

Fast CUDA Geomagnetic Map Builder

Delia Spridon(✉), Adrian Marius Deaconu, and Laura Ciupala

Department of Mathematics and Computer Science, Transilvania University, Brasov, BV 0500036, Romania
{delia.cuza,a.deaconu,laura.ciupala}@unitbv.ro

Abstract. In this paper, we use kriging techniques and inverse distance weighting (IDW) to generate geomagnetic maps in Romania. Kriging is a method of spatial interpolation that calculates unknown values based on statistical analysis of the correlation of known values in space. On the other hand, IDW estimates unknown values using a weighted average of nearby known values. Both methods are used in geophysics to interpolate Earth's magnetic field measurements in areas where no data is available, to fill gaps in existing data, or to generate maps for regions without discrete information. Each method has its advantages and limitations, and the selection of the appropriate method is dependent on factors such as the distribution of known data points and the properties of the underlying geophysical process. In this study, we used these interpolation methods to produce reliable and accurate maps of the Earth's magnetic field over a large spatial area. To speed up the computations and take advantage of the impresive computational power of GPUs, we used CUDA to parallelize the interpolation computations. The use of GPU programming for geomagnetic data interpolation has the potential to provide significant performance advantages, allowing us to analyze large datasets promptly and accurately.

Keywords: CUDA · IDW · kriging · geomagnetism

1 Introduction

Interpolation methods for geomagnetic data are used to estimate magnetic field values at locations where measurements are not available [1]. There are several interpolation methods that have been used for this purpose, each with its own strengths and weaknesses. Some of the interpolation methods for geomagnetic data are kriging, spline interpolation, radial basis functions, IDW, Artificial Neural Networks(ANN).

Kriging is a geostatistical interpolation method that can be used for interpolating discrete geomagnetic data using a variogram to model the spatial correlation of the data and estimates the values at unsampled locations using a weighted average of nearby measurements. Kriging is particularly effective for estimating data values in areas with high spatial variability.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 126–138, 2023.
https://doi.org/10.1007/978-3-031-36805-9_9

Spline interpolation is a non-parametric interpolation method that uses a mathematical function to fit a curve to the data. This method is particularly useful for data that has a smooth spatial variation.

Radial Basis Functions interpolation is a method that uses a set of radial basis functions to interpolate data values at unsampled locations and is particularly effective for data that has a strong directional dependence.

Inverse Distance Weighting (IDW) is a simple interpolation method that estimates values at unsampled locations based on the weighted average of nearby measurements. This method assumes that data values influence decreases with distance from known measurements by assigning higher weights to closer measurements. IDW is particularly useful for estimating values in areas with low spatial variability.

Artificial Neural Networks (ANNs) are machine learning models that can be used to interpolate data [2]. ANNs can learn complex spatial relationships in the data and are particularly useful for estimating values in areas with non-linear spatial variations.

Overall, the choice of interpolation method depends on the characteristics of the data and the specific research question being addressed. The interpolation methods for geomagnetic data continues to evolve, with new methods and approaches being developed and tested.

In this paper, we use kriging and IDW interpolation methods to obtain geomagnetic maps for Romania. The algorithms for this methods are also parallelised by using CUDA arhitecture on GPUs in order to obtain fast, high resolution geomagnetic maps for the studied area. The accuracy of the obtained maps are compared for the two interpolation methods, also the algorithms are analysed taking into account the execution time and the speed-up.

1.1 Geomagnetic Data

Geomagnetic data is the information gathered about the Earth's magnetic field. Geomagnetic data can be collected using a range of methods, including magnetic surveys conducted from aircraft, ships, or ground-based instruments [3,4]. Geomagnetic maps are graphical representations of this data that show variations in the strength and direction of the magnetic field at different locations on the Earth's surface [5]. Geomagnetic data and maps are used in a wide range of scientific, industrial, and commercial applications, from understanding the Earth's structure to identifying potential mineral and energy resources.

Geomagnetic data are essential for understanding the structure and dynamics of the Earth's interior [6]. The magnetic field is generated by the motion of molten iron in the Earth's outer core, and variations in the field can provide insights into the structure and behavior of the region [7]. For example, changes in the strength and direction of the magnetic field can indicate the presence of geological structures such as faults, magma chambers, and mineral deposits [8].

