INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2006; **50**:795–816
Published online 31 October 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1066

Optimization of the nonlinear convergence rate of the SIMPLER Picard algorithm for steady incompressible internal flows through enforcement of continuity projections

S. Kumar[‡] and P. J. Smith[*,†]

Department of Chemical Engineering, University of Utah, Salt Lake City, UT, U.S.A.

SUMMARY

We discuss the performance optimization of the Semi-Implicit Method for Pressure-Linked Equations—Revised (SIMPLER) Picard algorithm for steady incompressible internal flows. We discuss the nonlinear convergence of the Picard iteration as a function of the pressure and scalar potential continuity projections stemming from the SIMPLER algorithm, for three example problems. In particular, we discuss the choice of under-relaxation method, and choice of under-relaxation factors; the choice of the projection algorithm; and the required tolerance for the linear solve of the generalized Poisson equations for the pressure and scalar potential equations that arise from the projection operations. We conclude that the convergence of the nonlinear Picard iteration can be effectively controlled by the optimal enforcement of the continuity projections. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: projection methods; SIMPLER; Navier–Stokes equations; solvers; Picard iterations; convergence rate

1. INTRODUCTION

Computational fluid dynamics (CFD) has reached the status of an established technology in the field of chemical process design and analysis. This has been motivated by the fact that simple reactor models, such as those for the continuous stirred-tank reactor and the plug-flow reactor, are inadequate for predicting the mixing, which, in turn is crucial for predicting the extent of chemical reaction, in industrial reactors. Several commercial CFD computer codes

*Correspondence to: P. J. Smith, Department of Chemical Engineering, University of Utah, 50 S. Central Campus Dr. Room 3290, Salt Lake City, UT 84112, U.S.A.
†E-mail: smith@crsim.utah.edu
‡E-mail: skumar@crsim.utah.edu

Contract/grant sponsor: Department of Energy's HIPPS Program; contract/grant number: DE-FG22-949C9422
Contract/grant sponsor: Center for the Simulation of Accidental Fires and Explosions; contract/grant number: B341493

Received 6 December 2004
Revised 11 July 2005
Accepted 13 July 2005

Copyright © 2005 John Wiley & Sons, Ltd.

exist today (e.g. http://www.cd-adapco.com/products/starsolver.htm, http://www.fluent.com/software/fluent/index.htm, http://www.reaction-eng.com/modeling_tools/banff.html) for the solution of steady-state problems involving large-scale furnaces, reactors, smelters, and fluidized beds. For such accurate modelling of chemical reaction in large and complex reactors, it is necessary to represent two processes faithfully: the macroscopic mixing, that brings relatively large clumps of reactant fluids together, and the microscopic mixing, that brings the reactants from these clumps together at the molecular level. This paper deals with fluid mechanics, which is responsible for macromixing in chemical reactors.

An important concern that arises in the application of CFD codes to practical problems is that often these codes have difficulty in converging to a solution (a robustness issue) or do not converge fast enough (an efficiency issue). This paper focuses on robustness and efficiency issues arising from the application of the Semi-Implicit Method for Pressure-Linked Equations—Revised (SIMPLER) [1] technique to steady-state incompressible fluid flows.

The map of the rest of this paper is as follows. In Section 2, we present a concise description of the SIMPLER technique that is discussed in detail in Reference [2]. In Section 3, aspects of the nonlinear Picard iteration are discussed. In particular, an analysis of a particular variety of under-relaxation, introduced in Reference [1] and used in this paper, is presented. In addition, the solution of the linear systems of equations by a nonstationary scheme is discussed with a view to understanding the measurement of work used in the section on results (Section 4). Section 3 also discusses the convergence criterion used later in Section 4. Finally, overall conclusions on the performance of the SIMPLER algorithm, as a projection algorithm in the context of a nonlinear Picard solver, are drawn.

2. RECAPITULATION OF THE SIMPLER TECHNIQUE

The governing equations for laminar fluid flows are the Navier–Stokes (NS) equations [3]. We use the steady-state form of the NS equations, in which we neglect the time derivative. From hereon, the term 'NS equations' refers to this steady-state form. For steady-state turbulent flows, the Reynolds decomposition is applied and the NS equations are time-averaged, resulting in the Reynolds-averaged Navier–Stokes (RANS) equations [4]. Application of the Boussinesq hypothesis and use of the Prandtl–Kolmogorov hypothesis results [4] in the k–ε model, which looks very similar to the NS equations, with the difference being the addition of two extra PDEs and the use of an effective viscosity, instead of the laminar viscosity used in the NS equations. The effective viscosity is comprised of the laminar viscosity and a 'turbulent viscosity.' The turbulent viscosity depends on the 'turbulent kinetic energy' and the 'rate of dissipation of turbulent kinetic energy' (or 'dissipation' for short). The two additional PDEs referred to above are those for the turbulent kinetic energy and the dissipation. We will refer to both the NS and the RANS equations as simply the NS equations; the exact form will be evident from the context.

When discretized using a finite-volume technique on a staggered grid, the equations of conservation have the following form. The linearized, discrete momentum equations are

$$A^{\mathbf{V}_1}\mathbf{V}_1 = S^{\mathbf{V}_1} \equiv \widehat{S^{\mathbf{V}_1}} - G^{\mathbf{V}_1}\nabla_{f1}^{h}\mathbf{P} \tag{1}$$

$$A^{\mathbf{V}_2}\mathbf{V}_2 = S^{\mathbf{V}_2} \equiv \widehat{S^{\mathbf{V}_2}} - G^{\mathbf{V}_2}\nabla_{f2}^{h}\mathbf{P} \tag{2}$$

Copyright © 2005 John Wiley & Sons, Ltd.

and

$$A^{\mathbf{V}_3}\mathbf{V}_3 = S^{\mathbf{V}_3} \equiv \widehat{S^{\mathbf{V}_3}} - G^{\mathbf{V}_3}\nabla_{f3}^h\mathbf{P} \tag{3}$$

In these equations, $\mathbf{V}_i$ represents the velocity component in the ith direction, $\mathbf{P}$ is the pressure, and ∇_{fi}^h represents the ith component of the face-centred discrete gradient operator, ∇_f^h. $A^{\mathbf{V}_i}$ is the linearized coefficient matrix for $\mathbf{V}_i$, and $S^{\mathbf{V}_i}$ and is its source term, comprised of the nonlinear term $\widehat{S_i^{\mathbf{V}}}$ and the pressure gradient integrated over the finite volume, $G^{\mathbf{V}_i}\nabla_{fi}^h$. If the flow is turbulent, then the set of equations also includes the finite volume-integrated discrete equations for the turbulent kinetic energy and dissipation

$$A^{\mathbf{K}}\mathbf{K} = S^{\mathbf{K}} \tag{4}$$

and

$$A^{\mathbf{E}}\mathbf{E} = S^{\mathbf{E}} \tag{5}$$

The momentum equations can be written in terms of a single state vector

$$A^{\mathbf{V}}\mathbf{V} = S^{\mathbf{V}} = \widehat{S^{\mathbf{V}}} - G^{\mathbf{V}}\nabla_f^h\mathbf{P} \tag{6}$$

