University of Glasgow

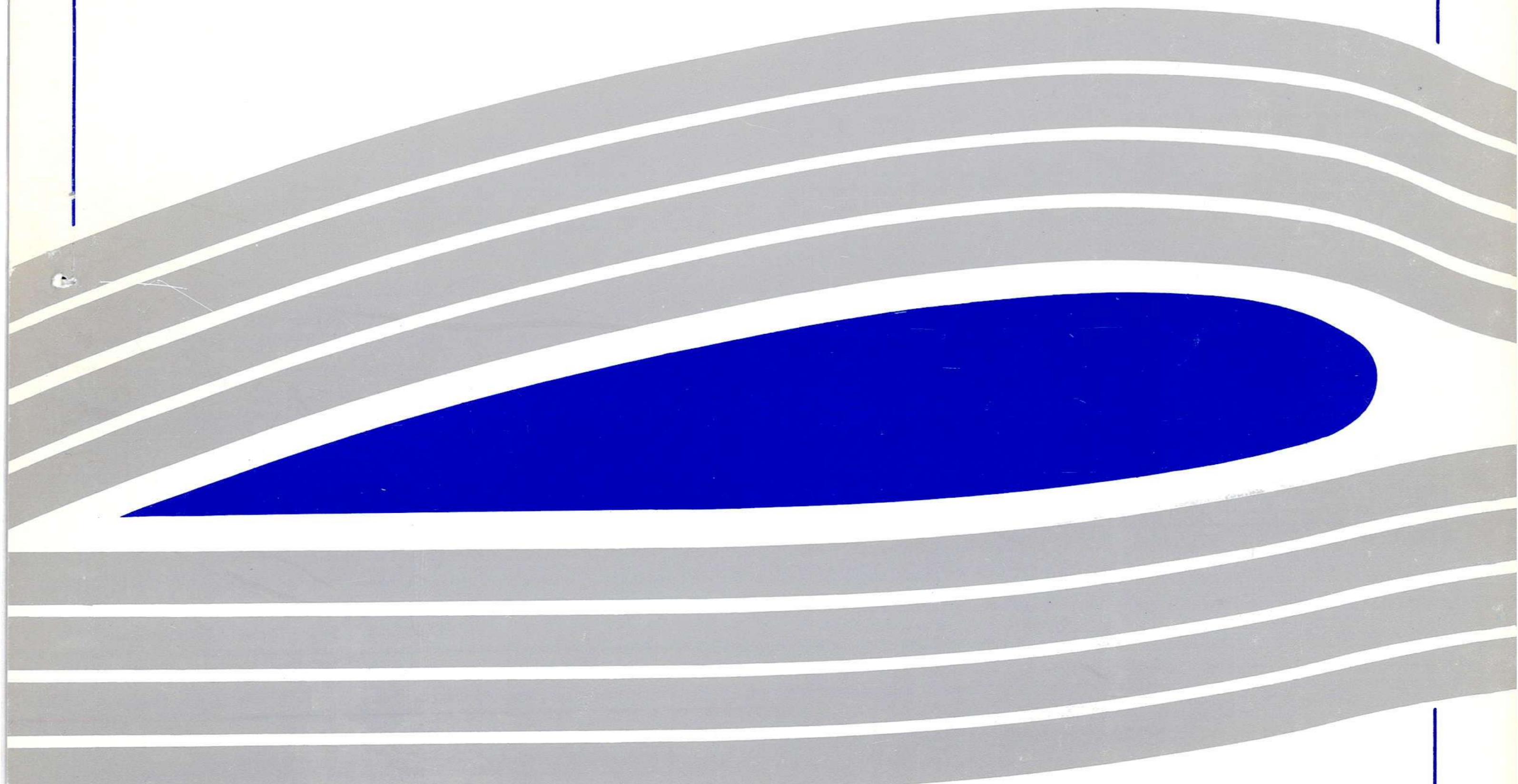
DEPARTMENT OF

AEROSPACE ENGINEERING

Application of the Highest Order Finite Difference Scheme to Solve Incompressible Navier-Stokes Equations

C. SHU and Bryan E. RICHARDS

G.U. Aero Report 9226



Application of the Highest Order Finite Difference Scheme to Solve Incompressible Navier-Stokes Equations

C. SHU and Bryan E. RICHARDS

G.U. Aero Report 9226

Dept. of Aerospace Engineering
University of Glasgow
Glasgow G12 8QQ

June, 1992

APPLICATION OF THE HIGHEST ORDER FINITE DIFFERENCE SCHEME TO SOLVE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

C. SHU[1] and B.E. RICHARDS[2]

[1]Dept. of Mechanical and Production Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 0511

[2]Dept. of Aerospace Engineering, University of Glasgow, Glasgow G12 8QQ, Scotland, U.K.

This paper introduces a global method of the highest order finite difference scheme for the discretization of any order derivative. The weighting coefficients in this scheme can be determined by a simple algebraic formulation or by a recurrence relationship. A multi-domain technique was also presented for treating more complex problems. Application of this scheme to solve 2D incompressible Navier-Stokes equations showed that accurate numerical results can be achieved using just a few grid points and requiring much less computational effort and storage.

