A Stochastic-based Optimized Schwarz Method for the Gravimetry Equations on GPU Clusters

Abal-Kassim Cheik Ahamed[1] and Frédéric Magoulès[1]

1 Introduction

By giving another way to see beneath the Earth, gravimetry refines geophysical exploration. In this paper, we evaluate the gravimetry field in the Chicxulub crater area located in between the Yucatan region and the Gulf of Mexico which shows strong gravimetry and magnetic anomalies. High order finite elements analysis is considered with input data arising from real measurements. The linear system is then solved with a domain decomposition method, namely the optimized Schwarz method. The principle of this method is to decompose the computational domain into smaller subdomains and to solve the equations on each subdomain. Each subdomain could easily be allocated to one single processor (i.e. the CPU), each iteration of the optimized Schwarz method involving the solution of the equations on each subdomain (on the GPU). Unfortunately, to obtain high speed-up, several tunings and adaptations of the algorithm should be carrefully performed, such as data transfers between CPU and GPU, and data structures, as described in [3, 2].

The plan of the paper is the following. In Section 2, we present the gravimetry equations. In Section 3, we introduce the optimized Schwarz method, followed in Section 4 by a new idea of using a stochastics-based algorithm to determine the optimized transmission conditions. An overview to the GPU programming model and hardware configuration suite is given in Section 5 for readers not familiar with GPU programming. Section 6 shows numerical experiments which clearly confirm the robustness, competitiveness and efficiency of the proposed method on GPU clusters for solving the gravimetry equations.

2 Gravimetry equations

The gravity force is the resultant of the gravitational force and the centrifugal force. The gravitational potential of a spherical density distribution is equal to $\Phi(r) = Gm/r$, with m the mass of the object, r the distance to the object and G the universal gravity constant equal to $G = 6.672 \times 10^{-11} m^3 kg^{-1} s^{-2}$. The gravitational potential at a given position x initiated by an arbitrary density distribution ρ is given by $\Phi(x) = G\int(\rho(x')/||x-x'||)dx'$ where x' represents one point position within the density distribution. In this paper, we consider only regional scale of the

[1] Ecole Centrale Paris, France, e-mail: akcheik@gmail.com, frederic.magoules@hotmail.com

gravimetry equations therefore we do not take into account the effects related to the centrifugal force. The gravitational potential Φ of a density anomaly distribution $\delta\rho$ is thus given as the solution of the Poisson equation $\Delta\Phi = -4\pi G\delta\rho$.

3 Optimised Schwarz method

The classical Schwarz algorithm was invented more than a century ago [16] to prove existence and uniqueness of solutions to Laplace's equation on irregular domains. Schwarz decomposed the irregular domain into overlapping regular ones and formulated an iteration which used only solutions on regular domains and which converged to a unique solution on the irregular domain. The convergence speed of the classical Schwarz algorithm is proportional to the size of the overlap between the subdomains. A variant of this algorithm can be formulated with non-overlapping subdomains and the transmission conditions should be changed from Dirichlet to Robin [6]. These absorbing boundary transmission conditions defined on the interface between the non-overlapping subdomains, are the key ingredients to obtain a fast convergence of the iterative Schwarz algorithm [5, 9]. Optimal transmission conditions can be derived but consists of non local operators and thus are not easy to implement in a parallel computational environment. One alternative is to approximate these operators with partial differential operators. This paper investigates an approximation based on a new stochastics optimization procedure.

For the sake of clarity, the gravimetry equations are considered in the domain Ω with homogeneous Dirichlet condition. The domain is decomposed into two non-overlapping subdomains $\Omega^{(1)}$ and $\Omega^{(2)}$ with an interface Γ. The Schwarz algorithm can be written as:

$$\begin{aligned}
-\Delta\Phi^{(1)}_{n+1} &= f, \quad \text{in} \quad \Omega^{(1)} \\
\left(\partial_\nu \Phi^{(1)}_{n+1} + \mathscr{A}^{(1)}\Phi^{(1)}_{n+1}\right) &= \left(\partial_\nu \Phi^{(2)}_{n} + \mathscr{A}^{(1)}\Phi^{(2)}_{n}\right), \quad \text{on} \quad \Gamma \\
-\Delta\Phi^{(2)}_{n+1} &= f, \quad \text{in} \quad \Omega^{(2)} \\
\left(\partial_\nu \Phi^{(2)}_{n+1} - \mathscr{A}^{(2)}\Phi^{(2)}_{n+1}\right) &= \left(\partial_\nu \Phi^{(1)}_{n} - \mathscr{A}^{(2)}\Phi^{(1)}_{n}\right), \quad \text{on} \quad \Gamma
\end{aligned}$$