When we decompose $A^{\mathbf{V}}$ into its diagonal and off-diagonal parts,

$$A^V = D^V - (L^V + U^V) \tag{7}$$

we get

$$D^{\mathbf{V}}\mathbf{V} = \left((L^{\mathbf{V}} + U^{\mathbf{V}})\mathbf{V} + \widehat{S^{\mathbf{V}}}\right) - G^{\mathbf{V}}\nabla_f^h\mathbf{P} = D^{\mathbf{V}}\hat{\mathbf{V}} - G^{\mathbf{V}}\nabla_f^h\mathbf{P} \tag{8}$$

which leads to

$$\mathbf{V} = \hat{\mathbf{V}} - \hat{D}\nabla_f^h\mathbf{P} \tag{9}$$

Multiplying both sides of (9) by the density, taking the cell-centred divergence, $\nabla_c^h\cdot$, and eliminating $\nabla_c^h \cdot (\rho\mathbf{V})$ using continuity, we get the pressure equation of the SIMPLER technique [1, 2]

$$\tilde{\Delta}^h\mathbf{P} \equiv \nabla_c^h \cdot (\rho\hat{D}\nabla_f^h)\mathbf{P} = \nabla_c^h \cdot (\rho\hat{\mathbf{V}}) \tag{10}$$

in which $\tilde{\Delta}^h$ is a generalized Laplacian. The scalar potential equation of the SIMPLER algorithm is derived by writing (9) for both an exact ($\mathbf{V}^*$) and an approximate ($\mathbf{V}$) velocity field, and by subtracting the two, assuming that $\hat{\mathbf{V}}$ is the same for the two cases, which leads to

$$\mathbf{V}' \equiv \mathbf{V}^* - \mathbf{V} = -\hat{D}\nabla_f^h(\mathbf{P}^* - \mathbf{P}) \tag{11}$$

Defining a scalar potential

$$\phi \equiv \mathbf{P}^* - \mathbf{P} \tag{12}$$

Copyright © 2005 John Wiley & Sons, Ltd.

and performing the same operations as for the pressure equation, we get the scalar potential equation

$$\tilde{\Delta}^h \phi = \nabla_c^h \cdot (\rho \mathbf{V}) \tag{13}$$

which gives us the velocity update

$$\mathbf{V}^* = \mathbf{V} - \hat{D} \nabla_f^h \phi \tag{14}$$

SIMPLER uses the velocity update (14) but not the pressure update, (12). As Kumar [2] points out, there is an important reason why the velocity update is acceptable even though the pressure update is not, and this reason is that (13) and (14) form a projection on the velocity field.

Based on this analysis, Kumar [2] proposes the following alternatives for the scalar potential field. First, one could formulate an orthogonal projection in an L^2 space:

$$(\rho \mathbf{V})^* = \rho \mathbf{V} - \nabla_f^h \phi \tag{15}$$

which gives the pure Poisson equation

$$\Delta^h \phi \equiv \nabla_c^h \cdot \nabla_f^h \phi = \nabla_c^h \cdot (\rho \mathbf{V}) \tag{16}$$

Another, *ad hoc*, alternative, is to look at (16) and realize that ϕ is a linear operator acting on $\rho \mathbf{V}$

$$(\rho \mathbf{V})^* = (I - \nabla_f^h (\Delta^h)^{-1} (\nabla_c^h \cdot))(\rho \mathbf{V}) \tag{17}$$

$$\phi = (\Delta^h)^{-1} (\nabla_c^h \cdot)(\rho \mathbf{V}) \tag{18}$$

which suggests that one could use an equation for ϕ of the form

$$\phi = \frac{1}{\kappa} \nabla_c^h \cdot (\rho \mathbf{V}) \tag{19}$$

κ could either be a constant or a variable. Kumar [2] suggests the possible alternatives

$$\kappa = 1 \tag{20}$$

and

$$\kappa = (D^{\mathbf{V}})^{-1} \tag{21}$$

Finally, one could decide to not solve a scalar potential equation at all, and rely on the pressure equation alone to enforce continuity on the velocity field. This is a weaker enforcement of continuity since, although it solves a generalized Poisson equation for the pressure, does not project the velocities back to a mass-conserving field.

Copyright © 2005 John Wiley & Sons, Ltd.

3. THE NONLINEAR PICARD ITERATION

The set of nonlinear equations to be solved is (1)–(3), (10), and, optionally, one of (16) or (19), and, in the case of turbulent flow, (4) and (5). (Here, we abandon (13) in favour of (16), based on the projection principles discussed in Reference [2].) There are many options on how to solve this system of nonlinear equations. We use the nonlinear Picard iteration as in Reference [1], by which we imply that we solve the equations in a sequential manner, updating coefficients and source terms between equations. The linearized equation for each variable is solved for that variable using an iterative linear solver until the *linear* residual of that equation drops below a satisfactory level. Once this is achieved, the variable is updated, the next equation is discretized and, the linearized equation for that variable is solved for, and so on.

What we are effectively doing is deriving measures of nonlinearity from the behaviour of linear systems. Needing to use the linear subproblems to measure nonlinearity, as opposed to relying on the nonlinear problem itself, is a limitation arising from using a Picard iteration as the nonlinear solver; this kind of solver does not provide information on nonlinearity, such as Jacobian information [5], which can be used [6, 7] to calculate the 'goodness' of a linearized approximation to the nonlinear system. Thus, our measure of nonlinear convergence (discussed in Section 3.3) is analysed as a function of the tightness of linear solves (discussed in Section 3.1), and the under-relaxation of linear equations (discussed in Section 3.2), both of which affect the linear update to the nonlinear equation. In addition, the nonlinearity of the system changes as we move from equation to equation, because the point around which the equation is linearized moves as we move from equation to equation due to the successive substitution in the Picard iteration.

Our performance analyses are strongly connected with the kind and implementation of linear solver used to solve the discretized equations. Therefore, in what follows, we give a brief description of linear solver ideas as they pertain to our results, although most of these ideas can be found, and in greater detail, in other sources (e.g. References [8, 9]). We also explain our metric for the convergence of the nonlinear iteration, since it is different from the metrics usually used in the literature for measuring nonlinear convergence.

3.1. Solution of the linear equations

The linearized equations for any variable can be written, in state space form (for the kinds of compact stencils analysed here; see Reference [2]) as

$$Ax = b \tag{22}$$

Given an initial guess, x_0, and an initial residual, r_0, defined as

$$r_0 = b - Ax_0 \tag{23}$$

the solution to (22) is the same [9] as the solution to

$$Ae = r_0 \tag{24}$$

where

$$x = x_0 + e \tag{25}$$

Copyright © 2005 John Wiley & Sons, Ltd.

In our case, the matrix A can be decomposed as

$$A = D - (L_1 + U_1 + L_2 + U_2 + L_3 + U_3) \tag{26}$$

in which D is the diagonal of the matrix A; L_1, L_2, L_3 are the three lower-triangular diagonals of A; and U_1, U_2, and U_3 are the three upper-triangular diagonals of A. Equation (24) is then operator-split to solve the system in each direction as a tridiagonal system (see Reference [2], for details on the implementation and mathematical representation). For example, one operator-split iteration can be written as

$$e^{l,1} = (D - L_1)^{-1}((U_1 + L_2 + U_2 + L_3 + U_3)e^l + r_0) \tag{27}$$

where l stands for the iteration number. One 'sweep,' defined here, is three successive tridiagonal solves, one in each direction. This amounts to one iteration of a steady ADI solve [8].

The ADI method is a stationary iterative method [8], and constructs the solution from a fixed residual, r_0, as opposed to nonstationary methods such as Generalized Minimal RESidual (GMRES) or Bi-Conjugate Gradient-STABilized (BiCGSTAB) [8, 9], which construct the solution from an additive combination of vectors based on the residual vector r_0 and the operator matrix A, such as $A^k r_0$ or $(A^{\mathrm{T}})^k r_0$, for whole number values of k. These methods are also known as Krylov methods [8]. The ADI method is then used as a preconditioner (approximate inverse) for the nonstationary iterative method. We use a variant of GMRES, called restarted GMRES, or GMRES (m), in which the solution is constructed from up to m vectors, each operated upon by an ADI preconditioning operation. If the error tolerance for the solution is achieved within m vectors, then the solution is calculated and the process is terminated. If not, the solution and residual are updated, and a new sequence of m vectors is generated, and so on, until the error tolerance or the maximum number of Krylov vectors specified is reached.