Geomagnetic data are also used to study the Earth's upper atmosphere and its interaction with the solar wind. The magnetic field acts as a shield, deflecting high-energy particles from the sun and protecting the Earth's atmosphere from erosion [9]. However, the magnetic field is not uniform, and variations in its

strength and direction can result in the formation of regions known as the Van Allen radiation belts, where charged particles are trapped and can cause damage to spacecrafts and satellites.

Geomagnetic data are also used in navigation and surveying. The magnetic field provides a reference for compasses and can be used to determine the orientation and position of objects on the Earth's surface. This information is used in a range of applications, from mapping and surveying to navigation and mineral exploration [4].

Geomagnetic maps are graphical representations of the Earth's magnetic field. They show variations in the strength and direction of the field at different locations on the Earth's surface, and can be used to identify geological structures, mineral deposits, and other features [5].

Geomagnetic maps are used in a range of applications, including mineral exploration, where variations in the magnetic field can indicate the presence of mineral deposits. For example, the Sudbury Basin in Canada is a large geological structure that contains a rich deposit of nickel, copper, and other minerals [10]. The basin is marked by a large magnetic anomaly, which was identified using magnetic surveys and helped to guide the exploration and development of the deposit.

Geomagnetic maps are also used in environmental monitoring, where changes in the magnetic field can indicate the presence of pollutants or other contaminants. For example, magnetic surveys have been used to map the extent of oil spills and other environmental disasters, providing valuable information for cleanup and mitigation efforts.

Geomagnetic maps are also used in geophysical research, where they can provide insights into the structure and behavior of the Earth's interior. For example, magnetic anomalies in the oceanic crust have been used to study the process of seafloor spreading and the formation of new oceanic crust.

Geomagnetic data and maps are essential tools for understanding the Earth's magnetic field and its diverse applications. Geomagnetic data provides insights into the structure and dynamics of the Earth's interior, while geomagnetic maps are used for navigation, geological mapping, and scientific research. Geomagnetic data and maps have practical applications in industry and commercial ventures, particularly in mineral exploration, energy development, and navigation.

1.2 Inverse Distance Weight

Inverse distance weighting (IDW) is a widespread adopted interpolation algorithm used to estimate values at unsampled locations based on measured values at nearby sample locations. The algorithm assumes that the values being estimated are a function of distance from the sample locations and that values closer to the sample locations have a greater influence on the estimated value than those farther away.

The basic idea behind IDW is to calculate a weighted average of the measured values at the sample locations, with the weights being proportional to the inverse of the distance between the unsampled location and each sample location. The weights are usually normalized so that they sum to one. The estimated value

at the unsampled location is then the weighted average of the measured values. The IDW algorithm can be expressed mathematically as follows:

Let $V(u)$ be the estimated value at the unsampled location u, $V(i)$ be the measured value at the ith sample location, located at coordinates $(x(i), y(i))$, and let $l(i)$ be the distance between the unsampled location and the ith sample location. Then, the estimated value at the unsampled location can be calculated using the following formula:

$$V(u) = (\sum_{i=1}^{k}[w(i) \cdot V(i)])/(\sum_{i=1}^{k} w(i)) \tag{1}$$

where $w(i) = 1/l(i)^p$, p is a positive exponent that controls the rate at which the weights decrease with distance and k is the total number of sample locations.

There are different ways to choose the value of the exponent p, but the most common values are 2 (inverse square distance weighting) and 3 (inverse cube distance weighting). IDW is a simple and computationally efficient algorithm, but it has some limitations. One of the main limitations is that it assumes that the values being estimated are a function of distance only, and it does not take into account other factors that may influence the values, such as topography, soil type, or land use. Another limitation is that it may produce unrealistic estimates at locations far from the sample locations, especially when the sample locations are unevenly distributed.

Despite its limitations, IDW is widely used in various fields, including geology, environmental science, and geography. It is often used as a baseline method for comparison with other interpolation algorithms, and it is also used for simple applications where high accuracy is not required.

1.3 Kriging

Kriging is a spatial interpolation algorithm that estimates unknown values at unobserved locations based on spatial correlation among observed values. It was first introduced by D. G. Krige in the 1950s s [11] and has since become a popular method for spatial prediction in geostatistics.

