tCSEM, implicit solution strategies of the EM-diffusion equation, like the Lanczos algorithm first introduced by (Druskin & Knizhnerman, 1994), became popular. In difference to time stepping, where every computed E-field depends on the full space solution of the previous time step, implicit solvers compute the electric field at any, arbitrary point in time. In a logarithmic time space, the E-field computation at many small, time steps can therefore be skipped. Lanczos algorithms are based on Krylov-spaces, a subspace approximation of the solution space. Next to polynomial Krylov spaces, much effort has been spent on the research of optimal Rational Krylov Space (RKS)((Börner, Ernst, & Spitzer, 2008),(Knizhnerman, Druskin, & Zaslavsky, 2009), (Druskin, Lieberman, & Zaslavsky, 2010), (Druskin & Simoncini, 2011), (Zaslavsky, Druskin, & Knizhnerman, 2011), (Börner, Ernst, & Güttel, 2014)). Typically in RKS the approximating dimension and therefore the number of iterations is drastically reduced, whereby every iteration itself becomes more expensive.
In terms of hardware, high performance GPUs are now available. While *Moore's Law* is not valid for CPUs anymore, the FLOPS of GPUs still increase. Here, we test our implementation on an old graphics card and a new one to quantify the improvement by hardware alone.

Solution strategy of the forward problem

The general solution strategy for polynomial Krylov spaces is described in (Druskin & Knizhnerman, 1994) and we describe it very shortly here. The spatial operators of the diffusion equation:

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

becomes discretized by central finite differences $\vec{\nabla} \times \vec{\nabla} \times (\sigma\mu)^{-1}\vec{E} \Rightarrow \mathbf{A}\vec{E}$, with $\mathbf{A} \in \mathbb{R}^{N\times N}$ and N being the model size, such that (1) is changed from a PDE to an ODE. Instead of discretizing time (like in explicit strategies) an exponential Ansatz function is chosen and spectrally decomposed:

$$\vec{E} = \vec{E}_0 exp(-t\mathbf{A}) = \sum_{i=1}^{N} \vec{z}_i exp(-t\lambda_i)\vec{z}_i^T\vec{E}_0 \tag{2}$$

with $\vec{E}_0$ denotes the initial source field and $\lambda_i, \vec{z}_i^T$ eigenpairs of $\mathbf{A}$. Since computation of eigenpairs is too expensive, $\mathbf{A}$ is projected by an orthonormalized

Krylov space $\mathbf{Q} \in \mathbb{R}^{\mathbf{N}\times\mathbf{k}}$ to the Ritz approximation $\mathbf{H}$:

$$\mathbf{QAQ}^T = \mathbf{H} \in \mathbb{R}^{k\times k} \tag{3}$$

Eigenpairs of $\mathbf{H}$ approximate those of $\mathbf{A}$ and can be used in (2). For $k \ll N$, the computation becomes much more efficient.
To compute $\mathbf{Q}$, it is very common to span the column vectors by the polynomial Krylov space:

$$\mathcal{K}^k_{poly}(\mathbf{A}, \vec{q}_0) = span(\vec{q}_0, \mathbf{A}\vec{q}_0, \mathbf{A}^2\vec{q}_0, \ldots, \mathbf{A}^{k-1}\vec{q}_0) \tag{4}$$

(Ruhe, 1984) has shown, that the most optimal subspace to approximate eigenspaces is the Rational Krylov subspace (RKS):

$$\mathcal{K}^k_{rat}(\mathbf{A}, \vec{q}_0, \vec{s}) = \tag{5}$$

$$span\left\{(\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_0, \ldots, \prod_{i=1}^{k}(\mathbf{A} - s_i\mathbf{I})^{-1}\vec{q}_0\right\}. \tag{6}$$

Instead of performing many matrix times vector multiplications like in (4), the matrix is shifted by poles $s_1, \ldots, s_k$ and inverted. The dimension of $\mathcal{K}_{rat}$ is much smaller than $\mathcal{K}_{poly}$ (between 14 to 22 instead of 1000 to 4000). For our implementation, $\vec{q}_0$ corresponds to the initial electric field $\vec{E}_0$.

