INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 18, 733–746 (1994)

FINITE VOLUME METHODS FOR LAMINAR AND TURBULENT FLOWS USING A PENALTY FUNCTION APPROACH

J. SIMONEAU* AND A. POLLARD†

Department of Mechanical Engineering, Queen's University at Kingston, Kingston, Ont., K7L 3N6, Canada

SUMMARY

A penalty function, finite volume method is described for two-dimensional laminar and turbulent flows. Turbulence is modelled using the k–ε model. The governing equations are discretized and the resulting algebraic equations are solved using both sequential and coupled methods. The performance of these methods is gauged with reference to a tuned SIMPLE-C algorithm. Flows considered are a square cavity with a sliding top, a plane channel flow, a plane jet impingement and a plane channel with a sudden expansion. A sequential method is employed, which uses a variety of dicretization practices, but is found to be extremely slow to converge; a coupled method, evaluated using a variety of matrix solvers, converges rapidly but, relative to the sequential approach, requires larger memory.

KEY WORDS Penalty function Finite volume Direct Solver D'Yakonov Laminar Turbulent (k–ε)

1. INTRODUCTION

A major factor influencing the speed and convergence of finite volume discretization methods is the handling of the pressure–velocity (PV) coupling that links the momentum and mass conservation equations.[1,2] Research to improve this PV coupling has resulted in developments that claim to improve basic solution algorithms such as the SIMPLE method of Patankar and Spalding.[3] Examples include SIMPLER,[4] SIMPLE-C[5] and FIMOSE.[2] These methods can solve the discretized equations either iteratively or sequentially. Alternatively, coupled methods can be used whereby the momentum and continuity equations are solved simultaneously using direct solvers (see e.g. Reference 6).

The applicability of the penalty function method to the solution of the Navier–Stokes equations has been demonstrated using finite element methods;[7,8] however, information on the use of penalty functions with finite difference and finite volume methods is scarce. Temam[9] showed that a finite difference method using a penalty formulation yields correct solutions as the penalty parameter increases to inifinity and the grid spacing decreases to zero. Braaten[10] presents encouraging results using a penalty function, finite volume method with a coupled solution algorithm. De Bremaecker[11] presented results for a finite difference method using a penalty function and demonstrated that the method can converge for a cavity flow at least.

* Now at Alcan International Ltd., Kingston Research and Development Centre, Kingston, Ont., Canada.
† Author to whom correspondence should be addressed.

CCC 0271–2091/94/080733–14

Received 29 January 1992
Revised 18 October 1993

Similar to the penalty function method is the artificial compressibility method.[12] There is some controversy as to the strict equivalence of the two methods.[13,14] Examples of the use of this method can be found in Reference 15.

The present paper describes the incorporation of penalty functions (which addresses the pressure–velocity coupling) into a finite volume code. The resulting equation sets are solved using two approaches: (i) an iterative or sequential solution method and (ii) coupled methods that link directly, through a solver or matrix operator, the velocities obtained from the application of penalty functions. The resulting methods are applied to both laminar and turbulent flows. The results are compared with those obtained using a standard finite volume solution algorithm.

2. THE PENALTY FUNCTION METHOD

Two-dimensional, incompressible, steady and isothermal fluid flows without body forces are governed by a set of differential equations where conservation of mass is ensured by

$$\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} = 0 \tag{1}$$

and conservation of momentum is governed by

$$\frac{\partial U^2}{\partial X} + \frac{\partial UV}{\partial Y} = -\frac{\partial P}{\partial X} + \frac{1}{Re}\left(\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2}\right), \tag{2}$$

$$\frac{\partial UV}{\partial X} + \frac{\partial V^2}{\partial Y} = -\frac{\partial P}{\partial Y} + \frac{1}{Re}\left(\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right). \tag{3}$$

These equations are written here in their non-dimensional two-dimensional conservative form with

$$U = \frac{u}{u_0}, \qquad X = \frac{x}{l_0}, \qquad Re = \frac{\rho u_0 l_0}{\mu}, \qquad P = \frac{p}{\rho u_0^2},$$

where V and Y are defined similarly and u_0 and l_0 are reference values defined for a particular flow.

The penalty function method was first introduced in the field of constrained minimization for the problem of a membrane under equilibrium, where the functional to be minimized is replaced by a modified functional containing a so-called penalty term, which acts to enforce the constraint indirectly.[8] In fluid mechanics, when the penalty function method is applied to the functional minimization associated with the Stokes problem of a flow at low velocity, the continuity equation is replaced by a slightly 'penalized' continuity equation.[7] The two-dimensional form is

$$\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} = -\frac{1}{\lambda} P. \tag{4}$$

The non-dimensional penalty parameter λ is a large positive[14] arbitrary constant (typically of order 10^5) ensuring that equation (4) is almost identical to equation (1). The same expression for pressure, equation (4), can be used in conjunction with the Navier–Stokes equations even if, unlike the Stokes problem, no functional form for them is known. Equation (4) provides an explicit expression between pressure and velocity. It should be noted that equation (4) may

provide an oscillatory pressure field.[16] However, in all the calculations to be presented, the pressure fields obtained are exactly equal to those obtained from other methods, none of which displayed any 'wiggles'.