The basic idea behind kriging is to estimate the unknown value $z(u)$ at an unsampled location u based on a weighted average of the observed values $z(x_i)$ at nearby sampled locations x_i. The weights are chosen to minimize the prediction error and are determined based on the spatial correlation structure of the data. The kriging weights are obtained by solving a system of linear equations that express the spatial autocovariance function of the data.

The kriging predictor for the unknown value $z(u)$ is given by:

$$z(u) = \sum w_i z(x_i)$$

where w_i are the kriging weights and $z(x_i)$ are the observed values at sampled locations x_i. The kriging weights are obtained by solving the following system of equations:

$$Cw = \lambda$$

where C is the spatial autocovariance matrix between the observed values at sampled locations x_i, w is the vector of kriging weights, and λ is a Lagrange multiplier that enforces the unbiasedness condition on the predictor.

The kriging algorithm can be extended to incorporate additional information, such as trend and measurement error, by modifying the kriging system of equations. There are several variants of kriging, including ordinary kriging, simple kriging, and universal kriging, each with different assumptions about the underlying spatial correlation structure.

Kriging has been widely applied in many fields, including geology, hydrology, ecology, and environmental sciences. It has proven to be a powerful tool for spatial prediction and uncertainty quantification [12,13].

1.4 GPU Programming

Graphics Processing Units (GPUs) are specialized hardware designed to perform complex mathematical computations in parallel, making them an ideal platform for high-performance computing (HPC) applications. Originally designed for graphics rendering, GPUs have been adapted for a variety of scientific, engineering, and financial applications that require large-scale parallelism. In order to take full advantage of GPUs, developers need to have a good understanding of GPU programming and frameworks such as CUDA (Compute Unified Device Architecture). GPU programming involves writing code that runs on both the CPU (central processing unit) and GPU. The CPU is responsible for managing the overall system and launching the GPU kernels, while the GPU is responsible for executing the computations in parallel. CUDA is a parallel computing platform and programming model that enables developers to write code in a language such as C or C++ and then compile and run it on NVIDIA GPUs.

CUDA provides several key features that make GPU programming more accessible to developers. First, it provides a compiler that translates CPU code into GPU code. This compiler also optimizes the code to take advantage of the GPU's architecture and capabilities. Second, CUDA provides a set of libraries that implement common functionality, such as linear algebra, signal processing, and image processing. These libraries are optimized for the GPU, and they can significantly improve the performance of GPU applications [14].

One of the most important concepts in CUDA programming is the kernel. A kernel is a function that executes on the GPU and is designed to perform a specific computation in parallel. Kernels are written in C or C++, and they use CUDA-specific language extensions to access GPU-specific functionality, such as shared memory and thread synchronization. CUDA kernels are executed on a grid of thread blocks, with each block containing multiple threads. The CUDA runtime manages the execution of the kernel across multiple blocks and threads, and it provides mechanisms for thread synchronization and memory management.

One of the main advantages of using CUDA for GPU programming is its ability to achieve significant speedups compared to traditional CPU-based implementations. This is because GPUs have a massively parallel architecture that allows them to execute thousands of threads simultaneously. For example, a GPU with 1,000 processing cores can execute 1,000 threads in parallel, while a CPU with 8 cores can only execute 8 threads in parallel. The increased parallelism of GPUs enables developers to perform computationally intensive tasks much faster than on a CPU. CUDA is widely used in many fields such as scientific computing, machine learning, image processing, and finance. In scientific computing, CUDA is used for performing simulations and modeling of complex physical systems such as fluid dynamics, astrophysics, and climate modeling. In machine learning, CUDA is used to train deep neural networks and perform large-scale data analytics. In image processing, CUDA is used for real-time video processing, image segmentation, and feature detection. In finance, CUDA is used for performing complex financial simulations and risk analysis.

CUDA is supported on a wide range of NVIDIA GPUs, including those in desktops, workstations, and servers. It is also supported on a variety of operating systems, including Windows, Linux, and macOS. NVIDIA provides a comprehensive set of documentation and resources for developers interested in learning CUDA, including online tutorials, sample code, and user forums.