In our solution technique, GMRES uses a specified total number of vectors, each of which requires a preconditioning operation. The total number of GMRES vectors used is therefore the total number of preconditioning operations (i.e. sweeps), and is a measure of the tolerance to which the linear system is solved, also referred to here as the tightness of the linear solve. As the tightness of the linear solve is increased (i.e. the tolerance is decreased), the number of GMRES vectors that will be needed to perform the solve also increases.

As mentioned in Reference [2], the projection equations help to maintain the mass flux field in a divergence-free space. The question to be answered is how tightly the projection equations need to be solved; in other words, how closely the mass flux needs to be maintained in a divergence-free space. This aspect will be studied in the section on results.

3.2. *Under-relaxation*

The updates to the variables that are obtained for a nonlinear equation are only locally valid in the vicinity of a linearization, and the values may need to be damped (under-relaxed) as a result. We use the form of under-relaxation used by Patankar [1]. This under-relaxation damps the updates for the nonlinear iteration through a modification to the linear system of equations. The reason as to why we use this particular form of under-relaxation is shown in what follows.

If we decompose the matrix A as

$$A = D - L - U \tag{28}$$

Copyright © 2005 John Wiley & Sons, Ltd.

where D is the diagonal part of A, and L and U are the strictly lower- and strictly upper-triangular parts of A, the Jacobi iteration [9] is defined as

$$De^{l+1} = (L + U)e^l + r^l \tag{29}$$

in which

$$r^l = Ae^l \tag{30}$$

Equation (29) can be rewritten as

$$e^{l+1} = D^{-1}((L + U)e^l + r^l) = D^{-1}(L + U + A)e^l = K_J e^l \tag{31}$$

Here, K_J is the convergence matrix for the Jacobi iteration. It follows that the correction vector after n iterations is

$$e^n = K_J^{n-1} e^1 = K_J^{n-1} D^{-1} r_0 = (D^{-1}(L + U + A))^{n-1} D^{-1} r_0 \tag{32}$$

Now, we define, as in Reference [1], an under-relaxation factor, α, such that

$$\tilde{D} = (1/\alpha)D \tag{33}$$

and

$$\tilde{b} = b + (1 - \alpha)\tilde{D}x_0 \tag{34}$$

The significance of the under-relaxation implied by (33) and (34) is that, for a Jacobi iteration, the solution yielded by this transformation corresponds to a scaling of the correction vector at *every* Jacobi iteration $l, \tilde{e}^l$, by the factor α, and thus the correction vector after n iterations, $\tilde{e}^n$, by α^n

$$\tilde{e}^n = \tilde{K}_J^{n-1} e^1 = \tilde{K}_J^{n-1} \left(\frac{D}{\alpha}\right)^{-1} r_0 \equiv \left(\left(\frac{D}{\alpha}\right)^{-1} (L + U + A)\right)^{n-1} \left(\frac{D}{\alpha}\right)^{-1} r_0 \tag{35}$$

i.e.

$$\tilde{e}^n = \alpha^n((D^{-1}(L + U + A))^{n-1} D^{-1} r_0) = \alpha^n K_J^{n-1} D^{-1} r_0 = \alpha^n e^n \tag{36}$$

For a value of $\alpha = 0.99$, this yields a reduction in the correction vector of about 0.82 after 20 iterations. This can be particularly severe when one considers that the linear solver is embedded within a nonlinear solver; in such a case, if one can approximate the nonlinear problem as a linear one, and ignore the effects of other variables on the convergence of the variable under consideration, the reduction after 20 nonlinear iterations, each with 20 Jacobi iterations, is about 0.99^{400}, or 0.018. This is an appreciable slowdown; in practice, the nonlinearity of the problem perturbs the point around which the linearization is done, and can considerably amplify the slowdown. This will be explored more in the section on results. This is the motivation for using this kind of under-relaxation, i.e. the fact that it is a particularly severe form of damping, and is therefore a very good test of the sensitivity of the nonlinear aspects of the problem to effects arising from the linear solutions to subproblems.

Copyright © 2005 John Wiley & Sons, Ltd.

The analysis for the Gauss–Seidel iteration is very similar. Specifically, the convergence matrix here is

$$K_{\mathrm{GS}} = (D - L)^{-1}(A + U) \tag{37}$$

and

$$e^1 = (D - L)^{-1} r_0 \tag{38}$$

The correction vector after n iterations is given by

$$e^n = (K_{\mathrm{GS}})^{n-1} e^1 = ((D - L)^{-1}(A + U))^{n-1}(D - L)^{-1} r_0 \tag{39}$$

For the under-relaxed iteration, we have

$$\tilde{e}^n = (\tilde{K}_{\mathrm{GS}})^{n-1} e^1 = \left(\left(\frac{D}{\alpha} - L\right)^{-1}(A + U)\right)^{n-1}\left(\frac{D}{\alpha} - L\right)^{-1} r_0 \tag{40}$$

i.e.

$$\tilde{e}^n = \alpha^n((D - \alpha L)^{-1}(A + U))^{n-1}(D - \alpha L)^{-1} r_0 \tag{41}$$

Thus, we see the same shortening of the correction vector; however, in this case, the length as well as the direction of the step vector (due to the α in $D - \alpha L$) change. Again, this can be significant over a large number of linear as well as nonlinear iterations when the linear problem is part of an overlying nonlinear problem.

3.3. Convergence criterion

We explain here our measure of convergence of the nonlinear iteration, which will be used to show our computational results. At each iteration, l, we first calculate an L^1 norm of the residual of each variable ϕ at each grid point

$$R_\phi^l \equiv \sum_{i,j,k} |r_\phi^l(i, j, k)| \tag{42}$$

This residual is normalized by an order-of-magnitude term, T_ϕ^l, for the variable ϕ at the iteration level l. This term, in turn, is taken to be the L^1 norm of a pointwise order-of-magnitude term, $t_\phi^l(i,j,k)$

$$T_\phi^l = \sum_{i,j,k} |t_\phi^l(i,j,k)| \tag{43}$$

The term $t_\phi^l(i,j,k)$, in turn, is calculated as the greater of the two terms $t_{1\phi}^l(i,j,k)$ and $t_{2\phi}^l(i,j,k)$, the former being the sum of the positive terms in the finite-difference equation at that point, and the latter being the sum of the negative terms in the finite-difference equation at that point. Thus

$$t_\phi^l(i,j,k) = \max(t_{1\phi}^l(i,j,k), |t_{2\phi}^l(i,j,k)|) \tag{44}$$

Copyright © 2005 John Wiley & Sons, Ltd.

The ratio R_ϕ^l/T_ϕ^1 is less than unity and is a measure of how small the residual is, relative to the size of the terms in the equation. In general, if $\varepsilon_{\text{mach}}$ is the precision of the machine, then the number of digits of precision of the machine is

$$D_{\text{mach}} = \log\left(\frac{1}{\varepsilon_{\text{mach}}}\right) \tag{45}$$

The ratio R_ϕ^l/T_ϕ^l can, therefore, be only as small as $10^{-D_{\text{mach}}}$. One can thus define, at each iteration l, the 'approach to convergence,' C_ϕ^l, of the variable ϕ

$$C_\phi^l \equiv D_{\text{mach}} + \log\left(\frac{R_\phi^l}{T_\phi^l}\right) \tag{46}$$

(The normalized residual, R_ϕ^l/T_ϕ^l, can be viewed as an $L1$ norm of the residual, $r_\phi^l(i,j,k)$, relative to a vector space weighted by $1/|t_\phi^l(i,j,k)|$.) As one approaches convergence, the residuals of the governing equations become smaller and smaller relative to the terms in the equations, and so R_ϕ^l/T_ϕ^l becomes closer to machine zero, and thus C_ϕ^l approaches zero. This makes the definition of a fully converged solution easy: when C_ϕ^l is zero, the equation for ϕ is converged. For the whole system of equations, an overall approach to convergence, C^l, is calculated at each iteration l, as follows:

$$C^l = \max_\phi(C_\phi^l) \tag{47}$$

The nonlinear method is fully converged when the overall approach to convergence is zero. In practice, this is rarely possible, and so a value is set by the user for how low C^l can be before it is considered to represent a converged solution.

