D E L F T U N I V E R S I T Y O F T E C H N O L O G Y

REPORT 89-68

A line relaxation smoother for the multigrid solution of the Boussinesq equations

S. Zeng and P. Wesseling

ISSN 0922-5641

Reports of the Faculty of Technical Mathematics and Informatics no. 89-68

Delft 1989

CONTENTS

ABSTRACT

A multigrid algorithm using a smoother with line relaxation is developed to overco convergence problems resulting from large cell aspect ratios. The Boussinesq equations a solved by this algorithm for a problem of free convection in a rectangular domain. Resu concerning convergence histories and computational cost are presented. The algorit control structure of the smoother is given, allowing the implementation of different types smoothings.

1. INTRODUCTION

A good smoother is of primary importance to the performance of a multigrid method. For the solution of the incompressible Navier–Stokes equations in primitive variables, Vanka's smoothing method SCGS (Symmetrical Coupled Gauss–Seidel) seems to be attractive, since it shows some advantages over other smoothing methods (Arakawa et al., (1988)). This smoothing method is a coupled method, which means that the pressure and velocities are updated simultaneously. A staggered grid is used and the equations are discretized by the finite volume method. The variables related to a cell, which are the four velocities on the cell faces, and the pressure (and the temperature as well if the Boussinesq equations are used) at the cell center, are arranged in one block and the resulting local system is solved after a certain simplification. However, Vanka's smoothing method does not work if the cell aspect ratio differs appreciably from 1. This is due to the fact that Vanka's smoothing method is of the collective point Gauss–Seidel type. When the variables are strongly coupled in a certain direction, as in the case of small or large cell aspect ratio, it is well–known that these variables should be updated simultaneously. Therefore, the failure of Vanka's smoothing method in such cases is not surprising. This is shown by Arakawa et al. (1988) in an investigation of fluid flow over a backward–facing step. A possible improvement is to use line relaxation, which is the subject of this paper.

The line relaxation smoother used here can be considered as an extension of Vanka's smoother. The variables related to a row or column of grid cells are updated simultaneously. Variables with the same indices are arranged in one block. The resulting matrix for one line is block–tridiagonal and is solved numerically by the standard backward–forward substitution. On the coarsest grid, since the smoother with line relaxation was found not to be a good solver, a direct solution method is employed.

Section 2 presents the line relaxation smoother. The discrete equations are also given, for completeness. Section 3 deals with the control structure of the program code for the smoother, which is designed in a general way. In section 4, two test cases of free convection are investigated and convergence histories are given. Section 5 summarizes the above results.

2. THE LINE RELAXATION SMOOTHER

2.1 *The partial differential equations*

The partial differential equations to be solved are the nondimensionalized Boussinesq equations:

$$\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right], \tag{2.1}$$

$$\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right] + \frac{Gr}{Re^2}\theta, \tag{2.2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{2.3}$$

$$\frac{\partial u\theta}{\partial x} + \frac{\partial v\theta}{\partial y} = \frac{1}{RePr}\left[\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right]. \tag{2.4}$$

The Reynolds number Re, the Grashof number Gr and the Prandtl number Pr are defined as $Re = \frac{U_0 a}{\nu}$, $Gr = \frac{\beta g a^3 (T_{max} - T_{min})}{\nu^2}$ and $Pr = \frac{\nu}{\chi}$. Here, U_0 is the characteristic velocity, a the characteristic length, ν the viscosity, β the thermal expansion coefficient, g the acceleration of gravity, χ the heat conductivity and T_{max} and T_{min} the maximum and the minimum temperature on the boundary, respectively. In our case, the characteristic length a is taken to be the height of the rectangular domain, and $U_0 = \nu\sqrt{Gr}/a$, so that $Re = \sqrt{Gr}$, which makes the source term in (2.2) of order one.

2.2 *The discrete equations*

The partial differential equations (2.1) – (2.4) are discretized by using the hybrid scheme on a staggered grid. The discretization was described in detail in our previous work (Zeng

and Wesseling, (1989a,b,c) and so is not discussed here anymore. The following presents just the results. The discretized Boussinesq equations can be written in a general form as below:

$$F^{eq} \equiv A_c^{eq}\varphi_c - A_e^{eq}\varphi_e - A_w^{eq}\varphi_w - A_n^{eq}\varphi_n - A_s^{eq}\varphi_s - \pi^{eq} - S^{eq} - f^{eq} = 0, \tag{2.5}$$

except for the continuity equation. The superscript *eq* is replaced by u for the x–momentum equation, v for the y–momentum equation and T for the energy equation. In cell (i,j), the coefficients A, the variable φ, π and S are given by

$$\begin{aligned}
&A_c^u = A_e^u + A_w^u + A_n^u + A_s^u ,\\
&A_e^u = \max[|C_{x+}^u|, D_{x+}^u|] - C_{x+}^u ,\\
&A_w^u = \max[|C_{x-}^u|, D_{x-}^u|] + C_{x-}^u ,\\
&A_n^u = \max[|C_{y+}^u|, D_{y+}^u|] - C_{y+}^u ,\\
&A_s^u = \max[|C_{y-}^u|, D_{y-}^u|] + C_{y-}^u ,\\
&\varphi_c = u_{i,j},\ \varphi_e = u_{i+1,j},\ \varphi_w = u_{i-1,j},\ \varphi_n = u_{i,j+1},\ \varphi_s = u_{i,j-1},\\
&\pi^u = (p_{i,j} - p_{i+1,j})/\delta x,\\
&S^u = 0
\end{aligned} \tag{2.6}$$

for the x–momentum equation,

$$\begin{aligned}
&A_c^v = A_e^v + A_w^v + A_n^v + A_s^v ,\\
&A_e^v = \max[|C_{x+}^v|, D_{x+}^v|] - C_{x+}^v ,\\
&A_w^v = \max[|C_{x-}^v|, D_{x-}^v|] + C_{x-}^v ,\\
&A_n^v = \max[|C_{y+}^v|, D_{y+}^v|] - C_{y+}^v ,\\
&A_s^v = \max[|C_{y-}^v|, D_{y-}^v|] + C_{y-}^v ,\\
&\varphi_c = v_{i,j},\ \varphi_e = v_{i+1,j},\ \varphi_w = v_{i-1,j},\ \varphi_n = v_{i,j+1},\ \varphi_s = v_{i,j-1},\\
&\pi^v = (p_{i,j} - p_{i,j+1})/\delta y,\\
&S^v = \frac{Gr}{2Re^2}(\theta_{i,j} + \theta_{i,j+1})
\end{aligned} \tag{2.7}$$

