The Effect of the Grid Aspect Ratio on the Convergence of Parallel CFD Algorithms

Y. F. Hu, J. G. Carter and R. J. Blake

Daresbury Laboratory, CLRC, Warrington WA4 4AD, United Kingdom

In this paper a parallel SIMPLE based incompressible axisymmetric CFD code is considered. It is found that the grid aspect ratio has a strong influence on the convergence rate of the code on several processors relative to that on one processor. This phenomenon is analysed and is attributed to the strong influence of the cell aspect ratios to the structure of the pressure correction equation — a Poisson-like equation. This in turn affects the effectiveness of the parallel preconditioner for the conjugate gradient algorithm used in solving the pressure correction equation. The problem is further confirmed by studying the Poisson equation. A remedy is suggested and demonstrated to improve the convergence of the parallel CFD algorithm.

1. PROBLEMS WITH LARGE ASPECT RATIO

The code considered is one for calculating steady-state incompressible turbulent chemical reacting pipe flow. A cell centred finite volume approach is used to discretise the equation. The coupling between velocity and pressure is dealt with using the SIMPLE procedure ([1]). The discretised momentum equations are solved with line implicit scheme (using TDMA for the resulting tridiagonal systems). The pressure correction equations, being symmetric, are solved using the conjugate gradient algorithm with Incomplete LU (ILU) factorization as the preconditioner. A fixed number of iterations is used, usually 2 for the momentum equation and 10 for the pressure correction equation.

The code was parallelised using the usual domain decomposition strategy. The computational domain is split into a number of subdomains and each of the subdomains is allocated to one processor. A 'halo' region is included in the subdomains and after some calculations on each subdomain, communication takes place to update the data stored in the halo regions. The TDMA is parallelised by each processor sweeping over the subdomain once, then updating the halo data. The conjugate gradient algorithm is easily parallelised except the preconditioner, for which a block diagonal ILU preconditioner is used. The use of block diagonal preconditioner means that each processor calculates the preconditioner based only on the local data it has, thus the operation is fully parallel. However by doing so the parallel conjugate gradient algorithm is no longer mathematically equivalent to the sequential algorithm.

It is found that for some partitionings of the grid, the parallel CFD algorithm takes a lot more iterations to converge, compared to the code running on one processor. This happens frequently when the cell aspect ratios of the grid are large, and in particular, when the domain is partitioned along the radial direction (also called y-split later on) of a long and thin pipe. In such cases the gain in going parallel is eroded by the degradation of the convergence rate.

To give an extreme example, Table 1 shows the results of using the parallel code on a simple pipe flow of 1 m long with a radius of 2.5 cm. The pipe lies parallel to the x-axis. The flow comes in from one end of the pipe and out of the other end, having a density of 1.29 and viscosity of 0.0001. The inlet velocity is 63.8 m/s, and the flow is assumed to be laminar. The size of the mesh is 64×16, that is, the grid is divided into 62 equal sections in x-direction (plus two dummy cells) and 14 in y-direction (plus two dummy cells). The domain is then split in various ways and assigned on to processors. For example, processor configuration 2×1 stands for the partitioning (x-split) illustrated by Figure 1.