When the pressure terms in the standard Navier–Stokes equations (2) and (3) are replaced by equation (4), the 'penalized' Navier–Stokes equations are obtained. In non-dimensional, conservative, Cartesian form these are

$$\frac{\partial U^2}{\partial X} + \frac{\partial UV}{\partial Y} = \lambda \frac{\partial}{\partial X}\left(\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y}\right) + \frac{1}{Re}\left(\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2}\right), \tag{5}$$

$$\frac{\partial UV}{\partial X} + \frac{\partial V^2}{\partial Y} = \lambda \frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y}\right) + \frac{1}{Re}\left(\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right). \tag{6}$$

The number of differential equations required to simulate a two-dimensional fluid flow is therefore reduced by one with only U and V as unknowns. Once the velocity field is known, the pressure field can be calculated using equation (4).

3. FINITE VOLUME DISCRETIZATION

As can be seen in equations (5) and (6), penalized Navier–Stokes equations have the same convection and diffusion terms as equations (2) and (3), implying that 'standard' finite volume discretization methods can be applied without modification; attention only needs to be paid to the pressure terms.

To obtain their finite volume discretized form, equations (2) and (3) are integrated over a control volume. A staggered grid is used. The discretization of the integrated equation for the U-momentum gives

$$a_P U_P = a_E U_E + a_W U_W + a_N U_N + a_S U_S + b, \tag{7}$$

where the source term is

$$b = (P_p - P_e)\Delta Y. \tag{8}$$

The notation utilized is illustrated in Figure 1. The coefficients are calculated using the hybrid scheme of Spalding.[17] Of course, other schemes can be utilized. Equations of a similar nature can be obtained for other co-ordinate directions.

The same discretization approach can be applied to equations (5) and (6). With reference to Figure 1, the resulting discretized equation is of the same form as equation (7), except that the source term (8) now depends on velocities. It is

$$b = \lambda\left[\Delta Y\left(\frac{U_E - U_P}{\delta X_e} - \frac{U_P - U_W}{\delta X_w}\right) + V_E + V_S - V_P - V_{SE}\right]. \tag{9}$$

Because the U-velocity variables in equation (9) already appear elsewhere in equation (7), source term linearization with a redistribution of terms can result in a more implicit set of discretized equations. A total of five variants involving up to 21 velocity components have been derived and tested.[14] However, that given below was found to be generally the most efficacious. For the penalized U-momentum the discretized equation is

$$\tilde{a}_P U_P = \tilde{a}_E U_E + \tilde{a}_W U_W + \tilde{a}_N U_N + \tilde{a}_S U_S + \tilde{b}, \tag{10}$$

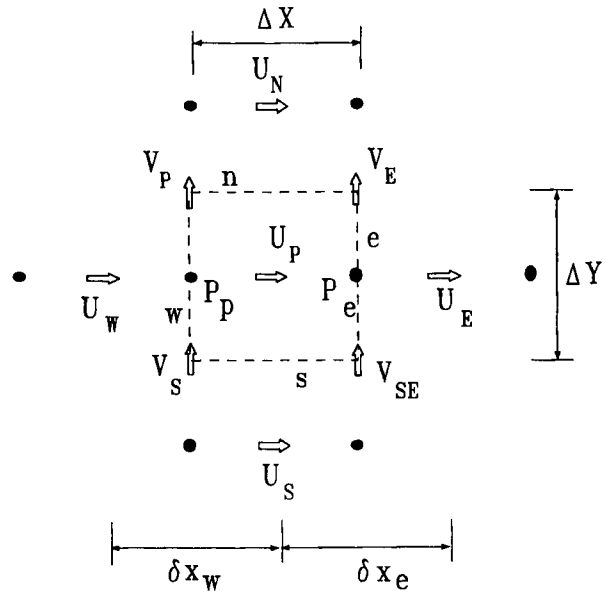

Figure 1. Finite volume typical U-velocity cell, notation and dimensions

with coefficients and source terms defined as

$$\tilde{a}_E = a_E + \lambda \frac{\Delta Y}{\delta X_e}, \qquad \tilde{a}_W = a_W + \lambda \frac{\Delta Y}{\delta X_w},$$

$$\tilde{a}_N = a_N, \qquad \tilde{a}_S = a_S,$$

$$\tilde{a}_P = \tilde{a}_E + \tilde{a}_W + \tilde{a}_N + \tilde{a}_S = a_P + \lambda \Delta Y \left(\frac{1}{\delta X_e} + \frac{1}{\delta X_w} \right),$$

$$\tilde{b} = \lambda (V_E + V_S - V_{SE} - V_P),$$

where those coefficients appearing without the tilde are from equation (7). An implicit formulation for the V-momentum can be written in a similar manner.

The iterative solution process for both equations is terminated when the normalized residual $R_\phi \leqslant 10^{-5}$ for both variables. This practice has been used extensively (see e.g. References 2 and 5). The velocity and pressure fields resulting from this choice of residual level have been found to accurately represent the flow situation under consideration.

4. SOLUTION ALGORITHMS

4.1. Penalty sequential algorithm (PENSA)

Equation (10) and its V-momentum counterpart can be solved at each grid point for U and V in sequence, one after the other. One example of this approach, called PENSA, is outlined below. It differs from more traditional algorithms in that no pressure correction equation is solved. The steps are as follows.