GPU programming has become an essential skill for developers in many fields due to the significant performance gains it offers. CUDA is a popular programming framework that enables developers to leverage the power of NVIDIA GPUs for general-purpose computing tasks. CUDA provides a comprehensive set of programming tools and libraries, and its parallel computing model enables developers to achieve significant speedups compared to traditional CPU-based implementations. With the increasing demand for high-performance computing in various fields, CUDA programming is expected to play an increasingly important role in the future.

Few recent researches show the advantages that CUDA can bring when is used for parallelizing interpolation methods. Thus, in [15], researchers proposed a CUDA-based parallel algorithm for IDW interpolation on big data sets and the results showed that the CUDA-based algorithm outperformed the traditional CPU-based algorithm in terms of computation time and accuracy. An other example is presented in [16], where researchers proposed a CUDA-based parallel algorithm for Kriging interpolation and concluded that the CUDA-based algorithm achieved significant speedups compared to the traditional CPU-based algorithm, especially for large-scale data sets. In [17], researchers proposed a CUDA-based framework for IDW and Kriging interpolation and showed also the time performance and accuracy improvement.

2 Methods

The aim of our paper is to obtain a geomagnetic map for Romania, by using IDW and kriging interpolation methods.The geomagnetic data were obtained from Romanian geomagnetic stations and with help of Physics Toolbox Sensor Suite app in more than 1300 GPS positions spreaded all over the country. The data was gather using Citizen Science action from Researchers Night 2018–2019 Handle with Science European project, financed from H2020, GA no. 818795/2018. The implemented algorithms were tested on a system with Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz, 16.0 GB RAM, NVIDIA GeForce RTX 2060 GPU and Windows 10 Pro operating system. These interpolation methods were accelerated by using CUDA programming for obtaining high resolution geomagnetic maps in real time. This tool could be used for example for monitoring geomagnetic modifications of a large area in order to identify the changes that could take place in Earth's structure or for identifying regions with certain magnetic properties.

The IDW algorithm pseudocode is presented below [18]:

Algorithm 1. IDW algorithm

$dy \leftarrow \frac{y_{max}-y_{min}}{n}$
$dx \leftarrow \frac{x_{max}-x_{min}}{m}$
$y \leftarrow y_{min}$
for $i \leftarrow 1$ to n **do**
  $x \leftarrow x_{min}$
  for $j \leftarrow 1$ to m **do**
    $g_{ij} \leftarrow v(x, y)$
    $x \leftarrow x + dx$
  end for
  $y \leftarrow y + dy$
end for

where g_{ij} are the points of a 2D grid of $m \times n$ interpolated values, $m, n \in N^*$ for a rectangular region given by coordinates $x_{min}, x_{max}, y_{min}, y_{max} \in R, (x_{min} < x_{max}, y_{min} < y_{max})$. To obtain the IDW estimated value by using Eq. 1, the distance between two points was calculated using the formula for distance on Earth.

To use CUDA for IDW, we first need to parallelize the algorithm. In IDW, we need to calculate the distance between the unsampled points and each of the sampled points. This distance calculation can be parallelized by assigning each GPU thread to a single unsampled point and calculating the distances to all sampled points.

Once we have calculated the distances, we need to calculate the weights for each sampled point based on its distance from the unsampled point. This weight calculation can also be parallelized by assigning each GPU thread to a single sampled point and calculating its weight for all unsampled points.

Finally, we can use the calculated weights to interpolate the values at the unsampled points. This interpolation step can also be parallelized by assigning each GPU thread to a single unsampled point and calculating its value based on the weighted average of the sampled point values.

The kriging algorithm was implemented following the next 4 steps:
Step 1. Calculation of semivariance points (Algorithm 2))
Step 2. Calculation of semivariance coefficients using least squares method [19]
Step 3. Calculation of interpolation weights (Algorithm 3)
Step 4. Calculation of interpolated values (Algorithm 4)

For parallelizing kriging algoritm using CUDA, multiple steps are necessary: the variogram calculation, the kriging matrix calculation, and the kriging weight calculation.

The variogram calculation involves computing the semivariance between all pairs of sampled points. This step can be parallelized by assigning each GPU thread to a single pair of sampled points and calculating their semivariance.

The kriging matrix calculation involves inverting a matrix that depends on the semivariances between the sampled points.

