Mecánica Computacional, Volumen XIX, pp. 91-96
Compilado por: F. Quintana y S. Felicelli
San Carlos de Bariloche, Noviembre de 2000

SOLVING THE POISSON EQUATION WITH MULTIGRID METHODS

Luis Cardón
INENCO - Instituto UNSa-CONICET
Universidad Nacional de Salta, Buenos Aires 177, 4400 Salta, Argentina
cardon@unsa.edu.ar

ABSTRACT

This paper reports the implementation of a full multigrid method for the solution of pressure like 3D Poisson equations for control volume discretization over structured grids. The method was tested on a four level 40x40x40 nodes grid and converges to $L_2 \sim 10^{-5}$ for the residual. The implementation is faster than the SIPSOL method used as a smoother, although there is space for performance improvements.

INTRODUCTION

This paper reports the development of a full multigrid solver for Poisson equation. This is intended to be used in a Large Eddy Simulation (LES) model for the Navier Stokes equation, also under development. LES requires accuracy in the time evolution and three dimensionality. The applications in sight requires geometrical complexity beyond the capabilities of many other fast solvers. Multrigrid methods appeared as the right choice.

In the past decade various solvers for NS and other equations are based on some sort of multigrid technique. Among them, we can cite Arad and Martinelly [1], Dawood and Burns [2], Muzaferija, Lilek and Peric [6] and many others.

THE POISSON EQUATION AS THE PRESSURE EQUATION IN FRACTIONAL STEPS METHODS

A Poisson like equation appears naturally in the application of fractional steps methods to the solution of Navier Stokes equations. Fractional steps methods let us solve these three equations in stages, solving for the velocities, disregarding the pressure term. This yields an approximate velocity field that does not satisfy continuity. Enforcement of continuity yields a Poisson like equation as follows.

Navier Stokes equations can be compactly expressed as

$$\frac{\partial u_i}{\partial t} = \mathcal{H}_i + \frac{\partial p}{\partial x_i} \tag{1}$$

where $\mathcal{H}_i$ stands for the convective, the diffusive and the sources terms,

$$\mathcal{H}_i = -\frac{\partial u_i u_j}{\partial x_j} + \frac{\partial 2(\mu) S_{ij}}{\partial x_j} + \rho f_i \tag{2}$$

For incompressible flow, the continuity equation is

$$\partial_i u_i = 0 \tag{3}$$

and can be written as a Poisson equation for pressure. For the numeric solution of these equations we are going to use a very simple fractional step method. Let us consider H_i be the spatial discretization of $\mathcal{H}_i$, and δ be the central difference operator (actually control volume approach), then the discretization equation for 1 is

$$\delta_t u_i = H_i + \delta_x p \tag{4}$$

where

$$\delta_t u_i = \frac{(\rho u_i)^{n+1} - (\rho u_i)^n}{\Delta t} \tag{5}$$

and

$$\delta_x p = \frac{p^{n+1} - p^n}{\Delta x} \tag{6}$$

the exact nature of the discretized $\mathcal{H}_i$ does not matter at this point. First we solve equation 1 for the velocity at time $n+1$, disregarding the pressure term. This approximate velocity doesn't satisfy the continuity equation, so it will be denoted with an asterisk,

$$\frac{(\rho u_i^*) - (\rho u_i)^n}{\Delta t} = H_i \tag{7}$$

The difference between u_i^* and u_i^{n+1} is

$$\frac{(\rho u_i^*) - (\rho u_i)^n + 1}{\Delta t} = \delta_x p \tag{8}$$

furthermore, taking the numerical divergence of this equation we have

$$\delta_x^2 p = \frac{1}{\Delta t}\left(\delta_x(\rho u_i^*) - \delta_x(\rho u_i)^{n+1}\right) \tag{9}$$

now we can enforce continuity at time level $n+1$ making $\delta_x(\rho u_i)^{n+1} = 0$. Finally we got

$$\delta_x^2 p = \frac{1}{\Delta t}\delta_x(\rho u_i^*) \tag{10}$$

Momentum equations can be solved explicitly or implicitly. In cases when we are interested in close time evolution, like in the case o Large Eddy Simulation, explicit methods are the adequate ones. Solving the Poisson equation can take 50% of the computation time for a complete time step. Thus, a fast solver is very important.