1. Calculate coefficients and source terms for the U-momentum at all grid points.
2. Solve this equation for the U-velocity components with an appropriate solver.
3. Calculate coefficients and source terms for the V-momentum equation at all grid points using new in-store values for U obtained in the previous step.
4. Solve this equation for the V-velocity components with an appropriate solver.
5. Using new V's from above, repeat all steps until a converged solution is obtained.
6. Calculate the pressure field using a discretized form of equation (4).

The solver used in this study is the well-known TDMA (tridiagonal matrix algorithm). This solver is adequate and no other more sophisticated solver is necessary (e.g. the strongly implicit methods of either Stone[18] or Schneider and Zedan[19]). No special treatment is required for the boundary conditions. PENSA requires less computer storage than the SIMPLE algorithm and its variants because neither pressure nor its correction are needed.

4.2. Testing PENSA

The performance of PENSA is briefly compared with that of a tuned SIMPLE-C algorithm in this section. The tuning of the SIMPLE-C algorithm involved searching for the minimum CPU times for a given grid, the optimum relaxation factors and the optimum overrelaxation parameter in the TDMA solver.[2] The problems considered are laminar flow in a square cavity with a sliding top and plane, laminar channel flow, both at $Re = 100$. Grid sizes of 10×10 and 15×8 were used for the cavity and channel flows respectively.

The results are given in Table I. As the penalty parameter is increased, the execution time and the number of iterations required to reach the $R_\phi \leqslant 10^{-5}$ limit for each equation also increase. Furthermore, the accuracy of the solution also increases with increasing λ, as indicated by the reference values for velocity and pressure listed in the table. As noted above, the accuracy is relative to the results obtained using the tuned SIMPLE-C algorithm. Indeed, for all penalty

Table I. Comparison of SIMPLE-C and PENSA for square cavity flow (10×10 grid) and channel flow (15×8 grid). U_{ref} and P_{ref} are reference velocity and pressure at grid points (i, j) 14, 2 (channel) and 5, 9 and 9, 9 for velocity and pressure respectively (cavity)[14]

	λ	No. of iterations	CPU time (s)	U_{ref}	P_{ref}
		Cavity flow			
SIMPLE-C	—	42	9·2	0·5534	0·2925
PENSA	1	153	13·2	0·5556	0·2845
	10	1559	103·2	0·5536	0·2918
	100	15521	999·1	0·5534	0.2924
	1000	155140	9917·4	0·5534	0·2925
		Channel flow			
SIMPLE-C	—	37	10·6	1·469	−1·013
PENSA	10	181	18·0	1·114	−0·532
	100	1594	131·1	1·406	−0·920
	1000	15589	1245·2	1·462	−1·003
	10000	155540	12479·0	1·468	−1·012

finite volume simulations presented, both the velocity and pressure fields were identical with the SIMPLE-C solution to four significant digits. Note again that the pressure fields displayed no oscillations as a result of applying equation (4).

The coupling between the momentum equations using PENSA is such that under relaxation did not aid convergence; however, slight overrelaxation improves the performance somewhat.[14] To assist in accelerating PENSA, a variable λ was introduced; however, this proved generally fruitless. This finding is not in accord with Bertin and Ozoe,[20] who applied a stepwise increase in λ in a finite element simulation of natural convection flows and found that convergence rates became enhanced with respect to those obtained using a fixed λ.

From Table I it is clear that as the penalty parameter increases, PENSA becomes exorbitantly expensive relative to the SIMPLE-C algorithm. The reason can be traced to the weak U–V coupling in PENSA.[14] PENSA in its present form is thus considered inappropriate for fluid flow calculations.

4.3. Penalty coupled algorithm (PENCA)

To ensure strong coupling between the penalized forms of the momentum equation, a direct solver can be used on the combined equations. The system of equations to be solved is

$$[A]\{u\} = \{b\}, \tag{11}$$

where $[A]$ is the matrix of coefficients and source terms for all equations, $\{u\}$ is the velocity vector containing both U and V-velocity variables and $\{b\}$ contains boundary conditions. This system can be replaced with advantage by

$$[A]\{\Delta u\} = \{b\} - [A]\{u^*\}, \tag{12}$$

where $\{u^*\}$ are velocities resulting from a previous iteration and $\{\Delta u\} = \{u\} - \{u^*\}$. The right-hand side of equation (12) vanishes when $\{u^*\}$ becomes the exact solution and the resulting $\{\Delta u\}$ will also vanish.

To solve either equation (11) or equation (12), various approaches were tested; however, when solving equation (12), the right-hand side quickly becomes small and the coefficients in [A] have values acceptably close to their final value regardless of the solver used. Since most direct solvers use an LU decomposition, there may be no need to continue factorization of $[A]$; after a few iterations it can remain fixed. The following system of equations results:

$$[A]^{\mathrm{F}}\{\Delta u\} = \{b\} - [A]\{u^*\}, \tag{13}$$

where $[A]^{\mathrm{F}}$ contains coefficients that remain fixed or frozen and which can be used for the remaining iterations. This was first proposed by D'Yakonov[21] and, when used subsequently in this paper, will be called D'Yakonov iteration. Equation (13) can be solved more rapidly than equation (12) and the accuracy of the solution is not affected, because convergence is still evaluated using updated coefficients that appear in $[A]$. For the results presented here $[A]^{\mathrm{F}}$ was fixed at the optimum stage in the iteration cycle for all flow test cases; this occurred after only a few iterations (i.e, from two to five for laminar flow and up to 22 for turbulent flow).