4. COMPUTATIONAL RESULTS

Following a description of the test cases, we discuss, in order, the dependence of the convergence history on under-relaxation factors, variations on the projection algorithm, and tightness of the linear solves. Once we determine the optimal under-relaxation factors for the pressure and scalar potential equations, we use that information in the rest of the analysis in this paper. The results shown for all calculations are optimal convergence histories; the ranges of values of under-relaxations and tightness of linear solves studied are far greater than the ones presented. The dependence of the convergence histories on under-relaxation factors and tightness of linear solves is only studied for the pressure and scalar potential equations, and not the momentum or the kinetic energy or dissipation equations. The tightness of the linear solves of variables other than the pressure and scalar potential equations is not very significant in terms of overall convergence rate, because the linear equations for these other variables are well-behaved and converge rapidly. The under-relaxation factors for these variables, however, affect the convergence rate. Since our focus here is on the pressure and scalar potential projection equations, these under-relaxation factors are kept constant for each case. Many of the convergence histories are shown with respect to both the CPU time and the number of nonlinear iterations. This is important because, if more efficient ways are used to solve the

Copyright © 2005 John Wiley & Sons, Ltd.

linear equations, the nonlinear convergence becomes more important than the linear convergence, and the variable that can measure the nonlinear convergence is the number of nonlinear iterations.

All calculations are performed using GMRES (30) as the linear solver.

4.1. Test cases and computational resources

The four test cases are based on one physical situation: the flow of a jet in a sudden expansion duct. The geometry is shown in Figure 1. The parameters for the flows are

$$V_{\text{ave}} = \frac{V(W1)^2}{W^2} \tag{48}$$

$$Re = \frac{V_{\text{ave}} W \rho}{\mu} \tag{49}$$

The four cases are as follows:

Case 1: $W = 0.2\,\text{m}$, $L = 2.0\,\text{m}$, $W1 = 0.06\,\text{m}$, $Re = 100$, laminar flow, $10 \times 10 \times 10$ uniform grid.

Case 2: $W = 0.2\,\text{m}$, $L = 2.0\,\text{m}$, $W1 = 0.06\,\text{m}$, $Re = 1000$, turbulent flow, $10 \times 10 \times 10$ uniform grid.

Case 3: $W = 0.2\,\text{m}$, $L = 3.41\,\text{m}$, $W1 = 0.086\,\text{m}$, $Re = 101\,386$, turbulent flow, $40 \times 20 \times 20$ nonuniform grid.

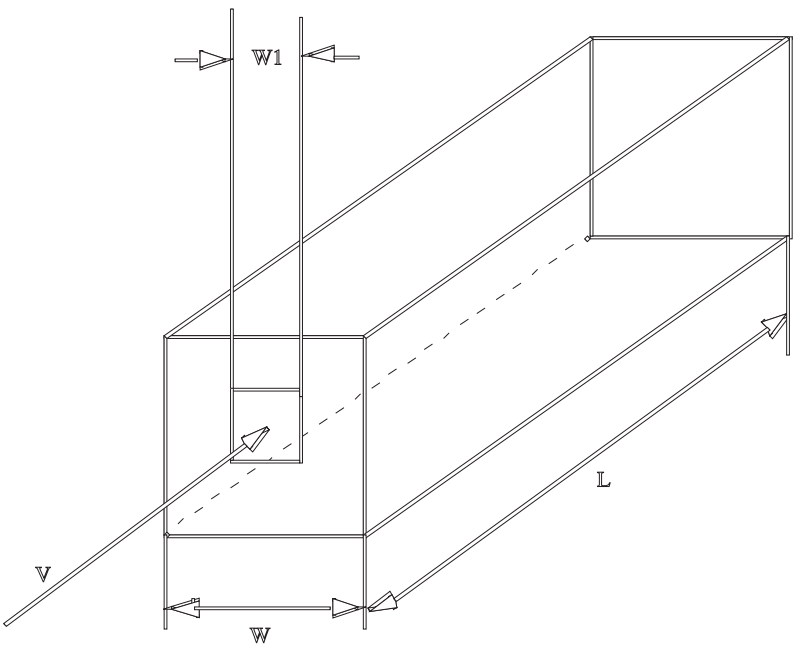

Figure 1. Geometry of cases considered:sudden expansion of a jet.

Copyright © 2005 John Wiley & Sons, Ltd.

Case 4: $W = 0.2\,\text{m}$, $L = 2.0\,\text{m}$, $W1 = 0.06\,\text{m}$, $Re = 100$, laminar flow, $60 \times 30 \times 30$ uniform grid.

The first three test cases are designed to be progressively more nonlinear in both the physics and the numerics. The fourth case is designed to provide a comparison to the first case with a more refined calculation. All calculations for Cases 1 and 2 were performed on a Silicon Graphics Octane with a MIPS R10000 250 MHz IP30 processor and 768 MB of RAM. All calculations for Case 3 were performed on a Silicon Graphics Origin 2000 with a MIPS R10000 250 MHz IP27 processor and 5.2 GB of RAM. Both these machines have 15.6 significant digits of accuracy in double precision. The calculations for Case 4 were performed on a Dell Precision 670 with an Intel Xeon processor and 3 GB of RAM. This machine has 18.9 digits of precision.

4.2. *Under-relaxation factors*

Figures 2–4 show the effect of the pressure under-relaxation factor (urfp) on the convergence history of the nonlinear Picard iteration for Cases 1–3, respectively. NSWPP, NSWPC, NSWPV, and NSWTK are the number of sweeps performed for the pressure, scalar potential, velocity components, and kinetic energy (and dissipation), respectively. The under-relaxation factors for the velocity components are kept at 0.7, and those for the kinetic energy and dissipation are kept at 0.85 for Cases 2 and 3. For the study of the pressure equation, the under-relaxation factor for the scalar potential is kept at 1.0. From our earlier discussion on the effect of under-relaxation on the scaling of the step, a slowdown in the convergence rate for the nonlinear iteration is expected. (For each nonlinear iteration, we expect a scaling of the step by a factor of only about 0.901 for a value of urfp of 0.995, in 20 linear iterations.)

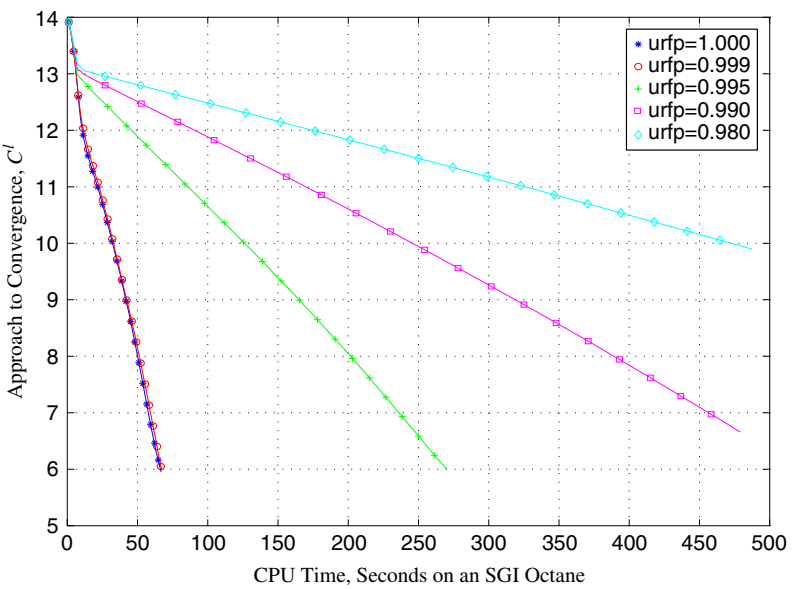

Figure 2. Effect of pressure under-relaxation, Case 1, NSWPV = 5, NSWPP = 20, NSWPC = 20.

