Numerical performance of compact fourth-order formulation of the Navier–Stokes equations

Ercan Erturk[1, 2, *, †]

[1]*Energy Systems Engineering Department, Gebze Institute of Technology, Gebze, Kocaeli 41400, Turkey*
[2]*Energy Systems Engineering Department, Bahcesehir University, Besiktas, Istanbul 34353, Turkey*[‡]

SUMMARY

In this study, the numerical performance of the fourth-order compact formulation of the steady 2-D incompressible Navier–Stokes (NS) equations introduced by Erturk *et al.* (*Int. J. Numer. Methods Fluids* 2006; **50**:421–436) will be presented. The benchmark-driven cavity flow problem will be solved using the introduced compact fourth-order formulation of the NS equations with two different line iterative semi-implicit methods for both second- and fourth-order spatial accuracy. The extra CPU work needed for increasing the spatial accuracy from second-order ($\mathcal{O}(\Delta x^2)$) formulation to fourth-order ($\mathcal{O}(\Delta x^4)$) formulation will be presented. Copyright © 2008 John Wiley & Sons, Ltd.

1. INTRODUCTION

In computational fluid dynamics field of study, high-order compact formulations are becoming more popular. Compact formulations provide more accurate solutions in a compact stencil.

In finite difference, in order to achieve fourth-order spatial accuracy, standard five-point discretization can be used. When a five-point discretization is used, the points near the boundaries have to be treated specially. Another way to achieve fourth-order spatial accuracy is to use high-order compact schemes. High-order compact schemes provide fourth-order spatial accuracy in a 3×3 stencil; hence this type of formulation can be used near the boundaries without complexity.

In the literature, Zhang [1], Dennis and Hudson [2], MacKinnon and Johnson [3], Gupta *et al.* [4], Spotz and Carey [5] and Li *et al.* [6] have demonstrated the efficiency of high-order

---

*Correspondence to: Ercan Erturk, Energy Systems Engineering Department, Gebze Institute of Technology, Gebze, Kocaeli 41400, Turkey.
†E-mail: ercanerturk@gyte.edu.tr, http://www.cavityflow.com
‡Partially affiliated with.

compact schemes on the streamfunction and vorticity formulation of 2-D steady incompressible Navier–Stokes (NS) equations for uniform grids. Also in the literature the studies of Ge and Zhang [7] and Spotz and Carey [8] are example studies on the application of the high-order compact scheme to nonuniform grids. The advantage of the high-order compact schemes is that for a given flow problem and for a chosen grid mesh, high-order compact formulation provides more accurate solutions ($\mathcal{O}(\Delta x^4)$) compared with standard second-order formulation ($\mathcal{O}(\Delta x^2)$). Also for a given flow problem, the same level of accuracy of the solution obtained by standard second-order formulation ($\mathcal{O}(\Delta x^2)$) using a certain grid mesh can be obtained with a smaller grid mesh when high-order compact ($\mathcal{O}(\Delta x^4)$) formulation is used.

Recently, Erturk and Gokcol [9] have presented a new fourth-order compact formulation. The uniqueness of this formulation is that the presented 'compact fourth-order formulation of the Navier–Stokes' (FONS) equations are in the same form as the NS equations with additional coefficients. In fact, the NS equations are a subset of the FONS equations and obtained when the additional coefficients are chosen as zero. Therefore, any numerical method that solves the NS equations can be easily applied to the introduced FONS equations to obtain fourth-order spatial accurate solutions. Moreover, the most important feature of the FONS equations is that any existing code with second-order accuracy ($\mathcal{O}(\Delta x^2)$) can easily be changed to provide fourth-order accuracy ($\mathcal{O}(\Delta x^4)$) by just adding some coefficients into the existing code at the expense of extra CPU work of evaluating these coefficients. This is an important feature because if one already has a second-order accurate code and wants to increase the accuracy of it to fourth order, instead of writing a new code, one can use the FONS equations and just by inserting some coefficient into the existing second-order code, the existing second-order code can turn into a fourth-order code. Therefore, the FONS equations introduced by Erturk and Gokcol [9] provide a very easy way to convert an existing second-order accurate code into a fourth-order accurate code and this is as simple as inserting some coefficients into the existing code. Of course, when this is done, the code will run a little slower because of the extra CPU work of evaluating the inserted coefficients. It will be good to estimate the CPU time needed for convergence of a converted fourth-order accurate code compared with the CPU time needed for a second-order accurate code.

Zhang [10] has studied the convergence and performance of iterative methods with fourth-order compact discretization schemes. To the best of the author's knowledge, in the literature there is no study that documents the numerical performance of high-order compact formulation of the NS equations compared to regular second-order formulation of the NS equations in terms of numerical stability and convergence for a chosen iterative method. In this study using the FONS equations introduced by Erturk and Gokcol [9], we will numerically solve the NS equations for both fourth-order ($\mathcal{O}(\Delta x^4)$) and second-order ($\mathcal{O}(\Delta x^2)$) spatial accuracy. This way we will be able to compare the convergence and stability characteristics of both formulations. In this study we will also document the extra CPU work that is needed for convergence when a second-order accurate code is converted into a fourth-order accurate code using the introduced FONS equations by Erturk and Gokcol [9]. The stability and convergence characteristics of both formulations and also the extra CPU work can show variation depending on the iterative numerical method used for the solution; therefore, in this study we will use two different line iterative semi-implicit numerical methods. Using these two numerical methods we will solve the benchmark-driven cavity flow problem. First, we will solve the cavity flow with second-order ($\mathcal{O}(\Delta x^2)$) spatial accuracy; then we will solve the same flow with fourth-order ($\mathcal{O}(\Delta x^4)$) spatial accuracy. We will document the stability characteristics such as the maximum allowable time increment

(Δt), convergence characteristics such as the number of iterations and the CPU time necessary for a chosen convergence criteria and also the extra CPU work that is needed to increase the spatial accuracy of the numerical solution from second order to fourth order using the FONS equations.

2. FOURTH-ORDER COMPACT FORMULATION

In non-dimensional form, steady 2-D incompressible NS equations in streamfunction (ψ) and vorticity (ω) formulation are given as

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega \tag{1}$$

$$\frac{1}{Re}\frac{\partial^2 \omega}{\partial x^2} + \frac{1}{Re}\frac{\partial^2 \omega}{\partial y^2} = \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} \tag{2}$$

where x and y are the Cartesian coordinates and Re is the Reynolds number.

