An Asynchronous Interface Relaxation Method for Multi-domain/Multi-physics Problems

Aigli Korfiati[a], Konstantis Daloukas[b], Panayiotis Alefragis[c], Panagiota Tsompanopoulou[b] and Spiros Likothanassis[a]

[a]*Department of Computer Engineering and Informatics, University of Patras, Patras, Greece, {korfiati, likothan}@ceid.upatras.gr*
[b]*Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece, {kodalouk, yota}@inf.uth.gr*
[c]*Computer & Informatics Engineering Department, Technological Educational Institute of Western Greece, Antirio, Greece, alefrag@teimes.gr*

Abstract. An approach for the solution of multi-domain and multi-physics problems is the application of an interface relaxation (IR) method to treat the solution on the common boundaries between domains of the original problem. This solution process is more efficient than other techniques, but still remains quite computationally intensive and the inherently parallel solution of the underlying problems does not scale to the overall method. This paper presents an asynchronous parallel algorithm of a specific IR method, named GEO. The performance results in terms of convergence speed and execution time demonstrate the efficiency of the proposed algorithm towards the solution of large-scale multi-domain and multi-physics problems.

Keywords: multi-domain/multi-physics problems, interface relaxation methods, asynchronous parallel algorithm.
PACS: 02.60.Cb, 02.60.Lj, 02.30.Jr, 07.05.Tp

INTRODUCTION

Solving composite Partial Differential Equations (PDEs) is an indispensable step in numerous scientific applications. However, this is a computationally and memory demanding process for large-scale differential equations. This is especially true for multi-domain and multi-physics problems where different PDE operators are applied on different subdomains. Domain decomposition techniques [1], [2] were primarily used to face such problems. They involve decomposition at the linear algebra level upon discretizing the domain and the equation with the desired method. Though, their main disadvantage is the non flexibility on the usage of different method per subdomain of the initial problem. Interface Relaxation methodology is an interesting alternative [3]–[5].

The IR methods' main advantage is that they treat a multi-domain and multi-physics problem as a loosely coupled system of sub-problems consisting of much simpler PDE problems concerning both the geometry and the differential operator. More specifically, in the IR methods the PDE domain is decomposed into subdomains, derived by the underlying physics or by high computational needs.

In this paper, we present an asynchronous parallel implementation of the GEO [6] IR method. The proposed methodology manages to overcome the traditional bottleneck of parallel IR methods and harness the underlying inherent parallelism of the solution of the subdomains, making the method scalable to large distributed parallel installation. A preliminary experimental evaluation of the asynchronous GEO demonstrates its scalability and efficacy for the solution of sizable multi-domain and multi-physics problems.

The rest of the paper is organized as follows. The proposed asynchronous algorithm and implementation details are described in Section 2, while Section 3 presents the experimental evaluation results. Finally, in Section 4 conclusions and future work are presented.

PROPOSED ALGORITHM

Sequential Algorithm

The PDE problem is divided in sub-problems, each of which discretized with the appropriate method. Boundary conditions have to be specified and applied to each sub-problem, as well as initial guesses on the interfaces between

the sub-problems. Then the solution and the gradient of each sub-problem are computed. After the solution step is performed, we have to get the values of the solution and the gradient on the interface points and compute the new relaxed values. With the GEO [6] IR method the new relaxed values on the interface points are obtained by adding to the old ones a geometrically weighted average of the normal boundary derivatives of the adjacent subdomains. Specifically, the solution at each iteration k+1 is given from the following equation:

$$u^{(k+1)}(x) = u^{(k)}(x) - \rho\left(\frac{\partial u_L^{(k)}(x)}{\partial n} - \frac{\partial u_R^{(k)}(x)}{\partial n}\right), k = 0,1,2,... \quad (1)$$

where, u is the computed solution on the interface point x, $\frac{\partial u_L^{(k)}(x)}{\partial n}$, $-\frac{\partial u_R^{(k)}(x)}{\partial n}$ are the values of the outward normal derivatives in the two adjacent subdomains and ρ is a relaxation parameter for convergence acceleration. These new relaxed values are then passed back to the sub-problems as updated values of the solution on the interfaces. Once the relaxed values of the interfaces are passed back to the sub-problems, a new iteration begins.

Synchronous Parallel Algorithm

GEO has a profound static parallel implementation, where each node represents a computational core that solves one subdomain. The computed PDE solutions and gradients on the interface points from each node are sent as a message to the node that handles the adjacent subdomain. When each node receives all adjacent interface points information, it computes the new relaxed interface points values, which serve as new estimates for the next iteration of the adjacent subdomains and sends them back as messages. When new interface values have been sent to all subdomains, the next iteration can begin in all nodes.

Asynchronous Parallel Algorithm

The main bottleneck of the parallel GEO algorithm emerges when the sub-problems have a lot of neighbors, which are of unequal size, and thus unequal computational workload. As a result, each sub-problem has to wait until all of its neighbors finish their solution step and new relaxed values have been computed for all the interfaces.

