DISCRETE AND CONTINUOUS
DYNAMICAL SYSTEMS SERIES S

doi:10.3934/dcdss.2020381

COMPUTATIONAL OPTIMIZATION IN SOLVING THE GEODETIC BOUNDARY VALUE PROBLEMS

Marek Macák*, Róbert Čunderlík
Karol Mikula and Zuzana Minarechová

Slovak University of Technology in Bratislava
Faculty of Civil Engineering, Department of Mathematics and Descriptive Geometry
Radlinskeho 11, Bratislava 810 05, Slovakia

Abstract. The finite volume method (FVM) as a numerical method can be straightforwardly applied for global as well as local gravity field modelling. However, to obtain precise numerical solutions it requires very refined discretization which leads to large-scale parallel computations. To optimize such computations, we present a special class of numerical techniques that are based on a physical decomposition of the computational domain. The domain decomposition (DD) methods like the Additive Schwarz Method are very efficient methods for solving partial differential equations. We briefly present their mathematical formulations, and we test their efficiency in numerical experiments dealing with gravity field modelling. Since there is no need to solve special interface problems between neighbouring subdomains, in our applications we use the overlapping DD methods. Finally, we present the numerical experiment using the FVM approach with 93 312 000 000 unknowns that would not be possible to perform using available computing facilities without aforementioned methods that can efficiently reduce a numerical complexity of the problem.

1. **Introduction.** A determination of the Earth's gravity field is usually formulated in terms of the geodetic boundary-value problems (BVPs). There exist various numerical approaches to solve such potential problems. In geodesy, the spherical harmonics-based methods are usually used for global gravity field modelling. They solve the problem in a frequency domain and nowadays they have become a very efficient and sophisticated tool. On the other hand, a development of high-performance computing facilities has also brought new opportunities for numerical solutions of the geodetic BVPs (GBVPs). Efficient numerical methods such as the boundary element method (BEM), finite element method (FEM) or finite volume method (FVM) can be also applied for global as well as local gravity field modelling. These discretization methods solve GBVPs in a space domain. In order to obtain precise numerical solutions, they require very refined discretizations that consequently leads to large-scale computations. Nonetheless, parallel implementations of algorithms and high-performance computations on clusters with distributed memory provide strong opportunities for high-resolution gravity field modelling.

2010 *Mathematics Subject Classification.* Primary: 65K05; Secondary: 49M27.
Key words and phrases. Geodetic boundary-value problem, finite volume method, MPI, OpenMP, domain decomposition methods, additive Schwarz method.

* Corresponding author: Marek Macák.

A numerical solution of GBVP and its precision is naturally depended on a level of the discretization of a computational domain. In this paper we present a possibility to increase such a level of the discretization by applying the domain decomposition (DD) method and parallelization. Consequently, we are able to perform computations on more detailed grids whose memory requirements would be much higher than a capacity of our available computational facilities. Namely, in our final numerical experiment we will perform large-scale parallel computations using the FVM approach with the horizontal resolution 1×1 *arc min* and 400 layers in the radial direction. Such a detailed grid yields 93 312 000 000 unknowns requiring 4.5 TB of distributed memory, however, our cluster consists of 1.7 TB.

The paper is organized as follows. In Section 2, we formulate the geodetic BVP. In Section 3, we summarize mathematical approaches to its solution and discuss the optimal linear solver for our approach. In Section 4, we deal with the speed-up of the solution by employing a parallelization and the DD method. In Section 5, we present numerical experiments.

2. Fixed gravimetric boundary-value problem.

To present the computational methods, we outline numerical solutions to the linearized fixed gravimetric BVP introduced and discussed e.g. in [15, 10, 6, 9, 18]:

$$\begin{aligned} \Delta T(\mathbf{x}) &= 0, \quad \mathbf{x} \in R^3 - S, && (1)\\ \langle \nabla T(\mathbf{x}), \vec{s}(\mathbf{x}) \rangle &= -\delta g(\mathbf{x}), \quad \mathbf{x} \in \partial S, && (2)\\ T(\mathbf{x}) &\to 0, \quad \text{as} \quad |\mathbf{x}| \to \infty, && (3) \end{aligned}$$

where Δ is the Laplace operator, $T(\mathbf{x})$ is the disturbing potential defined as the difference between the real $W(\mathbf{x})$ and the normal $U(\mathbf{x})$ gravity potential at any point $\mathbf{x}$, S denotes the Earth body, $\langle , \rangle$ represents the inner product of vectors, ∇ is the gradient operator, $\vec{s}(\mathbf{x}) = -\nabla U(\mathbf{x})/|\nabla U(\mathbf{x})|$ is the unit vector normal to the equipotential surface of the normal potential U at point $\mathbf{x}$, and $\delta g(\mathbf{x})$ is the so-called gravity disturbance.