Erturk and Gokcol [9] have introduced the FONS equations. The introduced FONS equations are the following:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega + A \tag{3}$$

$$\frac{1}{Re}(1+B)\frac{\partial^2 \omega}{\partial x^2} + \frac{1}{Re}(1+C)\frac{\partial^2 \omega}{\partial y^2} = \left(\frac{\partial \psi}{\partial y} + D\right)\frac{\partial \omega}{\partial x} - \left(\frac{\partial \psi}{\partial x} + E\right)\frac{\partial \omega}{\partial y} + F \tag{4}$$

where

$$A = -\frac{\Delta x^2}{12}\frac{\partial^2 \omega}{\partial x^2} - \frac{\Delta y^2}{12}\frac{\partial^2 \omega}{\partial y^2} - \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial^4 \psi}{\partial x^2 \partial y^2}$$

$$B = -Re\frac{\Delta x^2}{6}\frac{\partial^2 \psi}{\partial x \partial y} + Re^2\frac{\Delta x^2}{12}\frac{\partial \psi}{\partial y}\frac{\partial \psi}{\partial y}$$

$$C = Re\frac{\Delta y^2}{6}\frac{\partial^2 \psi}{\partial x \partial y} + Re^2\frac{\Delta y^2}{12}\frac{\partial \psi}{\partial x}\frac{\partial \psi}{\partial x} \tag{5}$$

$$D = \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial^3 \psi}{\partial x^2 \partial y} - Re\frac{\Delta x^2}{12}\frac{\partial \psi}{\partial y}\frac{\partial^2 \psi}{\partial x \partial y} + Re\frac{\Delta y^2}{12}\frac{\partial \psi}{\partial x}\frac{\partial^2 \psi}{\partial y^2}$$

$$E = \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial^3 \psi}{\partial x \partial y^2} - Re\frac{\Delta x^2}{12}\frac{\partial \psi}{\partial y}\frac{\partial^2 \psi}{\partial x^2} + Re\frac{\Delta y^2}{12}\frac{\partial \psi}{\partial x}\frac{\partial^2 \psi}{\partial x \partial y}$$

$$F = \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial \psi}{\partial y}\frac{\partial^3 \omega}{\partial x \partial y^2} - \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial \psi}{\partial x}\frac{\partial^3 \omega}{\partial x^2 \partial y} - \frac{\Delta x^2}{6}\frac{\partial^2 \psi}{\partial x^2}\frac{\partial^2 \omega}{\partial x \partial y}$$

$$+ \frac{\Delta y^2}{6}\frac{\partial^2 \psi}{\partial y^2}\frac{\partial^2 \omega}{\partial x \partial y} + Re\left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial \psi}{\partial x}\frac{\partial \psi}{\partial y}\frac{\partial^2 \omega}{\partial x \partial y} - \left(\frac{\Delta x^2}{12} - \frac{\Delta y^2}{12}\right)\frac{\partial \omega}{\partial x}\frac{\partial \omega}{\partial y}$$

$$- \frac{1}{Re}\left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\frac{\partial^4 \omega}{\partial x^2 \partial y^2}$$

As described briefly in Erturk and Gokcol [9], the numerical solutions of FONS equations (3) and (4) are fourth-order accurate to NS equations (1) and (2), *strictly* provided that second-order central discretizations shown in Table I are used and also *strictly* provided that a uniform grid mesh with Δx and Δy is used. We note that NS equations are a subset of FONS equations and obtained when the coefficients A, B, C, D, E and F in FONS equations are chosen as zero. The FONS equations are in the same form as the NS equations; therefore, any iterative numerical method applied to streamfunction and vorticity equation (1) and (2) can be easily applied to fourth-order streamfunction and vorticity equation (3) and (4). Moreover, if there is an existing code that solves the streamfunction and vorticity equation (1) and (2) with second-order spatial accuracy, by just adding some coefficients A, B, C, D, E and F into the existing code using the FONS equations, the same existing code can provide fourth-order spatial accuracy. A single numerical code for the solution; of the FONS equations can provide both second-order and fourth-order spatial accuracy by just setting some coefficients. Of course there is a payoff while switching from second-order to fourth-order spatial accuracy, that is, the extra cost of CPU work of calculating the coefficients A, B, C, D, E and F as defined in Equation (5).

Table I. Standard second-order central discretizations, $\mathcal{O}(\Delta x^2, \Delta y^2)$.

ϕ_x	$\dfrac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}$
ϕ_y	$\dfrac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}$
ϕ_{xx}	$\dfrac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2}$
ϕ_{yy}	$\dfrac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2}$
ϕ_{xy}	$\dfrac{\phi_{i+1,j+1} - \phi_{i-1,j+1} - \phi_{i+1,j-1} + \phi_{i-1,j-1}}{4\Delta x \Delta y}$
ϕ_{xxy}	$\dfrac{\phi_{i+1,j+1} - 2\phi_{i,j+1} + \phi_{i-1,j+1} - \phi_{i+1,j-1} + 2\phi_{i,j-1} - \phi_{i-1,j-1}}{2\Delta x^2 \Delta y}$
ϕ_{xyy}	$\dfrac{\phi_{i+1,j+1} - 2\phi_{i+1,j} + \phi_{i+1,j-1} - \phi_{i-1,j+1} + 2\phi_{i-1,j} - \phi_{i-1,j-1}}{2\Delta x \Delta y^2}$
ϕ_{xxyy}	$\dfrac{\phi_{i+1,j+1} - 2\phi_{i,j+1} + \phi_{i-1,j+1} - 2\phi_{i+1,j} + 4\phi_{i,j} - 2\phi_{i-1,j} + \phi_{i+1,j-1} - 2\phi_{i,j-1} + \phi_{i-1,j-1}}{\Delta x^2 \Delta y^2}$

3. FINITE DIFFERENCE EQUATIONS

For numerical solutions of the NS equations (1) and (2), the following finite difference equations provide second-order ($\mathcal{O}(\Delta x^2)$) accuracy:

$$\psi_{xx} + \psi_{yy} = -\omega \tag{6}$$

$$\frac{1}{Re}\omega_{xx} + \frac{1}{Re}\omega_{yy} = \psi_y\omega_x - \psi_x\omega_y \tag{7}$$

where subscripts denote derivatives as defined in Table I.

As explained in Erturk and Gokcol [9] the solution of the following finite difference equations are fourth-order ($\mathcal{O}(\Delta x^4)$) accurate to NS:

$$\psi_{xx} + \psi_{yy} = -\omega + A \tag{8}$$

$$\frac{1}{Re}(1+B)\omega_{xx} + \frac{1}{Re}(1+C)\omega_{yy} = (\psi_y + D)\omega_x - (\psi_x + E)\omega_y + F \tag{9}$$

where

$$
\begin{aligned}
A &= -\frac{\Delta x^2}{12}\omega_{xx} - \frac{\Delta y^2}{12}\omega_{yy} - \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\psi_{xxyy} \\[4pt]
B &= -Re\frac{\Delta x^2}{6}\psi_{xy} + Re^2\frac{\Delta x^2}{12}\psi_y\psi_y \\[4pt]
C &= Re\frac{\Delta y^2}{6}\psi_{xy} + Re^2\frac{\Delta y^2}{12}\psi_x\psi_x \\[4pt]
D &= \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\psi_{xxy} - Re\frac{\Delta x^2}{12}\psi_y\psi_{xy} + Re\frac{\Delta y^2}{12}\psi_x\psi_{yy} \\[4pt]
E &= \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\psi_{xyy} - Re\frac{\Delta x^2}{12}\psi_y\psi_{xx} + Re\frac{\Delta y^2}{12}\psi_x\psi_{xy} \\[4pt]
F &= \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\psi_y\omega_{xyy} - \left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\psi_x\omega_{xxy} - \frac{\Delta x^2}{6}\psi_{xx}\omega_{xy} \\[4pt]
&\quad + \frac{\Delta y^2}{6}\psi_{yy}\omega_{xy} + Re\left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\psi_x\psi_y\omega_{xy} - \left(\frac{\Delta x^2}{12} - \frac{\Delta y^2}{12}\right)\omega_x\omega_y \\[4pt]
&\quad - \frac{1}{Re}\left(\frac{\Delta x^2}{12} + \frac{\Delta y^2}{12}\right)\omega_{xxyy}
\end{aligned}
\tag{10}
$$