4.4. Selection of the solver for PENCA

In contrast with SIMPLE-C where a simple line solver is efficient, it was found that other approaches would be necessary for PENCA.[14] Three types of solvers are considered: iterative, direct and direct with iterative improvements. 'Iterative improvements' means that the solver

itself, once presented with equation (11), actually performs several inner iterations of a direct solver on the system given by equation (12).

Consider a domain of interest overlaid with control volumes. The order in which control volumes are visited can vary. Here two ways are presented. As shown in Figure 2(a) the numbering of the velocities can first proceed with all U-velocities followed by all V-velocities; alternatively, the velocities can be visited and numbered sequentially without regard to specific velocities, as shown in Figure 2(b). A sparse matrix system results if the practice illustrated in Figure 2(a) is used, whereas using the practice shown in Figure 2(b) gives rise to a banded matrix system. These matrix systems are shown in Figures 3(a) and 3(b) respectively, where P, N, etc. represent coefficients a_P, a_N, etc., λ is the penalty parameter and R is the right-hand side of the equation being solved.

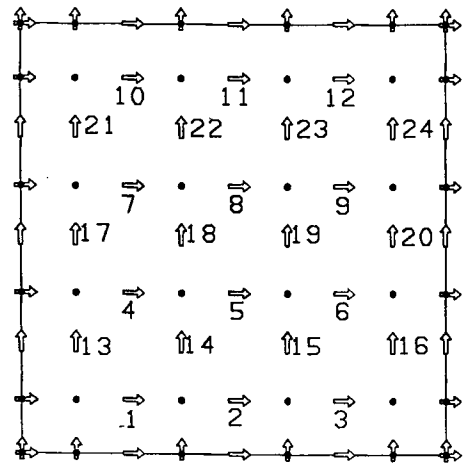

Figure 2(a). Velocity numbering for a sparse matrix

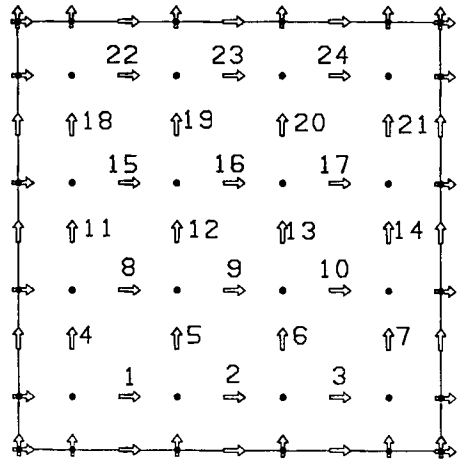

Figure 2(b). Velocity numbering for a banded matrix

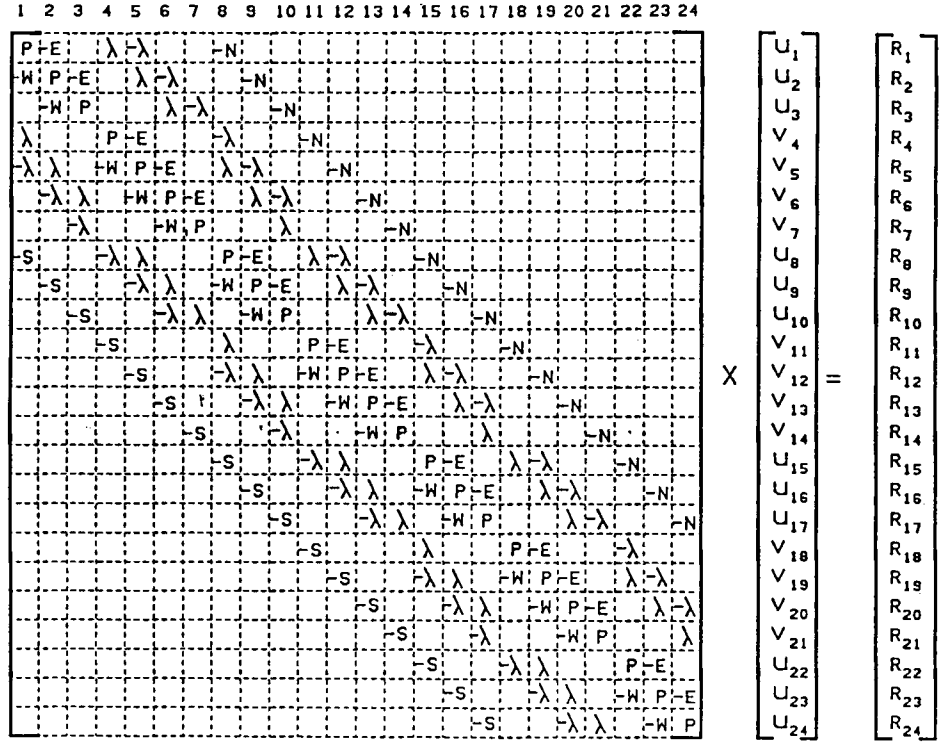

Figure 3(a). Sparse matrix system; note location of λ

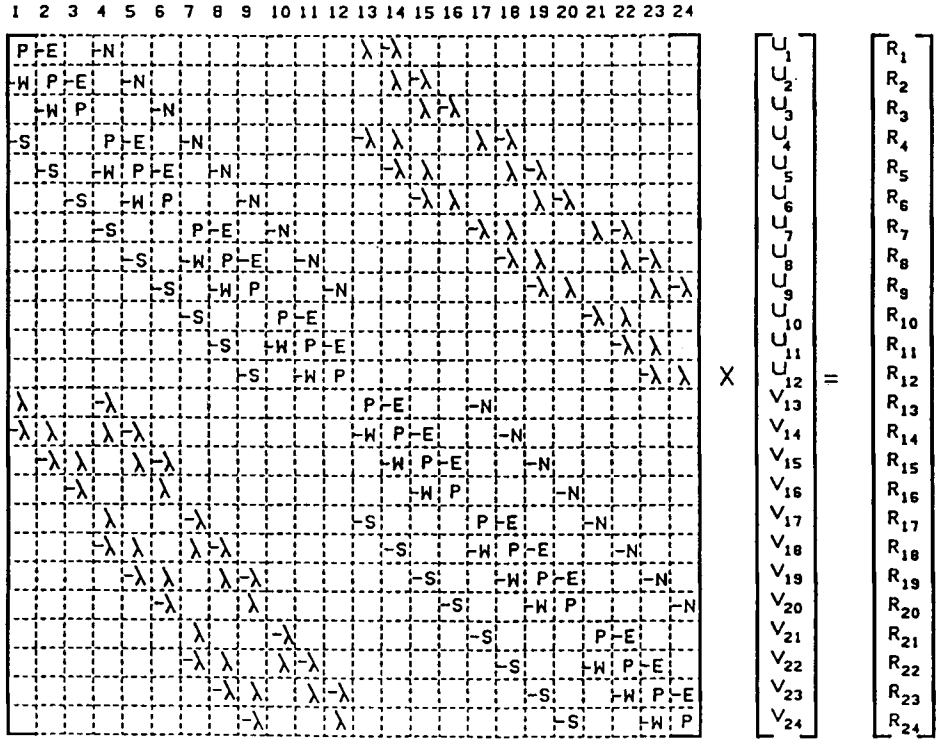

Figure 3(b). Banded matrix system; note location of λ

It is significant that the matrix structure permits the use of LU decomposition methods such as those found in the IMSL library, e.g. IMSL-LEQT2B and IMSL-LEQT2F (B, banded; F full), and of biconjugate gradient methods (ILUBCG2 and ILUCG2)[22], which include LU preconditioning. Note that the results from ILUCG2 are not presented here.[14]

