European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate, J. Périaux (Eds)
© TU Delft, The Netherlands, 2006

A HIGH SCALABILITY PARALLEL ALGEBRAIC MULTIGRID SOLVER

Marwan S. Darwish*, Tony Saad and Ziad Hamdan‡

*American University of Beirut, FEA, ME
Riad El Solh, Beirut 1107 2020, Lebanon
e-mail: darwish@aub.edu.lb
Web page: http://webfea.fea.aub.edu.lb/fea/me/cfd/

‡ Lebanese University, Faculty of Engineering
Ibn Sinaa Street, AL Oubbe, Tripoli, Lebanon
e-mail: hzia81@hotmail.com

Key words: Parallel Computing, Multigrid, Finite Volumes

Abstract. This paper deals with the implementation and performance analysis of a parallel Algebraic Multigrid Solver (pAMG) for a finite volume unstructured CFD code. The parallelization of the solver is based on the domain decomposition approach using the single program multiple data paradigm. The Message passing interface Library (MPI) is used for communication of data. An ILU(0) iterative solver is used for smoothing the errors arising within each partition at the different grid levels, and a multi-level synchronization across the computational domain partitions is enforced in order to improve the performance of the parallelized Multigrid solver. Two synchronization strategies are evaluated: in the first the synchronization is applied across the multigrid levels during the restriction step in addition to the base level, while in the second the synchronization is enforced during the restriction and prolongation steps. To increase robustness gathering of coefficients across partitions for the coarsest level is investigated. Tests on a number of grids from 100,00 to 800,000 elements for diffusion and advection problems have been conducted on up to 20 processors.

1 INTRODUCTION

Computational Fluid Dynamics (CFD) is an essential design tool in many industries (aerospace, automotive, chemical processing, power generation, etc …). At its basic level, CFD involves (i) a discretization step that translates a set of highly non-linear partial differential equations representing conservation principles (conservation of momentum, mass, energy, etc) into a sparse system of linearized algebraic equations, (ii) a solution step to solve the system of algebraic equations iteratively, iterative solvers are usually used for solving the system of equations (inner loop) because of their lower computational requirements in memory and CPU time[1]; finally (iii) because of nonlinearities, these two steps need to be performed repeatedly (outer loop)[2,3] until a final converged solution is reached. For transient problem the whole solution process is repeated as the solution marches in time (transient

loop).

With the continuous increase in complexity and size of CFD problems, techniques to accelerate this solution procedure have been the focus of intense effort over the past two decades. Work on resolving the inter-equation coupling of momentum and pressure efficiently either in a segregated[4,5] or coupled[6,7] manner yielded more robust and more efficient algorithms [8]. Work on the acceleration of the inner loop has led to the development of geometric multigrid methods[9,10,11], and later to algebraic multigrid methods (AMG)[12,13]. The AMG extended the main idea of geometric multigrid to a purely algebraic setting, yielding robustness and algorithmic simplicity. AMG can be used to build highly efficient and robust linear solver[14,15,16,17,18] by combining iterative solvers efficient in damping high frequency errors with multi-level grids that transform low frequency errors of the fine grids to high-frequency errors on coarser grids. This dual approach has been quite successful in tackling relatively large CFD simulations, but reaches its limits with large scale simulations for which the use of parallel processing becomes essential.

Parallel computing has undergone a small revolution of its own over the last decade. The continuously improving floating-point performance of the last few generations of micro-processors, and the availability of continuously cheaper high speed interconnection networks has meant that PC clusters (distributed memory) are increasingly being adopted as a cost effective alternative to classical parallel supercomputers (shared memory) for running large scale numerical simulations[19].

Parallel CFD codes can be written in several ways depending on the algorithmic implicitness/explicitness, the type of grid used (structured, unstructured, adaptive/non-adaptive), and the degree of coupling that exists or is incorporated into the implemented physical models. Many paradigms can be used, however, when the aim of the parallelization is to use clusters of interconnected processors[20], the Domain Decomposition approach is demonstrably the most effective.

In Domain Decomposition, the parallelization is enforced by dividing (partitioning) the domain of interest into a number of sub-domains or partitions (usually one for each processor). Each processor is then responsible for solving the computational problem within its sub-domain or partition. This is followed by a synchronization phase where neighboring sub-domains swap solution information to ensure consistency in the global solution of the original domain. This is equivalent to performing an inter-domain coupling at the outer loop; however, it is not sufficient if high scalability and robustness in performance are to be achieved. To overcome this shortfall, it is essential that an inter-domain coupling be performed also at the inner loop or linear solver level.

In the context of parallel multigrid solvers, a number of strategies can be followed to perform suitable synchronizations. On one extreme, synchronization can be performed at the finest mesh only, with the multigrid solver mainly playing the role of its sequential counterpart over the sub-domain or partition. On the other extreme, synchronization can be performed at each multigrid level, with the multigrid solver playing an additional role of smoother across partitions but at the expense of additional communication cost between partitions. Other methods include synchronization during the restriction phase or in both the restriction/prolongation phases. It is also worth noting that the number of communication

messages on coarse meshes is often nearly the same as that on fine meshes, although message lengths are much shorter. However, since most parallel architectures have high communication latencies as compared to current processor speeds, these can end up dominating coarse grid computations.

This paper reports on a number of techniques used in building a highly scalable and robust pAMG solver. Results from solving three test problems of various mesh sizes (100,000, 300,000, 500,00, 750,000 and 1,000,000 elements) are presented. The problems were selected to test the scalability and robustness of the pAMG solver.