Note that Equations (8) and (9) are in the same form as Equations (6) and (7) except with additional coefficients A, B, C, D, E and F. In Equations (8) and (9) if the coefficients A, B, C, D, E and F are chosen to be zero then the solution of these equations are second-order accurate ($\mathcal{O}(\Delta x^2, \Delta y^2)$) to NS equations, since when the coefficients are zero, Equations (8) and (9) are identical with Equations (6) and (7). However, in Equations (8) and (9) if the coefficients A, B, C, D, E and F are calculated as they are defined in Equation (10), then the solution of these Equations (8) and (9) are fourth-order accurate ($\mathcal{O}(\Delta x^4, \Delta x^2\Delta y^2, \Delta y^4)$) to NS equations. When FONS equations are

used, one can easily switch from second-order to fourth-order spatial accuracy using a single equation by just using the appropriate coefficients. Computationally, calculating the coefficients defined in Equation (10) when fourth-order accuracy is desired will require extra CPU work compared with second-order accuracy. In order to quantify the extra CPU work to switch from Equations (6) and (7) to Equations (8) and (9), i.e. switch from second-order accuracy to fourth-order accuracy, we solve the above equations with two different line iterative semi-implicit numerical methods and document the CPU time for comparison.

We note that Equations (8) and (9) are nonlinear equations; therefore, they need to be solved in an iterative manner. In order to have an iterative numerical algorithm we assign pseudo-time derivatives to Equations (8) and (9); thus we have

$$\frac{\partial \psi}{\partial t} = \psi_{xx} + \psi_{yy} + \omega - A \tag{11}$$

$$\frac{\partial \omega}{\partial t} = \frac{1}{Re}(1+B)\omega_{xx} + \frac{1}{Re}(1+C)\omega_{yy} - (\psi_y + D)\omega_x + (\psi_x + E)\omega_y - F \tag{12}$$

We solve Equations (11) and (12) in the pseudo-time domain until the solution converges to steady state.

One of the numerical methods we will use to solve Equations (11) and (12) is the alternating direction implicit (ADI) method. The ADI method is a very widely used numerical method and in this method a 2-D problem is solved in two sweeps while solving the equation implicitly in 1-D in each sweep. The reader is referred to [11–13] for details. When we apply the ADI method to solve Equation (11), first we solve the following tri-diagonal system in the x-direction:

$$\left(1 - \frac{\Delta t}{2}\delta_{xx}\right)\psi^{n+1/2} = \psi^n + \frac{\Delta t}{2}\psi_{yy}^n + \frac{\Delta t}{2}\omega - \frac{\Delta t}{2}A \tag{13}$$

Then we solve the following tri-diagonal system in the y-direction:

$$\left(1 - \frac{\Delta t}{2}\delta_{yy}\right)\psi^{n+1} = \psi^{n+1/2} + \frac{\Delta t}{2}\psi_{xx}^{n+1/2} + \frac{\Delta t}{2}\omega - \frac{\Delta t}{2}A \tag{14}$$

Similarly, when we apply the ADI method to solve Equation (12), we first solve the following tri-diagonal system in the x-direction:

$$\left(1 - \frac{\Delta t}{2}\frac{1}{Re}(1+B)\delta_{xx} + \frac{\Delta t}{2}(\psi_y + D)\delta_x\right)\omega^{n+1/2}$$

$$= \omega^n + \frac{\Delta t}{2}(1+C)\frac{1}{Re}\omega_{yy}^n + \frac{\Delta t}{2}(\psi_x + E)\omega_y^n - \frac{\Delta t}{2}F \tag{15}$$

Then we solve the following tri-diagonal system in the y-direction:

$$\left(1 - \frac{\Delta t}{2}\frac{1}{Re}(1+C)\delta_{yy} - \frac{\Delta t}{2}(\psi_x + E)\delta_y\right)\omega^{n+1}$$

$$= \omega^{n+1/2} + \frac{\Delta t}{2}\frac{1}{Re}(1+B)\omega_{xx}^{n+1/2} - \frac{\Delta t}{2}(\psi_y + D)\omega_x^{n+1/2} - \frac{\Delta t}{2}F \tag{16}$$

where δ_{xx} and δ_{yy} denote the second-order finite difference operators, and similarly δ_x and δ_y denote the first-order finite difference operators in the x- and y-directions, respectively, for example

$$\delta_x \theta = \frac{\theta_{i+1,j} - \theta_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x^2)$$
$$\delta_{xx} \theta = \frac{\theta_{i+1,j} - 2\theta_{i,j} + \theta_{i-1,j}}{\Delta x^2} + \mathcal{O}(\Delta x^2) \tag{17}$$

where i and j are the grid indices and θ denote any differentiable quantity.

The second numerical method we will use is the efficient numerical method proposed by Erturk *et al.* [14]. Following Erturk *et al.* [14], first using an implicit Euler approximation for the pseudo-time derivatives in Equations (11) and (12), we obtain the following finite difference formulations:

$$\left(1 - \Delta t \delta_{xx} - \Delta t \delta_{yy}\right)\psi^{n+1} = \psi^n + \Delta t \omega^n - \Delta t A^n \tag{18}$$

$$\left(1 - \Delta t \frac{1}{Re}(1+B)\delta_{xx} - \Delta t \frac{1}{Re}(1+C)\delta_{yy} + \Delta t(\psi_y + D)\delta_x - \Delta t(\psi_x + E)\delta_y\right)\omega^{n+1}$$
$$= \omega^n - \Delta t F^n \tag{19}$$

We note that Equations (18) and (19) are in fully implicit form and each equation requires the solution of a large banded matrix, which is not computationally efficient. Instead, we spatially factorize these Equations (18) and (19); thus, we obtain the following finite difference equations:

$$(1 - \Delta t \delta_{xx})\left(1 - \Delta t \delta_{yy}\right)\psi^{n+1} = \psi^n + \Delta t \omega^n - \Delta t A^n \tag{20}$$

$$\left(1 - \Delta t \frac{1}{Re}(1+B)\delta_{xx} + \Delta t(\psi_y + D)\delta_x\right)\left(1 - \Delta t \frac{1}{Re}(1+C)\delta_{yy} - \Delta t(\psi_x + E)\delta_y\right)\omega^{n+1}$$
$$= \omega^n - \Delta t F^n \tag{21}$$