Eqs. (1) - (3) represent an exterior BVP for the Laplace equation, i.e. the computational domain (outside the Earth) is infinite. So to apply FVM, we construct a computational domain Ω in the external space above the Earth, see [9]. The computational domain Ω is bounded by the bottom surface $\Gamma \subset \partial\Omega$ representing the approximation of the Earth's surface and the upper surface at the level of chosen satellite mission, where the Dirichlet-type boundary conditions (BC) for disturbing potential are generated from some satellite-only geopotential model. For our numerical experiments we will choose as Γ the surface area of the reference ellipsoid WGS84 and the upper boundary will be at the constant altitude 240 km above it, i.e., the computational domain Ω will be the space between two ellipsoids.

In the bounded domain Ω, we consider the fixed gravimetric BVP in the following form:

$$\begin{aligned} \Delta T(\mathbf{x}) &= 0, \quad \mathbf{x} \in \Omega, && (4)\\ \langle \nabla T(\mathbf{x}), \vec{s}(\mathbf{x}) \rangle &= -\delta g(\mathbf{x}), \quad \mathbf{x} \in \Gamma, && (5)\\ T(\mathbf{x}) &= T_{SAT}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega - \Gamma, && (6) \end{aligned}$$

where T_{SAT} represents the disturbing potential generated from the satellite-only geopotential model. Since in this case we deal with the solution in the bounded domain Ω, we do not prescribe regularity condition at infinity.

The boundary condition given by Eq. (2) or (5) represents the oblique derivative BC because the vector $\vec{s}$ does not coincide with the normal to the Earth's surface. For the sake of simplicity, we will consider it as the Neumann BC. It is because in all presented numerical experiments we use ellipsoidal approximation of the Earth's surface. Hence, the problem of oblique derivatives vanishes since $\vec{s} = \vec{n}_\Gamma$, where $\vec{n}_\Gamma$ is the normal to the computational domain Ω.

In all our numerical experiments, the bottom boundary Γ will be represented by the WGS84 reference ellipsoid and where the gravity disturbances transformed form the free-air gravity anomalies interpolated from the DTU10-GRAV [1] dataset will be prescribed. The upper boundary will be at the constant altitude 240 km above the reference ellipsoid where the Dirichlet BCs in form of the disturbing potential generated from the GOCO03S satellite-only geopotential model up to degree 250 [16] are applied.

3. Numerical solution of GBVP and optimal linear solver.

3.1. Numerical approaches for solving GBVP.

There exist various numerical approaches to solve such potential problems. The spherical harmonics based methods are usually used for the global gravity field modelling, c.f. [16, 20]. Recently, numerical methods like the BEM, the FEM, the finite difference method (FDM), FVM and others have been also applied for gravity field modelling. The BEM was innovatively applied by Klees in [13] and later developed using sophisticated tools to reduce its numerical complexity [14]. Later Čunderlík et al. [6, 7] presented the direct BEM formulation based on the collocation method for solving the linearized fixed gravimetric BVP. In case of the FEM, the pioneering work has been done by [17] and [21]. Later, the finite element technique for the solution of gravimetric BVPs with mixed BCs in 3D domains above the Earth's surface was studied in [9]. The FDM was applied by Keller in [12]. Other numerical approaches based on a weak formulation of the BVP and minimization of a quadratic functional were developed in [11] and [19].

The FVM aproach was recently applied in [18], where the horizontal resolution 5×5 *arc min* corresponding to $12 \times 360 \times 12 \times 180 = 9\ 331\ 200$ unknowns on the Earth's surface was presented. In this paper we present a similar FVM-based numerical experiment, where the horizontal resolution is increased into 1×1 *arc min* ($60 \times 360 \times 60 \times 180 = 233\ 280\ 000$ unknowns on the Earth's surface), because the available input data are provided in such detailed grids. To obtain numerical solutions as precise as possible, a discretization in the radial direction should be increased as well. This leads to large-scale parallel computations whose memory requirements are higher than the distributed memory of our cluster and needs to be reduced using the DD methods.