Finally, the kriging weight calculation involves computing the weights for each sampled point based on its distance and spatial correlation with the unsampled point. This step can be parallelized by assigning each GPU thread to a single unsampled point and calculating its weights for all sampled points.

3 Results and Discussions

The studied region is between 21° lon E and 29° lon E and between 41° lat N and 49° lat N. The obtained grids have 400×400, 800×800, 1200×1200 and 1600×1600 resolutions, so each grid point have approximatelly 2 km, 1 km, 0.75 km and, respectivelly, 0.5 km. In Fig. 1 and Fig. 2, 1 km resolution geomagnetic maps for Romania region obtained by IDW and, respectively, kriging interpolation are presented.

The accuracy of the results are presented in Table 1. Mean difference between interpolated and real value, standard deviation, median value and maximum difference are presented for each studied resolution, for both interpolation methods. The error was obtained eliminating one point at a time and calculating the difference between the interpolated value and the real value measured at each specific point. The results showed better accuracy for kriging interpolation method, for all studied resolution. For example, if for IWD the median value of error is between 4.476 and 4.895 μT, depending on resolution, for kriging method this value is between 2.871 and 3.687 μT. Moreover, in Fig. 3, the lower values of geomagnetic field mean error for kriging method can be observed.

In Table 2, the execution times on CPU and GPU, and the speed-up for the CUDA imeplemented algorithms are shown. As could be seen, the complex calculus from kriging method leads to an increased execution time for all resolutions, comparing with IDW (Fig. 4). The speed up obtained for IDW implementation is very high and it increases with the grid resolution, up to 104 times for

Algorithm 2. Calculation of semivariance points

▷ Input:
▷ Maximum distance between points: maxdist = 100 km
▷ Tolerance: toler = 2
▷ np = number of sample points
▷ P_i = coordinates of sample point i (i=0, np-1)
▷ v_i = value at sample point i (i=0, np-1)
▷ Ouput:
▷ ns = number of semivariance points
▷ ns = maxdist/toler
▷ s_k = semivariance values (k=0, ns-1)
▷ w_k = weights (k=0, ns-1)

for $i \leftarrow 0$ to $ns - 1$ **do**
  $s_k \leftarrow 0$
  $w_k \leftarrow 0$
end for
for $i \leftarrow 0$ to $np - 2$ **do**
  for $j \leftarrow i + 1$ to $np - 1$ **do**
    $d \leftarrow dist(P_i, P_j)$
    if $d < a + toler$ **then**
      $k \leftarrow \frac{d}{toler}$
      $s_k \leftarrow s_k + (v_i - v_j)^2$
      $w_k \leftarrow w_k + 1$
    end if
  end for
end for
for $i \leftarrow 0$ to $ns - 1$ **do**
  if $w_k > 0$ **then**
    $s_k = \frac{s_k}{2w_k}$
  end if
end for
$Sw \leftarrow \sum_{k=0}^{n_s} w_k$
for $i \leftarrow 0$ to $ns - 1$ **do** ▷ normalization of weights "w"
  $w_k \leftarrow \frac{w_k}{Sw}$
end for

Table 1. The interpolation methods accuracy.

Interpolation method	Resolution	Median error (μT)	Maximum error(μT)	Mean error(μT)	Standard deviation
IDW	400×400	4.895	9.21	3.600	0.275
	800×800	4.550	9.271	3.137	0.289
	1200×1200	4.495	9.268	3.065	0.254
	1600×1600	4.476	9.274	2.941	0.247
Kriging	400×400	3.687	7.21	3.102	0.312
	800×800	3.551	7.02	2.542	0.271
	1200×1200	3.215	6.82	2.14	0.252
	1600×1600	2.871	6.01	1.92	0.248