The advantage of Equations (20) and (21) is that each equation requires the solution of tridiagonal systems, which is computationally very efficient using the Thomas algorithm. However, spatial factorization introduces Δt^2 terms into the left-hand side (LHS) of Equations (20) and (21); these terms remain in the solution even at the steady state. To cancel out these Δt^2 terms due to the factorization, Erturk *et al.* [14] have added the same amount of Δt^2 terms to the right-hand side (RHS) of the equations so that the equations recover the correct physical representation at the steady state. The final form of the finite difference equations takes the following form:

$$(1 - \Delta t \delta_{xx})\left(1 - \Delta t \delta_{yy}\right)\psi^{n+1} = \psi^n + \Delta t \omega^n - \Delta t A^n + (\Delta t \delta_{xx})\left(\Delta t \delta_{yy}\right)\psi^n \tag{22}$$

$$\left(1 - \Delta t \frac{1}{Re}(1+B)\delta_{xx} + \Delta t(\psi_y + D)\delta_x\right)\left(1 - \Delta t \frac{1}{Re}(1+C)\delta_{yy} - \Delta t(\psi_x + E)\delta_y\right)\omega^{n+1}$$

$$= \omega^n - \Delta t F^n \left(\Delta t \frac{1}{Re}(1+B)\delta_{xx} - \Delta t(\psi_y + D)\delta_x\right)$$

$$\times \left(\Delta t \frac{1}{Re}(1+C)\delta_{yy} + \Delta t(\psi_x + E)\delta_y\right)\omega^n \tag{23}$$

The reader is referred to Erturk *et al.* [14] for details. The solution methodology for Equations (22) and (23) involves a two-stage time-level updating. For example, for the solution of Equation (22), we first solve for the introduced variable f in the x-direction in the following tri-diagonal system:

$$(1 - \Delta t \delta_{xx}) f = \psi^n + \Delta t \omega^n - \Delta t A^n + (\Delta t \delta_{xx}) (\Delta t \delta_{yy}) \psi^n \tag{24}$$

When the solution for f is obtained, the streamfunction variable is advanced into the next time level by solving the following tri-diagonal system in the y-direction:

$$(1 - \Delta t \delta_{yy}) \psi^{n+1} = f \tag{25}$$

Similarly, for the solution of Equation (23), we first solve for the introduced variable g in the x-direction in the following tri-diagonal system:

$$\left(1 - \Delta t \frac{1}{Re}(1+B)\delta_{xx} + \Delta t (\psi_y + D)\delta_x\right) g = \omega^n - \Delta t F^n \left(\Delta t \frac{1}{Re}(1+B)\delta_{xx} - \Delta t (\psi_y + D)\delta_x\right)$$
$$\times \left(\Delta t \frac{1}{Re}(1+C)\delta_{yy} + \Delta t (\psi_x + E)\delta_y\right) \omega^n \tag{26}$$

When the solution for g is obtained, the vorticity variable is advanced into the next time level by solving the following tri-diagonal system in the y-direction:

$$\left(1 - \Delta t \frac{1}{Re}(1+C)\delta_{yy} - \Delta t (\psi_x + E)\delta_y\right) \omega^{n+1} = g \tag{27}$$

Störtkuhl *et al.* [15] have presented an analytical asymptotic solution near the corners of cavity and using finite element bilinear shape functions they also have presented a singularity-removed boundary condition for vorticity at the corner points as well as at the wall points. For the boundary conditions, in both of the numerical methods described above we follow Störtkuhl *et al.* [15] and use the following expression for calculating vorticity values at the wall:

$$\frac{1}{3\Delta h^2} \begin{bmatrix} \cdot & \cdot & \cdot \\ \frac{1}{2} & -4 & \frac{1}{2} \\ 1 & 1 & 1 \end{bmatrix} \psi + \frac{1}{9} \begin{bmatrix} \cdot & \cdot & \cdot \\ \frac{1}{2} & 2 & \frac{1}{2} \\ \frac{1}{4} & 1 & \frac{1}{4} \end{bmatrix} \omega = -\frac{V}{\Delta h} \tag{28}$$

where V is the speed of the wall which is equal to 1 for the moving top wall and equal to 0 for the three stationary walls.

For corner points, we again follow Störtkuhl *et al.* [15] and use the following expression for calculating the vorticity values:

$$\frac{1}{3\Delta h^2} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & -2 & \frac{1}{2} \\ \cdot & \frac{1}{2} & 1 \end{bmatrix} \psi + \frac{1}{9} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & 1 & \frac{1}{2} \\ \cdot & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \omega = -\frac{V}{2\Delta h} \tag{29}$$
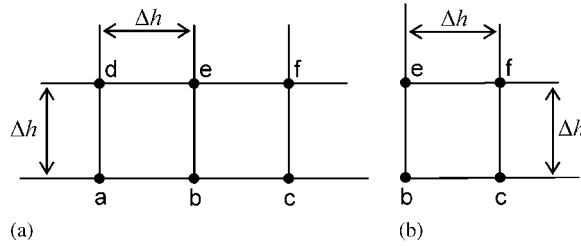
Figure 1. Grid points at the wall and at the corner: (a) wall points and (b) corner points.

where again V is equal to 1 for the upper two corners and it is equal to 0 for the bottom two corners.

In explicit notation, for the wall points shown in Figure 1, the vorticity is calculated as the following:

$$\omega_{\mathrm{b}} = -\frac{9V}{2\Delta h} - \frac{3}{2\Delta h^2}(\psi_{\mathrm{d}} + \psi_{\mathrm{e}} + \psi_{\mathrm{f}}) - \frac{1}{8}(2\omega_{\mathrm{a}} + 2\omega_{\mathrm{c}} + \omega_{\mathrm{d}} + 4\omega_{\mathrm{e}} + \omega_{\mathrm{f}}) \qquad (30)$$

Similarly, for the corner points also shown in Figure 1, the vorticity is calculated as the following:

$$\omega_{\mathrm{b}} = -\frac{9V}{2\Delta h} - \frac{3}{\Delta h^2}\psi_{\mathrm{f}} - \frac{1}{4}(2\omega_{\mathrm{c}} + \omega_{\mathrm{f}} + 2\omega_{\mathrm{e}}) \qquad (31)$$

The reader is referred to Störtkuhl *et al*. [15] for details on the boundary conditions.