In the asynchronous parallelization approach, the synchronization step is temporally relaxed and each subdomain is iteratively computed based on the currently available interface values provided by the neighboring sub-domains. The above scheme creates a message queue for each subdomain, with the queue filled with messages about new interface values from the neighboring subdomains. In the case that there are no new messages the subdomain solver waits. If the queue is not empty, all messages are consumed in a LIFO order for each interface and older messages are discarded. New values for each interface are locally computed and a new iteration of the PDE solver starts for the respective subdomain. At the end of each local iteration, the new local values computed at all interfaces are asynchronously sent to the respective neighbors. The process ends when for all interfaces all subdomains fail to find significant difference in the new values computed.

EXPERIMENTAL RESULTS

In order to evaluate the performance results of the proposed asynchronous algorithm, we have employed a simplified problem that consists of three sub-problems and two interfaces.

It is described as follows:

$$Lu \equiv \nabla^2 u(x,y) + \gamma^2 u(x,y) = f(x,y), (x,y) \in \Omega \quad (2)$$

$$u(x,y) = u^b(x,y), (x,y) \in \partial\Omega$$

where Ω is $\left[0,\frac{1}{3}\right]\times[0,2]\cup\left[\frac{1}{3},\frac{2}{3}\right]\times[1,2]\cup\left[\frac{2}{3},1\right]\times\left[\frac{1}{2},2\right]$, and $f(x,y)$ and $u^b(x,y)$ are selected such that the true solution is:

$$u(x,y) = e^{y(x+4)}x(x-1)(x-0.7)y(y-0.5) \quad (3)$$

The interfaces are at $x_1 = \frac{1}{3}, x_2 = \frac{2}{3}$ while $\gamma^2 = 2$. Seven different cases are formed examining different grid sizes, according to seven different values of the discretization parameter h, which is considered equal in both x and y direction. Each different grid size corresponds to a different number of interface points, increasing from 6 to 321. The left sub-problem is approximately four times larger than the middle, while the right one is approximately two times larger than the middle sub-problem. The two interfaces have the same number of points and therefore are of equal workload.

The experiments are performed in 3 different predefined virtual machines. The virtual machines used for the experiments have 4 virtual cores and 2GB RAM and are running in a XenServer virtualization environment installed on a server with 2 x Intel(R) Xeon(R) CPU E5-2620, 2.00GHz.

The asynchronous implementation of GEO is compared to a synchronous parallel implementation of GEO recently presented in [7] in terms of convergence speed and execution time.

The following figures depict the convergence history of the asynchronous GEO implementation on the internal points of the interfaces. FIGURE 1 (a) depicts the exact and the computed solutions on iterations 1, 3, 6, 8 on interface 1 for the case c1. FIGURE 1 (b) shows the convergence history of interface 1 and case c1 for the synchronous algorithm. The results can be compared and one can see the similarity of the computed solutions per IR iteration for both synchronous and asynchronous algorithms. Similar results hold for all the test cases and both interfaces proving the fast convergence of GEO.

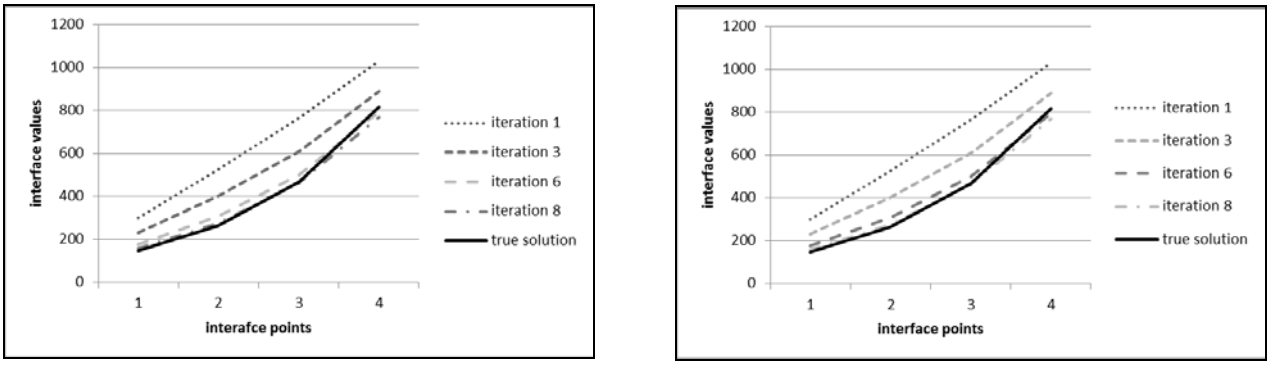

FIGURE 1. True solution and computed solutions on iterations 1, 3, 6, 8 on interface 1 for the case c1 in asynchronous **(a)** and synchronous **(b)** GEO.