Direct solvers considered include those from the Yale Sparse Matrix Package (YSMP),[23,24] which use an LU decomposition with compact storage but no pivoting. Compact storage is achieved by storing only the non-zero coefficients and using an efficient accounting system to store their location within $[A]$. In addition, a storage vector is reserved for the fill-in terms created during the decomposition process. This YSMP solver has been used in the past.[10,25] The general solution process of YSMP consists of optimal automatic reordering of $[A]$ to optimize the elimination process, symbolic decomposition to determine where the fill-in terms will occur during numerical decomposition, numerical decomposition whereby $[A]$ is decomposed into L–D–U matrices, and numerical solution consisting of forward and backward substitution. The first two steps need be done only once for a given system of equations. The efficiency of using automatic reordering was tested for both banded and sparse matrices and with and without D'Yakonov iterations. The results can be found in Tables II and III. It can be seen that automatic reordering applied to a banded matrix system using D'Yakonov iterations yields solutions in times much shorter than other methods.

The direct and semi-iterative solvers all terminate operation when the normalized residuals for the penalized momentum equations reach $R_\phi \leqslant 10^{-5}$. A sample of the relative performance of these solvers (YSMP, IMSL, conjugate gradient (CG),[26] etc.), including for comparison a line

Table II. Execution times (seconds), PENCA, using YSMP without D'Yakonov iterations, flow in a square cavity with a sliding top, $Re = 100$

	No reordering		With reordering	
Grid	Sparse $[A]$	Banded $[A]$	Sparse $[A]$	Banded $[A]$
10 × 10	37·9	7·7	5·9	5·7
12 × 12	141·6	17·0	11·9	11·5
14 × 14	419·1	34·4	23·3	20·4
18 × 18	—	115·3	57·8	71·0
22 × 22	—	283·5	152·9	136·7
30 × 30	—	—	498·1	434·1

Table III. Execution times (seconds), PENCA, using YSMP with D'Yakonov iterations, flow in a square cavity with a sliding top, $Re = 100$

	No reordering		With reordering	
Grid	Sparse $[A]$	Banded $[A]$	Sparse $[A]$	Banded $[A]$
10 × 10	17·6	4·5	3·9	3·8
12 × 12	61·2	8·9	7·3	6·8
14 × 14	174·6	17·1	12·6	11·6
18 × 18	—	48·8	27·6	32·6
22 × 22	—	115·1	67·1	61·2
30 × 30	—	—	206·1	181·2

Table IV. Execution times (seconds) using various solvers for simulating laminar flow in a square cavity with moving lid using PENCA