4. RESULTS

In order to quantify the extra CPU work needed when a second-order accuracy code is converted into a fourth-order accuracy code using the FONS equations introduced by Erturk and Gokcol [9], we have solved both second-order and fourth-order accurate equations (6)–(9) for the solution of the driven cavity flow. We consider the driven cavity flow for Reynolds numbers of $Re = 100, 1000$ and 3200, using a grid mesh of 128×128 ($\Delta x = \Delta y = \Delta h$). We note that since we are mainly interested in finding the ratio of CPU time needed for convergence of a fourth-order accuracy code to CPU time needed for convergence of a second-order accuracy code, the choice of grid mesh size is not important as the ratio will be the same whether a coarse or a fine grid mesh is used.

In solving the equations we decided to use the ADI method and the numerical method proposed by Erturk *et al*. [14]. By using two different numerical methods we would be able to see if the extra CPU work is dependent on the numerical method used. While doing this, as a corollary, we would also be able to compare the ADI method and the method proposed by Erturk *et al*. [14] in terms of numerical performance. In both of the numerical methods we use, for both second-order and fourth-order accuracy, the two equations, i.e. the streamfunction and the vorticity equations, are solved separately. In order to document the extra CPU work when a fourth-order accuracy is desired, what we do is, we first solve for second-order accuracy and solve Equations (6) and (7). Then keeping the number of grids, the time step Δt and boundary conditions the same, we

solve for fourth-order accuracy; thus we calculate and insert the coefficients A, B, C, D, E and F into the equations and solve for Equations (8) and (9). While we solve the same flow problem, i.e. the driven cavity flow, for both second- and fourth-order accuracy we document the necessary number of iterations and the CPU time needed for a certain defined convergence criterion. This way we would be able to compare the convergence characteristics of both second- and fourth-order formulations in terms of the number of iterations and the CPU time, and we would also be able to document the extra CPU time needed if a second-order code is converted into a fourth-order code using the FONS equations.

For the choice of the time steps in solving the governing equations, we decided to use different time steps, Δt, for streamfunction and vorticity equations. In both of the numerical methods we use, while solving both the streamfunction and vorticity equations, tri-diagonal matrices appear on the implicit LHS of the equations. When second-order accuracy is considered, in streamfunction equation the diagonal elements on the LHS matrices become $1 + (2\Delta t/\Delta h^2)$, also in vorticity equations the diagonal elements on the LHS matrices become $1 + (1/Re)(2\Delta t/\Delta h^2)$. We choose different time steps for streamfunction and vorticity equations, which would make the diagonal elements the same in both equations. Therefore, for streamfunction equation we use $\Delta t = \alpha \Delta h^2$ and for vorticity equation we use $\Delta t = \alpha Re \Delta h^2$, where α is a coefficient we can choose. For fourth-order accuracy we use the same time steps we use in second-order accuracy. By using the same time steps in second- and fourth-order accuracy we would be able to compare the numerical stability of the FONS equations and the NS equations. In order to do this first we solve both the NS and the FONS equations using the same time steps. Then we increase α, i.e. increase the time step Δt, and solve the NS and the FONS equations again. We continue doing this until at some Δt the solution does not converge. Therefore, we would document the maximum allowable Δt for convergence for both the NS and the FONS equations for a given Reynolds number and grid mesh. This maximum allowable Δt for convergence is indicative of the numerical stability. For example, using either numerical method, i.e. the ADI method or the Erturk method, we solve the same flow problem using both second- and fourth-order formulations. Therefore, for a chosen numerical method, the maximum allowable time step for second- and fourth-order formulations will be indicative of the numerical stability characteristics of the second-order formulation compared with that of the fourth-order formulation. Also, using either of the formulations, i.e. second- or fourth-order formulations, we solve the same flow problem using both the ADI method and the Erturk method. Therefore, for a chosen formulation, the maximum allowable time step for the ADI and the Erturk methods will be indicative of the numerical stability characteristics of the ADI method compared with that of the Erturk method.

Our extensive numerical studies show that the increase in the extra CPU work is dependent on the computer and the compiler used. In this study, we run the codes on a 64 bit HP ES45 machine with EV68 AlphaChip 1.25 GHz processors with HP Tru64 UNIX operating system. We run the codes with both compiled normally and also compiled using maximum compiler optimization (`-fast -O5`).

We start the iterations from a homogenous initial guess and continue until a certain condition of convergence is satisfied. As the measure of convergence to the same level, the residual of the equations can be used as it was also used by Erturk *et al.* [14]. However, we are solving two different equations, the NS and the FONS equations, and trying to compare the CPU time of convergence for each equation with the same level. Therefore, the residual of these equations may not show the same convergence level. Alternatively, we can use the difference of the streamfunction and vorticity variables between two time steps as the measure of convergence. However, the solutions

of the two different equations are slightly different since one is spatially second-order and the other is fourth-order accurate. Since the solutions are different, the difference of the streamfunction and vorticity variables between two time steps may not also show the same convergence level for those equations. Therefore, as convergence criteria we decided to use the difference of the streamfunction and vorticity variables between two time steps normalized by the previous value of the corresponding variable, such that

$$\text{Residual}_\psi = \max\left(\left|\frac{\psi_{i,j}^{n+1} - \psi_{i,j}^n}{\psi_{i,j}^n}\right|\right)$$

$$\text{Residual}_\omega = \max\left(\left|\frac{\omega_{i,j}^{n+1} - \omega_{i,j}^n}{\omega_{i,j}^n}\right|\right)$$

(32)

These residuals provide an indication of the maximum percent change in ψ and ω variables in each iteration step. In all of the data presented in this study, for both the solutions of the NS and the FONS equations, obtained using both the numerical methods, we let the iterations converge until both Residual$_\psi$ and Residual$_\omega$ are less than 10^{-8}. At this convergence level, this would indicate that the variables ψ and ω are changing less than 0.000001% of their value between two iterations at every grid point in the mesh.

Figures 2–4 show the streamline and vorticity contours of the driven cavity flow for $Re = 100, 1000$ and 3200, respectively, obtained using the method proposed by Erturk *et al.* [14] applied to FONS equations $(\mathcal{O}(\Delta x^4))$. We note that both second-order and fourth-order accurate solutions of the ADI method and the Erturk method agree well with the solutions found in the literature especially with Erturk *et al.* [9].

