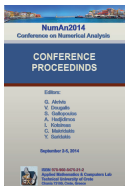
OpenAccess

Proceedings of the 6th International Conference on Numerical Analysis, pp 167-173

Contents lists available at AMCL's Digital Library

NumAn2014 Conference Proceedings

Digital Library Triton : http://lib.amcl.tuc.gr



Serial and Parallel Implementation of an Interface Relaxation Method

Aigli Korfiati[1], Panagiota Tsompanopoulou[2], and Spiros Likothanassis[1]

[1] Dep. of Computer Eng. and Informatics, University of Patras, Patras, Greece
[2] Dep. of Electrical and Computer Eng., University of Thessaly, Volos, Greece
{korfiati, likothan}@ceid.upatras.gr, yota@inf.uth.gr

Abstract. In the present paper an Interface Relaxation (IR) method, GEO, is implemented in FEniCS platform as a first step towards creating a complete multiphysics / multidomain problem solving environment. For the evaluation of the particular implementation, two different PDE problems are considered. A parallel implementation of the IR methodology using FEniCS is presented, as well as its performance comparison to the serial implementation. The performance results prove the significance of the parallel implementation towards the solution of large multiphysics / multidomain problems.

1 Introduction

The solution of large and composite Partial Differential Equations (PDE)s is a problem primarily faced with domain decomposition techniques [1, 2]. This approach involves decomposing at the linear algebra level after discretizing the domain and the equation with the desired method, i.e., Finite Differences (FD) or Finite Elements (FE). The main characteristic of these methods is the non flexibility on the selection of different method for each subdomain of the initial problem. Interface Relaxation (IR) methodology is an interesting alternative [3–5]. Here, the PDE domain is decomposed into subdomains, derived by the physics or for parallelization purposes while initial guesses are set on the interfaces between the subdomains. The subproblems are solved and new values on the interfaces are computed by particular IR methods (forcing the correct conditions for the problem) iteratively until convergence is succeeded.
Multidomain / multiphysics problem solving environments (PSE) implementing the interface relaxation methodology should be able to accommodate and incorporate a variety of existing PDE solvers and IR methods. These solvers should provide a minimum functionality including domain and PDE definition, mesh/grid generator, discretization scheme, evaluation of the solution and its derivatives at any point of the domain including the boundaries/interfaces. A complete list of existing software for the solution of differential equations can be found in [6].
According to our knowledge, a small number of implementations of IR methods can be found in the literature. Matlab has been used for seven IR methods, concerning one-dimensional problems [3] and for one concerning two-dimensional problems [7], not forming, though, a multidomain / multiphysics PSE. Two of these methods have been implemented in SciAgents Framework [8, 9] for two-dimensional problems. This

ISBN: 978-960-8475-22-9 http://lib.amcl.tuc.gr/handle/triton/49

implementation exploits the parallelism inherent in IR methodology using the Agents computing paradigm over a network of heterogeneous workstations. A second approach was accomplished with BOND agent middleware, [10]. Both SciAgents and BOND implementation used PELLPACK for their PDE solvers. GasTurbnLab [11, 12] is the latest complete approach. It is a multidisciplinary PSE for the gas turbine engine design based on the Grasshopper agent middleware and FORTRAN and C libraries. Last, a MATLAB toolbox that solves multidomain / multiphysics PDE problems is under construction, while a first stable version is presented in [13]. These PSEs highly depend on the agent paltforms and PELLPACK, revealing the need of a new implementation free of such constraints.
In this paper, a geometric (GEO) contraction based IR method [7] is implemented along with FEniCS. FEniCS [14] is a collection of free software for automated, efficient solution of differential equations and it is reliable, efficient, free and promising to be supported. Getting data on the interfaces and forcing new data as updated values for the boundary conditions is the main challenge of the IR methodology implementation and contribution of this paper. A parallel implementation of the GEO method using FEniCS and RabbitMQ (message-oriented middleware [15]), and its performance comparison to the serial implementation is presented in this paper too. The comparison of execution times for various problem sizes of a model problem, promises the high significance of the parallel GEO implementation towards the solution of big multiphysics / multidomain problems.

In Section 2 FEniCS platform and RabbiMQ tool are shortly described, while Section 3 contains the presentation of the IR methodology. Section 4 presents the implementation issues, while Section 5 contains the numerical results. Finally, in Section 6 the concluding remarks are presented.