3.2. Choice of the optimal linear solver.

Solution to the GBVP (4)-(6) by FVM leads to a large-scale linear system with a matrix that is nonsymmetric due to BC (5). In general, there does not exist an optimal solver for all types of matrices obtained by numerical discretization. Finding the solver with optimal computational time and memory requirements needs a detailed analysis. We can choose from two types of solvers:

- Stationary methods: Jacobi, Gauss - Seidel (GS) and Successive over-relaxation (SOR)

- Nonstationary methods: Conjugate Gradient, Generalized Minimal RESidual, Biconjugate Gradient Method (BiCG), Biconjugate gradient stabilized method (Bi-CGSTAB)

and their modifications. In a comparison to the nonstationary methods, a drawback of the stationary methods is a slow convergence, and advantage is small memory requirements since nonstationary methods need additional memory in addition to storage matrix coefficients and a right-hand side vector. A comparison of methods is shown in Table 1. As we can see, the most efficient method is Bi-CGSTAB.

Solver	**CPU time** [s]	**Number of iterations**
GS	703	10000
SOR	92	1136
BiCG	68	568
Bi-CGSTAB	41	348

TABLE 1. Efficiency comparison of the stationary and nonstationary methods in the experiment with 259 200 unknowns, tested on one CPU core.

In [22], the restarted BiCGstab(l) was introduced. It was designed to improve a convergence and stability of the solution. From Table 2, we can see that the modified solver decreases the number of iterations but increases the time and memory requirements for computing facilities. Size of the memory to store the matrix (stored in the Compressed Diagonal Storage format) and the right hand-side vector for the experiment with 4 374 000 unknowns was 52.84 MB.

Solver	**Number of iterations**	**CPU time** [s]	**Additional memory for solver [MB]**
Bi-CGSTAB	1053	403.82	184.26
BiCGstab(2)	554	494.14	258.02
BiCGstab(4)	272	629.01	405.46
BiCGstab(8)	130	860.86	700.34

TABLE 2. Efficiency comparison for various Bi-CGSTAB linear solvers in the experiment with 4 374 000 unknowns, tested on one CPU core.

Due to the minimal memory consumption, see Table 2, we use the Bi-CGSTAB as a linear solver in all following experiments.

4. Speed-up of solution.

4.1. Parallelization.

The speed up of numerical algorithms can be performed by a distribution of computations into several processes using the so-called Massively Parallel Processors' architecture together with the Message Passing Interface (MPI) [2] and Open Multi-Processing (OpenMP) programming framework [5].

The detailed study of the possibilities of the data management and communication between processes using MPI functions has been presented in [18]. MPI is a way to program on distributed memory devices. This means that the parallelism occurs where every parallel process is working in its own memory space in isolation from the others. OpenMP is a way to program on shared memory devices. This means that the parallelism occurs where every parallel thread has access to all the data. The OpenMP approach is very suitable for the multi-core CPU (multi-thread CPU), and, opposite to MPI, it is easy to implement and user does not need to manage a multiprocessor communication.

In Table 3, we present a comparison of the parallelization using MPI and OpenMP. In the case of MPI, we have increased the number of processes therefore increased the memory consumption due to parallelization with one layer overlapping. In case of OpenMP, the increased number of threads has not changed the memory consumption. Balancing between the numbers of MPI Processes and OpenMP Thread configurations is based on a real configuration of computing resources. In our case, we have 4 quad-core CPUs in one computing node.

MPI Processes	**OpenMP Threads**	**CPU time [s]**	**Speedup ratio**	**RAM [MB]**	**Memory increase**
1	1	403.82	-	237.108	-
	2	232.40	1.73		
	4	191.36	2.11		
	8	87.31	4.63		
	16	57.51	7.02		
2	1	216.84	1.86	245.868	+3.7%
	2	126.17	3.20		
	4	98.46	4.10		
	8	85.88	4.70		
4	1	114.01	3.54	266.040	+12.2%
	2	79.72	5.06		
	4	55.56	7.26		
8	1	79.34	5.09	308.456	+30.0%
	2	70.81	5.70		
16	1	59.51	6.78	390.068	+64.5%

TABLE 3. Comparison of Processes/Threads parallelization in the experiment with 4 374 000 unknowns, tested on 4 quad-core CPUs.

Due to the minimal CPU time 55.56 [s], see Table 3, we use the configuration 4 MPI Processes and 4 OpenMP threads in all following parallel computations.

4.2. **Domain decomposition.** To reduce large memory requirements of the problem, we apply the domain decomposition (DD) methods that are very efficient numerical methods for solving partial differential equations (PDEs) based on a physical decomposition of a global solution domain, see e.g. [4, 8]. In our study, we use the overlapping DD method because we use parallelization with overlapping and because there is no need to solve special interface problems between neighboring subdomains. For more details about the mathematical formulations of the overlapping DD methods, see [3, 4, 8].