Using both of the numerical methods, we solve the driven cavity flow using different coefficients for time (α), i.e. using different time steps, and document the CPU time and iteration number needed for the desired convergence level explained above. Table II shows the CPU time and
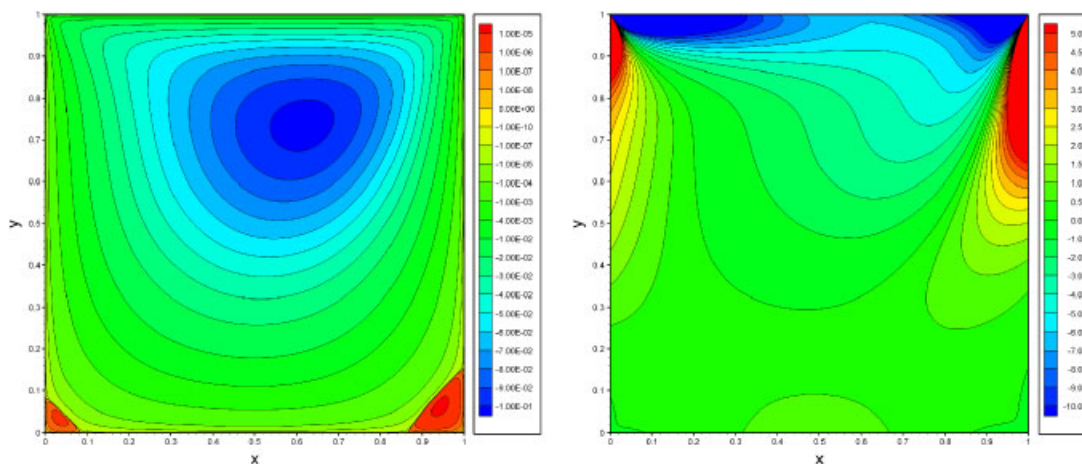

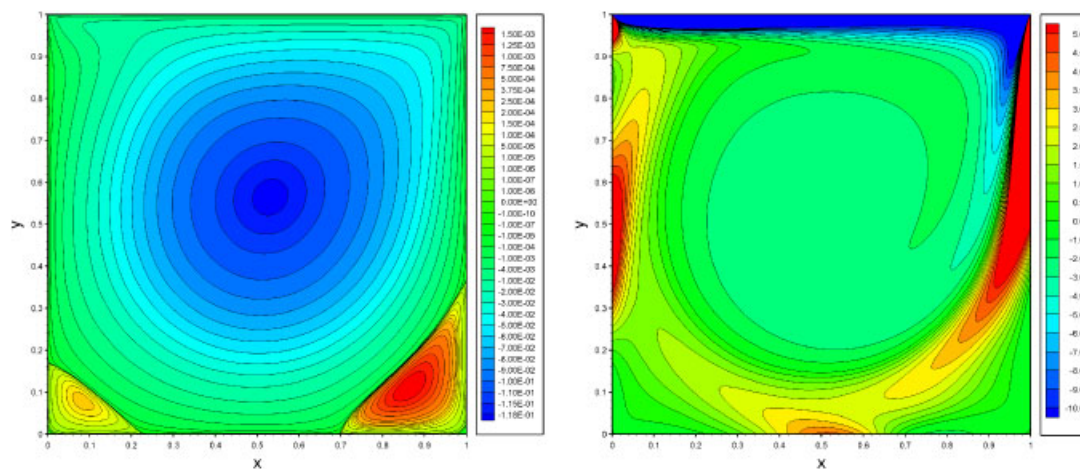Figure 2. Streamline and vorticity contours of driven cavity flow, $Re = 100$.

Figure 3. Streamline and vorticity contours of driven cavity flow, $Re = 1000$.
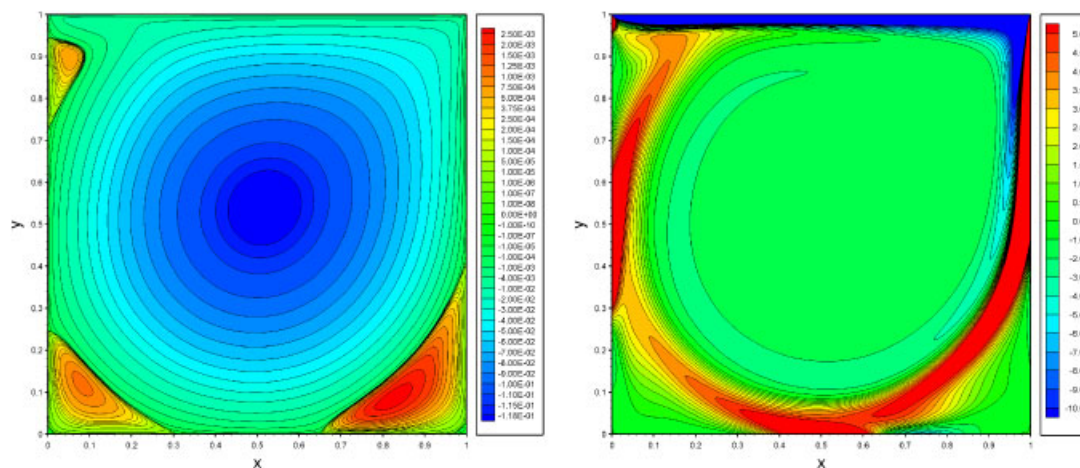
Figure 4. Streamline and vorticity contours of driven cavity flow, $Re = 3200$.

iteration numbers for the ADI method for different Reynolds numbers using various α values. Table III shows the same for the method proposed by Erturk *et al.* [14]. Looking at Tables II and III, for both of the numerical methods the number of iterations for convergence is almost the same for second-order and fourth-order accuracy. However, for both of the numerical methods the CPU time for fourth-order accuracy is greater than the CPU time for second-order accuracy as expected since the coefficients A, B, C, D, E and F have to be calculated at each iteration in fourth-order accuracy, which will result in an increase in the CPU time. The ratio of the CPU times of fourth-order accuracy to second-order accuracy shows the increase in CPU time when we switch from second-order accuracy to fourth-order accuracy. From Tables II and III we see that this ratio seems to increase slightly when Reynolds number increases.

Table II. Comparison of the CPU time and iteration number for second- and fourth-order accuracy obtained using the ADI method.

| Re | α | Second-order accuracy | | Fourth-order accuracy | | $\dfrac{\text{CPU}_{(\mathscr{O}\Delta x^4)}}{\text{CPU}_{(\mathscr{O}\Delta x^2)}}$ | $\dfrac{\text{Iteration no.}_{(\mathscr{O}\Delta x^4)}}{\text{Iteration no.}_{(\mathscr{O}\Delta x^2)}}$ |
		$\text{CPU}_{(\mathscr{O}\Delta x^2)}$	Iteration no.$_{(\mathscr{O}\Delta x^2)}$	$\text{CPU}_{(\mathscr{O}\Delta x^4)}$	Iteration no.$_{(\mathscr{O}\Delta x^4)}$		
100	0.6	182.57	25 393	321.83	25 091	1.76	0.99
	0.4	271.36	37 739	478.90	37 362	1.76	0.99
	0.2	531.62	73 932	942.35	73 506	1.77	0.99
1000	0.79	184.60	25 677	—	—	—	—
	0.78	187.07	26 013	336.72	26 260	1.80	1.01
	0.7	207.94	28 949	374.82	29 215	1.80	1.01
	0.6	242.14	33 682	435.98	34 007	1.80	1.01
	0.5	289.38	40 278	521.84	40 688	1.80	1.01
	0.4	359.96	50 116	649.25	50 662	1.80	1.01
	0.2	707.90	98 518	1279.43	99 831	1.81	1.01
	0.1	1384.97	192 629	2509.86	195 820	1.81	1.02
3200	0.1	1261.09	175 558	2398.06	187 023	1.90	1.07