for the y–momentum equation and

$$
\begin{aligned}
&A_c^T = A_e^T + A_w^T + A_n^T + A_s^T ,\\
&A_e^T = |C_{x+}^T|\max(S_{x+},0) - C_{x+}^T + D_{x+}^T,\\
&A_w^T = |C_{x-}^T|\max(S_{x-},0) + C_{x-}^T + D_{x-}^T,\\
&A_n^T = |C_{y+}^T|\max(S_{y+},0) - C_{y+}^T + D_{y+}^T,\\
&A_s^T = |C_{y-}^T|\max(S_{y-},0) + C_{y-}^T + D_{y-}^T,\\
&\varphi_c = \theta_{i,j},\ \varphi_e = \theta_{i+1,j},\ \varphi_w = \theta_{i-1,j},\ \varphi_n = \theta_{i,j+1},\ \varphi_s = \theta_{i,j-1},\\
&\pi^T = S^T = 0
\end{aligned}
\tag{2.8}
$$

for the energy equation. For the definition of the quantities $C_{x\pm}^u$, $C_{y\pm}^u$, $D_{x\pm}^u$, $D_{y\pm}^u$, $C_{x\pm}^v$, $C_{y\pm}^v$, $D_{x\pm}^v$, $D_{y\pm}^v$, $C_{x\pm}^T$, $C_{y\pm}^T$, $D_{x\pm}^T$, $D_{y\pm}^T$, $S_{x\pm}$ and $S_{y\pm}$, see Zeng and Wesseling (1989a). The discrete continuity equation reads

$$F^c \equiv \frac{1}{\delta x}(u_{i,j} - u_{i-1,j}) + \frac{1}{\delta x}(v_{i,j} - v_{i,j-1}) - f^c = 0 \tag{2.9}$$

The right hand side f of (2.5) and (2.9) is always zero on the finest grid, but could not be zero on coarser grids and is determined by restriction.

2.3 *The Jacobian for one line*

The line relaxation can be either of the x–line or the y–line type. For the x–line relaxation, the variables $\varphi_{i,j} = (u_{i,j},v_{i,j},p_{i,j},\theta_{i,j})^T$, i = 1, 2, $\cdots$, nx, j fixed, are updated simultaneously, and for the y–line relaxation, the variables $\varphi_{i,j}$, j = 1, 2, $\cdots$, ny, i fixed, are updated simultaneously.

a) The Jacobian for the x–line relaxation

At cell (i,j), the equations to be solved in x–line relaxation contain as unknowns only the variables of this cell and the two nearby cells (i–1,j) and (i+1,j). Therefore, the Jacobian is block–tridiagonal. In the equations for cell (i,j) at line j, the following unknowns occur:

$$
\begin{aligned}
&[(u_{i-1,j}),(u_{i,j},v_{i,j},p_{i,j}),(u_{i+1,j},v_{i+1,j},p_{i+1,j})],\\
&[(u_{i-1,j},v_{i-1,j}),(u_{i,j},v_{i,j},p_{i,j},\theta_{i,j}),(v_{i+1,j})],\\
&[(u_{i-1,j}),(u_{i,j},v_{i,j})],
\end{aligned}
$$

$$[(u_{i-1,j},\theta_{i-1,j}),(u_{i,j},v_{i,j},\theta_{i,j}),(\theta_{i+1,j})]$$

for the x–momentum equation, the y–momentum equation, the continuity equation and the energy equation, respectively. The derivatives of the equations with respect to the variables are given in Zeng and Wesseling (1989b).

b) The Jacobian for the y–line relaxation

This case is quite similar to case a). The unknowns for cell (i,j) at line i are

$$[(u_{i,j-1},v_{i,j-1}),(u_{i,j},v_{i,j},p_{i,j}),(u_{i,j+1})],$$
$$[(v_{i,j-1}),(u_{i,j},v_{i,j},p_{i,j},\theta_{i,j}),(u_{i,j+1},v_{i,j+1},p_{i,j+1},\theta_{i,j+1})],$$
$$[(v_{i,j-1}),(u_{i,j},v_{i,j})],$$
$$[(u_{i,j-1},\theta_{i,j-1}),(u_{i,j},v_{i,j},\theta_{i,j}),(\theta_{i,j+1})]$$

respectively for the momentum equations, the continuity equation and the energy equation. For the derivatives of these equations with respect to the variables, see also Zeng and Wesseling (1989b).

The resulting linear equation system can be written as

$$F'(\varphi_0)\varphi' = -F(\varphi_0), \tag{2.10}$$

where $\varphi = (u,v,p,\theta)^T$, φ_0 is the initial guess of the solution, F' the Jacobian of the discrete equations for the line considered and φ' the correction. Because the Jacobian is block–tridiagonal, it can be easily solved by the standard backward–forward substitution method. Underrelaxation is necessary and is performed by

$$\varphi^k := \varphi^k + \alpha_k \varphi'^k, \tag{2.11}$$

where

$$\alpha_k = \begin{bmatrix} \alpha_k^u & & & \\ & \alpha_k^v & & \\ & & \alpha_k^c & \\ & & & \alpha_k^T \end{bmatrix}. \tag{2.12}$$

Here, we take

$$\alpha_k^u = \alpha_k^v = \alpha_k^c = \alpha_k^T = \bar{\alpha}_k.$$

3. THE CONTROL STRUCTURE OF THE CODE

A general control structure for the smoother with line relaxation is developed which includes the x–line, the y–line, the alternating direction, the x–zebra, the y–zebra relaxation. The smoothing direction can be chosen forward or backward.