In the DD method, we decompose the global solution domain? into a set of M subdomains such that? $\Omega = \bigcup_{i=1}^{M} \Omega_i$, and we require an explicit overlap between each pair of neighboring subdomains. Then we denote common boundaries of neighbouring subdomains by $\widehat{\Gamma}_i$. We run an iterative solution procedure that starts with an initial guess T^0 (in our case we set $T^0 = 0$). One DD iteration consists of M sub-steps that have to be carried out in the sequence $i = 1, 2, ..., M$. For the sub-step number i we solve (4-6) restricted to subdomain Ω_i:

$$\Delta T_i^n(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega_i, \tag{7}$$

$$\langle \nabla T_i^n(\mathbf{x}), \vec{s}(\mathbf{x}) \rangle = -\delta g(\mathbf{x}), \quad \mathbf{x} \in \Gamma_i, \tag{8}$$

$$T_i^n(\mathbf{x}) = T_{SAT}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega_i - \Gamma_i - \widehat{\Gamma}_i, \tag{9}$$

$$T_i^n(\mathbf{x}) = T^*(\mathbf{x}), \quad \mathbf{x} \in \widehat{\Gamma}_i. \tag{10}$$

Here, the linear operator Δ has exactly the same form as in domain Ω. The artificial Dirichlet BC T^* in eq. (10) is updated by exchanging some data on $\widehat{\Gamma}_i$ from the neighboring subdomains. It is obvious that the solution T_i^n on $\widehat{\Gamma}_i$ is changing from iteration to iteration, while converging towards the true solution.

The Schwarz Method, which is suitable for solving subdomains one by one, is called the additive Schwarz method, see [3, 4, 8]. In the additive Schwarz method, the artificial Dirichlet BC is updated by using

$$T_i^n(\mathbf{x}) = T^{n-1}(\mathbf{x}), \quad \mathbf{x} \in \widehat{\Gamma}_i, \tag{11}$$

It means that the artificial Dirichlet BC is updated using solutions from all the relevant neighboring subdomains from the previous DD iteration. Therefore, the subdomain solution in the additive Schwarz method can be carried out completely independently.

In Table 4, we present a saving on memory costs by dividing the computational domain into a different number of subdomains solved on one CPU core. All comparisons are related to the solution obtained using one domain. Due to initialization of arrays, the CPU time oscillates since we use the different number of subdomains. As we can observe, the main advantage of this method is a saving on memory costs because the linear solver is handling a smaller domain, namely it is applied on one subdomain after another. So vectors, it is working with, have fewer elements and the whole solution is in RAM. We can also observe that there is no saving on computation time. If we partially saved solution on the disk, the computation time would be increased due to reading and writing to the file. Hence, we propose a small modification that is discussed below.

Number of subdomains	**CPU time [s]**	**Speedup ratio**	**RAM [MB]**	**Memory saving**
1	403.82	-	237.108	-
5	1651.68	0.24	89.868	-62.1%
10	907.99	0.44	71.308	-69.9%
15	856.04	0.46	65.248	-72.5%
30	854.24	0.47	57.816	-75.6%

TABLE 4. Efficiency comparison for the different number of subdomains in the DD experiment with 4 374 000 unknowns, tested on one CPU core.

For the case of 30 subdomains, we present the behaviour of the additive Schwarz method, where we have modified the number of iterations η, $\eta \in N$, of the linear solver in one DD iteration (see Table 5). On the contrary, in the Table 4 there is only one iteration of the linear solver in subdomain solutions. It means that the artificial Dirichlet BC is updated using solutions from all relevant neighbouring subdomains in every η iteration of the linear solver. As we can see in Table 5, it can bring a significant saving on CPU time. However, there arises a problem when the number of η is too big and solution in subdomains is solved with old (wrong) values on $\widehat{\Gamma}_i$. There is also another explanation of the optimality of the parameter η, i.e., by increasing number $\eta \in N$. Then the local problem will be solved more accurately and the number of global iterations will be decreasing. However, at some point of η, when the local solution will be enough accurate, the number of global iterations will be stagnated. This can explain an existence of an optimal point of η in terms of the computational time. Hence, in all following computations we will use the parameter $\eta = 15$.

In Figure 1a) we can see the solution after one DD iteration where $\eta = 15$. Figure 1b) depicts solution after 10^{th} DD iteration. We can observe that the initial boundaries between subdomains have vanished.