In the remainder of this article the Algebraic MultiGrid (AMG) Method is presented with some emphasis on the agglomeration or coarsening algorithm. The Domain Decomposition Method and inter-Domain synchronization strategies are then outlined in the context of the AMG. Finally, the different options implemented in pAMG are outlined and the algorithms used described. Results and conclusions drawn across a range of computational meshes and partitions are presented..

2 THE FINITE VOLUME METHOD

The Finite Volume Method (FVM) is a numerical technique aimed at the solution of partial differential equations (PDEs), especially tuned to those arising in fluid, heat, and mass transfer problems. The general PDE governing the transport of a conserved passive scalar has the following form

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{Transient\ Term} + \underbrace{\nabla\cdot(\rho\mathbf{u}\phi)}_{Convection\ Term} = \underbrace{\nabla\cdot(\Gamma\nabla\phi)}_{Diffusion\ Term} + \underbrace{S^{\phi}}_{Source\ Term} \tag{1}$$

where ρ is the density, t is the time, $\boldsymbol{u}$ is the velocity field, Γ is the diffusivity, and ϕ is the unknown scalar for which a solution is sought. Upon discretizing equation (1), our arbitrary control volume, an algebraic equation of the following form is obtained:

$$a_i\phi_i + \sum_{n=nb(i)} a_i^n\phi_j = b_i \tag{2}$$

where the coefficients depend on the specific schemes used in the discretization process. The solution of these equations is the purpose of the linear iterative solver.

3 MULTIGRID METHODS

While the standard iterative techniques are quite efficient at solving small and medium sized meshes, their rate of convergence deteriorates for larger sized meshes. Brandt[21] studied the intricacies associated with solving a large set of equations and after decomposing the error in the approximate solution into Fourier components, he found that while many of the iterative solvers are efficient in eliminating high frequency or oscillatory components of the error, they are inefficient in reducing the remaining low frequency smooth components of the error. These solver are said to possess the smoothing property, and are said to act as

smoothers.

Multigrid algorithm were originally introduced independently by Federenko[22,23] (Geometric Multigrid) and Poussin[24] (Algebraic Multigrid) in the 60s, and later gained popularity with the work of Brandt[21,25,26]. They are considered as one of the most efficient techniques for the numerical solution of PDEs, at least for sequential computers. While standard iterative solvers (Jacobi, Gauss-Siedel, SOR, ILU) are efficient in removing high frequency errors, they are incompetent in removing the remaining low frequency or smooth errors[27]. Multigrid methods overcome the decay in the convergence rate by using a hierarchy of coarse grids in addition to the one on which the solution is sought. The fundamental idea is that by restricting the problem to a coarser grid, the lower frequency errors now appear more oscillatory. Applying this restriction recursively, a grid is eventually reached that is coarse enough for the problem to be solved accurately.

The application of MG to a given problem follows two stages. In the first stage, the coarse grids and their connectivities are setup using an agglomeration or coarsening algorithm[28,29,30,31,32]. In the second stage, a multigrid cycling procedure is used with a smoother to yield the solution at the finest grid. Alternatively, the coarse grids can be generated independently of the fine grid.

3.1 Element Agglomeration

Three different approaches can be adopted for the Agglomeration or Coarsening algorithm. The first approach begins with a coarse mesh definition and generates finer grids by refinement[33,34]. The main advantage of this approach is that the inter-grid operators become simple because of grid nesting. Another advantage is the possibility of utilizing this setup in an adaptive procedure where the fine meshes are formed by adaptively refining the coarse meshes[33,35]. The principal disadvantage however is the dependence of the fine grid distribution on the coarse levels. The second approach uses non-nested grids either with a subset of fine grid points comprising the coarse meshes or with completely independent coarse and fine meshes[36,37,38]. In this case the inter-grid information transfer operators become very expensive to construct. Furthermore, for both of the above outlined approaches, generating coarse grids that truly represent complex geometries can be a difficult proposition. In the third approach, essential for the AMG, the coarse grids are generated through agglomeration of the fine grid control volumes[39,40]. The agglomeration procedure can be based on a geometric relation between the elements of the grid or on a conditional relation between the mutual coefficients of elements. In the current implementation we follow the latter approach.

To generate the coarse grid sequence, an agglomeration algorithm is used to selectively fuse fine grid element to form the agglomerated or coarse grid elements. This process is repeated until all fine grid elements are fused into coarse elements and so on. The agglomeration process is heuristic and a number of algorithms can be used. In general fine grid elements are visited one by one. For a given seed element, a maximum kmax-1 of its adjacent elements are fused is they satisfy the agglomeration criterion. If the number of fused elements is less than kmax-1, then the neighbors of the fused element are evaluated for fusing

until the kmax-1 elements have been fused to the seed element, giving rise to a coarse grid element. If a seed element fails to form a coarse element it is queued to the less populated coarse element among its neighbors.

In this work, the Directional Agglomeration Algorithm (DA) of Mavriplis[41,42] is used. The agglomeration proceeds by fusing available fine grid elements according to the connectivity strength of the elements adjacent to the seed element, and the connectivity strength is governed by the geometric shape of the elements. Since the agglomeration is geometrical it is performed only once before the start of the run.

3.3 Multigrid Cycles

The Multigrid cycle refers to the way by which coarse grids are visited during the solution process. Two categories of cycling methods in multigrid can be defined: fixed and flexible cycles[43]. Initially only the W cycles was used. During restriction, the solver performs 2 sweeps on fine levels and 5 sweeps on the coarsest level while during prolongation; the solver performs only 1 sweep on all visited levels. When the coarsest level coefficients are assembled across partitions a direct solver is used.