with n the iteration number, and ν the unit normal vector defined on Γ. The operators $\mathscr{A}^{(1)}$ and $\mathscr{A}^{(2)}$ are to be determined for best performance of the algorithm. Considering the case $\Omega = \mathbb{R}^2$, $f = 0$, and applying a Fourier transform, similar calculations as in [7] lead to the expression of the Fourier convergence rate, involving the quantities $\Lambda^{(1)}$ and $\Lambda^{(2)}$, which are the Fourier transforms of $\mathscr{A}^{(1)}$ and $\mathscr{A}^{(2)}$ operators. In this case, the choice $\Lambda^{(1)} := |k|$, and $\Lambda^{(2)} := |k|$ is optimal, since with this choice the algorithm converges in two iterations for two subdomains. Different techniques to approximate these non local operators with partial differential operators have been analyzed in recent years [5, 4, 7]. These techniques consist to define partial differential operators involving a tangential derivative on the interface such as: $\mathscr{A}^{(s)} := p^{(s)} + q^{(s)}\partial^2_{\tau^2}$, with s the subdomain number, $p^{(s)}$, $q^{(s)}$ two coefficients, and τ the unit tangential vector. The first results presented in [5, 9] use a zero order Taylor expansion of the non local operators to find $p^{(s)}$ and $q^{(s)}$. In [8] for convection diffusion equations, in [4] for Maxwell equation, in [7, 12] for the Helmholtz equation, and in [11, 10] for heterogeneous media, a minimization procedure has been used. The function to minimize, i.e., the cost function, is the maximum of the Fourier convergence rate for the frequency ranges considered, and the approach consists to determine the free parameters $p^{(s)}$ and $q^{(s)}$ through an optimization problem. Despite, analytic expressions of $p^{(s)}$ and $q^{(s)}$ can be determined for some specific problems, finding quasi-optimal coefficients numerically is also a good alternative [13]. Furthemore, since the evaluation of the cost function is quite fast and the dimension of the search space reasonable, a more robust minimization procedure could be considered, in the next section. Extension to non-regular geometry can be performed as described in reference [14].

4 Stochastic-based optimised transmission conditions

The stochastic minimization technique we propose to use now, explores the whole space of solutions and finds absolute minima; this technique is based on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). This algorithm is very robust [1], has good global search ability and does not need to compute the derivatives of the cost function. This algorithm only needs an initial search zone and a population size, even if the solution can be found outside of the initial search zone. The population size parameter is a trade-off between speed and global search. Meaning that, smaller populations lead to faster execution of the algorithm but have more chance to find a local minimum, and, larger sizes allow to avoid local minima better but need more cost function evaluations. For our purpose, a population size of 25 has been large enough to find the global minimum in a few second or less.

The main idea of the algorithm is to find the minimum of the cost function by iteratively refining a search distribution. The distribution is described as a general multivariate normal distribution $\mathrm{d}(m,C)$. Initially, the distribution is given by the user. Then, at each iteration, λ samples are randomly chosen in this distribution and the evaluation of the cost function at those points is used to compute a new distribution. When the variance of the distribution is small enough, the center of the distribution m is taken as solution. After evaluating the cost function for a new population, the samples are sorted by cost and only the μ best are kept. The new distribution center is computed with a weighted mean (usually, more weight is put on the best samples). The step size σ is a factor used to scale the standard deviation of the distribution, i.e., the variance of the search distribution is proportional to the square of the step size. The step size determines the “size” of the distribution. The covariance matrix C determines the “shape” of the distribution, i.e., it determines the principal directions of the distribution and their relative scaling. Adapting (or updating) the covariance matrix is the most complex part of the algorithm. While this could be done using only the current population, it would be unreliable especially with a small population size; thus the population of the previous iteration should also been taken into account.

5 Overview of GPU programming model

Parallel computation was generally carried out on Central Processing Unit (CPU) cluster until the apparition in the early 2000s of the Graphics Processing Unit (GPU) that facing the migration of the era of GPU computing. The peak performance of CPUs and GPUs is significanlty different, due to the inherently different architectures between these processors. The first idea behind the architecture of GPU is to have many small floating points processors exploiting large amount of data in parallel. This is achieved through a memory hierarchy that allows each processor to optimally access the requiered data. The gains of GPU computing is significantly higher for large size problem compared to CPU, due to the difference between these two architectures. GPU computing requires using graphics programming languages such as NVIDIA CUDA, or OpenCL. *Compute Unified Device Architecture* (CUDA) [15] programming model is an extension of the C language and has been used in this paper.