Algorithm 3. Calculation of semivariance coefficients using least squares method

▷ Input:
▷ Maximum distance between points: maxdist = 100 km
▷ np = number of sample points
▷ P_i = coordinates of sample point i (i=0, np-1)
▷ v_i = value at sample point i (i=0, np-1)
▷ Ouput:
▷ W_k = kriging interpolation weights (k=0, ns-1)
$np_2 \leftarrow 0$
for $i \leftarrow 0$ to $np - 1$ **do**
  for $k \leftarrow 0$ to $np - 1$ **do**
    if $i \neq k$ **then**
      $d \leftarrow dist(P_i, P_k)$
      if $d < maxdist$ **then**
        $Calculatesemivariance\sigma^1_{np2}usingd$
        $P2_{np2} \leftarrow P_i$
        $np_2 \leftarrow np_2 + 1$
      end if
    end if
  end for
  for $k \leftarrow 0$ to $np_2 - 1$ **do**
    for $l \leftarrow 0$ to $np_2 - 1$ **do**
      $h \leftarrow dist(P2_k, P2_l)$
      Calculate semivariance $\sigma^2_{k,l} = \sigma^2_{l,k}$ using h
    end for
  end for
  Solve the linear system of equations $\sigma^2_w W = \sigma^1$
  to calculate the weights $W_k(k = 0, np_2 - 1)$
end for

Algorithm 4. Calculation of interpolated values

▷ Input:
▷ nq = number of interpolated points
▷ Qi = coordinates of interpolated point i (i=0,nq-1)
▷ np = number of sample points
▷ vi = value at sample point i (i=0,np-1)
▷ Ouput:
▷ ui = interpolated value at point Qi (i=0,nq-1)
for $i \leftarrow 0$ to $nq - 1$ **do**
  Find the sample points at distance less than maxdist from Q_i
  u_i = interpolated value of the nearby values using the weights W
end for

1600×1600 grid. Eventhough, the speed up for kriging is not as high as for IDW (Fig. 5), for the highest studied resolution the execution time when running on GPU decreased 10 times comparing with CPU.

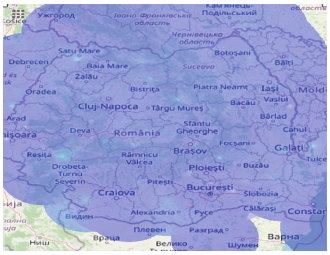

Fig. 1. Geomagnetic map obtained by IDW



Fig. 2. Geomagnetic map obtained by kriging

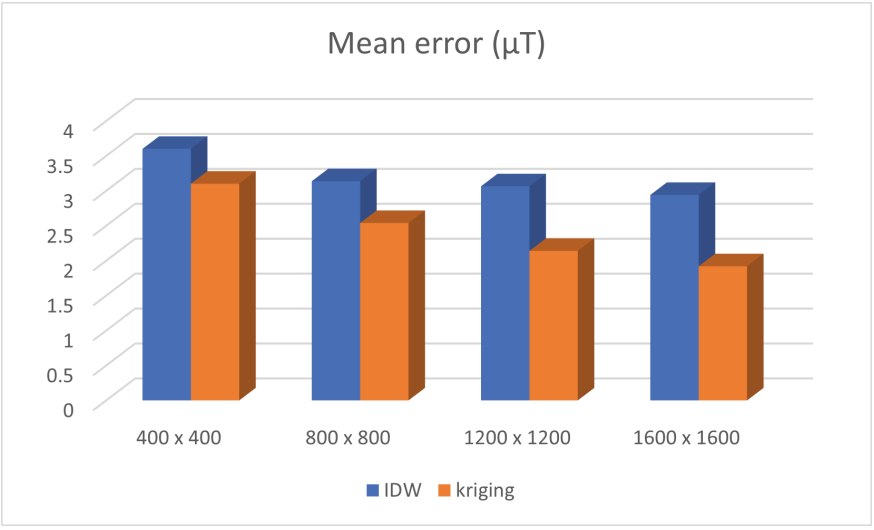

Fig. 3. Geomagnetic field mean error comparison between IDW and kriging

Table 2. The execution times and algorithms speedup on CUDA.