5. PARALLEL CFD

Domain decomposition following the SPMD (Single Program Multiple Data)[44] programming model present a natural parallelization strategy for CFD codes. This approach, usually called *coarse grain* parallelization, is more suitable when using a cluster of personal computers for parallel computations[45,46]. In this approach the computational domain is decomposed into a set of sub-domains or partitions. Each of these partitions are distributed along with the problem definition to the different processors where the same program is run in a sequential manner but working on a different part of the original data, while allowing for synchronization and data exchange across partitions as needed. The process thus begins with the partitioning of the computational domain, followed by an iteration of discretization and solution phases on the respective processors, along with inter-partition synchronization. Finally the solution fields on the different partitions are sent to the main processor for reconstruction on the original mesh and saved for further manipulation. The main steps are detailed below.

5.1 Mesh Partitioning

METIS[47] was used for the partitioning of the computational mesh as it combines flexibility, good load balancing properties, in addition to minimizing communication across partitions by minimizing partition boundaries[48]. The output from METIS is a list of all elements with their assigned partition number. These elements form the basis of the computational mesh of the partitions and are denoted as core elements. Core elements at the boundary of a partition can have neighboring elements that are part of another partition. These neighboring elements are denoted by shadow elements. A low shadow to core element count in each partition is essential to minimize inter-partition communication during the solution phase. The shadow elements are added to the core partitions.

5.2 Partition Agglomeration Strategies

If the agglomerated grids are not synchronized across partitions, the benefits of multigridding are lost with increasing partition number, as the solution will proceed from partition to partition in an explicit manner. A better approach is to synchronize the multigrid levels across the partitions

Two approaches can be followed for the parallelization of the multigrid coarsening scheme described previously, and these result in a degradation of performance with increased partitions, thus decreasing any potential for scalability.

In the first approach, the coarsening takes place on the original mesh. The agglomerated grids are constructed then the fine grid is partitioned and the agglomerated elements are associated with the respective partitions and distributed to the different processors. The problem in this approach occurs when elements are agglomerated across partitions; in this case special treatment is needed for these elements that can degrade the performance of the multigrid solver. The agglomeration can be forced to occur on a partition basis but then, in this case, following the second approach becomes more appropriate.

In the second approach, the agglomeration is performed on core elements and starts after the partitioning step, and is performed in parallel in the different partitions. This is repeated until the coarsest grid level is reached[49]. At each level, partitions exchange information at the interface regarding the agglomerated shadow and sender elements. This information is determined from the agglomerated core elements.

5.3 Multigrid Synchronization

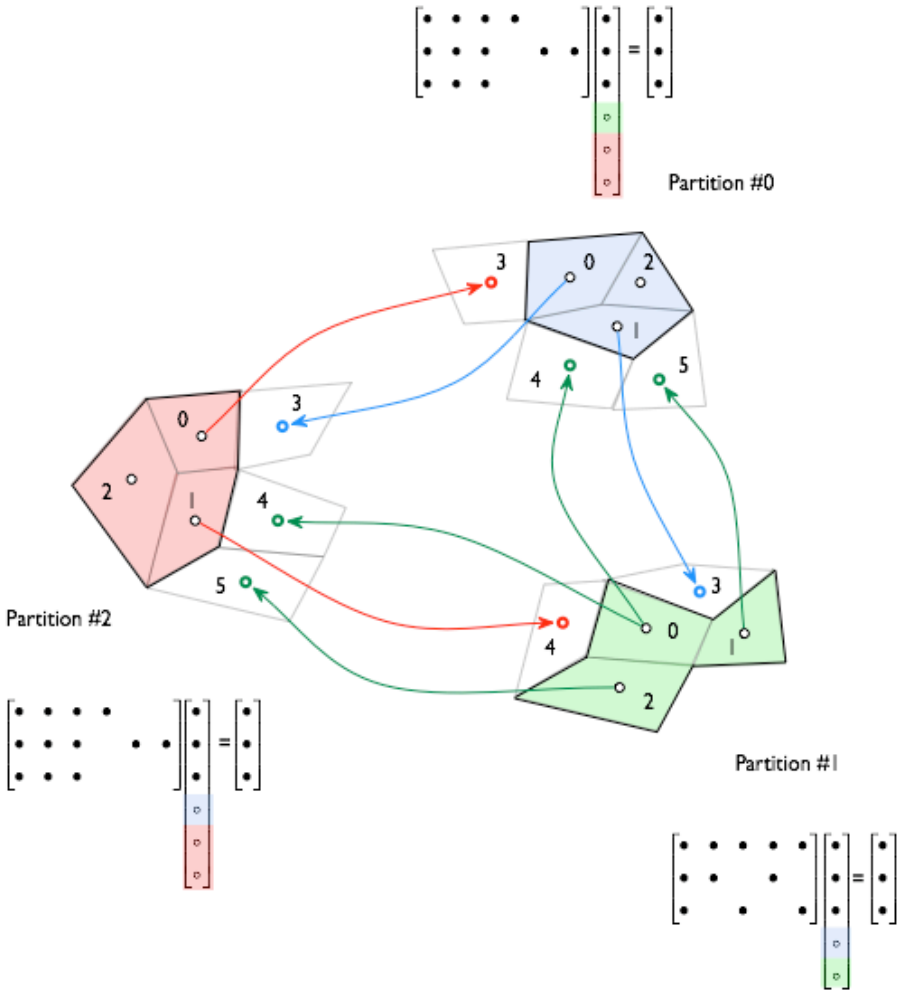
At each multigrid level the shadow elements are synchronized with their core elements. It is important to remember that if the benefits of synchronization do not offset its cost, then the performance of the pAMG will suffer. Two strategies were tested, in the first the synchronization is performed during the restriction step this is essential to ensure that the equations solved at each level represent the whole problem across the different partitions. The synhronization can also be performed during the prolongation step, this will ensure that some non-linearities are accounted for during the solution procedure.