1. INTRODUCTION

The numerical solutions of incompressible Navier-Stokes equations can usually be obtained by low order finite difference and finite element methods by using a large number of grid points. One disadvantage of these methods is that in some cases, the solutions are required at only a few specified points in the physical domain. But for a reasonable accuracy, these low order methods also require the use of a large number of grid points to obtain the solutions at those specified points. To seek a more efficient method to obtain the solutions using considerably smaller number of grid points, global methods may provide a promising way. Amongst them, the spectral and pseudospectral methods are widely used. But since these methods do not discretize the derivatives directly, they may be inconvenient to apply, especially for the case with nonlinear terms and high order derivatives. Furthermore, since the coordinates of grid points in spectral methods are usually taken as the roots of a specific function, a transformation between physical space and computational space is often required. As a result, the global method of the highest order finite difference scheme is attractive in application because it discretizes derivatives directly and is easier to apply than spectral methods. The key problem in the highest order finite difference scheme is how to determine the weighting coefficients for the discretization of any order derivative.

Recently, Bellman et al [1] presented an attractive method of differential quadrature (DQ) for the efficient solutions of partial differential equations. DQ is a type of global method, which discretizes the derivative using the same form as the highest order finite difference scheme. Key to DQ is the determination of weighting coefficients for any order derivative discretization. Bellman et al suggested two ways to determine the weighting coefficients of the first order derivative. The first way solves an algebraic equation system. The second uses a simple algebraic formulation, but with the condition of coordinates of grid points chosen as the roots of the shifted Legendre polynomial. From the publication, applications of DQ so far ([3]-[6]) usually use Bellman's first way to obtain the weighting coefficients because the coordinates of grid points can be chosen arbitrarily. But unfortunately, when the order of the system, i.e. the number of grid points, is large, the matrix of the system is ill-conditioned. Thus it is very difficult to obtain the weighting coefficients using this way when the number of grid points used is large. To overcome the drawbacks of DQ, the current authors [2] have developed the technique of generalized differential quadrature (GDQ), where the weighting coefficients of the first order derivative are determined by a simple algebraic formulation without any restriction on choice of grid points, and the weighting coefficients of the second and higher order derivatives are determined by a recurrence relationship. It has been proved in Ref. 2 that the algebraic equation system for the weighting coefficients derived from the highest order finite difference scheme is equivalent to that derived from GDQ. Thus the weighting coefficients for discretizing any order derivative in the highest order finite difference scheme can be given from the results of GDQ.

2. NUMERICAL METHOD

2.1 Highest Order Finite Difference Scheme

For brevity, the one-dimensional case is chosen for demonstration. Since any finite range can be transformed into the range of [0, 1] by a simple transformation, we will consider only the range [0, 1] hereafter. Supposing that there are N grid points in the whole domain, the discretization of the first order derivative of function f(x,t) with respect to x at x_i can be given by the *(N-1)*th order (i.e. the highest order) finite difference scheme as a linear sum of all the functional values at N grid points, which has the form

$$f_x(x_i,t)=\sum_{j=1}^{N} a_{ij}\cdot f(x_j,t) \tag{1}$$

for $i=1,2,\cdots,N,$

where $f_x(x_i,t)$ indicates the first order derivative of f(x,t) with respect to x at x_i, a_{ij} are the weighting coefficients. a_{ij} can be determined by the Taylor series expansion which is usually used in the design of the low order finite difference schemes. Using a Taylor series expansion, $f(x_j,t)$ can be expressed as

$$f(x_j,t)=f(x_i,t)+f^{(1)}(x_i,t)\cdot(x_j-x_i)+\cdots+ f^{(k)}(x_i,t)\cdot(x_j-x_i)^k/k!+\cdots+ f^{(N-1)}(x_i,t)\cdot(x_j-x_i)^{N-1}/(N-1)!+R_N \tag{2}$$

where $f^{(k)}(x_i,t)$ is the *k*th order derivative of f(x,t) with respect to x at x_i, R_N is the truncated error, which can be written as

$$R_N=f^{(N)}(\xi_j,t)\cdot(x_j-x_i)^N/N! \quad ,\xi_j\in[x_i,x_j] \tag{3}$$

Substituting eq. (2) into eq. (1) and keeping the *(N-1)*th order accuracy leads to the following algebraic equations

$$\begin{cases}\sum_{j=1}^{N} a_{ij}=0\\ \sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i)=1\\ \sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i)^k=0,\ k=2,\ 3,\ \cdots,\ N-1\end{cases} \tag{4}$$

for $i=1,2,\cdots,N.$

Equation set (4) is an equation system for determination of the weighting coefficients of the first order derivative in the highest order finite difference scheme. Similarly in the domain $[x_1, x_N]$, the *m*th order derivative of function f(x,t) with respect to x at x_i can be discretized by the highest order finite difference scheme as