A specific characteristic of GPU compared to CPU is the feature of memory used. Indeed, a CPU is constantly accessing the RAM, therefore it has a low latency at the detriment of its raw throughput. CUDA devices have four main types of memory: (i) *Global memory* is the memory that ensures the interaction with the host (CPU), and is not only large in size and off-chip, but also available to all threads (also known as compute units), and is the slowest; (ii) *Constant memory* is read only from the device, is generally cached for fast access, and provides interaction with the host; (iii) *Shared memory* is much faster than global memory and is accessible by any thread of the block from which it was created; (iv) *Local memory* is specific to each compute unit and cannot be used to communicate between compute units.

Threads are grouped into blocks and executed in parallel simultaneously, see Figure 1. A GPU is associated with a *grid*, i.e., all running or waiting blocks in the running queue and a kernel that will run on many cores. An ALU is associated with the thread which is currently processing. Threading is not an automated procedure. The developer chooses for each kernel the distribution of the threads, which are organized (*gridification* process) as follows: (i) threads are grouped into blocks; (ii) each block has three dimensions to classify threads; (iii) blocks are grouped together in a grid of two dimensions. The threads are then distributed to these levels and become easily identifiable by their positions in the grid according to the block they belongs to and their spatial dimensions. The kernel function must be called also providing at least two special parameters: the dimension of the block, *nBlocks*, and the number of threads per block, *nThreadsPerBlock*. Figure 1 presents the CUDA processing flow. Data are first copied from the main memory to the GPU memory, (1). Then the host (CPU) instructs the device (GPU) to carry out calculations, (2).

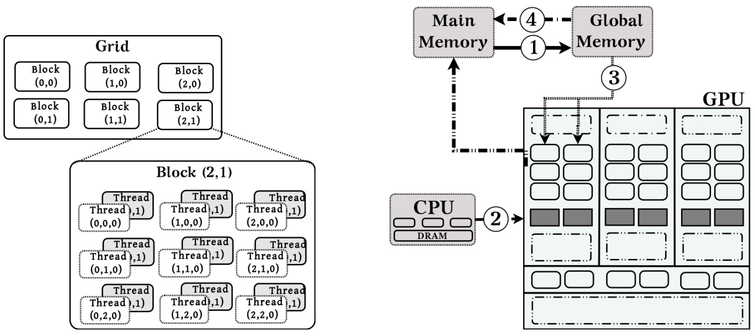

Fig. 1 Gridification of a GPU. Tthread, block, grid (left); GPU computing processing (right)

The kernel is then executed by all threads in parallel on the device, (3). Finally, the device results are copied back (from GPU memory) to the host (main memory), (4). To cope with this difficulty the implementation proposed in this paper uses some advanced tuning techniques developped by the authors, but the details are outside the scope of this paper, and the reader is refeered to [3, 2] for the computer science aspects of this tuning.

6 Numerical analysis

In this section, we report the experiments performed to evaluate the speed-up of our implementation. The Chicxulub impact crater, formed about 65 million years ago, is now widely accepted as the main footprint of the global mass extinction event that marked the Cretaceous/Paleogene boundary. Because of its relevance, in the last two decades, this impact structure has been used as a natural laboratory to investigate impact cratering formation processes and to indirectly infer global effects of large-scale impacts. The crater is buried under 1 km of carbonate sediments in the Yucatan platform. The crater is about 200 km in rim diameter, half on-land and half off-shore with geometric center at Chicxulub Puerto. The internal structure of the Chicxulub crater has been imaged by using several geophysical data sets from land, marine and aerial measurements.

In this paper we perform a finite element analysis of the gravimetry equation using the characteristics of the region provided by the measure. The domain consists of an area of $250 \times 250 \times 15$ in each spatial direction, and is discretized with high order finite element with a total of 19 933 056 degrees of freedom. This mesh is partionned in the x-direction. Each subdomain is allocated to one single processor (i.e. the CPU), each iteration of the optimized Schwarz method involving the solution of the equations inside each subdomain is allocated to one single accelerator (i.e the GPU). We compare the computational time of the optimised Schwarz method using one subdomain per CPU with the optimised Schwarz method using one subdomain per CPU and a GPU accelerator. For this particular model we performed calculations using our CUDA implementation of the Schwarz method with stochastics-based optimization procedure. The workstation used for all the experiments consists of 1 596 servers Bull Novascale R422Intel Nehalem-based nodes. Each node is composed of 2 processors Intel Xeon 5570 quad-cores (2.93 GHz) and 24 GB of memory (3Go per cores). 96 CPU servers are interconnected with 48 compute Tesla S1070 servers NVIDIA (4 Tesla cards with 4GB of memory by server) and 960 processing units are available for each server.