Grid	λ	IMSL-LEQT2F	IMSL-LEQT2B	YSMP	CG	ILUBCG2	Gauss–Seidel
6 × 6	10^1	2·4	—	—	4·3	—	10·9
	10^2	—	—	—	5·5	5·5	96·1
	10^3	2·4	—	—	7·4	6·2	946·2
	10^4	—	—	—	9·3	—	—
	10^5	2·4	11·3	—	4·4	9·4	—
10 × 10	10^1	—	107·3	3·5	208·8	—	155·7
	10^2	—	—	—	253·6	—	1525·9
	10^3	106·7	—	3·5	281·1	—	—
	10^4	—	—	—	275·7	—	—
	10^5	106·9	107·6	3·4	422·3	Diverges	—

iterative Gauss–Seidel solver, is given in Table IV. An examination of Table IV reveals that all entries were not attempted, because it became clear that some solvers were either overly expensive or no solution could be obtained. It should also be noted from Table IV that the CPU time required by some solvers was extremely sensitive to the value of λ as well as the grid density; although not shown, the only solver that displayed little sensitivity to the grid size was YSMP.[14] It is important to note that Table IV gives convergence histories that meet the convergence criterion $R_\phi \leqslant 10^{-5}$, that for $\lambda \geqslant 10^5$ the results become insensitive to the value of λ and that the results agree to within four significant digits with those obtained using SIMPLE-C.

The IMSL-based solvers were found to be reasonably robust but, relative to the YSMP direct solver, intolerably slow. The most efficient solver as a function of the magnitude of the penalty parameter λ can be readily identified as the YSMP solver. The conjugate gradient solvers performed poorly because of lack of diagonal dominance in the matrix $[A]$.[14]

From the data contained in Tables II–IV it is clear that reordering a banded matrix combined with D'Yakonov iterations is the most efficient combination for the YSMP solver.

5. LAMINAR AND TURBULENT FLOW TEST CASES

It is clear from Section 4.2 that PENSA is too CPU-bound. PENCA with non-symmetric YSMP and automatic ordering applied to a banded matrix is the fastest and is shown to be faster than a tuned SIMPLE-type algorithm (compare results for flow in a square cavity, 10 × 10 grid: from Table I SIMPLE-C requires 9·2 s, while from Table III PENCA requires only 3·8 s). Below, various laminar and turbulent flows are calculated to see whether this finding is general.

Four test cases are considered for both laminar and turbulent flows. The turbulence model is the high-Reynolds-number form of the k-ε model.[27] Wall functions are used to bridge the viscous near-wall region. PENCA ($\lambda = 10^5$ always) combined with the k-ε model is referred to as PENCAKE, whereby the equations for turbulence are solved sequentially using a line-by-line TDMA solver. The four cases, all of which use $Re \equiv \rho l_0 u_0/\mu$, are as follows:

(1) square lid-driven cavity flow, sides of dimension l_0, $Re = 100$ (laminar) and 10,000 (turbulent)
(2) developing flow in a plane channel with walls separated by distance l_0, $Re = 100$ (laminar) and 10,000 (turbulent), both with symmetry imposed at the channel centreline

(3) plane jet impingement upon a flat wall, $Re = 100$ (laminar) and 5000 (turbulent); the domain is $13l_0$ long and $2{\cdot}6l_0$ high and the jet thickness is l_0 with symmetry imposed along the jet centreline
(4) developing flow in a plane channel with a 2:1 sudden expansion, $Re = 100$ (laminar) and 5000 (turbulent); the domain is $8l_0$ long and $2l_0$ high with fluid entering through an entrance l_0 high and symmetry imposed along the axial centreline.

Three grid densities are used for each configuration, as listed in Table V. Inlet conditions are for uniform flow; outlet conditions are specified as zero axial gradient; wall conditions are either zero slip or wall functions. Turbulence inlet conditions are $k = 0{\cdot}005u_0^2$ and $\varepsilon = C_\mu k^{1{\cdot}5}/0{\cdot}03(l_o/2)$. Note that for the turbulence equations the underrelaxation was optimized for all cases. Tests indicate that mass flow rate corrections (see e.g. Reference 2) are beneficial in reducing the CPU time for all PENCA and SIMPLE-C.

Execution times for the four flows considered are given in Table VI. It is clear that PENCA and PENCAKE give solutions generally in less than half the time required by the tuned

Table V. Grid sizes used for laminar and turbulent flow investigations. Numbers indicate the grid nodes in the axial × cross-stream directions

Flow case	Coarse grid	Medium grid	Fine grid
Cavity	10 × 10	22 × 22	30 × 30
Channel	15 × 8	30 × 16	45 × 20
Impingement	8 × 15	17 × 28	22 × 41
Expansion	15 × 8	30 × 16	45 × 20

Table VI. Execution times (seconds) for optimally configured simulations. Penalty implies use of PENCA for laminar flows and PENCAKE for turbulent flows. Penalty parameter is 10^5 for all cases

Flow case	Laminar (L) or turbulent (T)	Penalty (P) or SIMPLE-C (S)	Coarse grid	Medium grid	Fine grid
Cavity	L	P	5·1	47·3	131·6
	L	S	9·4	93·6	249·8
	T	P	11·7	300·3	634·3
	T	S	50·0	501·6	1403·0
Channel	L	P	7·0	51·5	131·6
	L	S	10·6	106·0	382·4
	T	P	13·0	86·0	211·4
	T	S	16·5	151·4	608·6
Jet impingement	L	P	9·6	80·6	266·4
	L	S	39·2	276·7	651·7
	T	P	30·6	289·1	1112·0
	T	S	88·2	634·5	1246·6
Sudden expansion	L	P	7·9	58·9	144·2
	L	S	34·5	309·4	854·5
	T	P	30·9	191·5	546·0
	T	S	68·3	531·3	1620·9

SIMPLE-C algorithm. Furthermore, as the grid is refined, it is apparent that PENCA and PENCAKE give better relative performance with increasing grid densities. The relatively high CPU times required to solve the jet impingement problem using PENCAKE are due to the large number of iterations required prior to invoking D'Yakonov iterations. The execution times are of course dependent upon the number of equations to be solved and the time required to establish virtual storage within the YSMP solver.