The solver on the coarsest grid can limit the ultimate speedup that can be achieved in a parallel computation for two related reasons. First, the linear system at this level is generally small and the time required for communication may be higher that the time required to solve the system on a single processor. Second, the coarsest grid may couple all pieces of the global problem, and thus an accurate solution at this level is important, as is the global communication of the right hand side.

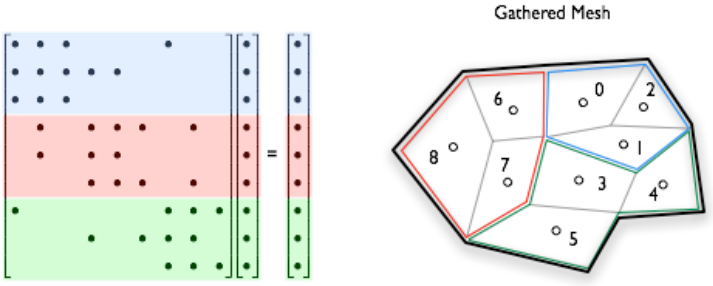
If the coarse grid is small enough, instead of solving in parallel, the coarsest grid problem may factored and solved on a single processor with the right hand side gathered and solution scattered to the other processors. Another option would be to solve the coarsest grid problem on all processors. This redundant form of the calculation does not require communication to distribute the results[50].

As the number of processors is increased, the coarsest grid problem may become too large. In this case, it might become preferable to do a parallel computation. However,

communication complexity can be reduced by solving with only a subset of the processors. Solving redundantly with a subset of the processors is again an option.



(a)



(b)

Figure 1 (a) coefficients in partitions (b) gathered coefficients

6. TEST PROBLEMS

In the first test, three diffusion problem illustrated in figure 2 were solved . The diffusion coefficient ratio vary between 1, 10 and 100 for the three problems. The importance of inter-partition coupling increasing with the diffusion ratios, as such the first test is used to evaluate the performance of the coefficient assembly techniques. The three problems were solved on grids with 100,000, 300,000, 500,00and 800,000 elements (control volumes). For each grid parallel runs on 2,4,6,8,10,12,14,16,18 and 20 partitions were performed on a cluster of ten G5 dual processor machines running at 2.0 GHz each with a memory of 512 MB per node, except for the master node with 2 GBytes.

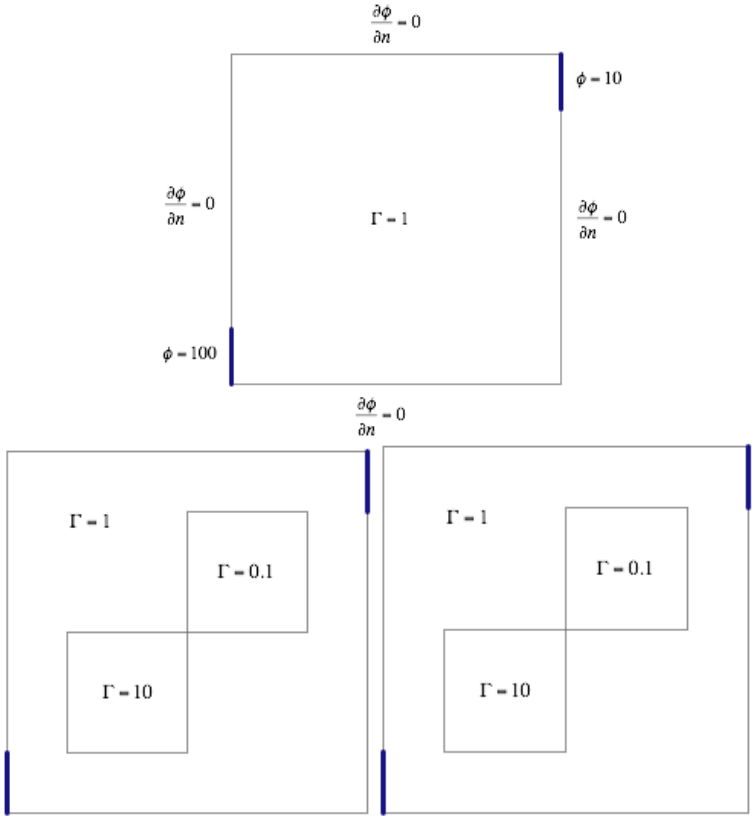


Figure 2: Test Problem 1

The interconnection network is a 100-MBits Ethernet switch. The runs were made until the residual was reduced to a value of 10^{-6} where the residual is defined as

$$r_P = \frac{b_P - \sum_{n=NB(P)} a_n \phi_n}{a_P} \tag{3}$$

Options A and B were used in the solution in test 1. The options are detailed in Table 1.