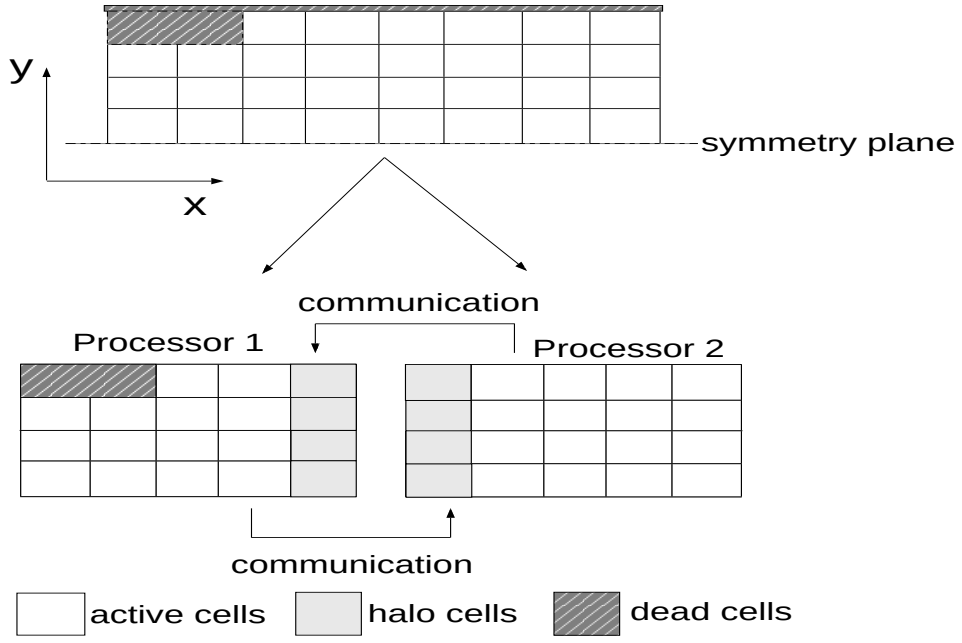

Figure 1

From the table, it is seen that if the domain is split along the axial direction (x-split), the number of iterations taken for the parallel algorithm to converge to a residual of 10^{-6} on several processors is almost the same as the number of iterations taken for the algorithm on one processor. However when the domain is split along the radial direction (y-split), or when box type partitioning is used, the number of iterations taken for the parallel algorithm increases considerably, in most case more than 3 times the number of iterations for the sequential case. This phenomenon is also observed for more

complex problems and for finer meshes, as well as when parallel TDMA is used instead of the parallel conjugate gradient algorithm to solve the pressure correction equations. Although for flow with turbulence, the deterioration in convergence is not as dramatic as that seen in Table 1.

Table 1
Number of iterations taken for the CFD code with different partitionings. Dimension of domain: $1m \times 2.5cm$.

Partitioning	1×1	2×1	4×1	8×1	1×2	2×2	4×2	1×4	2×4	4×4
Iteration Number	368	373	378	386	1200	1134	1162	5704	5614	5637

In the literature, little has been reported about this problem and its cause. Keyes ([2]) reported problems with y-split or box type split, and suggested that it was due to the physics of the particular flow problems studied. Many parallel CFD papers in the literature that worked on long and thin pipe-like geometries with high cell aspect ratios used the x-split only and, in doing so, avoided the problem associated with y-split. However, from the point of view of reducing the communication cost, box-type partitioning is usually more desirable than partitioning along one direction only. Using massively parallel machines, the number of processors may be more than the number of cells along any one direction. Thus, to utilise the parallel computers, a box-type split is unavoidable. Therefore it is very important to investigate the cause of this problem and to come up with some remedies.

2. ANALYSIS OF THE EFFECT OF CELL ASPECT RATIOS

In the SIMPLE procedure, it is necessary to solve a pressure correction equation. It is instructive to analyse the pressure correction equation, since it is a more difficult equation to solve than the momentum equation. The pressure correction equation at a cell P is of the form

$$c_P p'_P \;=\; c_E p'_E \;+ c_W p'_W \;+\; c_N p'_N \;+ c_S p'_S + d, \tag{1}$$

where c_E, c_W, c_N and c_S are coefficients that correspond to the east, west, north and south faces of the cell respectively, and

$$c_P \;=\; c_E \;+\; c_W \;+\; c_N \;+\; c_S.$$

Assuming the mesh is very fine and equi-spaced near the control volume to be considered , it is found that the pressure correction equation (1) is approximately equivalent to

$$\frac{\rho_P}{a_P}\{(\Delta y)^2(2p'_P - p'_E - p'_W) + (\Delta x)^2(2p'_P - p'_N - p'_S)\} = b, \tag{2}$$

with a_P a coefficient of the momentum equation and Δx and Δy the cell size along x and y directions respectively. Clearly (2) has the same form as a discretised Poisson equation.

If the cell aspect ratio $\alpha(=\Delta x/\Delta y)$ is large, then because of the effect of squares in (2),

$$\frac{c_E}{c_P} \approx \frac{c_W}{c_P} \approx \frac{1}{2(1+\alpha^2)} \approx 0,$$

$$\frac{c_N}{c_P} \approx \frac{c_S}{c_P} \approx \frac{\alpha^2}{2(1+\alpha^2)} \approx \frac{1}{2}.$$

As a result, the coupling between north and south is very strong while that between east and west is very weak. Assume the domain is x-split into two as in Figure 1. If a block diagonal preconditioner is used, the coupling between the east and west subdomains, that is, those coefficients c_E and c_W at the processor interface, is essentially ignored. This coupling is very weak anyway from previous analysis, thus its omission will have little adverse effect on the convergence rate. However if the y-split is used, the north-south coupling, which is very strong, is ignored when calculating the diagonally blocked preconditioner. As a result the preconditioner is not a very good preconditioner to the whole system. The usual fixed number of iterations of the conjugate gradient algorithm therefore may not solve the pressure correction equation to a satisfactory accuracy, and the whole CFD code converges slower. This is believed to be the main reason for the deterioration of the performance of parallel algorithms with y-split or box type splits experienced on a long and thin pipe. It is noted that the aspect ratio also effects the momentum equation, however the effect is relatively small compared with the pressure correction equation.

3. ANALYSIS USING POISSON EQUATION

Because the argument applies purely on the interface cells, it implies that if the cell aspect ratio near the interface is not large, the parallel code should not show as significant a deterioration in convergence rate. This is verified by numerical experiment.