Copyright © 2005 John Wiley & Sons, Ltd.

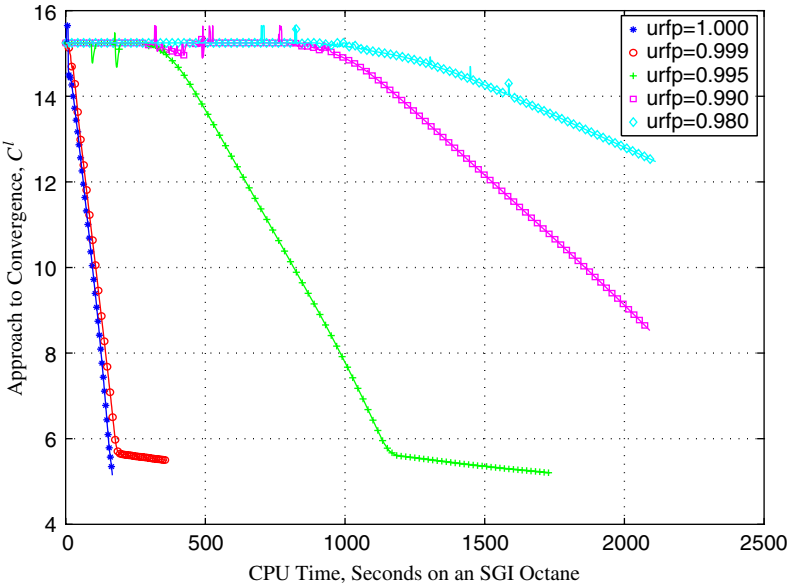

Figure 3. Effect of pressure under-relaxation, Case 2, NSWPV = 5, NSWTK = 5, NSWPP = 20, NSWPC = 20.

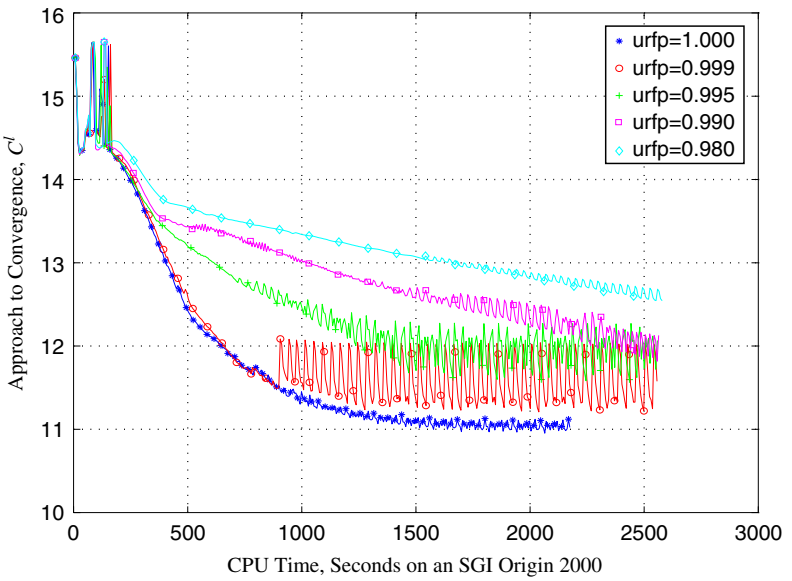

Figure 4. Effect of pressure under-relaxation, Case 3, NSWPV = 5, NSWTK = 5, NSWPP = 20, NSWPC = 20.

Copyright © 2005 John Wiley & Sons, Ltd.

(For clarity, not every point corresponding to a nonlinear iteration is shown in the graphs.) However, the dependence of the convergence rate on urfp is drastic. It is likely that this drastic performance is due to the fact that both the length *and* the direction of the solution vector are modified in each linear step.

A couple of features need to be observed here. First, in Cases 2 and 3, it can be seen that the convergence tails off after a certain point. We refer to this as the asymptotic convergence; this is dependent on the relative noise in the function evaluation and the range of scales in the problem. We show how several parameters in our study affect this value. In Case 2, this is reached at a value of about 5.5, and in Case 3, the best scenario value in all the simulations in this paper is about 11, about a 4.5 orders of magnitude reduction in the overall residual.

Second, in Case 3, we can see that, for an under-relaxation factor for the pressure of 0.999, the asymptotic approach to convergence starts to bounce between approximately 12 and 11.25, after it reaches a value of 11.5. Whenever the convergence history shows an oscillating behaviour, we refer to it as erratic convergence. This can either go down on average or reach an asymptotic value on average; in this case, we get an erratic asymptotic convergence.

Figures 5–7 show the effect of the scalar potential under-relaxation factor (urfc) on the convergence history for Cases 1–3, respectively. For these cases, urfp is kept at 1.0. That the nonlinear iteration should behave so well for even urfp = 0.01 (which, according to Section 3.2, yields a scaling of the step by a factor of 1.0e-40, in 20 linear iterations!) is surprising. As we can see, for Case 3, we again see erratic asymptotic convergence for values of the scalar under-relaxation factor other than 1, whereas the asymptotic convergence for an under-relaxation factor of 1 is 11, an order of magnitude less than that for under-relaxation factors other than 1.

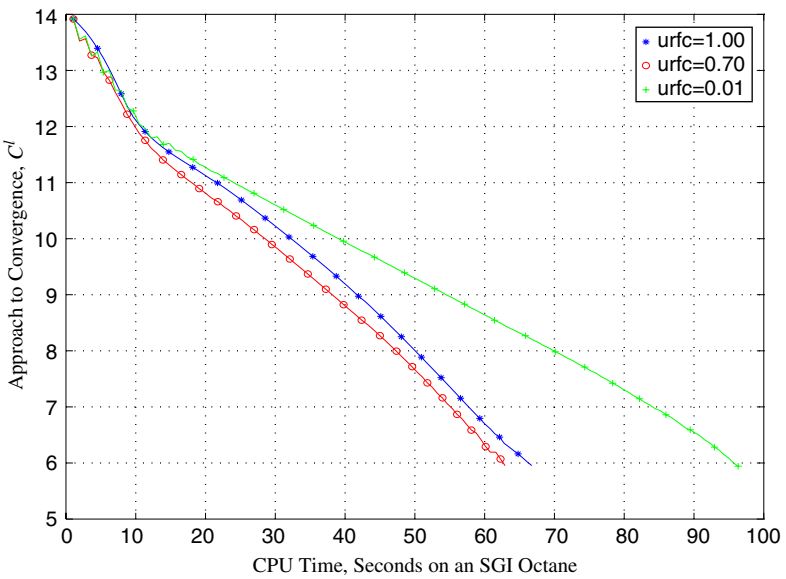

Figure 5. Effect of scalar potential under-relaxation, Case 1, NSWPV = 5, NSWPP = 20, NSWPC = 20.

Copyright © 2005 John Wiley & Sons, Ltd.