Option A	Synch multigrid levels during restriction (2 iterations) and prolongation (1 iteration) For coarsest levels assemble all coefficients on Master Node for inter-partition solution using direct solver
Option B	Synch multigrid levels during restriction (2 iterations) and prolongation (1 iteration) For coarsest levels 5 iterations
Option C	Synch during restriction only For coarsest levels 5 iterations

Table 1 : Options for inter-patition synchronization

Finally results are compared using two basic comparative parameters, the computational time **Speed-up**. To gain more insight into the results for each of the grids used, we also compute the shadow-to-core ratio. The shadow-to-core ratio for the used grids is shown in table 2. The shadow-to-core ratio for a grid increases with the number of partitions as the number of core elements decreases and the inter-partition boundaries increase. It can be viewed as a measure of the inter-partition communication. Generally it was found that the performance of the parallel solver starts to degrade when the shadow-to-core ration exceeded 2.33%.

	Number of Control Volumes			
Partitions	100,000	300,000	500,000	800,000
1	0.00%	0.00%	0.00%	0.00%
2	0.33%	0.39%	0.31%	0.26%
4	1.34%	0.85%	0.70%	0.55%
6	2.13%	1.63%	1.43%	0.99%
8	**2.33%**	2.03%	1.78%	1.37%
10	2.94%	**2.66%**	1.87%	1.56%
12	3.24%	2.72%	2.25%	1.73%
14	3.88%	3.08%	2.32%	1.88%
16	4.21%	3.33%	**2.57%**	1.98%
18	4.00%	3.40%	2.72%	2.25%
20	4.36%	4.02%	2.78%	**2.26%**

Table 2 : Shadow-to-Core ratio

Options A and B were used in the first test Results for the speedup is shown is figure 3.

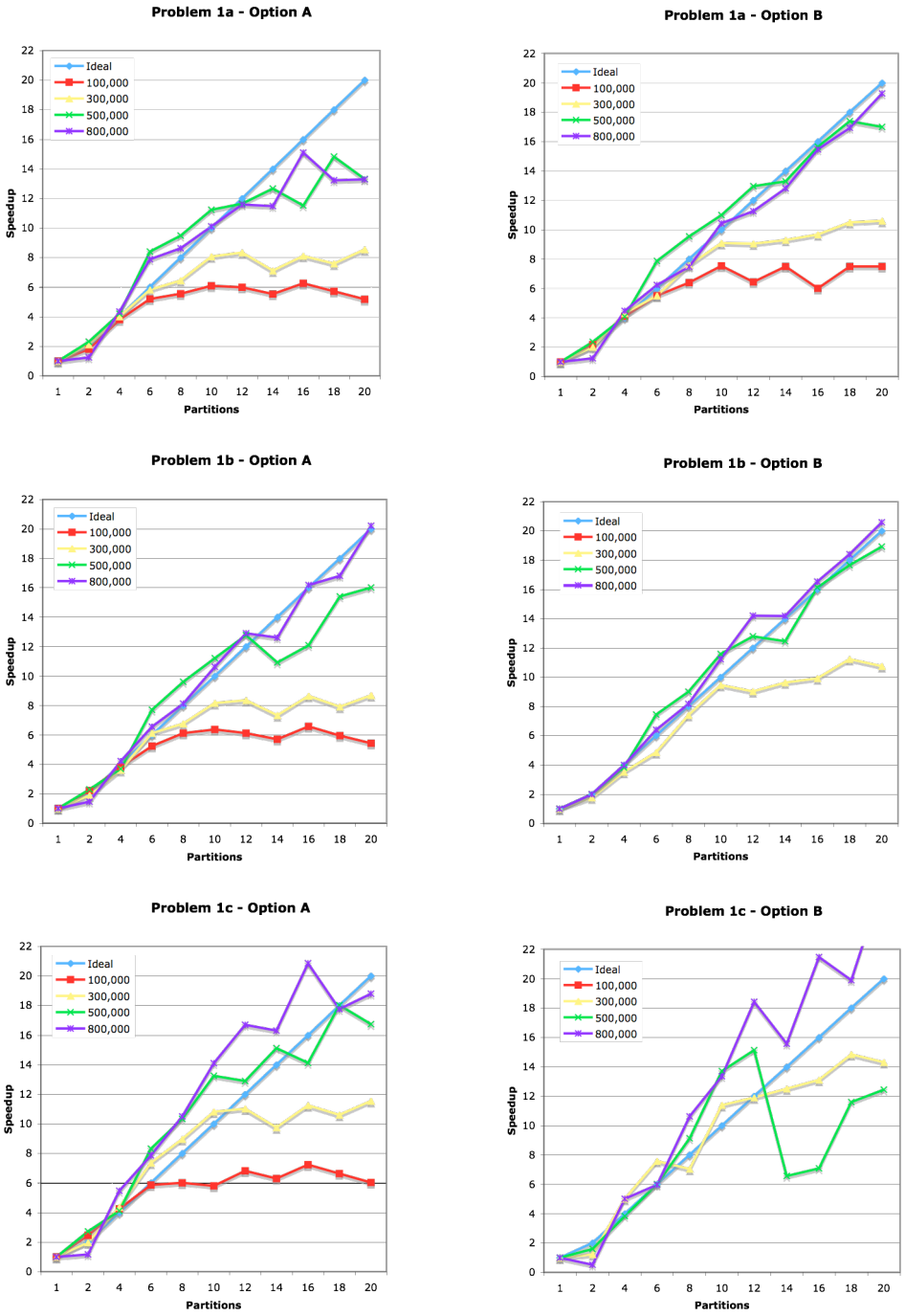


Figure 3: Speedup graphs for Problems 1a, 1b and 1c with options A and B

Figure 3 shows the speedup obtained for the four meshes using up to 20 processors. For the 100,000 and 300,000 element meshes the scalability degrades after 6-8 partitions, this is partly due to the large shadow-to-core ratio, The performance seems to improve as we move