We define four basic types of smoothing:

1) x–forward. This smoothing updates the variables $\varphi_{i,j}$ along lines x=constant, choosing the lines successively in the positive x–direction.

2) x–backward. As 1), but in the negative x–direction.

3) y–forward. In this type of smoothing, $\varphi_{i,j}$ is updated along lines y=constant, choosing the lines successively in the positive y–direction.

4) y–backward. As 3), but in the negative y–direction.

By these four basic types of smoothing, the smoothing methods mentioned above can be constructed. For example,

type 1 with j–increment=1 with j_{start}=1: the forward x–line relaxation,

type 1 with j–increment= 1 with j_{start}=1
followed by
type 2 with j–increment=–1 with j_{start}=ny
} : the symmetrical x–line relaxation,

type 1 with j–increment=1 with j_{start}=1
followed by
type 2 with j–increment=1 with i_{start}=1
: the alternating direction relaxation,

type 1 with j–increment=2 with j_{start}=1
followed by
type 1 with j–increment=2 with j_{start}=2
: the x–line zebra relaxation,

···, etc.

A smoothing sweep will be defined by combinations of these basic types of smoothing. The control structure is given below.

```
      subroutine SMTHLN(φ^k,f^k,ns,rsd,eps,stype,zebra,ninsth)
      R_cvl = norm(F^k(φ^k))
      if(zebra) then                                  : zebra=.true.: zebra–
          nzbr = 1                                    line relaxation is used
          dblstep = 2
      else
          nzbr = 0
          dblstep = 1
      end if                                          : do ns smoothings
5     do 10 nsmth = 1, ns                             : each smoothing
          do 15 ninner = 1, ninsth                    consists of ninsth
              if(stype(ninner).eq.1) then             basic smoothings
                  lnstart = 1
                  lnend = ny(k)
                  lnstep = 1
              else if(stype(ninner).eq.2) then
                  lnstart = ny(k)
                  lnend = 1
                  lnstep = –1
              else if(stype(ninner).eq.3) then
                  lnstart = 1
                  lnend = nx(k)
                  lnstep = 1
```

```
                else
                    lnstart = nx(k)
                    lnend = 1
                    lnstep = −1
                end if
                do 20 nz = 0, nzbr
                    do 25 ln = lnstart+nz·lnstep, lnend, lnstep·dblstep
                        call eqsln(k,ln,stype(ninner),φ^k,A^k,B^k)
                                                    : form the Jacobian
                                                    for one line
                        call bltrsy(φ'^k,A^k,B^k)   : solve the
                                                    resulting tri−
                                                    diagonal equation
                                                    system
                        φ^k := φ^k + α_k φ'^k       : update the lnth row
                                                    (if stype = 1 or 2) or
                                                    column (if stype = 3
                                                    or 4)
25                  continue
20              continue
15          continue
            R_cv2 = norm(F^k(φ^k))
            rsd(k) = R_cv2
            if(A) then                              : A=.true.: adaptive
                if(R_cv2/R_cv1.gt.η) goto 35        multigrid cycle
                if(rsd(k).lt.eps(k)) goto 36
            end if
            R_cv2 = R_cv1
10      continue
35      if(A.and.k.eq.1.and.rsd(k).gt.eps(k)) goto 5
36      return
        end
```

Here we give some more explanations. One smoothing sweep consists of *ninsth* basic smoothings, the types of which are specified by an array *stype*. The subroutine *eqsln* used above should have the following form:

subroutine eqsln(k,ln,stype,φ^k,A^k,B^k)
if(stype.eq.1.or.stype.eq.2) then

$$\left.\begin{aligned} A^k &= F'^k(\varphi^k) \\ B^k &= -F^k(\varphi^k) \end{aligned}\right\} \text{ for the } ln\text{th row}$$

else

$$\left.\begin{aligned} A^k &= F'^k(\varphi^k) \\ B^k &= -F^k(\varphi^k) \end{aligned}\right\} \text{ for the } ln\text{th column}$$

end if
return
end

The parameter η is the smoothing rate tolerance, rsd(k) (k= 1,$\cdots$,l) is the dynamic residual norm on grid k and eps(k) (k= 1,$\cdots$,l) is the residual norm tolerance on grid k. For detailed information on the choice of η, rsd and eps, see Zeng and Wesseling (1989a).

4. TEST PROBLEMS AND RESULTS

4.1 *Description of the test problems*

The geometry of the test problems is shown schematically in figure 4.1.

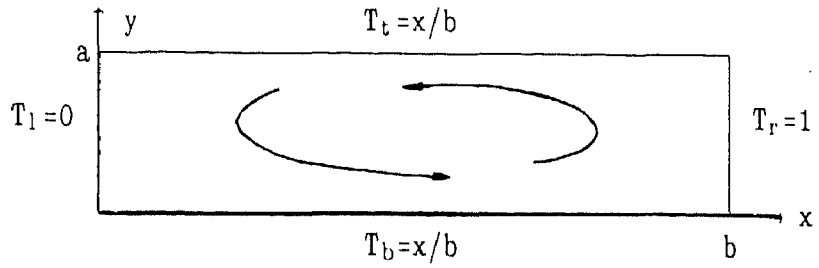

Figure 4.1 Geometry of the problems

The flow is driven thermally. the left and the right walls are kept at constant temperature;

the top and the bottom walls are perfectly conducting. The domain aspect ratio α = b/a will be taken to be 4 and 8. The Grashof number is $8 \cdot 10^4$ and the Prandtl number is 0.015.