Interpolation method	Resolution	Execution time on CPU (s)	Execution time on GPU (s)	Speed-up
IDW	400×400	13.7	0.355	38.6
	800×800	54. 1	0.724	74.4
	1200×1200	121.7	1.265	96.2
	1600×1600	212.8	2.044	104.1
Kriging	400×400	49.4	11.9	4.15
	800×800	190.5	33.0	5.77
	1200×1200	440.9	53.0	8.32
	1600×1600	757.1	72.4	10.45

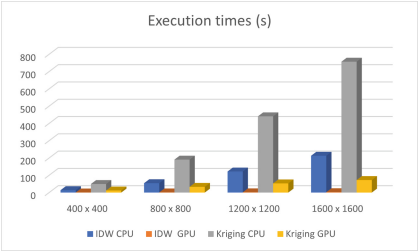

Fig. 4. Execution times of IDW and kriging on CPU and GPU

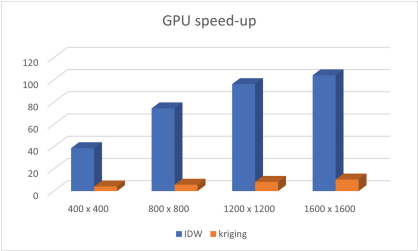

Fig. 5. GPU speed-up comparison between IDW and kriging

4 Conclusions and Perspectives

To conclude, both IDW and kriging interpolation methods could be used for obtaining high resolution geomagnetic maps. Using IDW algorithm, maps are obtained in a shorted execution time, but the accuracy of the obtained data is lower than the one obtined using kriging method. Both algorithms can be accelerated by CUDA programming, the speed-up for IDW being up to 10 times higher then the speed-up for kriging algorithm. By employing memory management techniques, the performance of CUDA implementation of interpolation algorithms can be improved, and better speedup could be achieved.

The use of CUDA for IDW and kriging interpolation has been used to improve the computational efficiency and accuracy of these techniques. This study demonstrates the potential of CUDA-based parallel algorithms and frameworks for improving the scalability and performance of IDW and Kriging interpolation on large-scale data sets. These results are expected to have practical implications for many other fields that rely on interpolation techniques, such as geology, meteorology, or environmental science.

References

1. Li, J., Oldenburg, D.W.: A comparison of interpolation methods for geomagnetic data. Geophys. J. Int. **181**(3), 1549–1560 (2010)
2. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
3. Blakely, R.J.: Potential Theory in Gravity and Magnetic Applications. Cambridge University Press, Cambridge (1996)
4. Campbell, W.H.: Introduction to Geomagnetic Fields. Cambridge University Press, Cambridge (1996)
5. Thébault, E., Finlay, C.C., Beggan, C.D., et al.: International geomagnetic reference field: the 12th generation. Earth Planets Space **67**(1), 79 (2015)
6. McElhinny, M.W., McFadden, P.L.: Paleomagnetism: Continents and Oceans. Academic Press, Cambridge (2000)

7. Kivelson, M.G., Russell, C.T.: Introduction to Space Physics. Cambridge University Press, Cambridge (2017)
8. Telford, W.M., Geldart, L.P., Sheriff, R.E.: Applied Geophysics. Cambridge University Press, Cambridge (1990)
9. Parker, R.L.: Geophysical Inverse Theory and Regularization Problems. Academic Press, Cambridge (2013)
10. Chave, A.D., Jones, A.G.: The Magnetotelluric Method. Cambridge University Press, Cambridge (2012)
11. Krige, D.G.: A statistical approach to some basic mine valuation problems on the Witwatersrand. J. South. Afr. Inst. Mining Metall. **52**(6), 119–139 (1951)
12. Cressie, N.: Statistics for Spatial Data, revised Wiley, New York (1993)
13. Journel, A.G., Huijbregts, C.J.: Mining Geostatistics. Academic Press, London (1978)
14. NVIDIA. CUDA C Programming Guide. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index
15. Liu, Q., Hu, C., Tian, Y.: Parallel implementation of IDW interpolation algorithm based on CUDA. J. Appl. Geophys. **189**, 104332 (2021)
16. Gao, S., Cao, J., Yang, J.: Parallel implementation of kriging interpolation based on CUDA. J. Comput. Appl. Math. **370**, 112665 (2020)
17. Ji, X., Wu, H., Wang, J.: A CUDA-based framework for IDW and Kriging interpolation. Int. J. Geogr. Inf. Sci. **34**(8), 1572–1587 (2020)
18. Ciupala, L., Deaconu, A., Spridon, D.: IDW map builder and statistics of air pollution in Brasov. Bull. Transilvania Univ. Brasov **1**(63), 247–256 (2021)
19. Oliver, M.A., Webster, R.: A tutorial guide to geostatistics: computing and modelling variograms and kriging. Catena **113**, 56–69 (2014)