from problem 1a to 1b to 1c with option A, this is due to the increased computational time needed to solve the problem sequentially as the diffusion ratio increases. With option B we also see an improvement in the performance of the solver, except for the fours runs in Problem 1c for mesh 500,000 and partitions 14 16 18 and 20. A look at table 4 explains this behaviour. The table lists for each computational mesh and partition number the Multigrid Level (MGL) used across the partitions and the Coarse Grid Elements (CGE) which is the total number of core elements across all the partitions, this last number is related to the number of MGL reached for the respective meshes and partitions. This number is computed based on the agglomeration algorithm, For the 500,000 mesh and partitions 14,16, 18 and 20 the total number of core elements at the coarsest level is around 40 since only 7 MGL are used, in contrast to the 800,000 computational mesh, where an extra multigrid level is used to lower the CGE number. As the CGE number increases we expect a degradation of performance with option B.

On the other hand since a direct solver is used in option A at the master node for the assembled coefficients, a large number of coarsest grid elements leads to a performance penalty. This could be resolved by using an iterative solver for the assembled coefficients.

	Computational meshes							
	100,000		300,000		500,000		800,000	
Partitions	MGL	CGE	MGL	CGE	MGL	CGE	MGL	CGE
1	8	5	8	5	9	3	9	4
2	8	4	8	5	9	3	9	4
4	8	7	8	8	8	10	9	4
6	8	9	8	6	8	11	8	14
8	8	18	7	23	8	11	8	18
10	7	26	7	21	8	11	8	16
12	7	25	7	24	8	12	8	13
14	7	26	7	24	**7**	**41**	8	16
16	7	37	7	23	**7**	**39**	8	17
18	7	24	7	23	**7**	**39**	8	19
20	7	30	7	21	**7**	**40**	8	20

Table 4 : Multigrid Levels and Number of Core Elements on Coarses Levels

Figure 4 shows the graph of efficiency vs. shadow-to-core, it is clear that beyond a certain shadow-to-core ratio the performance of the parallel solver degrade. This is to be expected as the communication cost increases with the number of shadow elements.

It can also be notes that for large meshes, there is a drop that occurs in performance for partition number of 2, this was found to be related to memory constraints on the slave nodes. For a two partition 800,000 mesh , each partition will have nearly 400,000 elements, this strains the memory of the slave nodes, leading to excessive use of virtual memory.