η	**CPU time [s]**	**Speedup ratio**
1	854.24	-
5	308.02	2.77
10	252.33	3.38
15	224.65	3.80
20	236.56	3.61
25	265.73	3.21

TABLE 5. Efficiency comparison for the different number η in the experiment with 4 374 000 unknowns for case of 30 subdomains, tested on one CPU core.

4.3. **Parallelization of the domain decomposition method.** Both, the parallelization and DD method have shown a significant contribution how to reduce computation time or memory costs. However, in the case when both methods are used together, it has to be explained how to cooperate the multi-processors with subdomains communication.

Based on our experiences in [18] and with respect to the size and shape of the Earth, an optimal and natural choice for data managements on multi-core processors with focus on communication, memory and time costs, is to split the computational domain in the latitudal direction. To speed-up computation on the multiple CPU, it is recommended to use OpenMP threads on one CPU core. To split the computational domain, we use the longitudal direction where the computational domain for each processor is split into multiple subdomains. For illustration of data management, see Figure 2.

In case of the parallel DD, we have the whole solution in RAM and we update it using partial solutions on subdomains. As the linear solver is handling a smaller domain, vectors, it is working with, have fewer elements, and a saving on memory costs is gained.

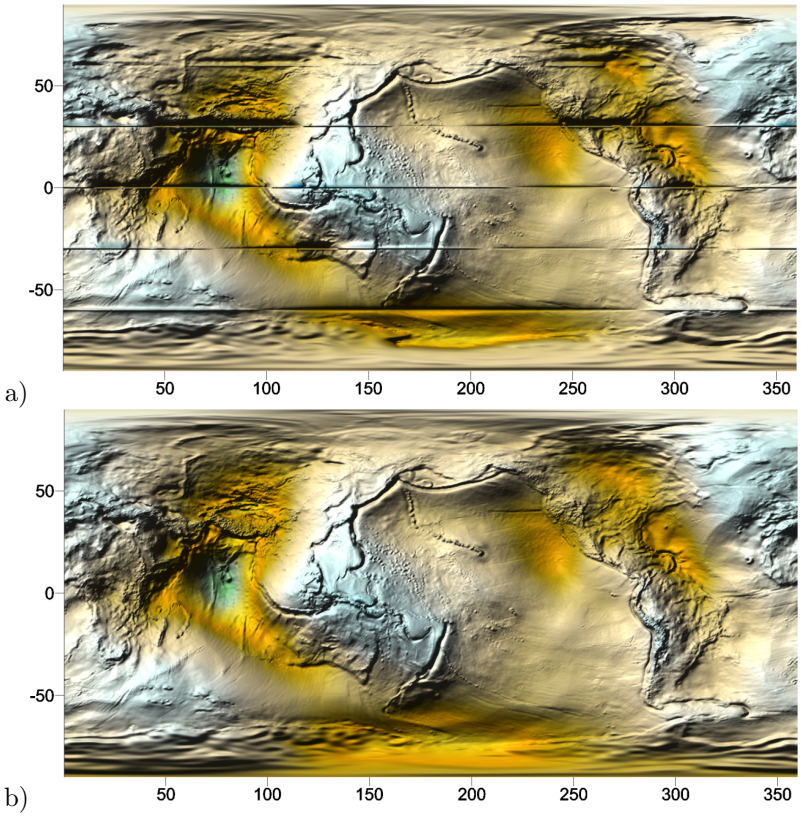

FIGURE 1. Illustration of solution after: a) 1^{st} iteration of domain decomposition with $\eta = 15$, b) 10^{th} iteration of domain decomposition with $\eta = 15$.

In Table 6, we present a saving on computation time and memory costs for the different number of subdomains when comparing with the solution obtained with parallel computation using 4 MPI processes, each with 4 OpenMP threads. All comparisons are related to the solution obtained using one domain. Based on the previous results (see Table 5), we choose $\eta = 15$ as an optimal parameter. One can observe that the increasing number of subdomains is saving the computation time until it reaches a specific number of subdomains when this trend changes. It is a consequence of increasing communication between processes.