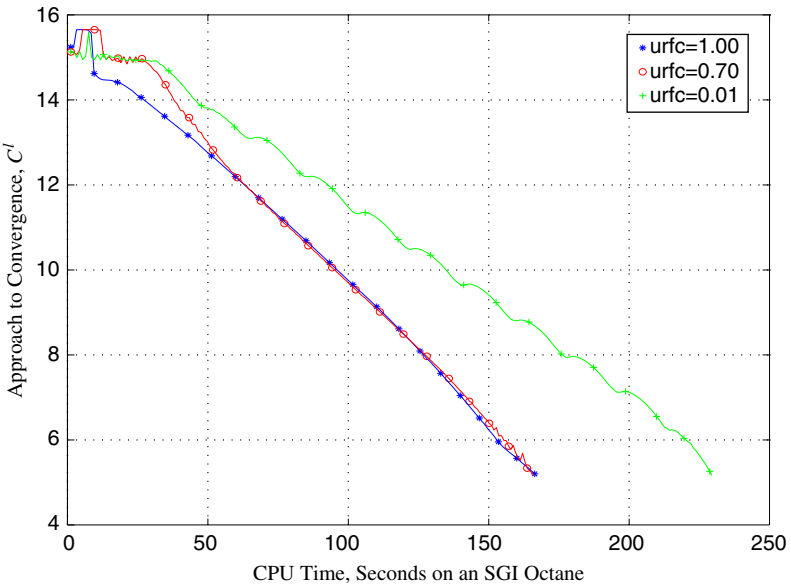

Figure 6. Effect of scalar potential under-relaxation, Case 2, NSWPV = 5, NSWTK = 5, NSWPP = 20, NSWPC = 20.

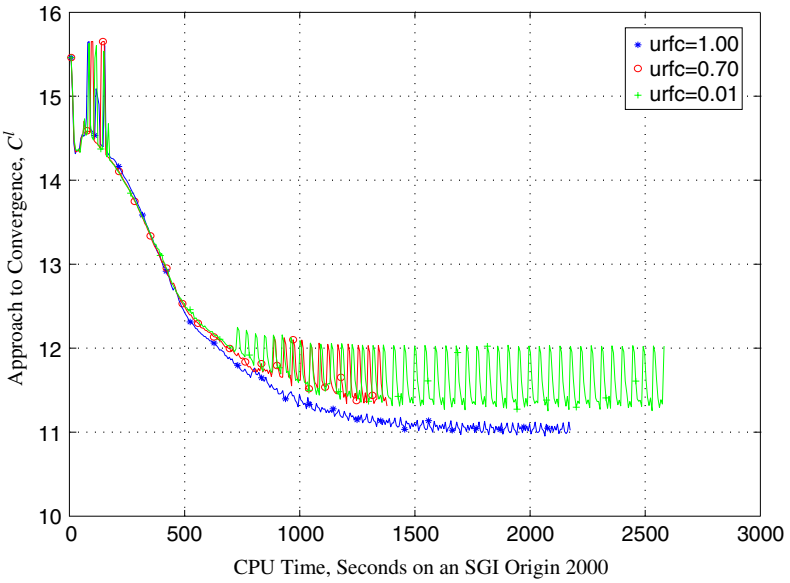

Figure 7. Effect of scalar potential under-relaxation, Case 3, NSWPV = 5, NSWTK = 5, NSWPP = 20, NSWPC = 20.

Copyright © 2005 John Wiley & Sons, Ltd.

The pressure equation yields an update to the pressure through enforcement of the continuity constraint in the SIMPLER algorithm, but does not correct the velocities to satisfy the continuity constraint. Thus, the projection of the mass fluxes into a divergence-free space is not complete. The pressure is used in the momentum equations to yield a velocity; thus, the velocities obtained from the momentum equation weakly satisfy the continuity constraint. However, the pressure obtained from the pressure equation is consistent with a divergence-free mass flux. The scalar potential equation does not correct the pressure, but strictly enforces the divergence-free condition on the mass fluxes. This suggests that the *strict* enforcement of the continuity constraint using the scalar potential (projection) equation is not absolutely essential. The next section explores the question of whether the scalar potential equation should be solved at all.

4.3. Projection algorithm variations

As discussed in Section 2, a variation on the SIMPLER strategy is to completely omit the scalar potential projection. Two other variants were discussed in that section: the use of (19), with a value of $\kappa = 1$ and with a value of κ equal to the inverse of the diagonal of the Laplacian operator (integrated over the control volume). This section studies the effects of such variations on the SIMPLER projection scheme on the overall nonlinear convergence.

Figure 8 shows the effect of the different projections on the convergence history of Case 1. In this figure, 'default' refers to the use of the projection defined by (15) and (16), with an underrelaxation of one for the scalar potential, 'alt1' refers to the use of (19) and (20), 'alt2' refers to the use of (19) and (21) and 'nopc' refers to the case where the scalar potential equation is not solved for at all. Figure 9 shows the convergence history for the

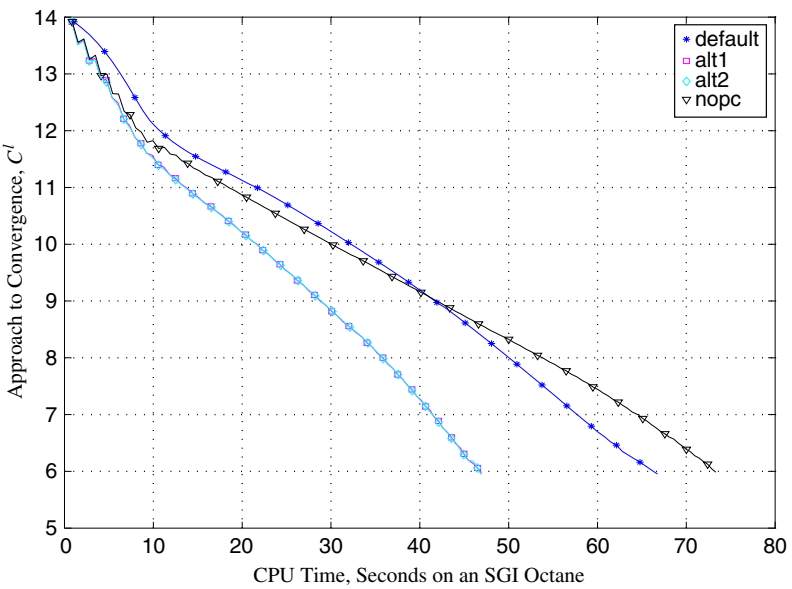

Figure 8. Effect of variation of projection algorithm, Case 1, NSWPV = 5, NSWPP = 20, NSWPC = 20.

Copyright © 2005 John Wiley & Sons, Ltd.

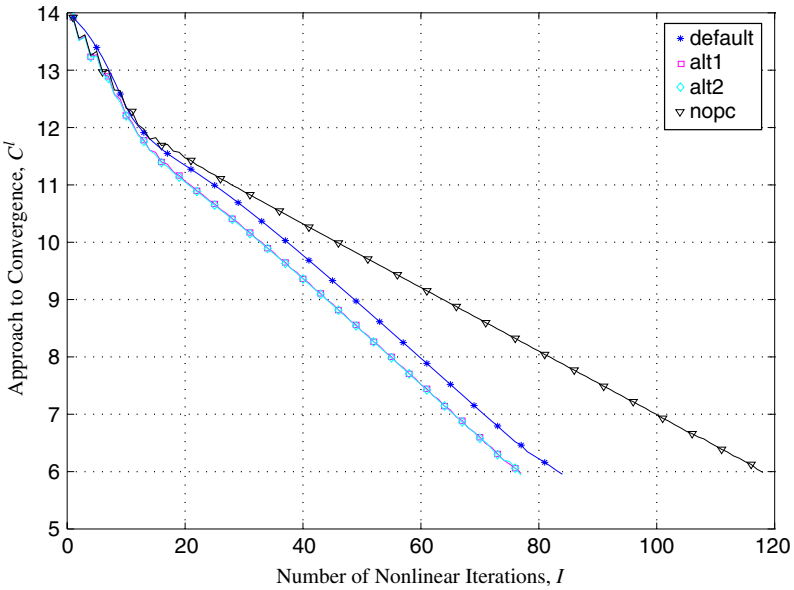

Figure 9. Effect of variation of projection algorithm, Case 1, NSWPV = 5, NSWPP = 20, NSWPC = 20.