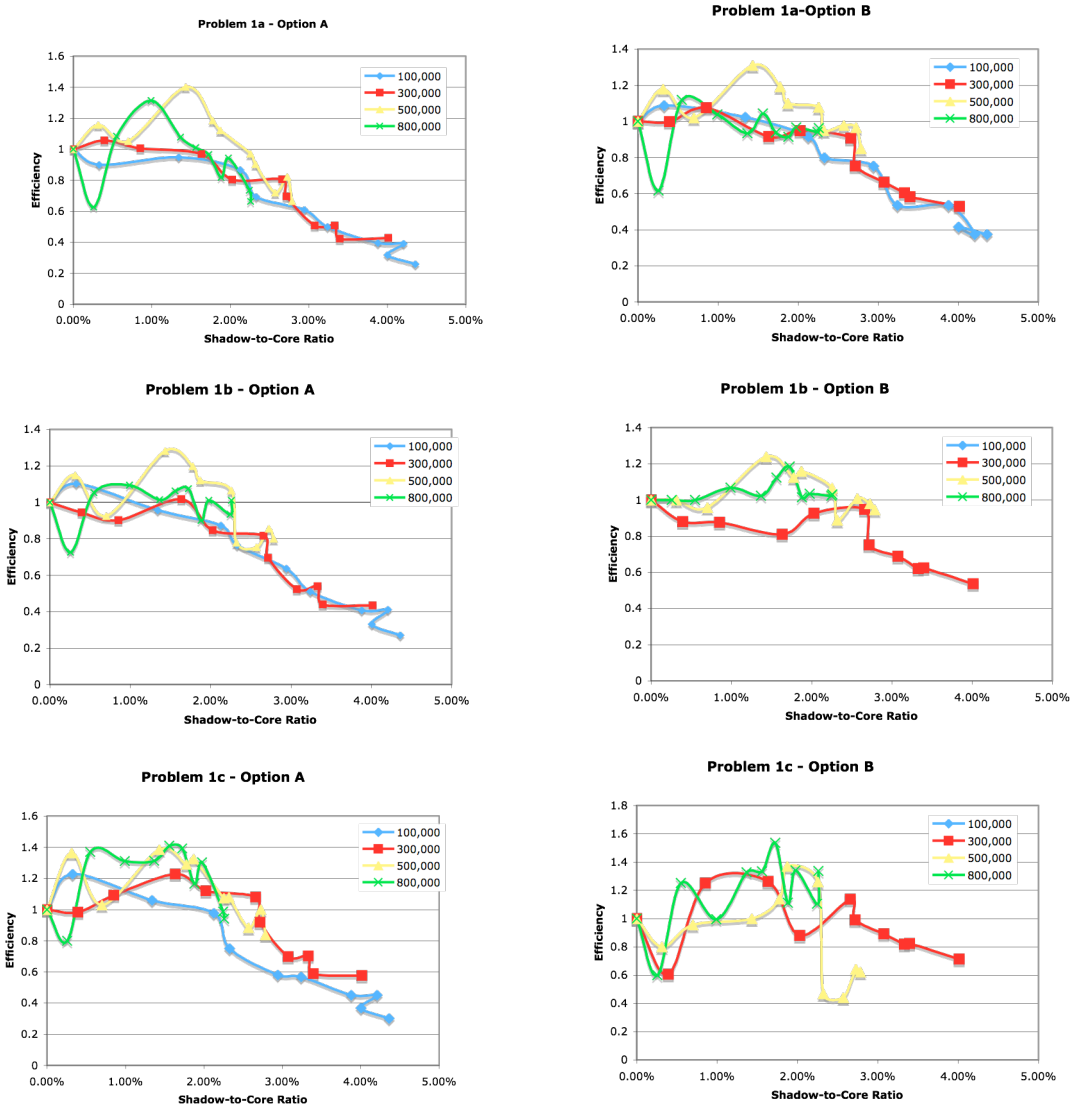


Figure 4: Efficiency vs. shadow-to-core ratio

Using option C resulted in divergence in all of the problems of test 1.

7. CONCLUSION

In this paper we have presented the performance of a pAMG solver. The performance of the parallel solver was shown to scale linearly for relatively large meshes with a shadow-to-core ratio above 2.33%, this was clearly shown using an efficiency vs. shadow-to-core graph. A technique for assembling the coarsest grid coefficients at the main node was shown to

improve the robustness of the code for cases involving sharp coefficient changes. However the cost of the assembly is relatively high and should be decreased.

The solver behaves extremely well for a range of processors. Future work will focus on optimizing the assembly process and solution process of coarsest grid level coefficients.

Acknowledgement

The authors wish to acknowledge the generous support of the University Research Board (URB) of the AUB and the Lebanese National Council for Scientific Research (LNCSR).

8. REFERENCES

[1] H. Martin Bucker, "Iteratively solving large sparse linear systems on parallel computers," NIC Serices, John Von Neumann Institute for Computing, Julich, **10**, 521-548, (2002).

[2] S.V. Patankar, Numerical heat transfer and fluid flow, Washington; New York: Hemisphere Pub. Corp.; McGraw-Hill, (1980).

[3] J.H. Ferziger and M. Peric, Computational methods for Fluid Dynamics, Berlin ; New York: Springer, (1999)

[4] Issa, R.I.,"Solution of the Implicit Discretized Fluid Flow Equations by Operator Splitting", Mechanical Engineering Report, FS/82/15, Imperial College, London, (1982).

[5] Van Doormaal, J. P. and Raithby, G. D."An Evaluation of the Segregated Approach for Predicting Incompressible Fluid Flows", ASME Paper 85-HT-9, Presented at the National Heat Transfer Conference, Denver, Colorado, August 4-7, (1985).

[6] Raw M. "Robustness of coupled algebraic multigrid for the Navier-Stokes equations", in Proceedings of the 34th Aerospace Sciences Meeting and Exhibit, Reno, NV, AIAA96-0297, (1996).

[7] Webster R., "Performance of Algebraic Multi-Grid Solvers based on unsmoothed and smoothed aggregation schemes", *Int. J. Numer Meth. Fluids*, **36**, 743-773, (2001).

[8] Moukalled F., Darwish M.S., A Unified Formulation of the Segregated Class of Algorithms for Fluid Flow at All Speed". Num. Heat Transfer, April 1999

[9] Brandt A. "Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Bounday Value Problems, in Proc. Of the Third Int. Conf. on Numerical Methods in Fluid Mechanics, Univ. Paris, 1972 H Cabannes, R. Teman eds. New York, Berlin, Heidelberg, (1973).

[10] Brandt A. “Multi-Level Adaptive Solutions to Boundary Vallue Problems”, *Math Comp,*, **31**, 333-390, (1977).

[11] Hackbusch W., “A Fast Numerical Method for Elliptic Boundaru VBalue Problems With Variable Coefficients”, in *2nd GAMM-Conf. Num. Methods Fluid Mech.*, Hirschel E.H. and Geller W. eds., Köln, 50-57, (1977).

[12] Brandt A. “Algebraic Multigrid Theory: The symmetric Case”, *Appl. Math. Comput*, **19**, 24-56, (1986).

[13] Brandt A., McCormick S.F., Ruge J.W. “Algebraic Multigrid for Automatic Multigrid Solutions with Application to Geodetic Computations”, Technical Report, Institute for Computational Studies, Fort Collins, Colorado, October, (1982).

[14] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteu S.F. McCormick, G.N. Miranda, and J.W. Ruge, “Robustness and scalability of algebraic multigrid”, *SIAM J. Sci. Stat. Comput.,* **21**, 1886-1908, (2000).

[15] Haase G., Langer U., Reitzinger S., Schöberl J., “Algebraic multigrid methods based on element preconditioning”. *International Journal of Computer Mathematics*, **78**, 575-598, (2004).

[16] Ruge J.W. and Stüben K.E.,”Finite Element Equations by Algebraic Multigrid (AMG). In: *Multigrid Methods for Integral and Differential Equations”,* H. Holstein, eds., The Institute of Mathematics and Its Applications Conference Series, Clarendon Press, Oxford, 169-212, (1985).

[17] Ruge J.W. and Stüben K., “Algebraic multigrid (AMG). In: Multigrid Methods”, *Frontiers in Applied Mathematics*, S.F. McCormick, ed., SIAM, Philadelphia, **3,** 73-130, (1987).

[18] Stüben K., “Algebraic multigrid (AMG): experiences and comparisons”, *Appl. Math. Comput.*,**13**, 419-452, 1983.

[19] Gropp W.D., Kauchik D.K., Keyes D.E., Smith B.F. “High performance parallel implicit CFD”, *Parallel Comput*ing, **27**, 337-362, (2001).