In Table 7, we present a comparison of all discussed methods. When we compare the serial and parallel computations using 4 MPI processes with 4 OpenMP threads (see Table 3), we observe a significant speed-up of the computation time, however, 11% increase of memory. When we compare the serial computation and the serial computation using the DD method, we observe a saving of approximately 44% of the computation time. Using a parallel version of the DD method, it is possible to save up to 95% of the computation time in comparison to serial computations. When we compare the parallel and serial versions of the DD method, we can observe that the parallel version has 92% less computation time, however, it needs 45% more of memory. To make a compromise between computation time and memory

Number of subdomains	CPU time [s]	Speedup ratio	RAM [MB]	Memory saving
1	55.56	-	266.040	-
5	55.52	1.00	115.508	-56.6%
10	28.47	1.95	97.568	-63.3%
15	17.44	3.18	91.156	-65.7%
30	18.67	2.97	84.128	-68.3%

TABLE 6. Comparison for the different number of subdomains using parallel DD method in the experiment with 4 374 000 unknowns with $\eta = 15$, tested on 4 quad-core CPUs.

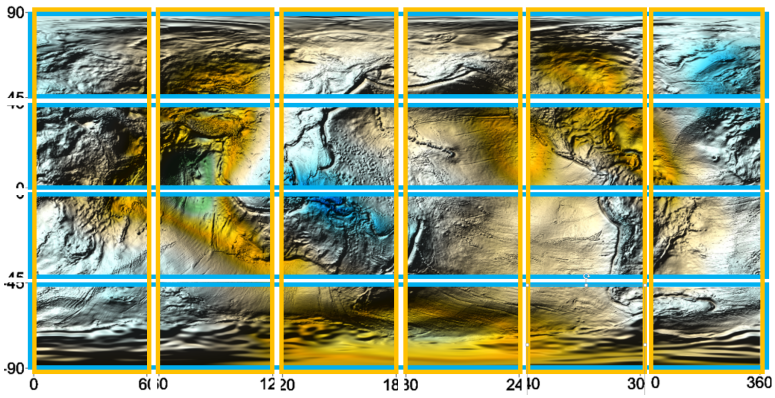

FIGURE 2. Illustration of data management in parallel DD implementations where blue color illustrate subdomains and yellow color illustrate parallelization.

costs, the parallel version of DD method is the most suitable method for large-scale computations.

Computation strategies	CPU time [s]	Speedup ratio	RAM [MB]	Memory saving
Serial without DD	403.82	-	237.108	-
Serial with DD	224.65	1.79	57.816	-75.6%
Parallel without DD	55.56	7.26	266.040	+10.8%
Parallel with DD	18.67	21.6	84.128	-64.5%

TABLE 7. Efficiency comparison for different computation strategies in the experiment with 4 374 000 unknowns where we use 30 subdomains and $\eta = 15$.

5. **Final numerical experiments.** The final experiments aim to demonstrate that we are able to solve GBVP on very detailed computational grids while reducing a numerical complexity of the problem using the DD method.

In the first experiment, the horizontal resolution has been 1×1 *arc min* that corresponds to 233 280 000 unknowns on the Earth's surface. The number of divisions in the radial direction has been determined by the available distributed memory of our cluster (1.7 TB). Hence, it has been set to 150, which has lead to 34 992 000 000 unknowns in the whole computational domain. The large-scale parallel computations were performed on 7 nodes with the following parameters: four 8-core CPUs with 256 GB RAM for each node. According to the NUMA architecture of the nodes, in our computations we used 56 MPI processes, each with 4 OpenMP threads.

In Table 8, we show a saving of the computation time and memory for the different number of subdomains solved with the parallel DD method. Due to the high time consumption, we present the computation time only for one DD iteration with parameter $\eta = 15$. In case of one subdomain, we present computation time for 15 iterations of the linear solver. One can see that this method brings a significant saving on memory but a slight increase in time due to increased communications between processes. The total number of DD iterations in the case of 10 subdomains was 86 and computation time was 5.925 10^4 [s] $\approx$ 17 hours.

No. sub. domains	**CPU time [s]**	**CPU time saving**	**RAM [GB]**	**Memory saving**
1	706.8	-	1 652	-
2	683.6	1.03	968	-41.4%
5	703.5	1.00	557	-66.3%
10	700.9	1.01	420	-74.5%
15	710.0	0.99	375	-77.3%
30	718.5	0.98	329	-80.0%

TABLE 8. Comparison for the different number of subdomains using Parallel-Domain decomposition method in the experiment with 34 992 000 000 unknowns, tested on 28 octo-core CPUs.