2 Foundation of the Problem

Aiming at the incorporation of the GEO method, along with other IR methods, in a complete problem solving environment concerning multidomain / multiphysics problems it was implemented in FEniCS. FEniCS provides classes and methods to specify the problem's subdomains properties (i.e., domain's geometry, PDE operator and boundary/interface conditions). Its methods can also generate and/or refine meshes (triangular elements) for each subdomain, solve the local PDE problems and show the computed results in the global domain and on the interfaces.
Sparse LU decomposition is used as the default solver in FEniCS programs. This is because it is robust for a few thousand unknowns in the equation system. However, sparse LU decomposition becomes slow and memory demanding in large problems. So FEniCS provides iterative methods such as preconditioned Krylov solvers, as well, which are faster and require much less memory. A complete list of the available Krylov solvers and preconditioners can be found in FEniCS documentation [14]. The actual solvers implementations that are brought into action depend on the choice of a linear algebra package. FEniCS interfaces several linear algebra packages, called linear algebra backends in FEniCS terminology. PETSc is the default choice, otherwise uBLAS,

Epetra (Trilinos) and MTL4 are other supported backends.
For the parallel implementation on the FEniCS platform, RabbitMQ was used. RabbitMQ is a lightweight, reliable, scalable and portable message broker. It gives applications a common platform to send and receive messages. RabbitMQ runs on all major operating systems and is easy to use. It supports a huge number of developer platforms among which Python that is the FEniCS platfom. RabbitMQ is based on the Advanced Message Queuing Protocol (AMQP). AMQP is a message protocol that deals with publishers and consumers. The publishers produce the messages; the consumers pick them up and process them. In the present paper, a Remote Procedure Call (RPC) system was constructed, based on RabbitMQ since there is a great need for communication during the parallel solution of the PDE problems.

3 IR methodology and GEO

Interface Relaxation methods, including GEO, were presented and studied in [3–5]. Consider the composite differential problem defined by

$$Lu = f \quad \text{in } \Omega \setminus \partial\Omega, \quad u = u^b \text{ on } \partial\Omega \tag{1}$$

where u^b is a prescribed function on the boundary, $\Omega \equiv \bigcup_{i=1}^{p} \overline{\Omega_i}$ and $\Omega_i, \quad i = 1, \cdots, p$ are open sets such that $\bigcap_{i=1}^{p} \Omega_i = \emptyset$. L is a differential operator which might be different in each subdomain Ω_i. Considering IR methodology, the above problem can be replaced with the following loosely coupled system of differential problems.

$$L_i u_i = f_i \quad \text{in } \Omega_i$$

$$G_{ij} u = 0 \quad \text{on } (\partial\Omega_i \cap \partial\Omega_j) \setminus \partial\Omega \quad \forall j \neq i, \qquad u = {u_i}^b \text{ on } \partial\Omega_i \cap \partial\Omega$$

where for $i = 1, \cdots, p, L_i, f_i$ and ${u_i}^b$ are the restrictions of L, f and u^b respectively on each subdomain Ω_i and G_{ij} is a condition on the interface between subdomains Ω_i and Ω_j which enforces proper coupling. This coupling is responsible for preserving the physical properties of the original problem (i.e., continuity, smoothness or jumping).
The PDE operators and the coupling can be of any kind. Nevertheless, this study is limited to the most common case of second order elliptic differential equations with smooth global solution. Thus continuity of the solution and its first (normal) derivative should be imposed on the interfaces. GEO is such a method and its convergence analysis is presented in [7]. 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,

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

where k is the iteration, 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 used to accelerate the convergence.

4 The implementation

GEO is implemented as a FEniCS program written in the Python programming language. The DOLFIN [16] library is used to import classes that help us create the problem's subdomains and generate meshes (triangular elements) on these subsomains. We then specify and apply the boundary conditions, as well as the initial guesses on the interfaces of the subdomains. The PDE problem has to be expressed as a variational problem and then defined in the program. After the computation of the solution, the computation of the gradient is also performed.
As mentioned above, creating the appropriate functions for getting the values of the solution and the gradient on the interface points (boundaries of the subproblems), computing the new relaxed values and passing them back to the subproblems as updated values for the interfaces was the main challenge of GEO implementation. The relaxed values on an interface point x are computed as in (2). Once the relaxed values of the interfaces are passed back to the subdomains, a new iteration begins.
Since GEO is inherently parallel, in its parallel implementation, each node solves one subdomain. The computed solutions and gradients on the interface points from each node (solving a subdomain which has more than one adjacent subdomains) are sent as a RabbitMQ message to the node that handles the adjacent subdomain. This node computes the new relaxed interface points values, which serve as input to its new iteration and sends them back as a RabbitMQ message, so as for the next iteration to begin in the other node, as well. This schema not involving separate nodes to treat the interfaces serves for the reduction of the number of the messages exchanged, in an effort to minimize the total communication time.