Table I. Dependence of convergence history on scalar potential under-relaxation.

	Case 2:$T(6)$	Case 2:$N(6)$	Case 3:$N(11.5)$
default	155	150	150
alt1	123	150	150
alt2	123	150	150
nopc	162	195	150

same comparison of solvers, but plotted against the number of overall nonlinear iterations. For this case, using alt1 and alt2 give the same result. Furthermore, they give slightly better convergence rates per nonlinear iteration, and since they do not require the solution of a Poisson equation, they are more efficient than the default projection method. It is also interesting to note what happens when the scalar potential equation is not solved at all. Figure 9 shows that the convergence rate per nonlinear iteration is better when the scalar potential equation is solved, but because of the extra work that is needed in solving the equation, the default projection method is only as efficient as not solving the scalar potential equation. In other words, the efficiency gained by solving the scalar potential equation is compensated for by the work needed to solve the scalar potential equation.

Cases 2 and 3 show similar trends, as can be seen in Table I. In these, T(6) and N(6) stand for the CPU time (in seconds) and number of nonlinear iterations needed in order to reduce C^l to a value of 6. The only difference (not shown here) from Case 1 is that when the scalar potential is not solved for at all, the convergence is somewhat wavy (Case 2) or noisy

Copyright © 2005 John Wiley & Sons, Ltd.

(Case 3). For Case 3, all four projection options take the same number of nonlinear iterations for most of the history, thereby implying that there is no benefit in the early period of the history of solving the scalar potential equation. However, the asymptotic level of convergence is erratic for all options other than the 'default' option, and bounces between approximately 11.25 and 12.

Thus, an overall conclusion about the different projection options, based on the above data, is that the scalar potential equation should be solved when possible. Even though only one case, *viz.* Case 3, shows the loss in the asymptotic level of convergence reached by the simulations due to the inaccuracy in solving the scalar potential equation, this case is also the most nonlinear and complex, and hence the closest to real situations involving complicated nonlinear behaviour. The data also suggest that the scalar potential equation may not need to be solved very accurately. This hypothesis will be explored in the next section.

4.4. Tightness of linear solves

The effect of the tightness of linear solves on the overall nonlinear convergence is evaluated for each case by varying two parameters over a range and observing the convergence histories. The two parameters are NSWPP and NSWPC, the maximum number of sweeps (or Krylov vectors, because GMRES is used as the linear solver) allowed for the solution of the pressure and scalar potential equations, respectively. The tighter the solve on the pressure equation, the more the solution of the pressure equation is consistent with an exact projection of the momentum equations into a divergence-free space. Similarly, a larger value of NSWPC means that the mass flux at the end of a nonlinear iteration is more divergence-free. For all the calculations here, the values of urfp and urfc are kept at unity.

The ranges of NSWPC and NSWPP used for the three cases are chosen such that they represent a good spread in the tightness of the solves. From previous experience, a range from 10 to 200 appears to be a fairly good one. For the simpler cases such as Cases 1 and 2, increasing the value of NSWPC or NSWPP to beyond 200 is pointless, since the equations are solved to machine precision with a value less than 200. For Case 3, the number of Krylov vectors needed to obtain machine precision is found to be much larger, on the order of a few hundred to a thousand. But it is not practical to use as many as 1000 vectors to solve a linear problem at each nonlinear iteration. (This is an example of a situation in which using a more efficient linear solver, such as a multigrid solver, might greatly decrease the amount of CPU time needed for solution of the linear equations. This would change the focus from the CPU time to the number of nonlinear iterations.) Thus, for all cases, 200 is chosen to be the upper value of the ranges. The lower value of 10 is obtained somewhat by trial and error, based on the idea that a minimum number of Krylov vectors would be needed to obtain a good solution. For Case 1, since the CPU time needed to converge each simulation is not very large, a greater range is chosen by making the minimum value of NSWPC to be as low as 2. Within the ranges chosen, the selection of individual values of NSWPP and NSWPC is fairly arbitrary.

For Case 1, the effect of solving the pressure equation to varying levels of accuracy is studied by using the following set of values of NSWPP: $\{5, 10, 20, 40, 80, 200\}$. This is done for a value of NSWPV $= 5$ and for each of the following values of NSWPC: $2, 3, 5, 10, 20, 40, 80,$ and 200. We observed that a certain 'break-even' value of NSWPP exists for each value of NSWPC; the convergence rate is optimal for this value of NSWPP. A value of NSWPP

Copyright © 2005 John Wiley & Sons, Ltd.

Table II. Effect of NSWPP on convergence rate, Case 1, NSWPV = 5, NSWPC = 5.

NSWPP	$T(6)$	$N(6)$
5	50	89
10	48	81
20	52	82
40	58	82
80	71	82
200	75	82

Table III. Effect of NSWPC on convergence rate, Case 1, NSWPV = 5, NSWPP = 10.

NSWPC	$T(6)$	$N(6)$
2	43	78
3	46	81
5	47	82
10	52	83
20	63	83

smaller than this break-even value yields erratic convergence, and a value greater than this value does not yield any improvement in the convergence rate; in fact, since more work is done per nonlinear iteration with a larger value of NSWPP, this makes the iteration inefficient. The NSWPP = 5 convergence history is rather erratic, whereas all the other curves are smooth, with the NSWPP = 10 curve being the most efficient. The reason for this behaviour is that all the simulations with a value of NSWPP greater than or equal to 10 take the same number of nonlinear iterations to converge to a given level. For all the NSWPC values reported above, the break-even value of NSWPP is found to be 10.

For each of the NSWPP values from the set $\{5, 10, 20, 40, 80, 200\}$, the effect of NSWPC on the nonlinear convergence is also studied for the range of NSWPC values: $\{2, 3, 5, 10, 20, 40, 80, 200\}$. It is found that the most efficient convergence is obtained with NSWPC = 2. The effect of varying NSWPP and NSWPC on the convergence history of Case 1 can be seen in Tables II and III.

Case 3 shows similar behaviour. For Case 3, the effect of the tightness of the linear solves is studied by simulating, for NSWPV = 5 and NSWTK = 5, the following range in both NSWPP and NSWPC: $\{10, 20, 80, 200\}$. This case, too, shows the break-even behaviour for NSWPP: below a certain value of NSWPP (= 20), the convergence is erratic. But beyond that value, further work does not improve efficiency. As in Case 1, the best value for NSWPC is the lowest used; in this case, 10. As in Case 1, further work on the scalar potential equation solve is wasted. The observations for Case 3 are summarized in Tables IV and V.

Case 2 is slightly different in that the lowest value of NSWPC is not the most efficient for all values of NSWPP. Using NSWPV = 5 and NSWTK = 5, the following range in both NSWPP and NSWPC is studied: $\{10, 20, 40, 80, 200\}$. Here, too, for NSWPP = 10 and 20,

Copyright © 2005 John Wiley & Sons, Ltd.

Table IV. Effect of NSWPP on convergence rate, Case 3, NSWPV = 5, NSWTK = 5, NSWPC = 10.

NSWPP	$T(11)$	$N(11)$
10	Not achieved	Not achieved
20	1750	300
80	3000	300
200	5000	300

Table V. Effect of NSWPC on convergence rate, Case 3, NSWPV = 5, NSWTK = 5, NSWPP = 20.