4.2 *Multigrid algorithm specification*

The nonlinear multigrid algorithm is almost the same as used in Zeng and Wesseling (1989a) except that the smoother with point relaxation is replaced by the line relaxation smoother discussed in this paper, and the solver on the coarsest grid by a direct solver used in Zeng and Wesseling (1989b). So, we do not present the algorithm here. More specifically, the line smoother used is the one using symmetrical y–line relaxation. As a comparison, some computations are also carried out by using the Vanka's point relaxation method in the multigrid algorithm used before, but on the coarsest grid the direct solver is employed. The maximum of iterations to solve the equations on the coarsest grid nsc = 10. The multigrid iteration number nit = 10. The number of pre–smoothings and the number of post–smoothings are both taken to be 1. The number of multigrid levels l varies from 3 to 6. Because the adaptive A–cycle described in Zeng and Wesseling (1989a) usually has better convergence than the V– and W– cycles, we choose the A–cycle as the multigrid cycle. The residual norm tolerance $\delta = 0.2$ and the smoothing rate tolerance $\eta = 0.64$. As usual, the coarse grid correction parameter $s_k = 1$. Nested iteration is adopted and the starting solution on the coarsest grid is zero. The factor used in the control of nested iteration is taken to be 0.1. The computations are done on a HP–825/S computer and on an Alliant FX–4 computer for cases taking more storage. No effort has been made to optimize the code for parallelization or vectorization.

4.3 *Results*

Figure 4.2–4.4 give the convergence histories for the domain aspect ratio α=4 and figures 4.5–4.7 give the convergence histories for α=8. The coarsest grid has 4×8 cells, which

means that the cell aspect ratio is 8 for α=4 and 16 for α=8, and the number of multigrid levels l varies from 3–5, which means that the finest grids have dimensions 16×32, 32×64 and 64×128, respectively. It seems that for α=8 the multigrid algorithm works better than for α=4, since the multigrid convergence behaviour shown in figures 4.5–4.7 has fewer wiggles and the algorithm still works on gird 64×128 (l=5) with the underrelaxation factor $\overline{\alpha}_k$=0.9. This does not indicate that the algorithm works better with large cell aspect ratio, because with the same Grashof number the temperature gradient is smaller if the domain aspect ratio is large, and the problem therefore is less difficult. Figures4.8 and 4.9 illustrate the streamlines for α=4 and α=8, plotted on grid 64×128. Next, for α=4, we take the coarsest grid to be 2×8 and the multigrid level to be 4–6, which give the finest grids 16×64, 32×128 and 64×256. The cell aspect ratio now is 16. The convergence histories are given in figures 4.10–4.12. Compared with figures 4.2–4.4, figures 4.10–4.12 do not show either significant improvements or deteriorations.

To see whether the computational cost can be diminished by taking the coarsest grid as coarse as possible, cases with the coarsest grid 2×4 and the grid level l=6 are investigated. These cases have the same number of cells on the finest grid as with the coarsest grid 4×8 and l=5. Figures 4.13 and 4.14 give the convergence histories. As we can see, figure 4.13 looks almost the same as figure 4.4 and figure 4.14 exactly the same as figure 4.7. In table 4.1 the computational cost is given of all the computations done above, expressed in the the number of work units (WU) and the CPU time (seconds) per work unit. The cost of solving on the coarsest grid in terms of WU has been determined by means of CPU time measurements. Table 4.1 shows that the computational cost changes little between the cases with the coarsest grid 4×8 and l=5 and those with the coarsest grid 2×4 and l=6, although the cost of a coarse grid solution increases rapidly with the number of cells in the coarsest grid. This is because the "exact" solution on the 4×8 coarsest grid provides a better coarse grid correction to the next finer grid (4×16) than the 2×4 and 4×8 grids. But

if the coarsest grid is visited very often in an adaptive cycle, then this situation may be reversed.

To reveal a possibility that parallelization or vectorization can be employed, the smoothing procedure of zebra type has been tested, because this smoother allows better parallelization and vectorization. Figures 4.15–4.17 give the convergence histories. It is clear that this smoother works.

For all of the above cases, the point collective Gauss–Seidel smoother (Vanka's smoothing method) does not work. But it is surprising that the line smoother does not work better than the point smoother when the cell aspect ratio is one. This is shown in figures 4.18 and 4.19, which present the convergence histories of the multigrid algorithm using the point smoother and the line smoother with the coarsest grid 8×2 and the multigrid level l=6. The cost of one work unit for the point smoother is smaller than for the line smoother. The reason why the line smoother does not surpass the points smoother when the cell aspect ratio is one is not clear.

5. CONCLUSIONS

A multigrid algorithm using a smoothing with successive line relaxation is investigated for free convection problems. This algorithm is developed to deal with cases in which the cell aspect ratio is relatively large, so that algorithms using smoothers with point relaxation fail to give good performance. For the cases studied, with the cell aspect ratio 8 and 16, the multigrid algorithm shows a reasonably good convergence. It is observed that the computational cost is not diminished by using a smaller number of cells on the coarsest grid, because more computational cost is spent on finer grids. The multigrid algorithm works also with a zebra line smoother, providing a possibility of parallelization or

vectorization. With the cell aspect ratio 1, it is found that the algorithm using the line relaxation smoother is not so good as the algorithm using the point relaxation smoother of Vanka.

ACKNOWLEDGEMENT The authours are grateful to P. Wilders for providing a code for solving block–tridiagonal equation systems. The authours would like to thank M. Daalmeijer for her help in typing this report.

REFERENCES

1. Arakawa, Ch., A.O. Demuren, W. Rodi and B. Schönung, *Application of multigrid methods for the coupled and decoupled solution of the incompressible Navier–Stokes equations.* In: M. Deville (ed.), Proc. 7th GAMM Conf. on Numer. Methods in Fluid Mech., Louvain–la–Neuve, Sept. 9–11, 1987. Notes on Numerical Fluid Mechanics 20, 1–8, Vieweg, Braunschweig, 1988.

2. Zeng, S. and P. Wesseling, *A multigrid method for the Navier–Stokes and Boussinesq eqautions.* Report 89–17, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1989a.

3. Zeng, S and P. Wesseling, *A multigrid method with defect correction for free convection problems at high Rayleigh numbers.* Report 89–27, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1989b.

4. Zeng, S. and P. Wesseling, *Numerical solution of a bifurcation problem for the Boussinesq equations at low Prandtl numbers by a multigrid method.* To appear, Report of the Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1989c.