5 Experiments and evaluation

In order to examine the method's correctness and the performance of the implementation, the following elliptic problem defined in [7] is considered:

$$\begin{aligned} Lu\left(x,y\right) \equiv -\nabla u\left(x,y\right) + \gamma^2 u\left(x,y\right) &= f\left(x,y\right), \quad (x,y) \in \Omega \\ u\left(x,y\right) &= u^b\left(x,y\right), \quad (x,y) \in \partial\Omega \end{aligned}$$

with $f\left(x,y\right)$ and $u^b\left(x,y\right)$ selected such that the true solution is:

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

In particular, two different PDE problems consisting of the above differential equation and boundary conditions and the two different domains depicted in Fig. 1 are studied. The interface points of the uniform problem are at $x_1 = \frac{1}{3}$ and $x_2 = \frac{2}{3}$ and of the non-uniform at $x_1 = \frac{1}{5}$ and $x_2 = \frac{1}{2}$ and $\gamma^2 = 2$. The experiments are performed for various values of the discretization parameter h, which is considered equal in both x and y direction. The resulting grid sizes are presented in Table 1, while the number of interface points in each case is equal to the number of points in y direction of the middle subdomain, i.e., increases from 6 to 641 points for both uniform and non-uniform cases. In the uniform case the left domain is the largest problem of the three with significant

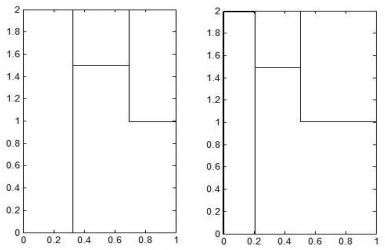


Fig. 1. Uniform (left) and Non-uniform (right) problems domains

Table 1. Considered test cases with their descritization step and grid size for left, middle and right domain for the Uniform and the Non-uniform problems.

		Uniform problem			Non-uniform problem		
case	h	left	middle	right	left	middle	right
c1	0.1	4x21	4x6	4x11	3x21	4x6	6x11
c2	0.05	8x41	8x11	8x21	5x41	7x11	11x21
c3	0.025	14x81	14x21	14x41	9x81	13x21	21x41
c4	0.0125	28x161	28x41	28x81	17x161	25x41	41x81
c5	0.00625	55x321	55x81	55x161	33x321	49x81	81x161
c6	0.003125	108x641	108x161	108x321	65x641	97x161	161x321
c7	0.0015625	214x1281	214x321	214x641	129x1281	193x321	321x641

difference of the others. In the non-uniform case, the right domain is the one with the heaviest task. All interfaces have the same number of points and therefore are of equal workload.

The convergence history is depicted in Fig. 2. Specifically, in the left graph, the max norm of the relative difference of successive iterants vs the number of iterations is shown for the interface $x_1 = \frac{1}{3}$ of the uniform problem for $h = 0.1, 0.05, 0.025$. As IR methodology promises, the rate of convergence is independent of the local discretization resolution h. For the same problem and interface, the exact and the computed solutions are plotted, in the right graph, for iterations $1, 3, 6, 10$ with $h = 0.05$. Similar results (not depicted due to space reasons) hold for all interfaces and both problems. In Table 2 the total execution times of the serial and the parallel implementation are presented. The serial execution is performed in a node with 4 Intel(R) Xeon(R) CPU E5-2620, 2.00GHz processors and 2GB RAM and the parallel in 3 nodes of the same configuration, in the Cloud Infrastructure of the Pattern Recognition Lab (CEID-UP).
The time of the serial executions contains the total time of computations of the three domains and two interfaces for 16 iterations. In the parallel executions, the total time corresponds to the time for the solution of the largest of the three subdomains along with the interface computations' time and the necessary communication, for 16 iterations. The test cases were set to explore the behavior of the parallel implementation and not for accuracy purposes. As the grids become finer, the work load increases significantly