SOLVING THE POISSON EQUATION

Many alternatives have been analyzed before choseosing a solver. Fourier methods, cyclic reduction, spectral methods. Early successful use of fractional step methods for Navier Stokes equations have been achieved by Kim and Moim [4]. They solve the Poisson equation with direct methods based on trigonometric expansion. This method is applicable when the equation has constant coefficients in space and the grid lines coincide with the boundaries. Different procedures appliy for periodic, homogeneous or inhomogeneous, Dirichlet, Neumann, or Robins boundaries condition. For the pressure Poisson equation, the coefficients are only related with the discretization mesh. A more general problem, like a heat conduction one, would need allowance of non constant coefficients. We also require the method to be applied to non uniform meshes.

Cyclic reduction o Fourier analysis and cycled reduction is applicable when the equation is separable and the mesh fits well with the boundaries, thus restrcting too much our applications.

Multigrid method is a class techniques to improve performance of iterative solvers, whose performance do well even compared with spectral methods when applied in explicit codes for Large Eddy Simulations, see Gavrilakis at al.,[5].

TWO GRID METHODS

The base module of many multigrid implementation, and of this one in particular, is the two meshes algorithm. Starting with a test field ϕ_0^h, we want to iteratively solve the following problem:

$$\delta^2 \phi^h = f^h \tag{11}$$

subject to boundary condition ϕ_Γ^h. After a number of n iteration steps we got the ϕ_n^h approximation to the solution. At this stage we can compute the residual or defect d^h

$$d^h = \delta^2 \phi_n^h - f^h \tag{12}$$

then the error of the solution is

$$\epsilon^h = \phi_n^h + \phi^h \tag{13}$$

The linearity of the operator δ^2 makes it possible to build the following problem

$$\delta^2 \epsilon^h = d^h \tag{14}$$

subject to $\epsilon^h = 0$ over Γ.

Iterative methods are fast to smooth grid size frequencies of the solution and very slow to any other, thus, to accelerate convergence, multigrid methods solve the error on a coarser grid. To do this the error problem is built on a grid of size H, twice as coarse as the first one. The source term, the defect, is restricted to the coarser grid

$$d^H = \mathcal{R} d^h \tag{15}$$

where $\mathcal{R}$ is the restriction operator. Finally we solve the following problem

$$\delta^2 \epsilon^H = d^H \tag{16}$$

with boundary condition $\epsilon^H = 0$ over Γ, starting with test field $\epsilon_0^H = 0$. On the H grid we look for the best solution we can afford. The error ϵ^H is prolonged to the fine grid in order to correct the solution ϕ_n^h, thus

$$\epsilon^h = \mathcal{P} \epsilon^H \tag{17}$$

where $\mathcal{P}$ is the prolongation operator. Then, we can correct the solution

$$\phi^h = \phi_n^h + \epsilon^h \tag{18}$$

If necessary, a new smoothing iteration can be done over ϕ using the corrected value of ϕ as the test field. Figure 1 schematize these ideas.

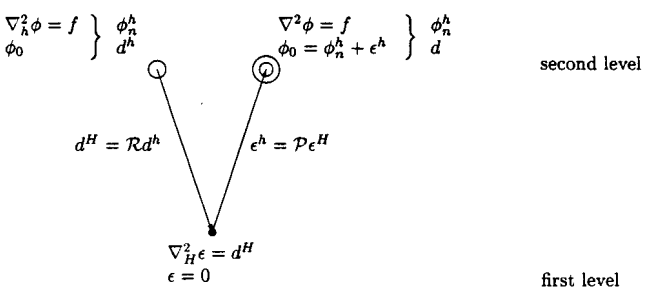

Figure 1: The two-grid algorithm.

IMPLEMENTATION

The differential operator in equation 10 is actually the result of the usual Patankar's control volumes method.