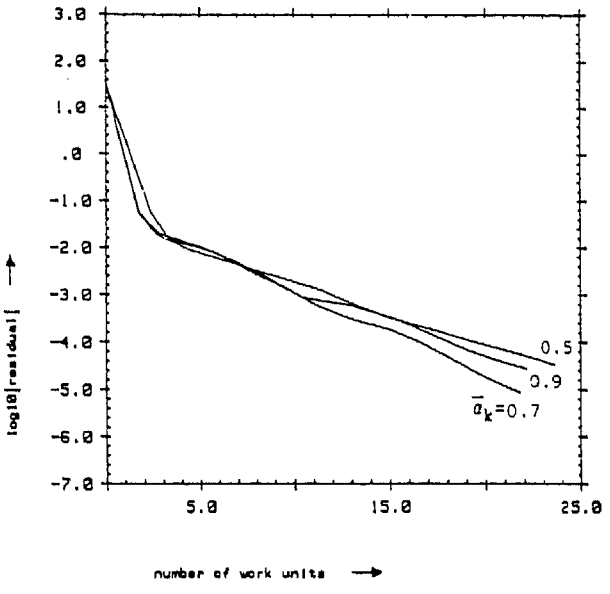

Figure 4.2 Convergence history, α=4, 4×8 cells on the coarsest grid, l=3

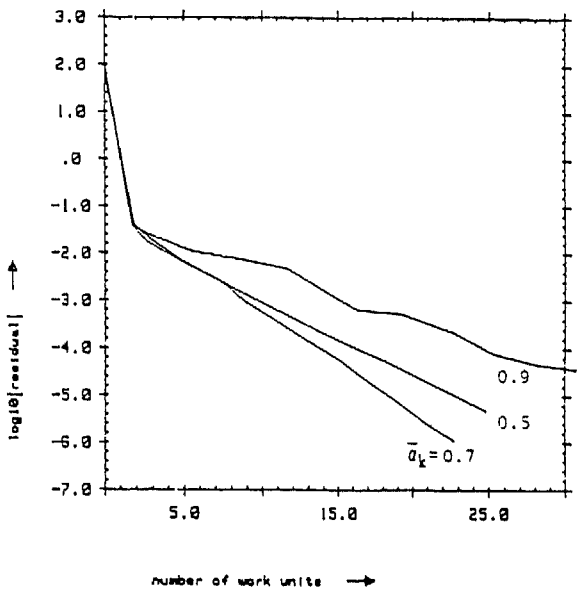

Figure 4.3 Convergence history, α=4, 4×8 cells on the coarsest grid, l=4

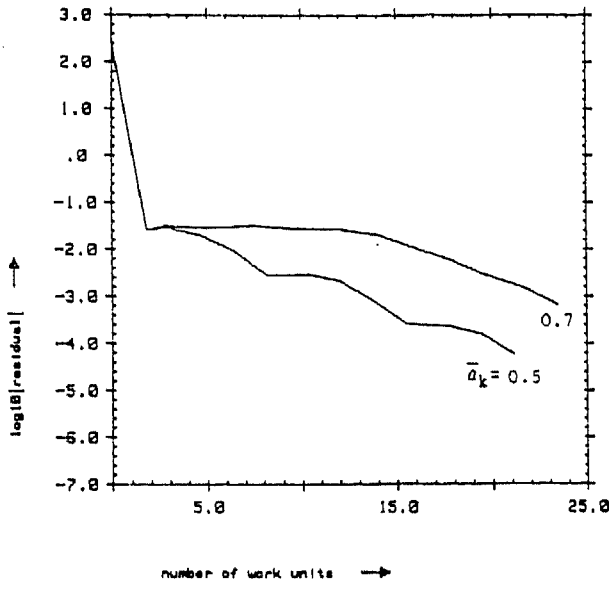

Figure 4.4 Convergence history, α=4, 4×8 cells on the coarsest grid, l=5

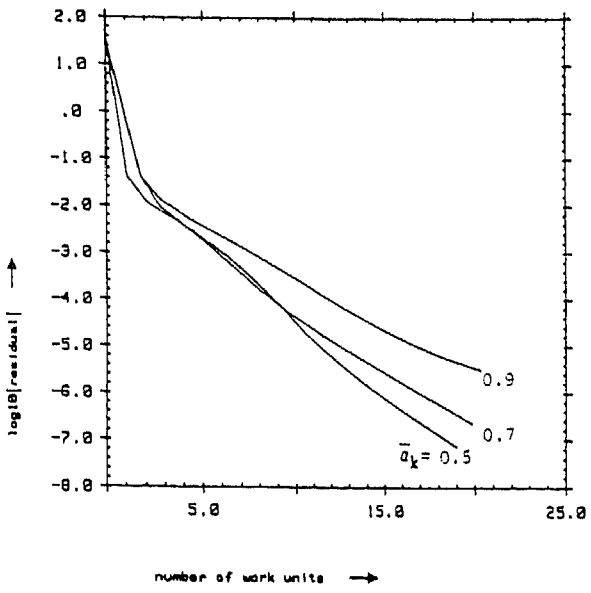

Figure 4.5 Convergence history, α=8, 4×8 cells on the coarsest grid, l=3

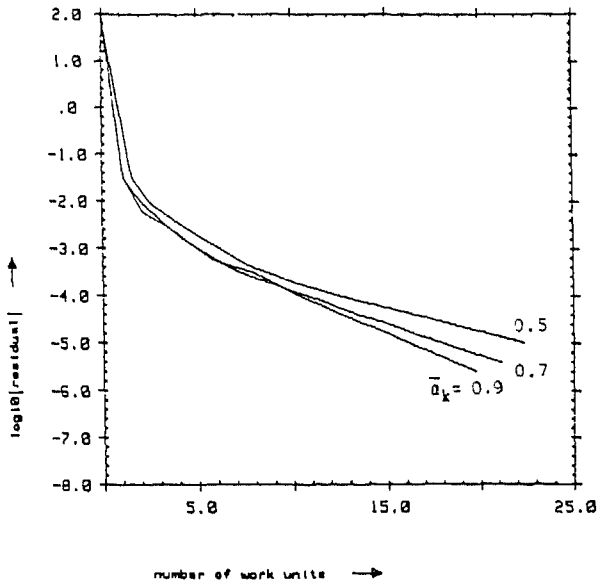

Figure 4.6 Convergence history, α=8, 4×8 cells on the coarsest grid, l=4

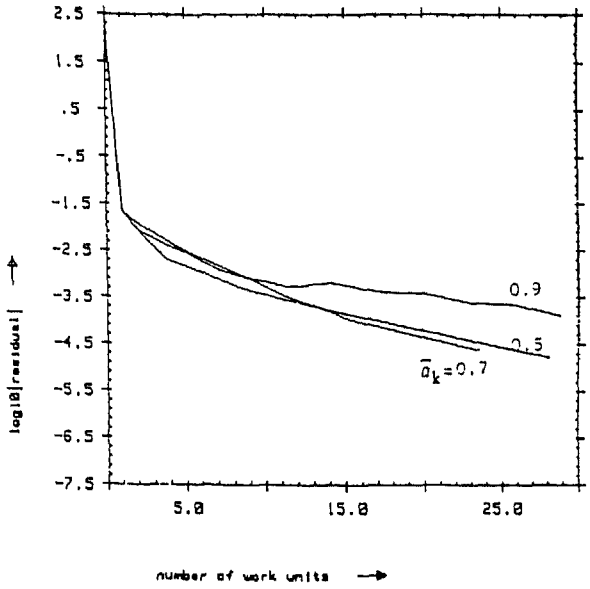