For the subdomain problems, the diagonal preconditioner conjugate gradient (PCG) is used and the coefficient matrices are stored in CSR format. We fix a residual tolerance threshold of $\varepsilon = 10^{-10}$ for PCG. *Alinea* [3, 2], our research group library, implemented in *C*++, which offers CPU and GPU solvers, is used for solving linear algebra system. In this paper, the GPU is used to accelerate the solution of PCG algorithm. PCG algorithm required the computation of addition of vectors (*Daxpy*), dot product and sparse matrix-vector multiplication. In GPU-implementation considered (Alinea library), the distribution of threads (*gridication*, differs with these operations. The gridification of *Daxpy*, dot product and sparse matrix-vector product correspond respectively to ($nBlocks = \frac{numb\_rows + numb\_th\_block - 1}{numb\_th\_block}$, $nThreadsPerBlock = 256$), ($nBlocks = \frac{numb\_rows + numb\_th\_block - 1}{numb\_th\_block}$, $nThreadsPerBlock = 128$) and ($nBlocks = \frac{(numb\_rows \times n\_th\_warp) + numb\_th\_block - 1}{numb\_th\_block}$, $nThreadsPerBlock = 256$), where $numb\_rows$, n_th_warp and $numb\_th\_block$ represent respectively the number of rows of the matrix, the number of threads per warp and the thread block size. We fix for all operations 8 threads per warp. GPU is used only for solving subdomain problems in each iteration. GPU experiment workstation Tesla S1070 has 4 GPUs of 240 cores. The number of computing units depends both on the size of the subdomain problem and the gridification that use 256 threads per threads and 8 threads per warp as introduced in [3, 2].

In our experiments, the CMA-ES algorithm considers as the cost function the Fourier convergence rate of the optimised Schwarz method. We consider for the CMA-ES the following stopping criteria of : a maximum of number iterations equal to 7200 and a residu threshold equal to 5×10^{-11}. Fig. 2 represents the convergence rate of the Schwarz algorithm in the Fourier space, respectively for the symmetric zeroth order (top-left), unsymmetric zeroth order (bottom-left), the symmetric second order (top-right) and unsymmetric second order (bottom-right) transmission conditions obtained from the CMA-ES algorithm. The Fourier convergence rate of the Schwarz method with one side (respectilely two sides) transmission conditions obtained from the CMA-ES algorithm is presented in Fig. 2 and Table 1. The distribution of processors is computed as follows: number of processors $=$ $2 \times$ number of nodes, where 2 corresponds to the number of GPU per node as available on our workstation. As a consequence, only two processors will share the bandwidth, which strongly improve the communications, especially the inter-subdomain communications. Table 2 presents the results done with double precision with a residu threshold, *i.e.* stopping criterion equal to 10^{-6}, for several number of subdomains (one subdomain per processor).

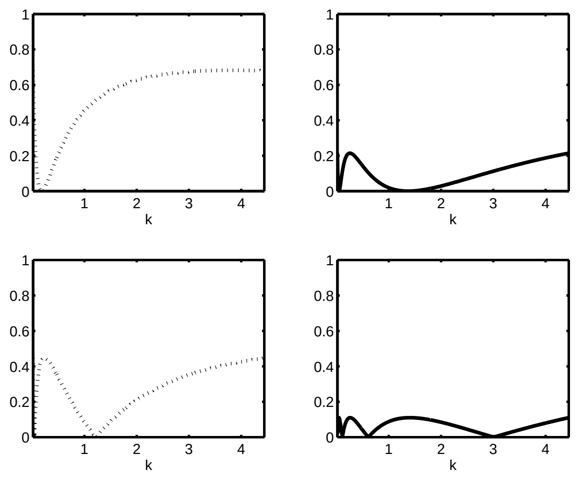

Fig. 2 Fourier convergence rate of the Schwarz algorithm

	$p^{(1)}$	$q^{(1)}$	$p^{(2)}$	$q^{(2)}$	ρ_{max}
oo0_symmetric	0.1826	0	0.1826	0	0.6823
oo0_unsymmetric	1.2193	0	0.0469	0	0.4464
oo2_symmetric	0.0471	0.7050	0.0471	0.7050	0.2143
oo2_unsymmetric	0.1081	0.3205	0.0231	1.5786	0.1101