$$f_x^{(m)}(x_i,t)=\sum_{j=1}^{N} w_{ij}^{(m)}\cdot f(x_j,t) \tag{5}$$

for $i=1,2,\cdots,N;\ m=2,3,\cdots,N-1,$

where $f_x^{(m)}(x_i,t)$ indicates the *m*th order derivative of f(x,t) with respect to x at x_i, $w_{ij}^{(m)}$ the weighting coefficients. Substituting eq. (2) into eq. (5), and keeping the *(N-m)*th order accuracy, we obtain

$$\begin{cases}\sum_{j=1}^{N} w_{ij}^{(m)}=0\\ \sum_{j=1}^{N} w_{ij}^{(m)}\cdot(x_j-x_i)^m=m!\\ \sum_{j=1}^{N} w_{ij}^{(m)}\cdot(x_j-x_i)^k=0, k=1,2,\cdots,N-1,\ k\neq m\end{cases} \tag{6}$$

Equation set (6) is an algebraic equation system for determining the weighting coefficients of the second and higher order derivatives in the highest order finite difference scheme. In the following, we will show that all these weighting coefficients can be determined by a simple algebraic formulation or by a recurrence relationship.

2.2 Weighting Coefficients of the First Order Derivative

For the efficient solution of a smooth problem, Bellman et al [1] introduced a technique of differential quadrature, which uses the same form as eq. (1) to discretize the first order derivative. They suggested two methods to determine the weighting coefficients a_{ij}. The first method is to let eq. (1) be exact for test functions

$g(x) = x^k,\ k = 0, 1, \cdots, N-1$, which leads to a set of algebraic equations as follows

$$\sum_{j=1}^{N} a_{ij} \cdot x_j^k = k \cdot x_i^{k-1} \tag{7}$$

for $i = 1, 2, \cdots, N;\ k = 0, 1, \cdots, N-1.$

This equation system has a unique solution because its matrix is of Vandermonde form. The second method is similar to the first one with an exception that the different test functions

$$g(x) = \frac{L_N(x)}{(x - x_k) \cdot L_N^{(1)}(x_k)},\ k = 1, 2, \cdots N, \tag{8}$$

are chosen, where $L_N(x)$ is the *N*th order Legendre polynomial and $L_N^{(1)}(x)$ the first order derivative of $L_N(x)$. By choosing x_k to be the roots of the shifted Legendre polynomial, Bellman et al obtained a simple algebraic formulation for a_{ij} with the condition that the coordinates of grid points should be chosen as the roots of an *N*th order Legendre polynomial. Applications of DQ in engineering so far [3]-[6] usually use Bellman's first method to obtain the weighting coefficients because it lets the coordinates of grid points be chosen arbitrarily. To overcome the drawbacks of DQ, the current authors [2] have introduced the technique of GDQ for determination of weighting coefficients. In the following, we will firstly prove that the algebraic equation system (7) given from GDQ is equivalent to the equation system (4) given from the highest order finite difference scheme, then use the results of GDQ to calculate the weighting coefficients.

It is obvious that the first equation of equation sets (7) and (4) are the same, i.e.

$$\sum_{j=1}^{N} a_{ij} = 0 \tag{9}$$

Furthermore, it can be shown that the second equation of the two systems are the same, i.e.

$$\sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i) - 1 = \sum_{j=1}^{N} a_{ij} \cdot x_j - 1 = 0 \tag{10}$$

Now, assuming that the first p+1 equations of the two systems are the same, that is

$$\sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^k = \sum_{j=1}^{N} a_{ij} \cdot x_j^k - k \cdot x_i^{k-1} = 0 \tag{11}$$

for $k = 0, 1, \cdots, p;\ i = 1, 2, \cdots, N,$

then using the binary formulation

$$(a-b)^p = a^p - c_p^1 \cdot a^{p-1} b + \cdots$$
$$+ (-1)^k c_p^k a^{p-k} b^k + \cdots + (-1)^p \cdot b^p \tag{12}$$

where c_p^k is the combination of p terms taken k at a time, and setting a = b = 1, the following expression will be obtained.

$$c_p^1 - c_p^2 + \cdots + (-1)^{k+1} c_p^k + \cdots + (-1)^{p+1} = 1 \tag{13}$$

Using eq. (12), the *(p+2)*th equation of equation set (4) can be written as

$$\sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^{p+1} = \sum_{j=1}^{N} a_{ij} \cdot x_j^{p+1} - c_{p+1}^1 \cdot x_i \cdot \left[\sum_{j=1}^{N} a_{ij} \cdot (x_j^p - \frac{1}{2} \cdot c_p^1 \cdot x_j^{p-1} \cdot x_i + \cdots + \frac{(-1)^p \cdot x_i^p}{p+1}) \right] \tag{14}$$

Substituting eqs.(11), (13) into eq. (14) leads to

$$\sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^{p+1} = \sum_{j=1}^{N} a_{ij} \cdot x_j^{p+1} - (p+1) \cdot x_i^p \tag{15}$$

Equation (15) demonstrates that the *(p+2)*th equation of the two systems are exactly the same. Since p is an arbitrary integer only if $p \le N-2$, it has been proved that the two systems (4) and (7) are the same. But although the weighting coefficients a_{ij} can be determined by the equation system (4) or by the equation system (7), the solution of either (4) or (7) is not easy to be obtained for a large N. We will use the results of GDQ to calculate them.