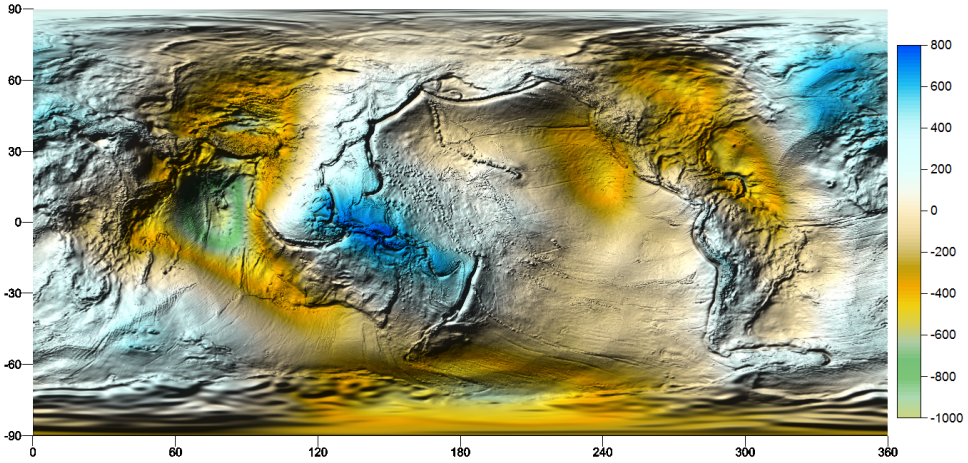

FIGURE 3. Global gravity field model with the resolution 1×1 *arc min* on the Earth's surface, $[m^2 s^{-2}]$.

Finally, in the second numerical experiment we have tried to increase a division in the radial direction as much as possible. Based on our experiences from the previous experiment, especially memory savings (Table 8, the last column), the number of divisions in the radial direction has increased to 400. The horizontal resolution has been the same, i.e., 1×1 *arc min*. Such a grid has consisted of 93 312 000 000 unknowns in the whole computational domain. To solve such a problem, it would require 4.5 TB of memory. By applying the DD method with 30 subdomains, it has allowed us to perform large-scale parallel computations using only 1 TB of the distributed memory of our cluster. Based on the NUMA architecture of the nodes, in our computations we used 56 MPI processes, each with 4 OpenMP threads. The computation time for 15 iterations was 2428 [s] which is much higher than in the previous experiment. It is due to 2.6 higher communications between processes. The total number of DD iterations was 236 and computation time was 5.7301 10^5 [s] $\approx$ 159 hours. The obtained numerical solution of GBVP on the Earth's surface is depicted in Figure 3.

Conclusions. In this paper we have presented a possibility to solve the geodetic BVP using the FVM approach on very detailed computational grids. We have demonstrated that applications of the domain decomposition (DD) method and their parallelization have allowed increasing a level of the discretization significantly. Consequently, we have been able to perform computations on more detailed grids. Namely, in our final numerical experiment we performed large-scale parallel computations with 93 312 000 000 unknowns in the whole computational domain. To solve such a problem, it would require 4.5 TB of memory. By applying the DD method with 30 subdomains, it required only 1 TB of the distributed memory of our cluster.

To reach such a level of the discretization, the FVM-approach is more efficient than the FEM approach. It is due to fact that FVM for such 3D problems results in a 7-point stencil while FEM in a 27-point stencil. It means that the FEM approach would require 17.3 TB of memory.

As a linear solver, the BiCGSTAB seams to be optimal for solving such kind of potential problems. Testing of the restarted BiCGSTAB has shown that the solver decreases the number of iterations, however, increases the time and memory requirements. Therefore, we recommend using the BiCGSTAB without restarting (Section 3.2).

In the case of the Additive Schwarz Method, the optimal choice of the parameter η for updating boundary conditions obtained from neighbouring solutions has been tested. Numerical experiments has shown that its optimal value is 15 (Section 4.2).

A parallel version of the DD method performed on our cluster has indicated that balancing between the numbers of the MPI Processes and OpenMP Thread configurations is based on a real configuration of the nodes with the NUMA architecture. Therefore, we used 4 OpenMP threads for every MPI process which resulted in the highest speed-up. To make a compromise between computation time and memory costs, the parallel version of DD method has been shown as the most suitable choice (Section 4.3).

All these findings have contributed to our effort to solve GBVP numerically using the FVM approach and to achieve its numerical solution as precise as possible.

Acknowledgments. Funded by the Government of Slovakia through an ESA Contract under the PECS (Plan for European Cooperating States), namely through the

PECS contract SK2-08: "GOCE-based high-resolution gravity field modelling in a space domain (GOCE-numerics)". The view expressed herein can in no way be taken to reflect the official opinion of the European Space Agency. This work was also supported by Grants APVV-15-0522 and VEGA 1/0486/20. We would like to also thank to the anonymous reviewers whose comments have greatly improved this manuscript.