Implementation on GPU

The implementation of the polynomial method on GPU, described in (Sommer et al., 2013), gave a significant speedup compared with a CPU optimized code from Schlumberger (compare black & green line in Fig.1 above). The Krylov-dimension k was set to values between 1000 and 4000. Therefore, the eigenpairs of $\mathbf{H}$ of this size had be to solved by an eigensolver of the CULA library.
For spanning the RKS, the matrix had to be shifted and inverted, which was achieved by using Conjugate Gradients (CG). Many ways of preconditioning were tried, but turned out to be suboptimal. It can be shown, that even without preconditioning, a good speedup can be achieved.
We investigated run-times of our code for and old and new graphics card, to quantify the progress in hardware development. Run-times for our implementation and other codes are depicted in Fig.1. The black curve corresponds to the CPU optimized code *sldmem* from Schlumberger, the green one to the polynomial method on an old GPU, the blue one to the new code on an old card and the red one to the new code on a new card. The figure below shows speedups for the different codes and hardware with respect to each other. Run-time for this model is between 1 to 3 seconds. Compared with our previous code on the old GPU, the speedup is constant around 20. A new GPU reduces run-time by a factor of 3.

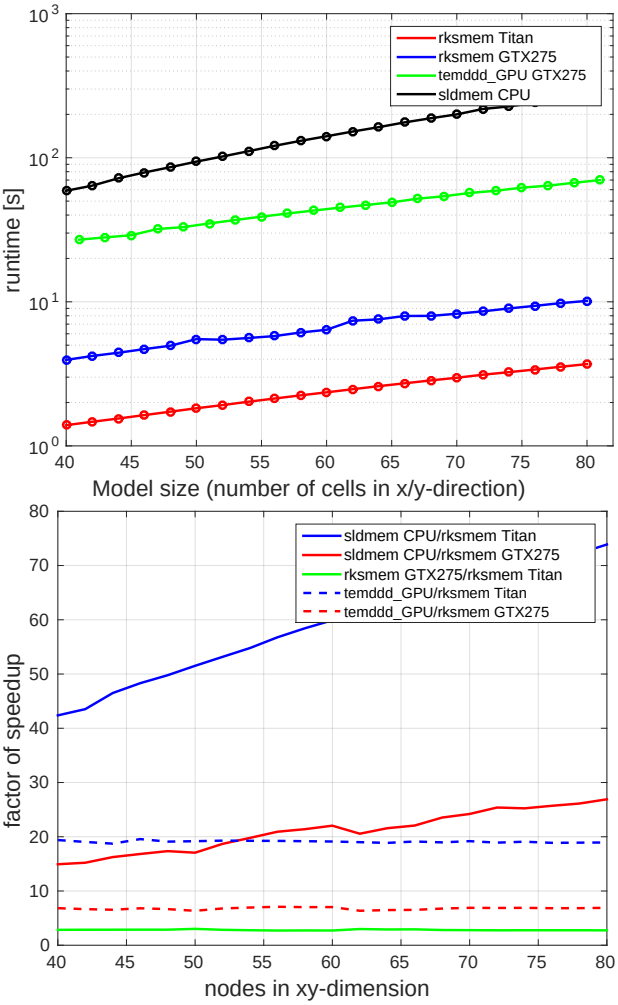

Figure 1: Up: run-times of *sldmem* (black), *temddd_GPU* on GTX275 (green), *rksmem* on GTX275 (blue) and *rksmem* on Titan (red) in dependence of model sizes in x/y-direction between 40 and 80. Krylov dimension was set to 3000 for SLDM based codes and to 20 for RKSM based codes. Down: Speedups of the codes to each other, to reveal the impact of different algorithms and architectures on runtime.

Sensitivity calculation

The implementation of the aforementioned RKS algorithm was extended to calculate sensitivities $\mathbf{J} \in \mathbb{R}^{N_{model} \times N_{data}}$. We followed thereby an approach of (Zaslavsky et al., 2011). There, a derivative is applied to the Ansatz eq.(2) (which gives the electric field) with respect to conductivity ($\sigma = \sigma_{i,j,k}$), yielding, by definition, the Jacobian

$$\mathbf{J} = \frac{\partial \vec{E}}{\partial \sigma} \approx \mathbf{E}^T \mathbf{B} \tag{7}$$