In FIGURE 2 the max norm of the relative difference of successive solutions on interface points vs the number of iterations $r_i = \frac{\|u_{i+1} - u_i\|_\infty}{\|u_{i+1}\|_\infty}$ is presented. It concerns interface 1 for the case c1 for both the proposed asynchronous GEO and the synchronous GEO. Although for both algorithms the initial values on the interfaces u_0 are the same, this is not depicted in the figure because the first value is $r_1 = \frac{\|u_2 - u_1\|_\infty}{\|u_2\|_\infty}$. Again, similar results hold for all the test problems and both interfaces.

TABLE 1 presents the comparison of the execution times of the two implementations of GEO. For the synchronous implementation, the total time corresponds to the time for the solution of the three subdomains along with the interfaces computations' time and the necessary communication, for 15 iterations, while in the asynchronous implementation only the middle subdomain performs 15 iterations. The comparison of the execution times proves the high significance of the asynchronous implementation, especially as grids become finer and thus, problems get larger.

In order to study and clarify the contribution of the proposed algorithm, as instant steps for the employed test problem, we plan to analyze the timestamps of each IR iteration, for each interface, through the execution of the global problem, for both the synchronous and the asynchronous GEO. We will consider the relative error norm on the interfaces per iteration in order to depict the differences in the behavior of the two GEO. Finally, we contemplate to study how the workload of each node is lay out through the execution time of the global problem, by separating and indicating the time for the PDE solver, IR method, communication time and idle time for both synchronous and asynchronous algorithms.

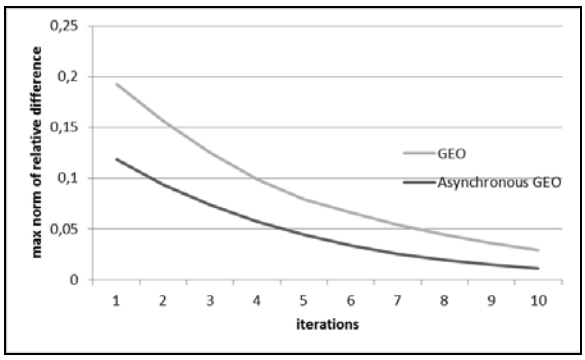

FIGURE 2. Max norm of the relative difference of successive solutions on interface 1 points vs the number of iterations for the case c1 for GEO and asynchronous GEO.

TABLE 1. EXECUTION TIMES OF ASYNCHRONOUS AND SYNCHRONOUS GEO

	c1	c2	c3	c4	c5	c6	c7
ASYNCHRONOUS GEO	12.12	12.361	11.81	7.727	14.805	38.022	256.084
SYNCHRONOUS GEO	3.495	4.068	6.14	13.595	39.808	154.633	723.731

CONCLUSION

In this paper an asynchronous parallel algorithm for the solution of multi-domain/multi-physics problems using the Interface Relaxation method is presented. Experimental results show that temporally relaxing the synchronization step between subdomains does not degrade the convergence properties of the solutions process and significantly lowers the wall clock execution time of the algorithm. During the experiments, more than 300% speed-up was achieved compared to the synchronous parallel implementation. Future work will include the integration of a work stealing scheme for better utilization of computational resources for problems where subdomains outnumber computational cores and extensive experimental tests with multiple scenarios in the utilization of the computational resources, a larger number of domains varying in size and difficulty and different scenarios in the composition of the original multi-domain problem.

ACKNOWLEDGMENTS

The present research work has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS. Investing in knowledge society through the European Social Fund (MIS 379416).

REFERENCES

1. Chan, T.F., Mathew, T.P., “Domain decomposition algorithms”, Acta Numerica 1994, Cambridge University Press, Cambridge, 61-143 (1994)
2. Keyes, D., Gropp, W., “A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation”, SIAM J. Sci. Statist. Comput. 8 s166-s202 (1987)
3. Rice, J. R., Tsompanopoulou, P., Vavalis, E., “Interface relaxation methods for elliptic differential equations”, Applied Numerical Mathematics 32 2, 219–245 (2000)
4. Rice, J.R., Tsompanopoulou, P. Vavalis, E.A., “Fine Tunning Interface Relaxation Methods for Elliptic Differential Equations”, Applied Numerical Mathematics, 43(4), 459–481 (2002)
5. Tsompanopoulou, P., Vavalis, E., “An Experimental Study of Interface Relaxation Methods for Composite Elliptic Differential Equations”, Applied Mathematical Modelling, 32 1620–1641 (2008)
6. Tsompanopoulou, P., Vavalis, E., “Analysis of an interface relaxation method for composite elliptic differential equations”, Journal of Computational and Applied Mathematics 226 2, 370– 387 (2009)
7. Korfiati, A., Tsompanopoulou, P., Likothanassis, S., “Serial and Parallel Implementation of an Interface Relaxation Method”, Proceedings of the 6th International Conference on Numerical Analysis, pp 167-173 (2014)