Solving the three-dimensional linear magnetometry problem on GPU

*E. N. Akimova (Ural Federal University and Krasovskii Institute of Mathematics and Mechanics of UrB RAS), V. E. Misilov (Ural Federal University and Krasovskii Institute of Mathematics and Mechanics of UrB RAS), V. O. Miftakhov (Ural Federal University)

SUMMARY

The paper is devoted to construction and implementation of the cost-efficient algorithm for solving the three-dimensional problem of finding the vertical component of the magnetic field generated by a rectangular parallelepiped with a variable unidirectional magnetization. The algorithm uses the Toeplitz-block-Toeplitz structure of the matrix of the discretized problem to reduce the amount of calculations and computing time. The algorithm was implemented on GPU using the CUDA technology. A series of numerical experiments were performed to study the efficiency of the parallel algorithm.
Statement of the Forward Magnetometry Problem

Let us introduce a Cartesian coordinate system where plane x0y coincides with the Earth’s surface, and axis z is directed downwards (Fig. 1). Assume that there are no magnetization anomalies outside the parallelepiped \( \Pi = \{(x, y, z): a \leq x \leq b, c \leq y \leq d, e \leq z \leq f\} \), and the magnetization \( M(x, y, z) \) inside this parallelepiped is unidirectional. Thus, we can let \( M(x, y, z) = M(x, y, z) \cdot \mathbf{M} \), where \( \mathbf{M} = (\mathbf{M}_x, \mathbf{M}_y, \mathbf{M}_z) \) is a given unit vector of the magnetization direction and \( M(x, y, z) \) is a given intensity of magnetization.

The problem is to find vertical magnetic field \( B_z(x, y, 0) \) measured at Earth’s surface generated by the body \( \Pi \).

The field generated by parallelepiped \( \Pi \) is given by the following equation (Blakely, 1996; Martyshko, 1999):

\[
B_z(x, y, z) = C_m \int_{\Pi} M(x', y', z') \varphi(x, y, z, x', y', z') dx' dy' dz',
\]

where \((x, y, z)\) is the observation point outside of \( \Pi \), \((x', y', z')\) is the point of integration within \( \Pi \), and \( C_m = \frac{\mu_0}{4\pi} = 10^{-7} \text{ H/m} \) is the magnetic constant.

Discretization of magnetized parallelepiped model

The approximation for the integral operator of formula (1) is constructed as follows. The discretization of the model is performed by dividing the variables of integration into the rectangular grid \( n \times m \times p: (x_i, y_j, z_k), i = 1..n, j = 1..m, h = 1..p \) with steps \( \Delta x, \Delta y, \) and \( \Delta z \) for the body \( \Pi \) as shown in Fig. 2. Magnetization values are defined by grid function \( M_{ijk} \).

Let us assume that each rectangular parallelepiped (cuboid) \( K_{ijh} = \{(x, y, z): x_i \leq x \leq x_{i+1}, y_j \leq y \leq y_{j+1}, z_h \leq z \leq z_{h+1}\} \) have constant magnetization \( M_{ij} = M(x_i, y_j, z_h) \). Then, the magnetic field at point \((x_k, y_l, 0)\) is given by

\[
B(x_k, y_l, 0) = B_{kl} = \sum_{i=1..n} \sum_{j=1..m} \sum_{h=1..p} M_{ijh} \varphi_{ijkl}, \quad k = 1..n, l = 1..m,
\]
where $\varphi_{ijkl}$ is the magnetic field at point $k, l$ due to cell $i, j, h$ with unit magnetization $\mathbf{M}$ (see Akimova, 2018; 2019):

$$
\varphi_{ijkl} = -C_m \left[ M_x \ln \left( (\eta - y_l) + \sqrt{(\xi - x_k)^2 + (\eta - y_l)^2 + \zeta^2} \right) + \\
+ M_y \ln \left( (\xi - x_k) + \sqrt{(\xi - x_k)^2 + (\eta - y_l)^2 + \zeta^2} \right) + \\
+ M_z \tan^{-1} \left( \frac{(\xi - x_k)(\eta - y_l)}{\sqrt{(\xi - x_k)^2 + (\eta - y_l)^2 + \zeta^2}} \right) \right],
$$

(3)

We can now rewrite equation (2) in the matrix form

$$
Au = f,
$$

where matrix $A$ of dimension $mn \times mnp$ consists of elements $a_{h,m,n+j,n+i,l,n+k} = \varphi_{ijkl}$, $i = 1..m, j = 1..n, h = 1..p, k = 1..m, l = 1..n$, magnetization vector $u$ of dimension $mnh$ consists of elements $u_{h,m,n+j,n+i} = M_{ij}$ is magnetizations vector, and field values vector $f$ of dimension $mn$ consists of elements $f_{j,n+i} = B_{ij}$.