Table III. Comparison of the CPU time and iteration number for second- and fourth-order accuracy obtained using the Erturk method.

| Re | α | Second-order accuracy | | Fourth-order accuracy | | $\dfrac{\text{CPU}_{(\mathscr{O}\Delta x^4)}}{\text{CPU}_{(\mathscr{O}\Delta x^2)}}$ | $\dfrac{\text{Iteration no.}_{(\mathscr{O}\Delta x^4)}}{\text{Iteration no.}_{(\mathscr{O}\Delta x^2)}}$ |
		$\text{CPU}_{(\mathscr{O}\Delta x^2)}$	Iteration no.$_{(\mathscr{O}\Delta x^2)}$	$\text{CPU}_{(\mathscr{O}\Delta x^4)}$	Iteration no.$_{(\mathscr{O}\Delta x^4)}$		
100	0.6	228.76	25 419	367.15	25 115	1.60	0.99
	0.4	339.95	37 763	545.53	37 386	1.60	0.99
	0.2	666.38	73 953	1072.67	73 528	1.61	0.99
1000	1.89	98.85	10 963	—	—	—	—
	1.75	106.45	11 820	176.40	12 082	1.66	1.02
	1.6	115.97	12 888	192.65	13 191	1.66	1.02
	1.2	153.90	17 091	254.16	17 406	1.65	1.02
	0.6	303.67	33 734	497.10	34 067	1.64	1.01
	0.4	451.49	50 165	741.81	50 717	1.64	1.01
	0.2	888.35	98 564	1457.44	99 884	1.64	1.01
	0.1	1733.51	192 673	2854.97	195 871	1.65	1.02
3200	1.2	155.51	17 290	261.89	17 986	1.68	1.04

It seems that for both of the numerical methods, at a given Reynolds number, the ratios of CPU time and iteration number for second- and fourth-order accuracy are constant and independent of the time step, i.e. α.

For the ADI method, in Table II, when the order of accuracy is increased to fourth order from second order, the CPU time increases almost 1.76 times for $Re = 100$. This number increases as the Reynolds number increases and the increase in CPU time becomes 1.90 times for $Re = 3200$.

For the method proposed by Erturk *et al.* [14], in Table III, when the order of accuracy is increased to fourth order from second order, the CPU time increases almost 1.60 times for $Re=100$ and it is almost 1.68 times for $Re=3200$.

We note that when the order of accuracy is increased from second order to fourth order, the 1.6 and 1.68 times increase in CPU time for $Re=100$ and 3200, respectively, for the method proposed by Erturk *et al.* [14] are less than the equivalent 1.76 and 1.90 times increase in CPU time for the same Reynolds numbers for the ADI method. This shows that the extra CPU time needed for fourth-order accuracy when FONS equations are used is dependent on the numerical method used, and the extra CPU time for the method proposed by Erturk *et al.* [14] is much lower than the CPU time for the ADI method.

In Tables II and III, comparing the two methods, for the *same* Reynolds numbers and for the *same* α values (Δt), the iteration numbers for convergence are almost the same for both the ADI method and the method proposed by Erturk *et al.* [14]; however, the CPU time for the ADI method is less than that of the method proposed by Erturk *et al.* [14]. The reason for this is that in the method proposed by Erturk *et al.* [14] on the RHS of the finite difference equations more terms have to be calculated at each iteration and this increases the CPU time compared with that of the ADI method.

For faster convergence one can use larger time steps, if the numerical method used has a higher numerical stability limit. Therefore, for a numerical method, the maximum allowable time step (Δt) for convergence gives an indication of the numerical stability limit of the numerical method. Since in this study we have used two different numerical methods for the same flow problem, we decided to compare the numerical stability limit of the two methods applied to both the NS and the FONS equations, by finding the maximum allowable time step for convergence for both numerical methods. In order to find the maximum allowable time step for convergence, for a given Reynolds number we solve the second- and fourth-order equations using both of the numerical methods several times while increasing α in 0.01 increments each time, until the solution no longer converges.

Table IV. Comparison of the CPU time and iteration number for second- and fourth-order accuracy obtained using the ADI method, compiled with (`-fast -O5`) optimization.

Re	α	Second-order accuracy		Fourth-order accuracy		$\dfrac{\text{CPU}_{(\mathscr{O}\Delta x^4)}}{\text{CPU}_{(\mathscr{O}\Delta x^2)}}$	$\dfrac{\text{Iteration no.}_{(\mathscr{O}\Delta x^4)}}{\text{Iteration no.}_{(\mathscr{O}\Delta x^2)}}$
		$\text{CPU}_{(\mathscr{O}\Delta x^2)}$	$\text{Iteration no.}_{(\mathscr{O}\Delta x^2)}$	$\text{CPU}_{(\mathscr{O}\Delta x^4)}$	$\text{Iteration no.}_{(\mathscr{O}\Delta x^4)}$		
100	0.6	128.56	25 393	165.61	25 091	1.29	0.99
	0.4	191.07	37 739	246.44	37 362	1.29	0.99
	0.2	374.29	73 932	484.75	73 506	1.30	0.99
1000	0.79	130.40	25 677	—	—	—	—
	0.78	131.81	26 013	173.11	26 260	1.31	1.01
	0.7	146.70	28 949	192.82	29 215	1.31	1.01
	0.6	170.49	33 682	224.27	34 007	1.32	1.01
	0.5	204.44	40 278	267.97	40 688	1.31	1.01
	0.4	253.79	50 116	333.91	50 662	1.32	1.01
	0.2	499.28	98 518	658.93	99 831	1.32	1.01
	0.1	975.98	192 629	1291.06	195 820	1.32	1.02
3200	0.1	889.31	175 558	1233.30	187 023	1.39	1.07

Table V. Comparison of the CPU time and iteration number for second- and fourth-order accuracy obtained using the Erturk method, compiled with (`-fast -O5`) optimization.

Re	α	Second-order accuracy		Fourth-order accuracy		$\dfrac{\text{CPU}_{(\mathscr{O}\Delta x^4)}}{\text{CPU}_{(\mathscr{O}\Delta x^2)}}$	$\dfrac{\text{Iteration no.}_{(\mathscr{O}\Delta x^4)}}{\text{Iteration no.}_{(\mathscr{O}\Delta x^2)}}$
		$\text{CPU}_{(\mathscr{O}\Delta x^2)}$	Iteration no.$_{(\mathscr{O}\Delta x^2)}$	$\text{CPU}_{(\mathscr{O}\Delta x^4)}$	Iteration no.$_{(\mathscr{O}\Delta x^4)}$		
100	0.6	124.71	25 419	165.94	25 115	1.33	0.99
	0.4	184.99	37 763	247.33	37 386	1.34	0.99
	0.2	361.82	73 953	486.49	73 528	1.34	0.99
1000	1.89	53.51	10 963	—	—	—	—
	1.75	58.01	11 820	80.14	12 082	1.38	1.02
	1.6	62.94	12 888	87.18	13 191	1.39	1.02
	1.2	83.98	17 091	114.99	17 406	1.37	1.02
	0.6	165.19	33 734	226.39	34 067	1.37	1.01
	0.4	245.57	50 165	335.01	50 717	1.36	1.01
	0.2	483.04	98 564	661.49	99 884	1.37	1.01
	0.1	937.74	192 673	1284.31	195 871	1.37	1.02
3200	1.2	83.91	17 290	117.73	17 986	1.40	1.04