[20] V. Dolean and S. Lanteri, "Parallel multigrid methods for the calculation of unsteady flows on unstructured grids: algorithmic aspects and parallel performances on clusters of PCs," *Parallel Computing*, **30**, 503-525, 4. (2004).

[21] Brandt A., Multi-level adaptive solutions to boundary value problems, *Mathematics of Computations*, **31**, 333-390. (1977).

[22] Fedorenko R.P., A relaxation method for solving elliptic difference equations, *Computational Mathematics and Mathematical Physics*, **4**, 1092-1096, (1964).

[23] Fedorenko R.P.,The speed of Convergence of ote iterative process, Computational Mathematics and Mathematical Physics, **4**, 227-235, (1964)

[24] Poussin F. V. 'An Accelerated Relaxation Algorithm for Iterative Solution of Elliptic Equations', *SIAM J. Num. Anal*., **5**, 340-351, (1968).

[25] Brandt A., Guide to Multigrid Development, in *Lecture Notes in Mathematics*, Ed. A. Dold, B. Eckermann, (1982).

[26] Brandt A., "Multigrid Techniques, Guide with applications to fluid dynamics", (1984).

[27] Briggs W.L., "A Multigrid Tutorial", *Society of Industrial and Applied Mathematics*, (1987).

[28] Elias S.R., Stubley G.D., Raithby G.D., An Adaptive Agglomeration Method For Additive Correction Multigrid, *Int. J. Num. Methods in Eng.*, **40**, 887-903, (1997).

[29] Mavriplis D., Venkatakfrishnan V., Agglomeration multigrid for viscous turbulent flows", AIAA paper 94-2332, (1994).

[30] Ruge J.W., Stüben K., Algebraic multigrid. In Multigrid Methods, SIAM Conference, S.F. McCormick, Ed., 73-130, (1987).

[31] Vanek P. Mandel J., Brezina M., "Algebraic Multigrid by Smoothed Aggregation for second order and forth order elliptic problems", *Computing*, **56**, 179-196, (1996).

[32] Lonsdale R.D., "An algebraic multigrid scheme for solving the Navier-Stokes equations on unstructured meshes", in Taylor C., Chin J.H. and Homsy G.M. (eds) *Numerical Methods in Laminar and Turbulent Flow*, Pineridge press, Swansea, U.K., **7**, 1432-1442, (1991).

[33] Perez E., "A 3D Finite Element Multigrid Solver for the Euler Equations", INRIA Report# **442**, Sept. (1985).

[34] Connell S.D., Braaten D.G., "A 3D Unstructured Adaptive Multigrid Scheme for the Euler Equations", AIAA J., **32**, 1626-1632, (1994).

[35] Parthasarathy V., Kallinderis Y. "New Multigrid Approach for Three-Dimensional Unstructured Adaptive Grids", *AIAA J.*, **32**, 956-963, (1994).

[36] Peraire J. Peiro J., Morgan K., "A 3D Finite-Element Multigrid Solver for the Euler Equations", *AIAA Paper 92-0449*, Jan., 1992.

[37] Mavriplis D.J., "Three-Dimensional Multigrid Euler Equations Solver", *AIAA J*., **29**, 2086-2093, (1991).

[38] Mavriplis D.J., "Three-Dimensional Multigrid Reynolds-Averaged Navier-Stokes Solver for Unstructured Meshes", *AIAA J.*, **33**, 445-453, (1995).

[39] Koobus B. Lallemand M.G., Dervieux A. “Unstructure Volume-Agglomeration MG: Solution of the Poisson Equation”, INRIA Report# **1946**, (1993).

[40] Lallemand M., Steve H. and Dervieux A., “Unstructured Multigridding by Volume Agglomeration: Current Status”, *Computer and Fluids*, **21**, 397-433, (1992).

[41] Mavriplis D.J., “Directional Agglomeration Multigrid Techniques for High Reynolds Number Viscous Flow Solvers”, *AIAA J.,* **37**, 393-415, (1999).

[42] Mavriplis D.J., “Directional agglomeration multigrid techniques for high-Reynolds-number viscous flows”, *AIAA J.*, **37**, 1222–1230, (1999).

[43] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, (2001).

[44] B. Wilkinson and C.M. Allen, Parallel programming : techniques and applications using networked workstations and parallel computers, Upper Saddle River, N.J.: Prentice Hall, (1999).

[45] L.C. Dutto, W.G. Habashi and M. Fortin, "An algebraic multilevel parallelizable pre-conditioner for large-scale CFD problems", *Comput.Methods Appl.Mech.Eng.*, **149**, 303-318, (1997).

[46] D. Kwak, C. Kiris and C.S. Kim, "Computational challenges of viscous incompressible flows", *Comput. Fluids*, **34**, 283-299, (2005).

[47] G. Karypis and V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs", *Journal of Parallel and Distributed Computing*, **48**, 96-129, (1998).

[48] Kevin McManus, "A strategy for mapping unstructured mesh computational mechanics programs onto distributed memory parallel architectures," School of Computing and Mathematics Science, University of Greenwich, London, UK, vol. 1, pp. 1-213, 1995. 1995

[49] A. Krechel and K. Stüben., “Parallel algebraic multigrid based on subdomain blocking”. *Parallel Computing*, **27**,1009–1031, (2001).

[50] Gropp W. D., “Parallel computing and domain decomposition”, in T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., *Fifth Conference on Domain Decomposition Methods for Partial Differentia equations*, 349-362, (1992).