**Optimization of the algorithm**

In work (Akimova, 2019), we constructed and implemented the cost-efficient technique for calculating and storing the coefficient matrix for the case of one z-layer, i.e., $p = 1$. This technique uses the fact that the coefficients $\varphi$ are equal for the same spatial configuration of the observation point $(k, l)$ and grid element $(i, j)$. It means that the matrix $A$ is Toeplitz-block-Toeplitz one, and we need to compute and store only $(2m - 1)(2n - 1)$ elements while full storage required $m^2n^2$.

For the case of three-dimensional grid, the matrix $A$ will consist of several square blocks, each of them corresponding to one z-layer. Each of these blocks is also a Toeplitz-block-Toeplitz matrix. Thus, instead of calculating and storing $m^2n^2p$ elements, we need only $(2m - 1)(2n - 1)p$. 
Parallel implementation and numerical experiments

For solving the forward problem, the parallel algorithm was developed for graphics processors using CUDA technology. We use the matrix $A$ in a «compressed» form, so, only unique elements are computed. The computation consists of four nested loops. Outer ones are distributed between CUDA threads. After this, the computed matrix is multiplied by the magnetization intensity vector $u$. In this operation, the matrix columns are distributed between threads.

The model problem of finding the magnetic field generated by parallelepiped of $200 \times 200 \times 11$ km$^3$ located between depths of $H_1 = 10$ km and $H_2 = 21$ km. The magnetization direction is $\vec{M} = (0.71; 0.71; 0.71)$ A/m.

To study the performance of the developed algorithm, we have constructed a set of test problems with various grid sizes: $64 \times 64 \times 64$, $128 \times 128 \times 128$, and $256 \times 256 \times 256$.

The intensity of magnetization $M(x, y, z)$ is shown in Fig. 3. The computed magnetic field $B_z$ is shown in Fig. 4.

![Figure 3: Synthetic intensity of magnetization](image1)

![Figure 4: Synthetic magnetic field](image2)
The table contains the computing times for model problem with various grid sizes for three implementations.

1) Unoptimized serial algorithm implemented on the Intel Xeon E5-2650 processor.
2) Optimized serial algorithm (using TBT structure of matrix) implemented on the Intel Xeon E5-2650 processor.
3) Optimized parallel algorithm on the NVIDIA Tesla K40 GPU.

**Table** Computing times of parallel algorithm

<table>
<thead>
<tr>
<th>Size of problem</th>
<th>Computing time on Xeon E5-2650 (without TBT), seconds</th>
<th>Computing time on Xeon E5-2650 (using TBT), seconds</th>
<th>Computing time on Tesla K40 (using TBT), seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>64³</td>
<td>150</td>
<td>0.5</td>
<td>0.22</td>
</tr>
<tr>
<td>128³</td>
<td>4940</td>
<td>10.3</td>
<td>4.7</td>
</tr>
<tr>
<td>256³</td>
<td>158000</td>
<td>353.7</td>
<td>109</td>
</tr>
</tbody>
</table>

**Conclusion**

The cost-efficient algorithm is developed for solving the forward linear three-dimensional problem of finding the magnetic field by parallelepiped with unidirectional magnetization which is variable by intensity. The algorithm exploits the Toeplitz-block-Toeplitz structure of the coefficients matrix to significantly reduce the computing time. The parallel algorithm for solving the problem is constructed and implemented on the GPU using CUDA technology. A series of model problems were solved.

Taking into account the Toeplitz-block-Toeplitz structure of the matrix reduces the computing time by five hundred times. The computing time on GPU is three times lower than CPU for the grid size $256 \times 256 \times 256$. Later, we plan to use the developed algorithm to implement the method for solving the inverse magnetometry problem of finding the magnetization intensity using known magnetic field.

**References**


Akimova, E. N., Misilov, V. E., Miftakhov, V. O. [2018] A numerical technique for solving the linear forward magnetic problem for a horizontal layer with variable magnetization. *17th International Conference on Geoinformatics - Theoretical and Applied Aspects*. [https://doi.org/10.3997/2214-4609.201801861](https://doi.org/10.3997/2214-4609.201801861)