For the ADI method the maximum allowable α for convergence was 0.79 for second-order accuracy and it was 0.78 for fourth-order accuracy for $Re=1000$. For the method proposed by Erturk *et al.* [14] the maximum α values was 1.89 for second-order accuracy and it was 1.75 for fourth-order accuracy. This would indicate that one can use much larger time steps in the Erturk method compared with the ADI method, for example, for $Re=1000$ the method proposed by Erturk *et al.* [14] allows to use 2.4 times larger time step for the NS equations and 2.2 times larger time step for the FONS equations than the ADI method. From this we can conclude that the Erturk method has better numerical stability characteristics compared with the ADI method. When the maximum allowable time steps are used, the required CPU time for the method proposed by Erturk *et al.* [14] is almost 0.53 of the required CPU time for the ADI method. This means that the method proposed by Erturk *et al.* [14] converges almost twice faster than the ADI method when the maximum allowable time steps are used.

Comparing the numerical stability of the NS and the FONS equations, we see that for a chosen numerical method the FONS equations have slightly less stability limit than the NS equations. For example, for the ADI method and for $Re=1000$ the value of 0.79 for maximum allowable α for convergence for second-order accuracy drops down to 0.78 when fourth-order accuracy is used. Also, for the Erturk method and for $Re=1000$ the maximum allowable α value of 1.89 for second-order accuracy drops down to 1.75 if we switch to fourth-order accuracy. This would indicate that for fourth-order formulations the maximum allowable time step for convergence is lower than the maximum allowable time step for convergence for second-order formulations.

We then decided to run the same codes compiled by using the maximum compiler optimization (`-fast -O5`). Tables IV and V document the CPU time and iteration numbers when compiler optimization is used for the same conditions documented in Tables II and III, respectively. Comparing the numbers in Tables IV and II for ADI method, compiler optimization decreases the necessary CPU time for convergence about 0.71 times for second-order accuracy and about 0.51 times for fourth-order accuracy. Also comparing the same in Tables V and III for the method proposed

by Erturk *et al.* [14], compiler optimization decreases the necessary CPU time for convergence about 0.54 and 0.45 for second-order and fourth-order accuracy, respectively. These numbers show that compiler optimization decreases the CPU time significantly and for the numerical method proposed by Erturk *et al.* [14] the codes almost run twice faster in terms of CPU time when compiler optimization is used.

5. CONCLUSIONS

The FONS equations introduced by Erturk and Gokcol [9] are in the same form as the NS equations; therefore, any numerical method that solves the NS equations can be easily applied to the FONS equations in order to obtain fourth-order accurate solutions ($\mathcal{O}(\Delta x^4)$). One of the features of the introduced FONS equations is that any existing code that solves the NS equations with second-order accuracy ($\mathcal{O}(\Delta x^2)$) can be easily altered to provide fourth-order accuracy ($\mathcal{O}(\Delta x^4)$) just by adding some coefficients into the existing code using the FONS equations. This way, the accuracy of any second-order code can be easily increased to fourth order; however, there is a payoff for this increased accuracy, that is the extra CPU time for calculating the added coefficients.

In this study, we have solved the NS equations and the FONS equations to document the extra CPU time necessary for convergence when an existing second-order accurate code is altered to provide fourth-order accurate solutions. For this we have used two different numerical methods. We find that the extra CPU time is slightly dependent on the numerical method used. For the ADI method to obtain fourth-order accurate solutions of driven cavity flow, the CPU time increases 1.8 times compared with second-order accurate solutions for $Re = 1000$. Also for the numerical method proposed by Erturk *et al.* [14], with the cost of 1.64 times the CPU time necessary for second-order accuracy, a fourth-order accurate solution can be obtained for $Re = 1000$ using the FONS equations.

The FONS equations introduced by Erturk and Gokcol [9] provide a very easy way of obtaining fourth-order accurate solutions by just adding some coefficients into an existing second-order accurate code, at the expense of a minor increase in the CPU time.

REFERENCES

1. Zhang J. An explicit fourth-order compact finite difference scheme for three-dimensional convection–diffusion equation. *Communications in Numerical Methods in Engineering* 1998; **14**:209–218.
2. Dennis SC, Hudson JD. Compact h^4 finite difference approximations to operators of Navier–Stokes type. *Journal of Computational Physics* 1989; **85**:390–416.
3. MacKinnon RJ, Johnson RW. Differential-equation-based representation of truncation errors for accurate numerical simulation. *International Journal for Numerical Methods in Fluids* 1991; **13**:739–757.
4. Gupta MM, Manohar RP, Stephenson JW. A single cell high order scheme for the convection–diffusion equation with variable coefficients. *International Journal for Numerical Methods in Fluids* 1984; **4**:641–651.
5. Spotz WF, Carey GF. High-order compact scheme for the steady streamfunction vorticity equations. *International Journal for Numerical Methods in Engineering* 1995; **38**:3497–3512.
6. Li M, Tang T, Fornberg B. A compact fourth-order finite difference scheme for the steady incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 1995; **20**:1137–1151.
7. Ge L, Zhang J. High accuracy iterative solution of convection diffusion equation with boundary layers on nonuniform grids. *Journal of Computational Physics* 2001; **171**:560–578.
8. Spotz WF, Carey GF. Formulation and experiments with high-order compact schemes for nonuniform grids. *International Journal of Numerical Methods for Heat and Fluid Flow* 1998; **8**:288–303.

9. Erturk E, Gokcol C. Fourth order compact formulation of Navier–Stokes equations and driven cavity flow at high Reynolds numbers. *International Journal for Numerical Methods in Fluids* 2006; **50**:421–436.
10. Zhang J. On convergence and performance of iterative methods with fourth-order compact schemes. *Numerical Methods for Partial Differential Equations* 1998; **14**:263–280.
11. Peaceman DW, Rachford Jr HH. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics* 1955; **3**:28–41.
12. Anderson DA, Tannehill JC, Pletcher RH. *Computational Fluid Mechanics and Heat Transfer*. McGraw-Hill: New York, 1984.
13. Chung TJ. *Computational Fluid Dynamics*. Cambridge University Press: Cambridge, 2002.
14. Erturk E, Corke TC, Gokcol C. Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers. *International Journal for Numerical Methods in Fluids* 2005; **48**:747–774.
15. Störtkuhl T, Zenger C, Zimmer S. An asymptotic solution for the singularity at the angular point of the lid driven cavity. *International Journal of Numerical Methods for Heat and Fluid Flow* 1994; **4**:47–59.