Further tests with the use of parallel conjugate gradient algorithms on the solution of Poisson equation, and subsequent eigenvalue analysis, also confirms that it is the large aspect ratios of the interface cells between processors that degrade the performance of the algorithms.

In one such experiment, a Poisson equation on a rectangular domain $[0, XL] \times [0, YL]$ in the (x, y) space is considered. The length of the rectangle along x is fixed to $XL = 1$ and the length along y varies between $YL = 100$ to $YL = 0.001$. The domain is divided into 64×64 cells of equal size. The derivatives on the boundary are assumed to be known. The right hand side of the Poisson equation is set by assuming that the solution is $p(x, y) = x^2 + y^2 + xy$. The number of iterations taken for the residual to become less than 10^{-5} is reported in Table 2, together with the cell aspect ratios. Several partitionings are tested, including splitting the domain into two equal halves with the line $x = XL/2$ (x-split) or with the line $y = YL/2$ (y-split), and box type split with two lines $x = XL/2$, $y = YL/2$. The processor configuration corresponding

to these partitioning are 2×1, 1×2 and 2×2 respectively. As can be seen from the table, with the increase of the cell aspect ratio, y-split takes more and more iterations to converge, x-split generally takes less and less iterations, while the performance of box type split usually follows the worst of that of the former two types of partitionings.

Table 2
The influence of changing domain size to the iteration number of parallel conjugate gradient algorithm on the Poisson equation

XL	YL	Grid size	Aspect ratio	1× 1	1×2	2× 1	2× 2
1	100	64×64	0.01	67	67	226	226
1	10	64×64	0.1	73	73	135	135
1	1	64×64	1	58	86	86	71
1	0.1	64×64	10	63	120	64	120
1	0.01	64×64	100	64	217	64	217

Table 3
The iteration number when the grid is stretched near $y = 0.5$. The aspect ratio shown here is the cell aspect ratio near $y = 0.5$. The grid is uniform elsewhere.

XL	YL	Grid size	Aspect ratio	1× 1	1×2	2× 1	2× 2
1	1	64×64	0.033	64	64	62	65
1	1	64×64	0.1	81	82	92	90
1	1	64×64	1	58	86	86	71
1	1	64×64	10	73	104	86	111
1	1	64×64	30	72	139	86	147

In order to confirm that it is only the aspect ratios of the interface cells that affect the convergence of the parallel codes, in another test the Poisson equation with $XL = 1$, $YL = 1$ and with mesh size of 64×64 is solved, but instead of using a regular mesh, here the grid along x is still equi-spaced, but the grid lines next to the line $y = 0.5$ are shifted, so as to give required aspect ratio of the cells next to the line $y = 0.5$. The rest of the grid are regularly spaced. The results are listed in Table 3, where the aspect ratios shown refer to that of the cells next to the line $y = 0.5$. Clearly with the increase of the cell aspect ratios near the interface, even thought the aspect ratios of the cells that are not next to the interface are close to unity, y-split still takes more and more iterations compared to the sequential case. This confirms that it is the aspect ratios of the cells next to the interface which affect the convergence of the parallel conjugate algorithm relative to the sequential algorithm. The behavior of x-split and that of box type split is also very easy to explain with the same argument.

4. SOME SOLUTIONS

The cause of the performance degradation of the parallel conjugate gradient algorithm on grid with large aspect ratios readily give a possible remedy. Since it is the large

aspect ratios of the cells near the interface of the subdomains that affect the parallel code, it is proposed to coarsen the grid near the interface, thus reducing the cell aspect ratios, in order to improve the convergence.

The procedure is as follows. Denoting by a coarse grid a grid that is coarsened near the interface where the cell aspect ratio is large. Whenever a pressure correction equation is to be solved, the coarse grid equation is first formed and a few iterations of the conjugate gradient algorithm are used to get a good approximation of the solution. This is then interpolated to the original grid and a few CG iterations are applied again to smooth out the interpolation error. The total number of CG iterations will be kept the same as when coarsening is not used.

The coarse grid problem is formed as follows. Since the pressure correction equation behaves like a Poisson equation, for the purpose of calculating the coefficients of the pressure correction equation on the coarsened grid, it is assumed that the equation is of the form

$$c_1(x,y)\frac{\partial^2 p}{\partial x^2} \; + \; c_2(x,y)\frac{\partial^2 p}{\partial y^2} = a(x,y).$$