with $\mathbf{E} = \mathbf{E}(\vec{t}, \vec{\lambda}, \vec{z}_1, \ldots, \vec{z}_k) \in \mathbb{R}^{N_{data} \times 2k}$ and $\mathbf{B} = \mathbf{B}(\vec{\lambda}, \vec{z}_1, \ldots, \vec{z}_k, \frac{\partial \vec{\lambda}}{\partial \sigma}, \frac{\partial \vec{z_1}}{\partial \sigma}, \ldots, \frac{\partial \vec{z_k}}{\partial \sigma}) \in \mathbb{R}^{N_{model} \times 2k}$. Whereby $\vec{\lambda}$ and $\vec{z}_1, \ldots, \vec{z}_k$ are eigenpairs of the corresponding Ritz approximation $\mathbf{H}$. The calculation of the derived Ritz-eigenpairs is deduced from the Rayleigh quotient formula:

$$\frac{\partial \lambda_i}{\partial \sigma} = \vec{z}_i^T \frac{\partial \mathbf{A}}{\partial \sigma} \vec{z}_i \qquad \frac{\partial \vec{z}_j(l)}{\partial \sigma} = \sum_{j=1, j \neq i}^{N} \frac{\vec{z}_j^T \frac{\partial \mathbf{A}}{\partial \sigma} \vec{z}_i}{\lambda_i - \lambda_j} \vec{z}_j(l) \tag{8}$$

with l being the element index.
The computation of the Ritz-pairs is done by a Block-Krylov-Space (BKS)

$$\begin{aligned} &\mathcal{K}^k_{block}(\mathbf{A}, \vec{q}_0, \vec{s}) = \\ &span\{(\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_0, \ldots, (\mathbf{A} - s_1\mathbf{I})^{-1}\vec{q}_m, \\ &\prod_{i=1}^{k}(\mathbf{A} - s_i\mathbf{I})^{-1}\vec{q}_0, \ldots, \prod_{i=1}^{k}(\mathbf{A} - s_i\mathbf{I})^{-1}\vec{q}_m\} \end{aligned}$$

whereby $\vec{q}_0$ is the initial source field and $\vec{q}_1, \ldots, \vec{q}_m$ are canonical vectors corresponding to receivers locations, for m receivers. The BKS preserves the adjoint principle.

Implementation of Sensitivity calculation

The computation of the BKS was implemented in a similar way like the RKS for the forward problem. Most run-time is spent for deriving the Ritz-pairs. Writing the Ritz vectors in a matrix $\mathbf{Z} = [\vec{z}_0, \ldots, \vec{z}_k]$ allows to compute all derivatives with eq.(8) through multiplying a sparse with a dense matrix ($\frac{\partial \mathbf{A}}{\partial \sigma} \cdot \mathbf{Z} = \mathbf{R}$) and multiplication of the resulting matrix $\mathbf{R}$ with another dense matrix ($\mathbf{Z}^T \cdot \mathbf{R}$).
First tests show, that Jacobians computed in this way looks very similar to brute-forced Jacobians. Problematic is the high run-time of eq.(8), and the memory transfer of $\frac{\partial \mathbf{A}}{\partial \sigma}$ on the GPU. A possible remedy to this problem would be to solve $\frac{\partial \mathbf{A}}{\partial \sigma}$ on GPUs.

Acknowledgments

Benchmark results for the *sldmem* code were provided by courtesy of the University of Toronto. We thank Michael Zaslavsky for helpful advises about the sensitivity calculation. This work was done within the SUGAR project.

References

Börner, R. U.-., Ernst, O. G., & Güttel, S. (2014). Three-Dimensional Transient Electromagnetic Modeling Using Rational Krylov Methods. *The University of Manchester*.

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

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.

Druskin, V., Lieberman, C., & Zaslavsky, M. (2010). On Adaptive Choice of Shifts in Rational Krylov Subspace Reduction of Evolutionary Problems. *siamj*, *32*, 2485-2496.

Druskin, V., & Simoncini, V. (2011). Adaptive Rational Krylov Spaces for Large-Scale Dynamical Systems. *sacl*, *32*, 2485-2496.

Hölz, S., Swidinsky, A., Sommer, M., Jegen, M., & Bialas, J. (2015). 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), 224.

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

Ruhe, A. (1984). Rational krylov sequence methods for eigenvalue computation. *LINEAR ALGEBRA AND ITS APPLICATIONS*, *58*, 391–405.

Sommer, M., Hölz, S., Moorkamp, M., Swidinsky, A., Heincke, B., Scholl, C., et al. (2013). GPU parallelization of a three dimensional marine CSEM code. *Computers & Geosciences*, *(58)*.

Zaslavsky, M., Druskin, V., & Knizhnerman, L. (2011). Solution of 3d time-domain electromagnetic problems using optimal subspace projection. *geophys*, *76*.