Figure 4.7 Convergence history, α=8, 4×8 cells on the coarsest grid, l=5

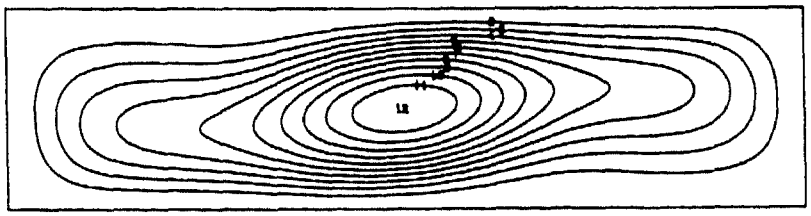

Figure 4.8 Streamlines, α=4, on grid 64×128

Figure 4.9 Streamlines, α=8, on grid 64×128

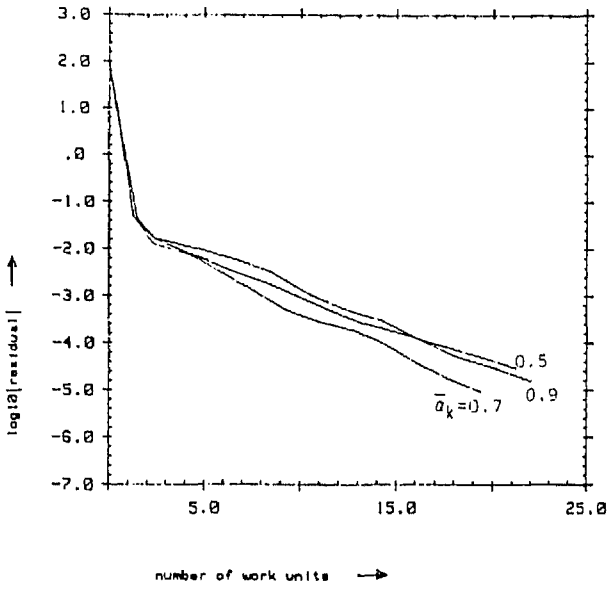

Figure 4.10 Convergence history, α=4, 2×8 cells on the coarsest grid, l=4

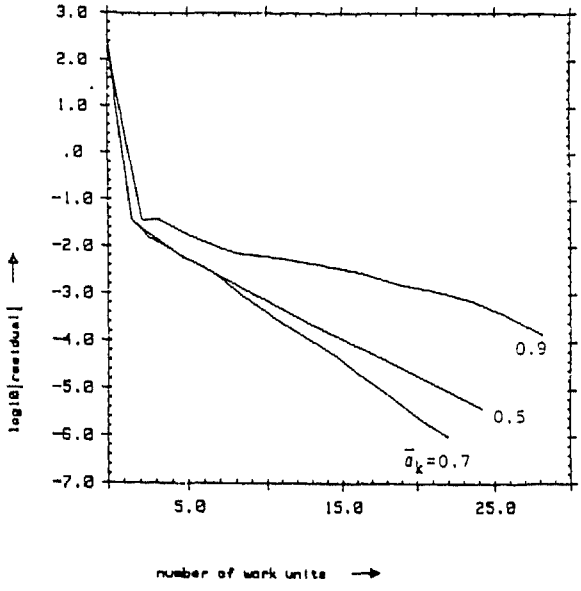

Figure 4.11 Convergence history, α=4, 2×8 cells on the coarsest grid, l=5

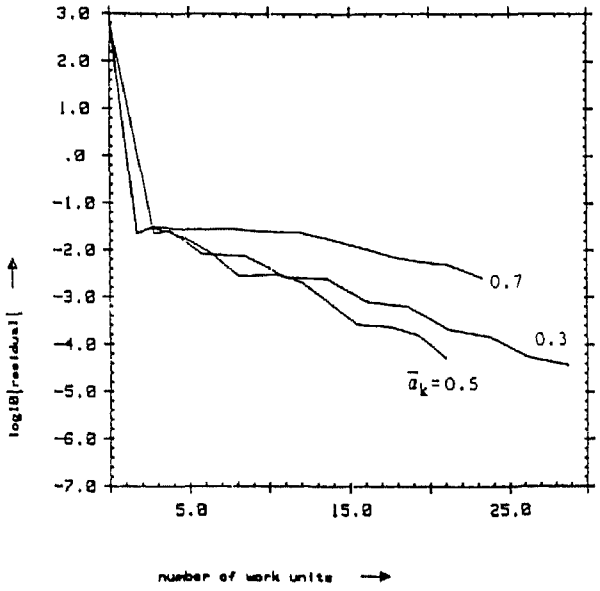

Figure 4.12 Convergence history, α=4, 2×8 cells on the coarsest grid, l=6

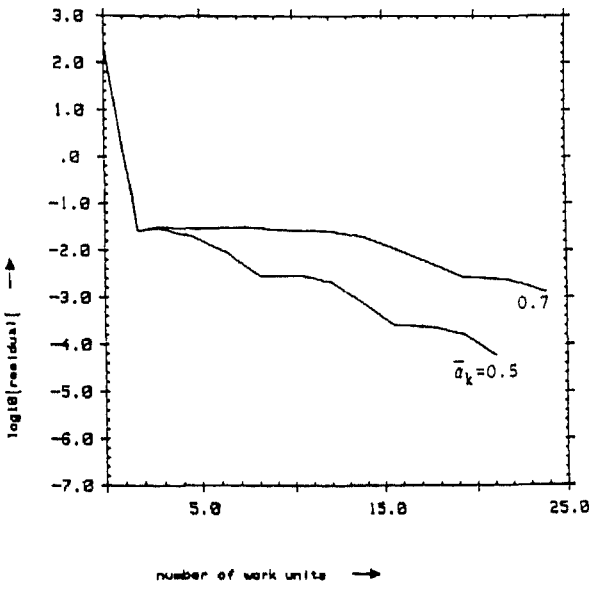