NSWPC	$T(11)$	$N(11)$
10	1750	300
20	1750	300
80	3000	300
200	5000	300

Table VI. Effect of NSWPC on convergence rate, Case 2, NSWPV = 5, NSWTK = 5, NSWPP = 80.

NSWPC	$T(6)$	$N(6)$
10	200	155
20	180	135
40	205	125
80	260	125
200	420	125

the lowest value of NSWPC, namely 10, performs the best, requiring just as many nonlinear iterations as the other simulations with larger values of NSWPC. However, for NSWPP = 40 and 80, the break-even value of NSWPC seems to shift from 10 to 20 (for NSWPP = 40) to 40 (for NSWPP = 80); in other words, when more work is done on the scalar potential equation, the simulation takes fewer nonlinear iterations to converge, up to a point. Again, with a more efficient linear solver, what would be more important is the dependence of the convergence history on the number of nonlinear iterations, and thus the trends generally agree with Cases 1 and 3. This is shown in Table VI.

The convergence results for Case 4 are presented in Tables VII and VIII. These show that there exists a break-even point in terms of number of CPU time to converge by 6 orders of magnitude which is NSWPP = 10, NSWPC = 20. This is close to yielding the fewest nonlinear iterations. Looking further, we see that for all values of NSWPP, moderate values of NSWPC (around 20) give the minimum CPU time. Break-even points for NSWPP for a given NSWPC are also moderate (10–30). For a very low value of NSWPP, it may be deleterious to 'over-solve' the scalar potential equation, as can be seen for values of NSWPP of 5 and NSWPC

Copyright © 2005 John Wiley & Sons, Ltd.

Table VII. Effect of NSWPP and NSWPC on number of nonlinear iterations, $N(13)$, Case 4.

NSWP	$P=5$	$P=10$	$P=20$	$P=30$	$P=40$	$P=80$
$C=5$	674	382	323	371	362	836
$C=10$	509	319	370	398	412	246
$C=20$	412	253	232	325	293	236
$C=30$	524	348	272	209	264	259
$C=40$	586	304	221	200	178	174
$C=80$	Not achieved	471	291	221	227	202

Table VIII. Effect of NSWPP and NSWPC on CPU Time, $T(13)$, Case 4.

NSWP	$P=5$	$P=10$	$P=20$	$P=30$	$P=40$	$P=80$
$C=5$	1067	681	715	991	1089	3891
$C=10$	936	649	879	1101	1300	1169
$C=20$	923	603	639	1074	1025	1121
$C=30$	1390	982	918	771	1079	1368
$C=40$	1790	955	810	829	835	1021
$C=80$	Not achieved	2427	1534	1244	1350	1529

of 80. Note that $T(13)$ for this case represents a reduction in the initial residual by 6 orders of magnitude, since the machine used has roughly 19 digits of precision.

The overall conclusions regarding the tightness of linear solves for the scalar potential equation is that the scalar potential equation needs to be solved only to a very coarse tolerance. This is in agreement with Sections 4.2 and 4.3, which suggest that whereas the under-relaxation factors for the scalar potential can be very large, and whereas coarse alternatives to the scalar potential equation can sometimes do well, the solution (albeit a very coarse one) of the scalar potential is still necessary. This highlights the importance of projecting the mass flux into a divergence-free space; a small correction has a large impact on the solution.

The overall conclusion regarding the tightness of linear solves for the pressure is that while there seem to be break-even points for NSWPP for all the cases, these are usually not very large numbers, and beyond that the less work one does in solving these equations, the more efficient one is in converging to the solution.

5. CONCLUSIONS

We have studied the sensitivity of the convergence of the nonlinear Picard method to the maintenance of the mass fluxes in a divergence-free space.

The variables that affect the projection of the mass fluxes to a divergence-free space, that are studied are: the values of the under-relaxation factors, the choice of projection algorithm, and the tightness of linear solves, for both the pressure and the scalar potential equations. The conclusions from this study are framed in the context of projection ideas for the pressure and the scalar potential.

Copyright © 2005 John Wiley & Sons, Ltd.

The overall conclusions about the SIMPLER algorithm can be summarized as follows.

The first conclusion is that the nonlinear convergence is very sensitive to the correct solve of the pressure equation. In terms of nonlinear iterations, there is a point of diminishing returns, beyond which more work done on the pressure equation does not yield an improvement in the nonlinear convergence. This can be significant in terms of CPU time, because of the expense concomitant with the solution of a Poisson equation. The determination of the correct values of the number of sweeps for the pressure, and hence the enforcement of the mass flux projection, is not obvious. More work is needed before this can be correctly predicted.

The second conclusion is more powerful than the first; that the nonlinear convergence is only weakly dependent on the linear convergence of the scalar potential (also referred to in the literature as the pressure correction) equation, in the SIMPLER algorithm. The nonlinear convergence is not strongly sensitive even to the presence or absence of a scalar potential equation. However, the studies do indicate that a coarse solve of the scalar potential equation is beneficial to the convergence of the nonlinear iteration; failure to include it can potentially affect the asymptotic convergence of the nonlinear iteration. This sensitivity is found to be stronger in more complex cases. This has implications for real cases, which are far more complex than the most complicated case shown here.

These conclusions suggest that the mass flux projection inherent in the pressure equation, although not strictly enforced via the Hodge decomposition for the mass fluxes (but applied to the pressure), is sufficient, in large part, to maintain the mass fluxes in a divergence-free space. This implies that the weak maintenance of the mass fluxes in a divergence-free space, through the solve of the momentum equations, is sufficient for a practical simulation. The addition of a scalar potential definitely improves the convergence rate; but this is to be expected, since tighter enforcement of the mass flux projection helps to improve the asymptotic convergence. The second projection step is expensive because of the time involved in solving an extra Poisson equation; however, this can be overcome by using a more efficient solver, such as a multigrid solver.

The benefit of this study, however, is that one does not necessarily need Jacobian information or similar such information stemming from the theory of nonlinear equations [5] in order to correctly determine the appropriate update of the linearized step for optimal convergence; the optimal enforcement of the continuity projections is enough to determine optimal convergence of the NS equations.

ACKNOWLEDGEMENTS

This work was sponsored in part by the Department of Energy's HiPPS Program, Contract Number DE-FG22-949C9422 and in part by the Center for the Simulation of Accidental Fires and Explosions (C-SAFE), a Department of Energy Accelerated Strategic Computing Initiative (ASCI) Program Research Center, Contract Number B341493.

REFERENCES

1. Patankar SV. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill: New York, 1980.
2. Kumar S. A comparative analysis of solution approaches for steady-state three-dimensional internal flows. *Ph.D. Dissertation*, Department of Chemical and Fuels Engineering, University of Utah, Salt Lake City, UT, 1999.
3. Bird RB, Stewart WE, Lightfoot EN. *Transport Phenomena*. Wiley: New York, 1960.
4. Wilcox DC. *Turbulence Modeling for CFD*. DCW Industries: La Cañada, CA, 1993.

Copyright © 2005 John Wiley & Sons, Ltd.

5. Dennis Jr JE, Schnabel RB. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics: Philadelphia, 1996.
6. Eisenstat SC, Walker HF. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing* 1996; **17**:16–32.
7. Dembo RS, Eisenstat SC, Steihaug T. Inexact Newton methods. *SIAM Journal on Numerical Analysis* 1982; **19**:400–408.
8. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company: New York, 1996.
9. Kelley CT. Iterative methods for linear and nonlinear equations. *Frontiers in Applied Mathematics Series*, vol. 16. Society for Industrial and Applied Mathematics: Philadelphia, PA, 1995.

Copyright © 2005 John Wiley & Sons, Ltd.