It is well known that a continuous function in the interval [0, 1] can be approximated by an infinite polynomial accurately in accordance with the Weierstrass polynomial approximation theorem. In practice, a truncated finite polynomial may be used. Following this approach, it is supposed that any smooth function in the interval [0, 1] can be approximated by a *(N-1)*th order polynomial. And it is easy to show that the polynomial of degree less than or equal to N-1 constitutes an N-dimensional linear vector space V_N with respect to the operation of addition and multiplication. From the concept of linear independence, the bases of a linear vector space can be considered as a linearly independent subset which spans the entire space. Here if $r_k(x),\ k=1,2,\cdots,N$, which are in the space V_N, are the base polynomials, any polynomial in V_N can be expressed uniquely as a linear combination of $r_k(x),\ k=1,2,\cdots,N$. And if all the base polynomials satisfy a linear constrained relationship such as eq. (1), so does any polynomial in the space. In the linear vector space, there may exist several sets of base polynomials. Each set of base polynomials can be expressed uniquely by another set of base polynomials. It is found that, if the base polynomial $r_k(x)$ is chosen to be x^{k-1}, or taken the same form as eq. (8), the same results given by Bellman et al can be achieved.

For generality, GDQ chooses the base polynomial $r_k(x)$ to be the Lagrange interpolated polynomial

$$r_k(x)=\frac{M(x)}{(x-x_k)\cdot M^{(1)}(x_k)} \tag{16}$$

where $M(x)=(x-x_1)\cdot(x-x_2)\cdots(x-x_N)$

$$M^{(1)}(x_k)=\prod_{j=1,\,j\neq k}^{N}(x_k-x_j)$$

$x_1,x_2,\cdots,x_N$ are the coordinates of grid points, and can be chosen arbitrarily.

For simplicity, we set

$$M(x)=N(x,x_k)\cdot(x-x_k),\ k=1,2,\cdots,N \tag{17}$$

with $N(x_i,x_j)=M^{(1)}(x_i)\cdot\delta_{ij}$, where δ_{ij} is the Kronecker operator. Thus we have

$$M^{(m)}(x)=N^{(m)}(x,x_k)\cdot(x-x_k)+m\cdot N^{(m-1)}(x,x_k)$$
$$for\ m=1,2,\cdots,N-1;\ k=1,2,\cdots,N, \tag{18}$$

where $M^{(m)}(x),\ N^{(m)}(x,x_k)$ indicate the mth order derivative of $M(x)$ and $N(x,x_k)$. Substituting eq. (16) into eq. (1) and using eq. (18), we obtain

$$a_{ij}=\frac{M^{(1)}(x_i)}{(x_i-x_j)\cdot M^{(1)}(x_j)}\ ,\ for\ j\neq i \tag{19a}$$

$$a_{ii}=\frac{M^{(2)}(x_i)}{2M^{(1)}(x_i)} \tag{19b}$$

for $i,j=1,2,\cdots,N$.

Equation (19) is a simple formulation for computing a_{ij} without any restriction on choice of grid point x_i. Actually, if x_i is given, it is easy to compute $M^{(1)}(x_i)$, thus a_{ij} for $i\neq j$. It can be applied in a straightforward way for both uniform and non-uniform grids. The calculation of a_{ii} is based on the computation of the second order derivative $M^{(2)}(x_i)$ which is not easy to obtain. On the other hand, from the equation system (4), a_{ii} can be obtained from the following formulation

$$\sum_{j=1}^{N} a_{ij}=0 \tag{20}$$

2.3 Weighting Coefficients of the Second and Higher Order Derivatives

For the second order derivative, we introduce the following linear constrained relationship

$$f_{xx}(x_i,t)=\sum_{j=1}^{N} b_{ij}\cdot f(x_j,t) \tag{21}$$
$$for\ i=1,2,\cdots,N,$$

where $f_{xx}(x_i,t)$ is the second order derivative of f(x,t) with respect to x at x_i, and Lagrange interpolated polynomials are chosen as the base polynomials. Using the same approach as for the first order derivative and formulation (18), (19), the weighting coefficients b_{ij} are given by

$$b_{ij}=2\cdot a_{ij}\left(a_{ii}-\frac{1}{x_i-x_j}\right),\ for\ j\neq i \tag{22a}$$

$$b_{ii}=\frac{M^{(3)}(x_i)}{3M^{(1)}(x_i)} \tag{22b}$$
$$for\ i,\ j=1,2,\cdots,N.$$