Figure 4.13 Convergence history, α=4, 2×4 cells on the coarsest grid, l=6

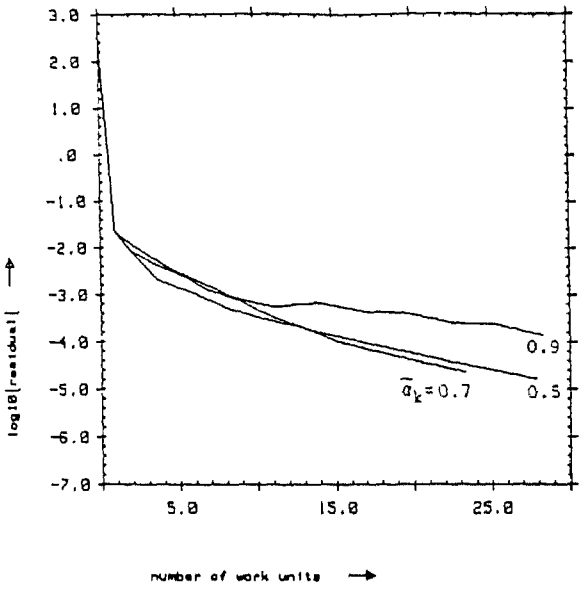

Figure 4.14 Convergence history, α=8, 2×4 cells on the coarsest grid, l=6

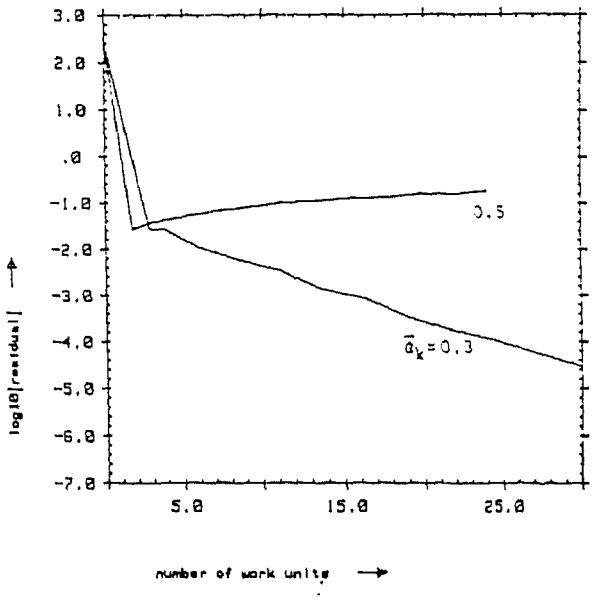

Figure 4.15 Convergence history, α=4, 2×4 cells on the coarsest grid, l=6, zebra line relaxation

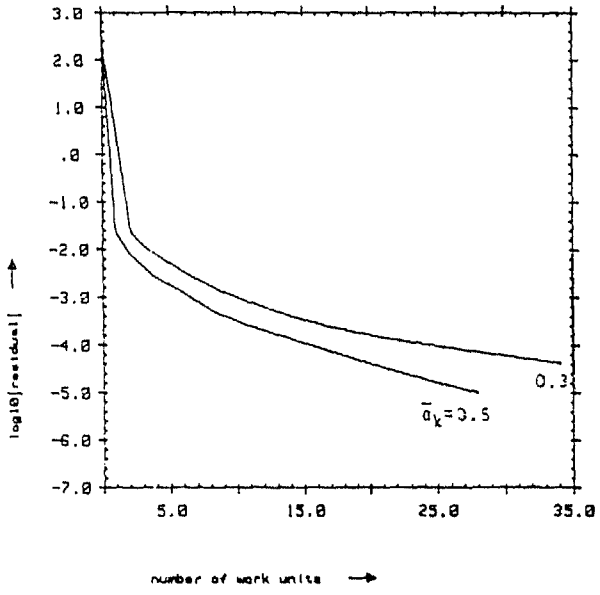

Figure 4.16 Convergence history, α=8, 2×4 cells on the coarsest grid, l=6, zebra line relaxation

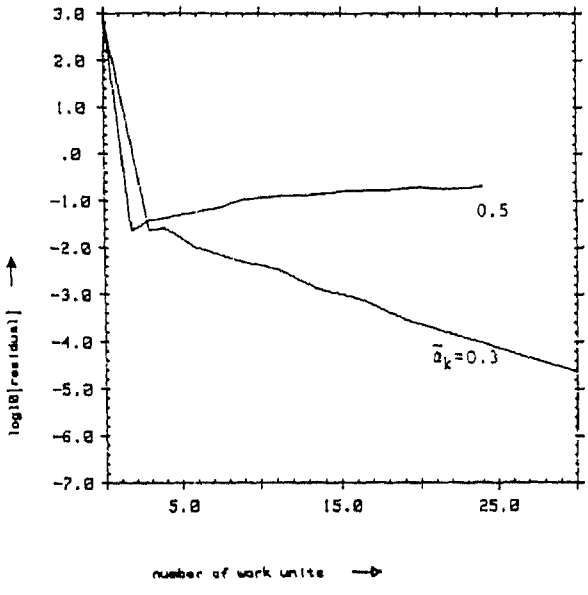

Figure 4.17 Convergence history, α=4, 2×8 cells on the coarsest grid, l=6, zebra line relaxation