To build differential operator on the coarse grid we use the so called **coarse discretization approximation.** In this practice the δ_H^2 operator obtained by discretizing the differential equation on the coarse grid. Here the coarse discretization approximation is also *compatible coarsening*, the coarsening follows the same kind of discretization technique used for the fine grid problem. This is a desirable property of an algorithm to be used in connection with a Navier Stokes solver.

The coarse grid approximation is easier to implement, as it takes advantage of the same routine that builds the δ^h coefficients matrix. Additionally it allows the change of the restriction or interpolation operators without any change in the coefficients matrix.

The restriction operator $\mathcal{R}$

Most of the complexities of the implementation is in the the data structure to handle two grids, and in the full multigrid approach any number of grids. For this, we have used the data structure of a program published by Ferziguer and Peric [3]. Also we use their mesh generator and the solver used as the smoother. All the rest of the implementation of the multigrid algorithm was programmed from scratch.

The prolongation operator $\mathcal{R}$

To prolongue the values from the coarse to the fine grid, a prism defined by eight node in the coarse grid is taken. The prism enclose eight control volumes of the fine grid. The values of the later are interpolated from the first using threelinear interpolation. Formulas of common use in finite element practice are used. Implementation is done with two subroutines; **ftoc6** which swept the coarse grid grouping control volumes in packets of eight and **inter6** which does the actual interpolation. In this way, it is easy to change to some extent the interpolation method, if so is required.

The same approach is used for the restriction. In this case, eight control volumes in the fine grid will collapse in one control volume in the coarse grid. The eight nodal values on the fine grid are used to interpolate threelinearly a nodal value in the coarse one. The same subroutine **inter6** is used by **ftoc6** to do the job. Although we have used the same algorithm for interpolate top to bottom or bottom to top, this is not a requirement of the method.

The two level grid approach is also called a "V" cycle, and we can do it any time it is necessary. The "V" cycle is schematize in figure 1. V cycles can be done in many levels too. The subroutine **VCY** implements them on any number of levels and let control the number of smoothing swepts on the descending branch, called pre-swepts, and on the ascending branch, called pos-swepts. It also let control the number of swepts on the lower level grid.

FULL MULTIGRID METHODS

It is well known that efficiency of iteration methods improves when a test field near the solution is chosen. It si possible to estimate a good test field on a coarse grid and then prolongue its value to a finer grid where the "actual calculation of the problem" will be done. On the fine grid we can use the two grid algorithm described before.

A further step is to go up any number of levels to reach the ultimate fine size of grid we desire for the calculation. This algorithm is schematize in figure 2.

A full multigrid algorithm with any number of "V" cycles at each level and any number of levels was implemented (although this must be set at compilation time). A subroutine called **MGSIP2** implement the initial test calculation on the first level and lunch a nested iteration of "V" cycles on all the levels of grids. **MGSIP2** can be used with any smoother, among them, Gauss Seidel, SIPSOL, CGSTAB , but so far it has been tested with the first two.

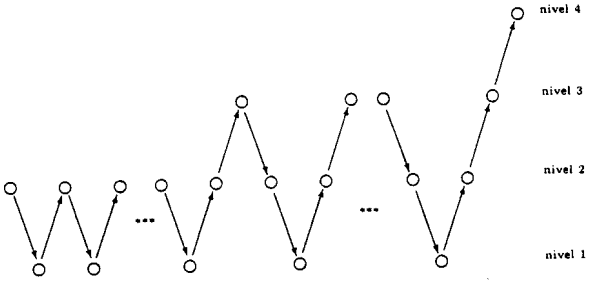

Figure 2: The full multigrid algorithm.

TEST PROBLEM RESULT

For the purpose of checking the accuracy and performance of the implementation, we have first solved a pressure like Poisson equation subject to Neumann boundary conditions. Two sources have been tested: a sinusoidal function $f = sin(\pi x)sin(\pi y)sin(\pi z)$ and a uniform source located in a centered small cube. The right solution has been achieved in both cases. Only the second one is reported here.