When $j\neq i$, b_{ij} can be calculated from a_{ij} easily. Also, from the results of the highest order finite difference scheme, i.e. the equation system (6), b_{ii} can be given by

$$\sum_{j=1}^{N} b_{ij} = 0 \tag{23}$$

For the case of discretization of the higher order derivative, eq. (5) can be applied. To deduce a recurrence relationship for the weighting coefficients, the following linear constrained relationship is also applied

$$f_x^{(m-1)}(x_i,t) = \sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot f(x_j,t) \tag{24}$$

$$for\ i = 1,2,\cdots,N;\ m = 2,3,\cdots,N-1.$$

Substituting eq. (16) into eqs. (5), (24), and using eqs. (18), (19), a recurrence formulation is obtained as follows

$$w_{ij}^{(m)} = m \cdot \left(a_{ij} \cdot w_{ii}^{(m-1)} - \frac{w_{ij}^{(m-1)}}{x_i - x_j} \right) \quad , \ j \neq i \tag{25}$$

$$for\ i,\ j = 1,2,\cdots,N;\ m = 2,3,\cdots,N-1.$$

where a_{ij} is the weighting coefficients of the first order derivative described above. Again, from the equation system (6), $w_{ii}^{(m)}$ can be obtained by

$$\sum_{j=1}^{N} w_{ii}^{(m)} = 0 \tag{26}$$

For the multi-dimensional case, it is shown in Ref. 2 that if the grid is structured, then the discretization of spatial derivatives in each direction can be treated using the same fashion as in the one-dimensional case.

2.4 Multi-Domain Technique

It is supposed that the physical domain of a problem is represented by Ω, and the boundary by Γ. The multi-domain technique, firstly, decomposes the domain Ω into several subdomains Ω_i, $i = 1,2,\cdots,K$, where K is the number of subdomains. In each subdomain, a local mesh is generated with stretching near the boundary and the local highest order finite difference scheme is applied, in the same fashion as the application of the scheme in a single domain. Each subdomain may have a different number of grid points. The solutions for interior grid points are independent for each subdomain. Globally, the information exchange between subdomains is required. This can be done across the interface of subdomains. Any complex geometry can be transformed into a rectangular domain or a combination of the rectangular subdomains, by the technique of grid generation. Here we thus consider only a rectangular domain for the demonstration without losing generality.

For the solution at the interface, there are several ways available. In the present application, a patched interface, which keeps the function and its normal derivative to be continuous across the interface, was adopted. As shown in Fig. 1, Γ_{ij} is the interface between the subdomains Ω_i and Ω_j.

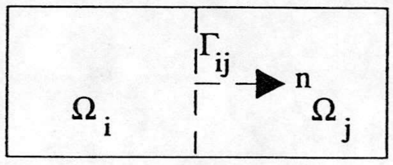


Fig. 1 Topology of a Patched Interface

Keeping the function and its normal derivative to be continuous across the interface, we obtain

$$f(x_N^i) = f(x_1^j) \quad on\, \Gamma_{ij} \tag{27}$$

$$f_n(x_N^i) = f_n(x_1^j) \quad on\, \Gamma_{ij} \tag{28}$$

where $f(x_N^i)$, $f(x_1^j)$ represent the values of the function f at the interface of the i subdomain and the j subdomain, and $f_n(x_N^i)$, $f_n(x_1^j)$ the values of the first order derivative of f with respect to n at the interface.

For the cases selected for study, each subdomain is rectangular. Then the normal direction to the interface is parallel to one coordinate axis in the local coordinate system. For simplicity, this coordinate axis can be assumed as the x axis, and along this direction, there are N grid points in the i subdomain and M grid points in the j subdomain. The weighting coefficients of the first order derivative along the x direction are written as a_{mn}^i in the i subdomain and a_{mn}^j in the j subdomain. Thus, using the technique described above, eq. (28) can be written as

$$\sum_{k=1}^{N} a_{Nk}^i \cdot f(x_k^i) = \sum_{k=1}^{M} a_{1k}^j \cdot f(x_k^j) \tag{29}$$

Using eq. (27), and setting $f(x_N^i) = f(x_1^j) = \bar{f}$, we obtain

$$\bar{f} = \frac{\sum_{k=1}^{N-1} a_{Nk}^i \cdot f(x_k^i) - \sum_{k=2}^{M} a_{1k}^j \cdot f(x_k^j)}{a_{11}^j - a_{NN}^i} \tag{30}$$

where $\bar{f}$ is the value of the function f at the interface Γ_{ij}, which exchanges the information between the subdomains, and $f(x_k^i), f(x_k^j)$ represent the values of the function f at x_k^i in the i subdomain and x_k^j in the j subdomain. Along the interface, the function is C^1 continuity, and may not satisfy the governing equations.