After discretisation, the equation is of the form (1) with

$$c_E = \frac{(c_1)_e \Delta y}{(\delta x)_e}, \tag{3a}$$

$$c_W = \frac{(c_1)_w \Delta y}{(\delta x)_w}, \tag{3b}$$

$$c_N = \frac{(c_2)_n \Delta x}{(\delta y)_n}, \tag{3c}$$

$$c_S = \frac{(c_2)_s \Delta x}{(\delta y)_s} \tag{3d}$$

Now if two cells A (at (i, j)) and B (at (i, j+1)) are lumped into one cell C, then according to (3), assuming

$$(c_1)_e^A (\Delta y)^A + (c_1)_e^B (\Delta y)^B = (c_1)_e^C (\Delta y)^C$$

and

$$(c_1)_w^A (\Delta y)^A + (c_1)_w^B (\Delta y)^B = (c_1)_w^C (\Delta y)^C,$$

the coefficients of the discretised equation on the coarse cell can be calculated using that on the fine cells. That is,

$$(c_E)^C = (c_E)^A + (c_E)^B,$$

$$(c_W)^C = (c_W)^A + (c_W)^B,$$

$$(c_N)^C = \frac{(c_N)^B (\delta y)_n^B}{(\delta y)_n^C},$$

$$(c_S)^C = \frac{(c_S)^A(\delta y)_s^A}{(\delta y)_s^C}.$$

If more than one cells are lumped to form a cell, the coefficients of the coarse cell can be calculated in a similar way.

The source terms on a coarse cell are calculated by lumping the source terms on the fine cells which form the coarse cell.

The idea is implemented and is found to improve the convergence of the parallel algorithms for the Poisson equations as well as the convergence of the parallel CFD code for pipe flow.

Table 4
Iteration number and CPU time of the CFD code with various processor configurations and with or without coarsening on problem 2

Processor configuration	Iterations	Total cpu time	Communication time
1×1	5740	3243	0
1× 2	5840	2067	309
1×2 coarsened	5420	1905	267
2×1	5680	1794	182
2×2	5840	1253	335
2×2 coarsened	5300	1167	320
1×4	5700	1547	519
1×4 coarsened	5480	1483	520
4×2	5820	988	404
4×2 coarsened	5240	920	382
2×4	13300	2448	1065
2×4 coarsened	5480	1026	457
8×1	5700	933	358
4×4	5720	899	470
4×4 coarsened	5460	871	445
8×2	6100	854	419
8×2 coarsened	5320	765	381
16×1	5560	804	396

To give an example, turbulent flow in a sudden expansion pipe is considered. The pipe is 1.0 m long and has a radius of 0.1 m. The inlet volecity is 64 m/s, density is 1.29 and viscosity 0.0001. A 64×32 equi-spaced grid is used. The results are listed in Table 4.

As can be seen from the table, partitioning along the y direction tends to increase the number of iterations needed for the algorithm to converge, although the adverse effect is less dramatic as in the case of the laminar flow problem. Coarsening produces some of the best iteration numbers and least CPU time.

Coarsening however has its limitation. If the cell aspect ratio is extremely large, then lumping a few cells together will not reduce the aspect ratio by very much. How

to implement the coarsening procedure on irregular grid in 3D also needs further investigation.

5. CONCLUSIONS

The problem associated with high aspect ratios when working on parallel incompressible CFD algorithms are analysed. The degradation in convergence comes only when the domain is decomposed along a direction where the cell aspect ratios are high, and is attributed to the strong coupling in the pressure correction equations between subdomains.

Coarsening as a remedy to the problem has been shown to be useful.

Not reported here, the use of block correction ([3-5]) was also tested and found to improve the convergence slightly. Overlapped block diagonal preconditioner was suggested ([6]) to improve the pure block diagonal preconditioner, this and similar ideas were also tried on the current problems but were found to make little improvement.

Currently the coarsening strategy is being compared with other techniques such as the use of Schur complement type preconditioners ([7-8]).

REFERENCES

1. S. V. Patankar, Numerical Heat Transfer and Fluid Flow, McGraw-Hill, 1980.
2. D. E. Keyes, Domain decomposition methods for the parallel computation of reacting flows, Computer Physics Communication, 53 (1989), 181-200.
3. A. Settari and K. Aziz, A generalization of the additive correction methods for the iterative solution of matrix equations, SIAM Journal of Numerical Analysis, 10 (1973), 506-521.
4. S. V. Patankar, A calculation procedure for two-dimensional elliptic situations, Numerical Heat Transfer, 4 (1981), 409-425.
5. B. R. Hutchinson, P. F. Galpin and G. D. Raithby, Application of additive correction multigrid to the coupled fluid flow equations, Numerical Heat Transfer, 13 (1988), 133-147.
6. G. Radicati and Y. Robert, Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor, Parallel Computing, 11 (1989), 223-239.
7. T. F. Chan and T. P. Mathew, The interface probing technique in domain decomposition, SIAM Journal of Matrix Analysis and Applications, 13 (1992), 212-238.
8. J. H. Bramble, J. E. Pasciak and A. H. Schaz, The construction of preconditioners for elliptic problems by substructuring, Mathematics of Computation, 47 (1986), 103-134.