The data from Table VI do not indicate the rate at which the solutions can be obtained. An example of the residual reduction for laminar flow in a square cavity, coarse grid (see Table V) is given in Figure 4, where results for PENSA, PENCA and SIMPLE-C are displayed. It is quite evident that PENCA achieves almost one order of magnitude reduction in the residual for every iteration while other algorithms require at least 10 iterations. An interesting comparison is the effect of the stringency of the convergence criterion on the performance of the algorithms. For the cases considered so far the residual was $R_\phi \leqslant 10^{-5}$, where ϕ refers to the equation being solved. If $R_\phi \leqslant 10^{-8}$ is used on medium-sized grids (see Table V) to increase the accuracy to more than four significant digits, the percentage increase in execution time for SIMPLE-C is, on average for all four cases, more than twice that for PENCA and PENCAKE.[14] Note that again the velocity and pressure fields obtained are exactly the same regardless of the algorithm used.

It has been found[14] that in these tests with a YSMP solver the magnitude of λ has little effect on the number of iterations required to obtain solutions; moreover, the CPU time is also essentially independent of λ. The storage requirements for PENCAKE and SIMPLE-C are given in Table VII, from which it is clear that the storage requirements depend almost linearly on the grid size for all methods, but for the penalty function formulations the storage overhead is on average about five times that of a SIMPLE-type algorithm. Also given in the table is the storage required for the fill-in terms of the YSMP solver. That is, YSMP requires storage for the non-zero elements in $[A]$. It also requires a storage vector to handle the factorization.

Other tests have been performed to evaluate PENCAKE.[14] These include effects of grid aspect ratios, convergence criterion, bandwidth of $[A]$, optimum number of factorizations of $[A]$ prior to invoking D'Yakonov iterations, optimum underrelaxation parameters in SIMPLE-C, costs

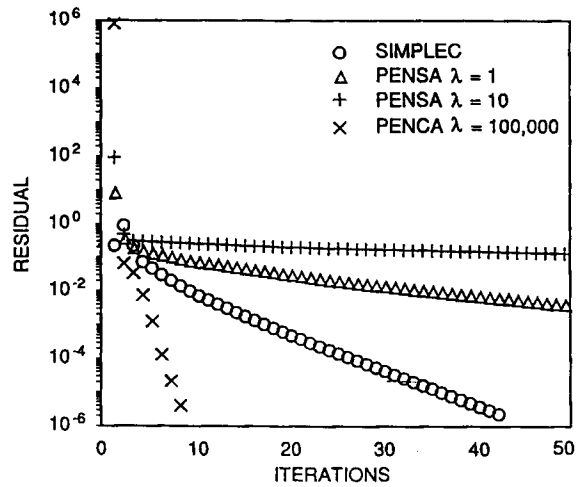

Figure 4. Residual reduction for coarse grid, laminar flow in a square cavity as a function of iteration count

Table VII. Total and fill-in storage (kilowords) required for various grid densities. Note that grid density groupings have products similar in magnitude

	PENCA or PENCAKE		SIMPLE-C	
Grid	Storage	Fill-in	Storage	Fill-in
10 × 10	20	9·7	9	0
15 × 8	22	11·4	11	0
8 × 15	23	11·8	11	0
17 × 28	163	116·3	41	0
30 × 16	162	116·0	41	0
22 × 22	177	129·1	42	0
45 × 20	383	273·2	76	0
30 × 30	390	317·6	76	0
22 × 41	388	303·0	76	0

including both execution times and storage costs using charging algorithms from different Canadian universities, etc.

6. CONCLUSIONS AND RECOMMENDATIONS

The work presented here compares the execution time and storage requirements of the well-known SIMPLE-C algorithm with those of the new algorithms PENSA and PENCA, all of which have been applied to both laminar and turbulent flows. The following conclusions can be drawn.

1. A penalty-function, finite volume method combined with a suitable solver can be used to simulate both laminar and turbulent fluid flows with similar accuracy to that obtained with a conventional finite volume method.
2. A sequential penalty function formulation (PENSA) is unacceptably slow for fluid flow calculations.
3. Gauss–Seidel, conjugate gradient, preconditioned conjugate gradient and IMSL direct solvers are too slow and/or require excessive computer storage when applied to the coupled system of penalized form of the Navier-Stokes equations.
4. YSMP with automatic reordering of a banded matrix is adequate for the efficient solution of the flow with the penalty coupled algorithm PENCA.
5. PENCA and PENCAKE are faster than a tuned SIMPLE-C algorithm.
6. The SIMPLE-C algorithm requires less storage than PENCA or PENCAKE.

It is recommended that PENSA not be abandoned: ways need to be found to increase the U–V coupling. A custom solver that uses to advantage the sparse and banded nature of the coefficient matrix needs to be devised. Extensions of the method to three-dimensional flows, flows with buoyancy and the inclusion of higher-order difference schemes should be given priority to further evaluate this method.