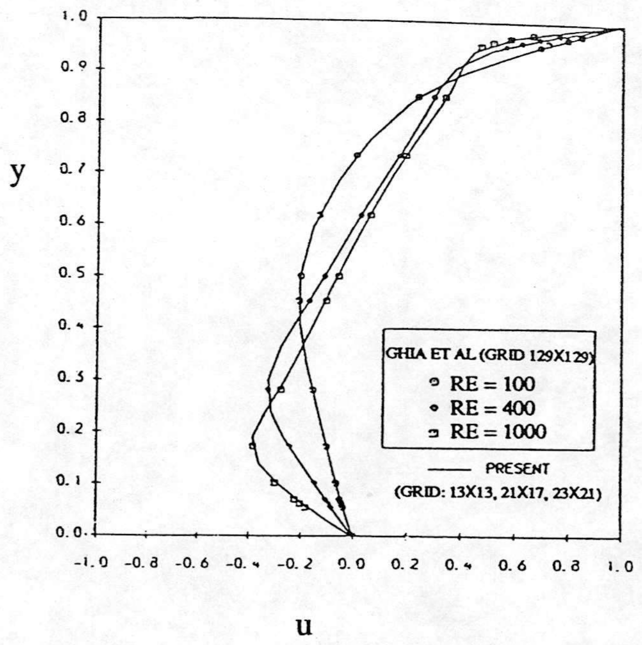
3. NUMERICAL RESULTS

For the application of the highest order finite difference scheme to simulate incompressible viscous flows, the vorticity-stream function formulation was taken as the governing equation. The boundary condition for vorticity at the solid boundary is given from the stream function formulation. And two components of velocity at the solid boundary give two boundary conditions for the stream function. One is of Dirichlet type, another is of Neumann type. After discretizing the derivative in the Neumann type boundary condition by the highest order finite difference scheme, these two boundary conditions can be combined to give two-layer boundary conditions for the stream function. After all the spatial derivatives are discretized by the highest order finite difference scheme, the resultant set of ordinary differential equations for vorticity are then solved by the 4th order Runge-Kutta scheme, and the set of algebraic equations for the stream function are solved by LU decomposition. Two test examples are demonstrated as follows.

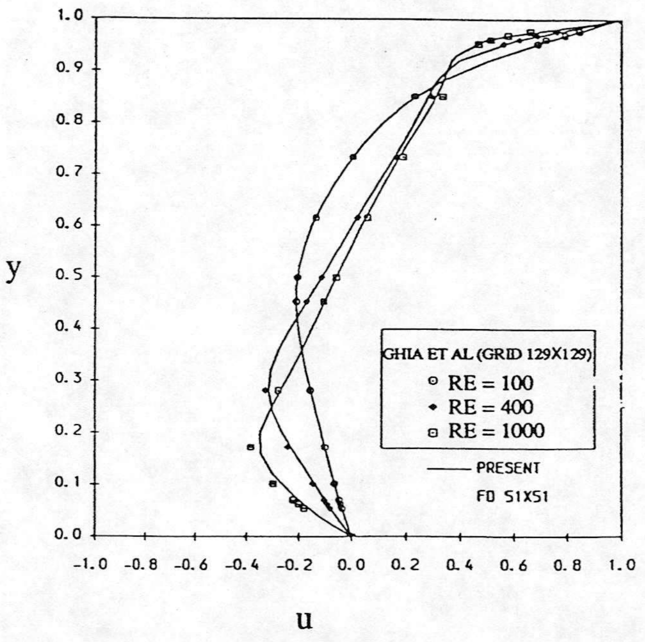
3.1 The Driven Cavity Flow

The driven cavity flow problem is a standard test case, which is often chosen to validate new numerical techniques. There are a variety of numerical results available for comparison. For example, the velocity profile through the geometrical centre of the cavity are presented by Ghia et al [7]. For present numerical simulation by the highest order finite difference scheme, the solutions were obtained in the Reynolds number range from 100 to 1000. The mesh sizes used are 13×13, $17\times15, 21\times17$ and 23×21 for Reynolds numbers of 100, 200, 400, 1000. The initial values for all the variables in the interior points are taken to be zero.

For direct comparison of the highest order finite difference scheme with the conventional low order finite difference scheme, numerical results using a second order time-split MacCormack finite difference scheme for the vorticity equation and a preconditioning technique of the strongly implicit procedure (SIP) for the stream function, were also obtained using a uniform grid of mesh size of 51×51. By numerical experiment, the allowable



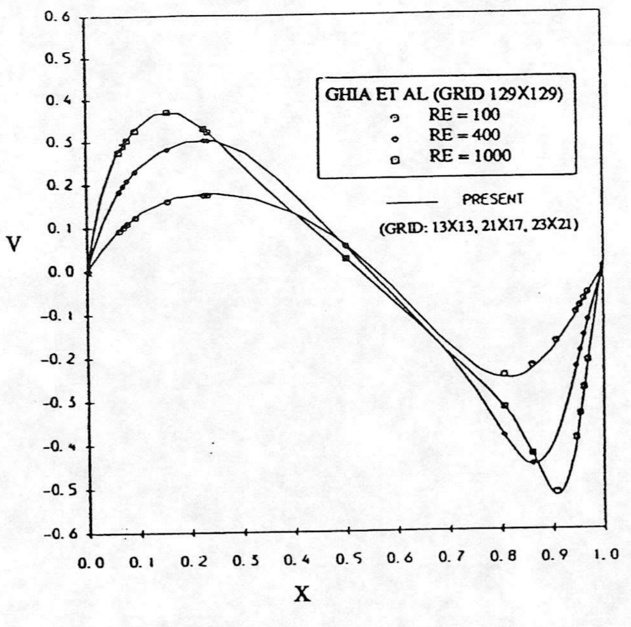
(a) Results from the Highest Order FD Scheme