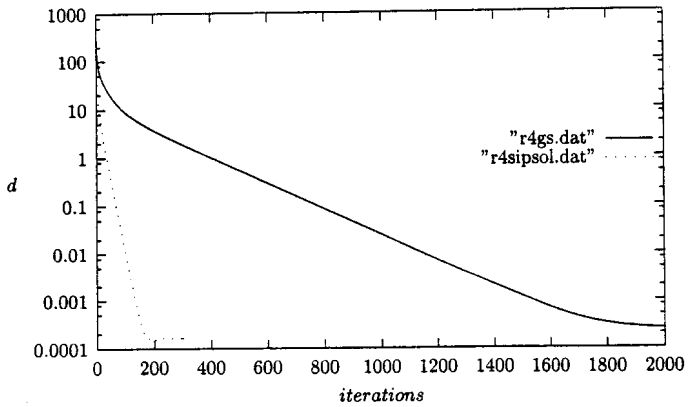

Figure 3: Evolution of the residue with iteration number

Table 1 shows the summary results of six typical runnings with the four level full multigrid algorithm on an $40 \times 40 \times 40$ grid. The program was compiled with g77 on a Linux System running on an AMD K6 II processor at 350 Mhz, with 251MB of RAM. The program was previously compiled with f90 and ran over a $80 \times 80 \times 80$ mesh (five level full multigrid) on a Linux dual processor PC. We could not reproduce the last calculation on our PC probably because a poor configuration.

We can compare performance with SIPSOL results shown in figure 3. To get an L_2 norm of the residue to 0.001 there is a need of 45 iterations, approximately. They take about 14.53 seconds. Compared with run 2, 4 or 5, even though MG-SIPSOL is better, there is not much of a difference. To get $L_2 = 0.0002$ it is required 65 iterations with SIPSOL, and it takes 16.88 seconds, while run 7 shows that the incremental time requiered by MG-SIPSOL get the same accuracy is very small. On a finer grid these results show even better.

Run	V cycles	pre swepts	post swepts	bottom swepts	L_1	L_2	time, sec.
1	1	1	1	1	0.47	0.008	9.16
2	2	1	1	1	0.16	0.001	10.48
3	3	1	1	1	0.10	0.0005	11.71
4	2	1	1	10	0.15	0.001	10.53
5	2	1	1	20	0.15	0.001	10.63
6	2	1	2	20	0.09	0.0005	10.88
7	2	1	5	20	0.04	0.0002	11.72
8	3	1	5	20	0.023	0.0001	13.47
9	4	1	5	20	0.015	$7.9\ 10^{-5}$	15.24
10	7	1	5	20	0.005	$2.8\ 10^{-5}$	20.63
11	10	5	5	20	0.0002	$1.2\ 10^{-6}$	31.54

Table 1: Comparison among various full multigrid schemes.

CONCLUSION

A set of subroutines that implement various kinds of multigrid methods have been implemented for 3D Poisson like equations. These subroutines satisfy the basic requirement of convergence to the machine accuracy level. They compare favorably against SIPSOL in the test problem. We can expect that on finer grids, the properties of multigrids methods will prevail against the use of a stand alone smoother. Also, we think that there is ample space for improvement and optimization.

References

[1] E. Arand and Luigi Martinelli. Application of a parallel, multigrid algorithm for large eddy simulation. In K. Hanjalic and T.W.J. Peeters, editors, *Turbulence, Heat and Mass Transfer 2, Proceedings of the second international symposium on turbulence, heat and mass transfer, Delf, The Netherlands*, pages 499–508, 1997.

[2] A. S. Dawood and P. J. Burns. Steady state three-dimensional convective heat transfer in a porous box via multigrid. *Numerical Heat Transfer, Part A*, 22:167–198, 1992.

[3] J. H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. Springer, 1996.

[4] J. Kim and P. Moin. Application of a fractional-step method to incompressible mavier-stokes equations. *Journal of Computational Physics*, 59:308–323, 1985.

[5] P.R. Voke S. Gavrilakis, H.M. Tsai and D.C. Leslie. Simulation methods for low reynolds number channel flow. *Numerical Methods in Laminar and Turbulent Flow*, 5, Part 1:435–445, 1987.

[6] S. Muzaferija Z. Lilek and M. Peric. Efficiency and accuracy aspects of a full-multigrid simple algorithm for three-dimensional flows. *Numerical Heat Transfer, Part B*, 31:23–42, 1997.