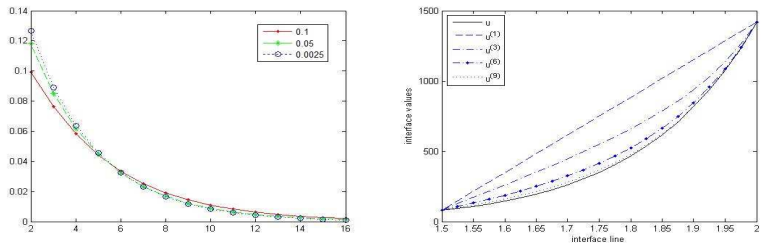


Fig. 2. Relative successive differences (left) and exact and computed solutions on interface points (right)

Table 2. Total execution times of serial and parallel implementation

	c1	c2	c3	c4	c5	c6	c7
				Uniform Problem			
Serial	1.900	2.664	5.216	12.883	40.883	178.217	996.348
Parallel	3.495	4.068	6.140	13.595	39.808	154.633	723.731
				Non Uniform Problem			
Serial	2.294	2.854	5.111	13.251	37.266	179.561	1057.644
Parallel	3.720	4.264	5.732	10.633	34.172	126.607	918.838

while the communication time increase with an order of magnitude less. This is the main reason that gain is noticed for fine grids.

6 Conclusions

This paper presents both serial and parallel implementation of GEO in FEniCS, as a first effort to build an environment for the solution of multidomain / multipysics problems. Parallel implementation seems to be invaluable, especially for large problems.
Nearby, further experimentation with more complex mutidomain / multiphysics PDEs, even finer grids and different solvers on each subdomain is planned. More IR methods are to be implemented as well. Communication issues on different machines are under study, while high and low level parallelism will be exploited too. Next steps also include creating a functionality in FEniCS, where the users will be able to: (i) define complex multiphysics PDEs, (ii) select the appropriate PDE solvers for the domains and IR methods for the intefaces and (iii) visualize the computed solution of the global 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, in: 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. Young, R., MacPhedran, I. 2006: Internet Finite Element Resources. Online. Available: $http://homepage.usask.ca/\ ijm451/finite/fe\_resources/fe\_resources.html$ (accessed July 11, 2014)
7. 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)
8. Drashansky T.: An Agent-Based Approach to Building Multidisciplinary Problem Solving Environments. PhD Thesis, Purdue University, Computer Science Department, (1996)
9. Rice, J.R., Tsompanopoulou, P., Vavalis, E.A.: SciAgents Tool: User's Guide. Tech. Rpt. TR-98-043, Dept. Computer Sciences, Purdue Univ., (1998)
10. Bölöni, L., Marinescu, D.C., Rice, J.R., Tsompanopoulou, P., Vavalis, E.A.: Agent Based Scientific Simulation and Modelling. Concurancy: Practice and Experience, 12, 845–861 (2000)
11. Markus, S., Houstis, E., Catlin, A., Rice, J., Tsompanopoulou, P., Vavalis, E., Gottfried, D., Su K., Balakrishnan, G.: An Agent-Based Netcentric Framework for Multidisciplinary Problem Solving Environments. International Journal of Computational Engineering Science, 1, 33–60 (2000)
12. Houstis, E.N., Catlin, A.C., Tsompanopoulou, P., Gottfried, D., Balakrishnan, G., Su, K., Rice, J.R.: GASTURBNLAB: A Multidisciplinary Problem Solving Environment for Gas Turbine Engine Design on a Network of Non-Homogeneous Machines. J. of Comp. and Applied Mathematics, 149(1), 83-100 (2002)
13. Chalkias, C.: Implementation of a Distributed System for the Solution of MultiDomain / MultiPhysics Problems, Diploma Thesis, (2013), Dep. of Electrical and Computer Eng., Univ. of Thessaly. Online. Available: $http://www.inf.uth.gr/wp-content/uploads/formidable/Chalkias\_konstantinos.pdf$
14. Logg, A., Mardal, K. A., Wells, G. N. et al.: Automated Solution of Differential Equations by the Finite Element Method. Springer, (2012)
15. Rabbitmq, 2014. Online. Available: $www.rabbitmq.com/documentation.html$ (accessed July 11, 2014).
16. Logg, A., Wells G. N.: DOLFIN: Automated Finite Element Computing. ACM Transactions on Mathematical Software 37, 2 (2010)