ACKNOWLEDGEMENTS

The authors would like to thank the Royal Military College (RMC) of Canada for their support while J.S. was a faculty member. Support for this research was also provided by the Natural Science and Engineering Research Council of Canada through grants to A.P. All the computations were performed on the RMC Honeywell DPS-8/70, running FORTRAN 77, release D00, with automatic double precision.

REFERENCES

1. G. D. Raithby and G. E. Schneider, 'Numerical solution of problems in incompressible fluid flow: treatment of the velocity–pressure coupling', *Numer. Heat Transfer*, **2**, 417–440 (1979).
2. B. R. Latimer and A. Pollard, 'Comparison of pressure–velocity coupling solution algorithms', *Numer. Heat Transfer*, **8**, 635–652 (1985).
3. S. V. Patankar and D. B. Spalding, 'A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows', *Int. J. Heat Mass Transfer*, **15**, 1787–1806 (1972).
4. S. V. Patankar, 'A calculation procedure for two-dimensional elliptic situations', *Numer. Heat Transfer*, **4**, 409–425 (1981).
5. J. P. Van Doormaal and G. D. Raithby, 'Enhancements of the SIMPLE method for predicting incompressible fluid flows', *Numer. Heat Transfer*, **7**, 147–163 (1984).
6. S. P. Vanka, 'Block-implicit calculation of steady turbulent recirculating flows', *Int. J. Heat Mass Transfer*, **28**, 2093–2103 (1985).
7. J. N. Reddy, 'On penalty function methods in the finite element analysis of flow problems', *Int. j. numer. methods fluids*, **2**, 151–171 (1985).
8. J. N. Reddy, 'Penalty-finite-element analysis of 3-D Navier–Stokes equations', *Comput. Methods Appl. Mech. Eng.*, **35**, 87–106 (1985).
9. R. Temam, 'Une methode d'approximation de la solution des equations de Navier–Stokes', *Bull. Soc. Math. Fr.*, **96**, 115–152 (1968).
10. M. E. Braaten, 'Development and evaluation of iterative and direct methods for the solution of the equations governing recirculating flows', *Ph.D Thesis*, University of Minnesota, 1985.
11. J. C. De Bremaecker, 'Penalty solution of the Navier–Stokes equations', *Comput. Fluids*, **15**, 275–280 (1987).
12. A. J. Chorin, 'A numerical method for solving incompressible viscous flow problems', *J. Comput. Phys.*, **2**, 12–26 (1967).
13. T. M. Shih, C. H. Tan and B. C. Hwang, 'Equivalence of artificial compressibility method and penalty-function method', *Numer. Heat Transfer*, **15**, 127–130 (1989).
14. J. Simoneau, 'The numerical calculation of laminar and turbulent flows using penalty-function finite-volume methods', *Ph.D. Thesis*, Department of Mechanical Engineering, Queen's University at Kingston, 1990.
15. S. E. Rogers and D. Kwak, 'Upwind differencing scheme for time-accurate incompressible Navier–Stokes equations', *AIAA J.*, **28**, 253–262 (1990).
16. T. J. R. Hughes, W. K. Liu and A. Brooks, 'Finite element analysis of incompressible viscous flows by penalty function formulation', *J. Comput. Phys.*, **30**, 1–60 (1979).
17. D. B. Spalding, 'A novel finite difference formulation for differential expressions involving both first and second derivatives', *Int. j. numer. methods eng.*, **4**, 551–559 (1972).
18. H. L. Stone, 'Iterative solution of implicit approximations of multidimensional partial equations', *SIAM J. Numer. Anal.*, **5**, 530–558 (1968).
19. G. E. Schneider and M. Zedan, 'A modified strongly implicit procedure for the numerical solution of field problems', *Numer. Heat Transfer*, **4**, 1–19 (1981).
20. H. Bertin and H. Ozoe, 'Technique for rapid convergence of the penalty finite-element method with a modified Galerkin scheme and its application to natural convection', *Numer. Heat Transfer*, **10**, 311–325 (1986).
21. E. G. D'Yakonov, 'An iterative method for solving systems of finite difference equations', *Dokl. Akad. Nauk SSSR*, **138**, 522–525 (1961).
22. A. E. Koniges and D. V. Anderson, 'ILUBCG2: a preconditioned biconjugate gradient routine for the solution of linear asymmetric matrix equations arising from 9-point discretisation', *Comput. Phys. Commun.*, **43**, 297–302 (1987).
23. S. C. Eisenstat, M. C. Gursky, M. H. Schultz and A. H. Sherman, 'Yale Sparse Matrix Package I—The symmetric codes', *Rep. 112*, Department of Computer Science, Yale University, 1977.
24. S. C. Eisenstat, M. C. Gursky, M. H. Schultz and A. H. Sherman, 'Yale Sparse Matrix Package II—The non-symmetric codes', *Rep. 114*, Department of Computer Science, Yale University, 1977.
25. S. P. Vanka and G. K. Leaf, 'Fully-coupled solution of pressure-linked fluid flow equations', *Rep. ANL-83-73*, Argonne National Laboratory, University of Chicago, 1983.
26. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes—The Art of Scientific Computing*, Cambridge University Press, Cambridge 1986.
27. B. E. Launder and D. B. Spalding, 'The numerical computation of turbulent flows', *Comput. Methods Appl. Mech. Eng.*, **3**, 269–289 (1974).