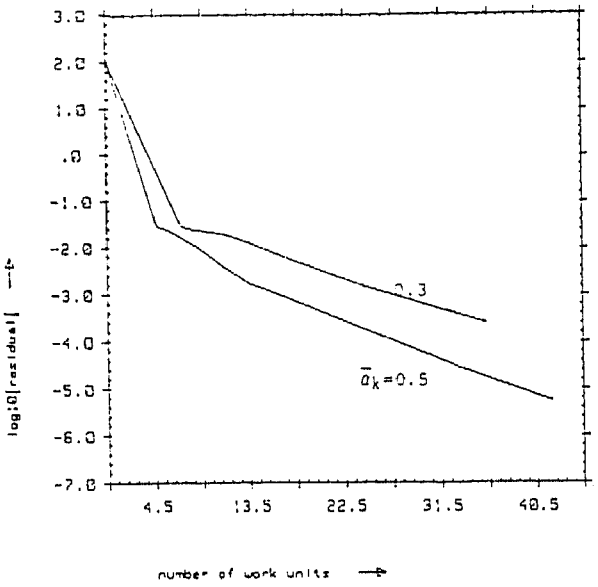

Figure 4.18 Convergence history, α=4, 8×2 cells on the coarsest grid, l=6, point smoother

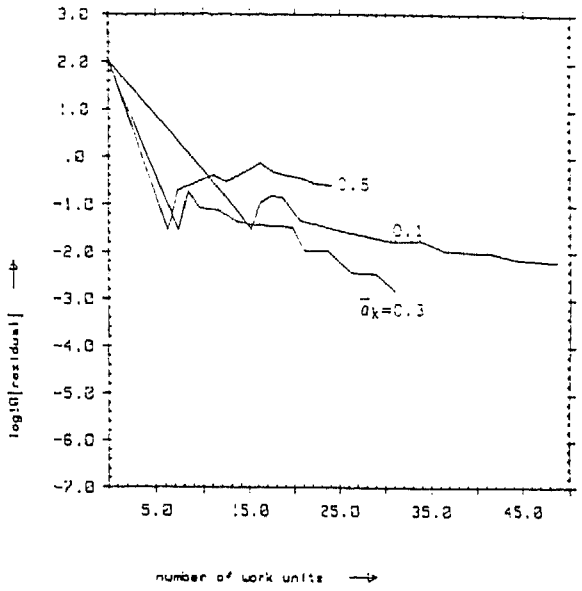

Figure 4.19 Convergence history, α=4, 8×2 cells on the coarsest grid, l=6, line smoother

Table 4.1 The computational cost expressed in the number of work units and the CPU time (seconds) per work unit

a. The domain aspect ratio a=8

$n_x \times n_y$ on l=1	l	$\bar{a}_k$	WU	CPUt/WU	smoothing method	computer
4×8	3	.5 .7 .9	23.6 21.8 21.2	15.2 15.1 15.2	symmetrical y-line	HP-825/S
	4	.5 .7 .9	24.8 22.7 30.7	62.2 63.1 61.9		
	5	.5 .7 .9	21.1 23.4 div*	179.0 178.9 /		Alliant FX-4
2×4	6	.5 .7	21.0 23.8	181.6 180.8		
		.3 .5	30.0 div	178.9 /	zebra y-line	
2×8	4	.5 .7 .9	21.2 19.4 22.0	31.5 31.7 32.3	symmetrical y-line	HP-825/S
	5	.5 .7 .9	24.2 21.9 28.1	128.7 127.7 127.5		
	6	.3 .5 .7	28.7 21.0 23.2	361.7 365.5 366.2		Alliant Fx-4
		.3 .5	29.9 div	362.7 /	zebra y-line	

* div means divergence

b. The domain aspect ratio a=8

$n_x \times n_y$ on l=1	l	$\bar{a}_k$	WU	CPUt/WU	smoothing method	computer
4×8	3	.5 .7 .9	20.3 19.9 19.0	15.4 15.6 15.4	symmetri-cal y-line	HP-825/S
	4	.5 .7 .9	22.3 21.2 19.8	62.8 63.0 63.7		
	5	.5 .7 .9	28.1 23.5 28.8	178.1 180.2 178.5		Alliant FX-4
2×4	6	.5 .7 .9	27.9 23.3 28.3	181.7 180.6 180.7		
		.3 .5	34.1 28.0	179.3 180.5	zebra y-line	

Table 4.2 Number of work units spent on each grid level for two different grid systems

a. The domain aspect ratio a=4

WU ($\bar{a}_k$=.5)	WU ($\bar{a}_k$=.7)	level		WU ($\bar{a}_k$=.5)	WU ($\bar{a}_k$=.7)
			$(2\frac{1}{4})$	0.049	0.040
0.511	0.596	$(2\frac{1}{8})$	2	0.289	0.246
0.875	0.906	2	3	0.953	0.922
2.438	2.938	3	4	2.500	3.125
6.250	8.000	4	5	6.250	8.500
11.00	11.00	5	6	11.00	11.00
21.07	23.44	Total		21.04	23.83

b. The domain aspect ratio a=8

WU ($\bar{a}_k$=.5)	WU ($\bar{a}_k$=.7)	WU ($\bar{a}_k$=.9)	level		WU ($\bar{a}_k$=.5)	WU ($\bar{a}_k$=.7)	WU ($\bar{a}_k$=.9)
				$(2\frac{1}{4})$	0.063	0.029	0.009
0.746	0.475	0.698	$(2\frac{1}{8})$	2	0.445	0.215	0.207
1.729	0.906	1.500	2	3	1.734	0.906	1.500
5.125	3.125	5.125	3	4	5.125	3.125	5.125
9.500	8.000	10.50	4	5	9.500	8.000	10.50
11.00	11.00	11.00	5	6	11.00	11.00	11.00
28.09	23.51	28.82	Total		27.87	23.27	28.34