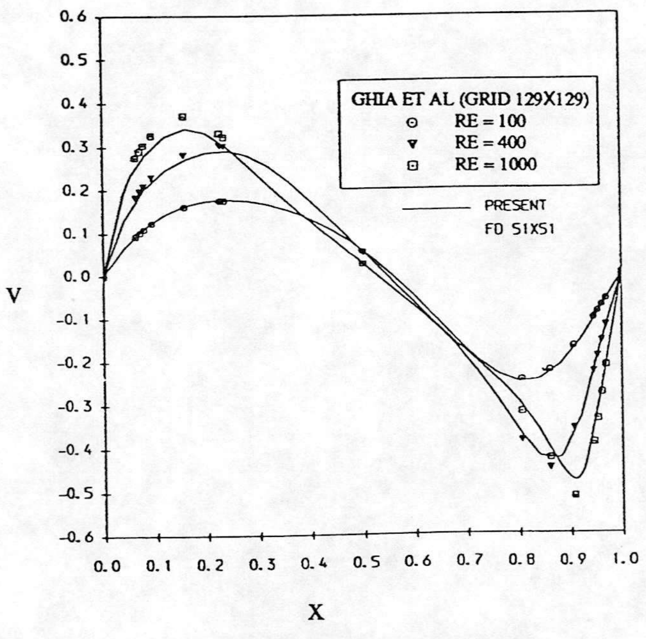
(b) Results from the Second Order FD Scheme

Fig.2 Horizontal Velocity Profile through the Geometrical Center of the Cavity

maximum time step size was used. Fig. 2 displays the computed horizontal velocity profiles along the vertical line through the geometric center of the cavity for Reynolds numbers of 100, 400, 1000. Fig. 3 shows the vertical velocity profiles along the horizontal line through the geometric center of cavity. The numerical results given by Ghia et al [7] were also included in the figures for comparison. These results are based on a fine mesh of 129×129. It is clear from Fig. 2 and 3 that the



(a) Results from the Highest Order FD Scheme



(b) Results from the Second Order FD Scheme

Fig.3 Vertical Velocity Profile through the Geometrical Center of the Cavity

highest order finite difference results are more accurate than the second order finite difference results even though considerably fewer grid points were used in the highest order scheme. Furthermore, the CPU time required by the highest order finite difference results is much less than that by the second order finite difference results. Table I shows the CPU time (seconds) required for the driven cavity flow simulation by both the highest order and the second order finite difference schemes on the IBM 3090 vector machine.

Table I Comparison of CPU Time (seconds) for the Driven Cavity Flow Simulation

Re	100	200	400	1000
Highest Order FD	4.27	6.69	16.99	33.79
2nd order FD	442.73	536.98	601.50	732.90

3.2 The Flow past a Square Step

Now considered is the flow in a channel containing a square step in which the step is located fairly close to the inlet. The flow past a square step with a "flat" inlet velocity distribution rather than a fully developed parabolic profile, is a more challenging problem for numerical simulation since in this case, not only the two sharp corners of the step produce vorticity singularities, but also the boundary condition at the inlet introduces other vorticity singularities. Some researchers have reported difficulty in numerical simulation of this problem. For example, Hughes et al [8] claimed that the conventional Galerkin finite element method produced the spurious wiggles in the velocity vectors upstream of the step, and suggested the use of an upwind scheme which then generated the solution without wiggles. Leone and Gresho [9] studied this problem exhaustively using a velocity-pressure formulation and the conventional Galerkin method. They claimed that, when a coarse mesh is used the inlet wiggles may be caused more by the presence of the step than the inlet leading edge singularity, and when the finer mesh is used, most of the inlet wiggles disappear, only small deviations appearing near the top singularity of the inlet leading edge. They thus suggested that this difficult problem should be solved on a fine grid.