REFERENCES

[1] O. B. Andersen, The DTU10 Gravity field and Mean sea surface, *Second International Symposium of the Gravity Field of the Earth (IGFS2), Fairbanks, Alaska*, (2010).
[2] Y. Aoyama and J. Nakano, RS/6000 SP: Practical MPI programming, *IBM.*, (1999), `http://www.redbooks.ibm.com`.
[3] X. Cai, Overlapping domain decomposition methods, *Advanced Topics in Computational Partial Differential Equations*, (2003), 57–95.
[4] T. F. Chan and T. P. Mathew, Domain decomposition algorithms, *Acta Numerica*, **3** (1994), 61–143.
[5] B. Chapman, G. Jost and R. Pas, Using OpenMP: Portable shared memory parallel programming, *The MIT Press, Scientific and Engin Edition*, (2007).
[6] R. Čunderlík, K. Mikula and M. Mojzeš, Numerical solution of the linearized fixed gravimetric boundary-value problem, *Journal of Geodesy*, **82** (2008), 15–29.
[7] R. Čunderlík and K. Mikula Direct BEM for high-resolution global gravity field modelling, *Studia Geophysica et Geodaetica*, **54** (2010), 219–238.
[8] V. Dolean, P. Jolvet and F. Nataf, *An Introduction to Domain Decomposition Methods. Algorithms, Theory, and Parallel Implementation*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015.
[9] Z. Fašková, R. Čunderlík and K. Mikula, Finite element method for solving geodetic boundary value problems, *Journal of Geodesy*, **84** (2010), 135–144.
[10] P. Holota, Coerciveness of the linear gravimetric boundary-value problem and a geometrical interpretation, *Journal of Geodesy*, **71** (1997), 640–651.
[11] P. Holota, Neumann's boundary-value problem in studies on Earth gravity field: Weak solution, *50 years of Research Institute of Geodesy, Topography and Cartography*, **50** (2005), 49–69.
[12] W. Keller, Finite differences schemes for elliptic boundary value problems, *Bulletin IAEG*, **1** (1995), Section IV.
[13] R. Klees, *Loesung des Fixen Geodaetischen Randwertprolems mit Hilfe der Randelementmethode*, Ph.D thesis, Muenchen, 1992,
[14] R. Klees, M. van Gelderen, C. Lage and C. Schwab, Fast numerical solution of the linearized Molodensky problem, *Journal of Geodesy*, **75** (2001), 349–362.
[15] K. R. Koch and A. J. Pope, Uniqueness and existence for the geodetic boundary value problem using the known surface of the earth, *Bulletin Géodésique (N.S.)*, **46** (1972), 467–476.
[16] T. Mayer-Gürr and et al., The new combined satellite only model GOCO03s, *International Symposium on Gravity, Geoid and Height Systems GGHS 2012*, (2012).
[17] P. Meissl, *The Use of Finite Elements in Physical Geodesy*, Geodetic Science and Surveying,Report 313, The Ohio State University, 1981.
[18] Z. Minarechová, M. Macák, R. Čunderlík and K. Mikula, High-resolution global gravity field modelling by the finite volume method, *Studia Geophysica et Geodaetica*, **59** (2015), 1–20.
[19] O. Nesvadba, P. Holota and R. Klees, A direct method and its numerical interpretation in the determination of the gravity field of the Earth from terrestrial data, *Proceedings Dynamic Planet 2005, Monitoring and Understanding a Dynamic Planet with Geodetic and Oceanographic Tools*, **130** (2007), 370–376.
[20] N. K. Pavlis, S. A. Holmes, S. C. Kenyon and J. K. Factor, The development and evaluation of the Earth Gravitational Model 2008 (EGM2008), *Journal of Geophysical Research: Solid Earth*, **117** (2012), 1–38.
[21] B. Shaofeng and B. Dingbo, The finite element method for the geodetic boundary value problem, *Manuscripta Geodetica*, **16** (1991), 353–359.
[22] G. L. G. Sleijpen and D. R. Fokkema, BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum, *Electron. Trans. Numer. Anal.*, **1** (1993), 11–32.

[23] M. Šprlák, Z. Fašková and K. Mikula, On the application of the coupled finite-infinite element method to geodetic boundary-value problem, *Studia Geophysica et Geodaetica*, **55** (2011), 479–487.

Received December 2018; 1st revision October 2019; final revision March 2020.

E-mail address: macak@math.sk
E-mail address: cunderli@svf.stuba.sk
E-mail address: mikula@math.sk
E-mail address: minarechova@math.sk