Table 1 Optimized coefficients obtained from *CMA-ES* algorithm

#subdomains	**#iterations**	**cpu time (sec)**	**gpu time (sec)**	**SpeedUp**
32	41	11 240	1 600	**7.03**
64	45	5 360	860	**6.23**
128	92	6 535	960	**6.81**

Table 2 Comparison of the implementation of our method on CPU and GPU

7 Conclusion

In this paper, we have presented a stochastic-based optimized Schwarz method for the gravimetry equation on GPU Clusters. The effectiveness and robustness of our method are evaluated by numerical experiments performed on a cluster composed of 1 596 servers Bull Novascale R422Intel Nehalem-based nodes where 96 CPU servers are interconnected with 48 compute Tesla S1070 servers NVIDIA (4 Tesla cards with 4GB of memory by server). The presented results range from 32 up-to 128 subdomains show the interest of the use of GPU technologies for solving large size problems, and outline the robustness, performance and efficiency of our Schwarz domain decomposition method with stochastic-based optimized conditions.

Acknowledgments

The authors acknowledge partial financial support from the OpenGPU project (2010-2012), and GENCI (Grand Equipement National de Calcul Intensif) for the computer time used during this long-term trend.

References

1. Auger, A., Hansen, N.: Tutorial CMA-ES: evolution strategies and covariance matrix adaptation. In: GECCO (Companion), pp. 827–848 (2012)
2. Cheik Ahamed, A.K., Magoulès, F.: Fast sparse matrix-vector multiplication on graphics processing unit for finite element analysis. In: HPCC-ICESS, pp. 1307–1314. IEEE Computer Society (2012)
3. Cheik Ahamed, A.K., Magoulès, F.: Iterative methods for sparse linear systems on graphics processing unit. In: HPCC-ICESS, pp. 836–842. IEEE Computer Society (2012)
4. Chevalier, P., Nataf, F.: Symmetrized method with optimized second-order conditions for the Helmholtz equation. In: Domain decomposition methods, 10 (Boulder, CO, 1997), pp. 400–407. Amer. Math. Soc., Providence, RI (1998)
5. Després, B.: Domain decomposition method and the Helmholtz problem.II. In: Second International Conference on Mathematical and Numerical Aspects of Wave Propagation (Newark, DE, 1993), pp. 197–206. SIAM, Philadelphia, PA (1993)
6. Després, B., Joly, P., Roberts, J.E.: A domain decomposition method for harmonic Maxwell equations. In: Iterative methods in linear algebra, pp. 475–484. North-Holland, Amsterdam (1992)
7. Gander, M.J., Magoulès, F., Nataf, F.: Optimized Schwarz methods without overlap for the Helmholtz equation. SIAM J. Sci. Comput. **24**(1), 38–60 (2002)
8. Japhet, C., Nataf, F., Rogier, F.: The optimized order 2 method. Application to convection-diffusion problems. Future Generation Computer Systems **18**(1), 17–30 (2001)
9. J.D.Benamou, Després, B.: A domain decomposition method for the Helmholtz equation and related optimal control problems. J. of Comp. Physics **136**, 68–82 (1997)
10. Maday, Y., Magoulès, F.: Improved ad hoc interface conditions for Schwarz solution procedure tuned to highly heterogeneous media. Applied Mathematical Modelling **30**(8), 731–743 (2006)
11. Maday, Y., Magoulès, F.: Optimized Schwarz methods without overlap for highly heterogeneous media. Comput. Methods in Appl. Mech. Eng. **196**(8), 1541–1553 (2007)
12. Magoulès, F., Ivànyi, P., Topping, B.: Convergence analysis of Schwarz methods without overlap for the helmholtz equation. Computers & Structures **82**(22), 1835–1847 (2004)
13. Magoulès, F., Ivànyi, P., Topping, B.: Non-overlapping Schwarz methods with optimized transmission conditions for the Helmholtz equation. Computer Methods in Applied Mechanics and Engineering **193**(45-47), 4797–4818 (2004)
14. Magoulès, F., Putanowicz, R.: Optimal convergence of non-overlapping Schwarz methods for the Helmholtz equation. Journal of Computational Acoustics **13**(3), 525–545 (2005)
15. Nvidia Corporation: CUDA Toolkit Reference Manual, 4.0 edn. Available on line at: `http://developer.nvidia.com/cuda-toolkit-40` (accessed on September 29, 2012)
16. Schwarz, H.A.: ber einen grenzbergang durch alternierendes verfahren. Vierteljahrsschrift der Naturforschenden Gesellschaft **15**, 272–286 (1870)