Following the work of Hughes et al, it is attempted to simulate the developing flow in a one unit high channel containing a step located 1.2 units from the inlet which is 0.4 units high and 0.4 units across. The problem definition and the computational domain are shown in Fig. 4, where the whole computational domain is decomposed into 5 subdomains with 4 interfaces. The inlet boundary condition is a "flat" velocity profile, u=1 and v=0, except that the no-slip condition, u=v=0 occurs on the top and bottom surfaces. For the present numerical simulation, the outlet location is chosen as 8 units from the inlet. Numerical experiment showed that the accurate numerical results can be obtained by using the mesh sizes of 15×13 for domain I and II, 7×13 for domain III, and 21×13 for domain IV and V. This configuration is shown in Fig. 5. The present results agree well with those given by Leone and Gresho [9]. Fig. 6 shows the streamlines for Reynolds numbers of 50, 100, 150, 200, 250, where the values of these streamlines are 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0, -0.1, -0.01, -0.001 and the window for plotting these streamlines in the x direction is from x=0.0 to x=6.0. Clearly, it is shown that no wiggles appear in the flow field except for very small wiggles caused by the top singularity of the channel leading edge, which appear near the top corner of the inlet (streamlines have a small contraction towards mid-channel). This agrees well with the analysis of Leone et al and demonstrates that the mesh sizes used are fine enough to get accurate results. The lengths of the upstream and downstream separation zone for the various Reynolds numbers are shown in Table II, where $\bar{x}_{up} = x_{up} / h, \bar{x}_{do} = x_{do} / h, \quad x_{up}, x_{do}$ represent the lengths of the upstream and downstream separation zones, and h is the height of the step. For each Reynolds number, numerical results were obtained within 4 minutes of CPU time on the IBM 3090 vector machine.

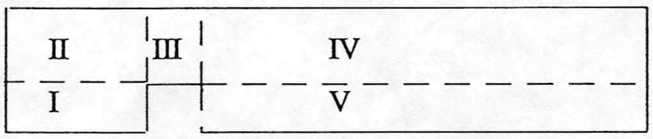


Fig.4 Problem Definition and Computational Domain for a Square step Problem

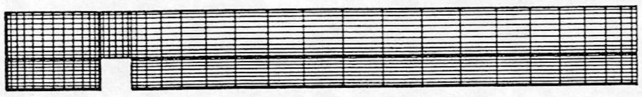
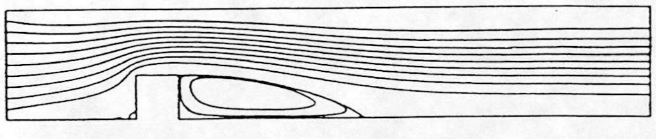
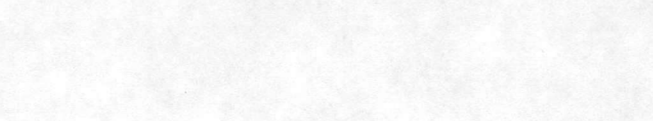


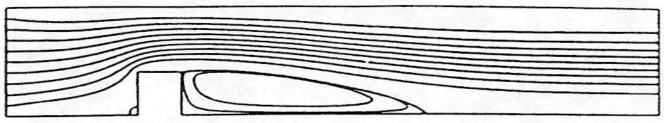
Fig. 5 Meshes for Flow past a Square Step



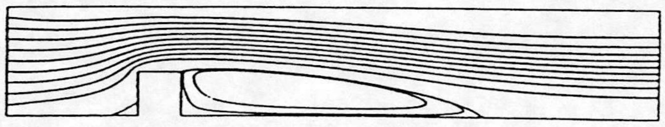
(a) Re = 50



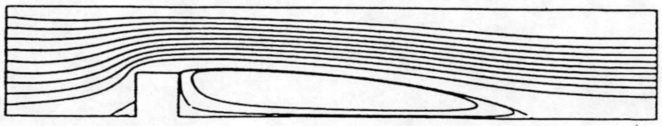
(b) Re = 100



(c) Re = 150



(d) Re = 200



(e) Re = 250

Fig. 6 Streamlines past a Square Step

Table II Length of the Separation Zone for a Square Step Problem

Re	25	50	85	100	150	200	250
$\bar{x}_{up}$	0.1749	0.1749	0.1749	0.1757	0.5846	0.5771	0.5846
$\bar{x}_{do}$	1.5771	2.6701	3.8876	4.3501	5.7549	7.0636	7.9824

REFERENCES

1. R. Bellman, B.G. Kashef and J. Casti, J. Comput. Phys. 10 (1972), 40-52.
2. C. Shu, PhD Thesis, University of Glasgow, July 1991.
3. F. Civan and C.M. Sliepcevich, J. Math. Anal. Appl. 93 (1983), 206-221.
4. F. Civan and C.M. Sliepcevich, J. Math. Anal. Appl. 101 (1984), 423-443.
5. J.O. Mingle, J. Math. Anal. Appl. 60 (1977), 559-569.
6. S.K. Jang, C.M. Bert, and A.G. Striz, Int. J. Numer. Methods Eng. 28 (1989), 561-577.
7. U.Ghia, K.N. Ghia, and C.T. Shin, J. Comput. Phys. 48 (1982), 387-411.
8. T.J.R. Hughes, W.K. Liu and A. Brooks, J.Comput. Phys. 30 (1979), 1-60.
9. Jr., J.M. Leone and P.M. Gresho, J. Comput. Phys. 41 (1981), 167-191.