INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 12, 491–506 (1991)

HIGHER-ORDER DIFFERENCE APPROXIMATIONS OF THE NAVIER–STOKES EQUATIONS

PAOLO LUCHINI

Istituto di Gasdinamica, Facoltà di Ingegneria, University of Naples, Naples, Italy

SUMMARY

A discretization scheme is presented which, unlike the standard higher-order finite difference and spline methods, does not give rise to unphysical solution modes and boundary conditions. Practical application of this scheme is achieved via the DCMG algorithm recently developed by the same author, which turns out to be able to find a converged solution of the ψ–ζ Navier–Stokes equations in about the same time for high-order as for low-order discretization schemes. Examples are presented for the driven cavity problem to explore the accuracy of the new method. Finally, a local analysis is performed of the corner singularities which exist in driven cavity flow, and their effect on the overall accuracy of the solutions obtained by polynomial interpolation methods is investigated.

KEY WORDS Higher-order Finite differences Navier–Stokes

1. INTRODUCTION

Whereas in the solution of initial value problems for systems of ordinary differential equations the fourth-order Runge–Kutta method is a *de facto* standard, in boundary value problems approximation schemes of higher than second order are much less frequently adopted for ordinary, and only very seldom for partial, differential equations. For example, several higher-order differencing schemes for the Navier–Stokes equations, based on splines and other polynomial interpolations, were reviewed by Rubin and Khosla[1] in 1977 but met with little fortune in the following years.

One reason for this lack of success was certainly the loss of performance encountered by the numerical solution methods that the experimenters tried to apply to these higher-order difference formulations, generally variants of line relaxation or ADI. In addition, it was generally not easy to enforce exact discrete conservation properties in the higher-order formulations, whereas the experience with standard low-order differences shows that such conservative formulations tend to yield higher precision. It is therefore likely that the gain due to the more precise difference formulae was somewhat offset by the loss due to lack of discrete conservativeness, as is also confirmed by our results shown below. Giannattasio and Napolitano in a recent paper[2] also indicate as a possible factor the ineffectiveness of higher-order methods in the presence of singularities such as those which affect corners in the classical driven cavity problem.

In fact, the choice of an ADI or line relaxation solution algorithm also posed a major restriction on the polynomial interpolations judged to be feasible in Rubin and Khosla's presentation. In order for such a method not to require an exceedingly long execution time, the parameters of the chosen approximate representation (e.g. spline coefficients) had to be determinable from the inversion of a system having a block-tridiagonal coefficient matrix, and Rubin and

0271–2091/91/050491–16$08.00

Received August 1989
Revised May 1990

Khosla purposely considered only schemes having this property. A characteristic of all the higher-order schemes that result as feasible is that they require a larger number of boundary conditions than the differential problem. These additional boundary conditions are generally determined by using either the differential equations themselves, and possibly their derivatives, or extrapolations from the interior points. The choice of additional boundary conditions is delicate since they can sometimes make the difference equation system unstable even when the differential problem is not, and they might have been the cause of the loss in performance noticed in Reference 2 when the convergence rate of the higher-order method was compared with that of the standard second-order one.

Multigrid algorithms have considerably widened the range of feasible discretization choices. In particular, the DCMG (deferred correction multigrid) algorithm of Reference 3 turns out to be able to solve a variety of higher-order discrete formulations of a given problem in about the same computing time required for a low-order formulation, and without placing any special requirement upon the difference equations.

Using this algorithm we have experimented with a scheme which shares with the one-dimensional Runge–Kutta method the property of not requiring any more boundary conditions than the differential problem itself does.

2. THE DCMG ALGORITHM

The deferred correction multigrid (DCMG) algorithm, for the details of which we shall refer the reader to Reference 3, is based on the concept of imperfect Newton iteration. It takes two difference formulations of the same problem, one accurate but difficult to cope with and another low-order but such that an easy and fast relaxation method is available, and performs an approximate Newton iteration process by conceptually solving at each iteration a linear equation system which has the derivative matrix of the low-order difference equations as coefficient matrix and the sign-changed residuals of the high-order ones as known terms, using for this purpose a multigrid[4] algorithm based on the available relaxation procedure for the low-order formulation. In practice, no derivative matrix need ever be calculated explicitly, because the process is realized through the full-approximation storage[4] technique by just adding correction terms, similar to those that are already applied to the coarser levels of the multigrid structure, to the finest level as well, and calculating these terms as the difference of the low-order and high-order residuals, much as though the high-order equations represented an even finer multigrid level.

A peculiar characteristic of the DCMG method is that the high-order equations, in spite of being the very ones which are being solved, are used sparingly in the calculation. Indeed, they only appear in a single subroutine which calculates their residuals, and this subroutine is executed only once per multigrid cycle. It follows that the high-order equations, even if complicated, have comparatively little effect on total computation time and that it is very easy to experiment with different high-order formulations by simply placing the relevant difference equations in that single subroutine.

A suitable first-order formulation of the Navier–Stokes equations which makes an explicit Gauss–Seidel relaxation routine usable as the 'engine' of the multigrid cycle was described in Reference 3. The DCMG algorithm using this routine was there seen to be as fast as the multigrid programmes based on implicit ADI or line relaxation routines at solving the second-order conservative form difference formulation of the ψ–ζ Navier–Stokes equations for the driven cavity problem.

Here we shall describe the use of the DCMG programme to test some new higher-order difference formulations of the Navier–Stokes equations. Notice that the only modification that

must be operated in order to do so is the replacement of the single routine that calculates the residuals of the selected formulation of the equations and boundary conditions.

3. BLOCK-POLYNOMIAL DISCRETIZATION

The fact that the solutions of the differential equation problems of mathematical physics are almost everywhere analytic, i.e. admit a Taylor series expansion, implies that a local nth-order polynomial interpolation obtained from a discrete representation of the solution is in principle able to approximate it with an error which goes down with mesh size h asymptotically as h^{n+1}. The choice of a polynomial interpolation suitable for the solution of a differential equation problem, however, is anything but unique, and when relatively high-order polynomials are considered a wide range of possibilities opens up.

Existing methods of discretization may be classified into three categories: strictly finite difference methods, which use a one-dimensional polynomial fitted to points in a neighbourhood of any given mesh point to evaluate derivatives of the solution at that point; spline methods, which use a polynomial spline fitted to all the points located along a given co-ordinate line to evaluate derivatives of the solution at each point along that line (see e.g. References 1 and 5); finite element methods, which use a multidimensional polynomial fitted to all the points which fall inside or on the boundary of a certain portion (finite element) of the calculation region to evaluate derivatives at all the points where this is needed inside that same region. (To be sure, finite element programmes most often—but not always, see e.g. Reference 6—use integrated weak formulations of the differential equations and evaluate integrals rather than derivatives, but this is irrelevant in the present context.)

Methods of the first and second class require special treatment of the boundaries, since neither the additional points necessary for finite difference formulae nor the derivatives of the main unknowns which are used as auxiliary variables in spline methods are known there. These added unknowns at the boundary must be determined through 'additional boundary conditions', which may be obtained from extrapolation formulae or by taking derivatives of the differential equations themselves. This difficulty is intrinsically related to another, more subtle, one: the difference equations turn out to be of a higher order than the differential equations they represent (which is why they require additional boundary conditions) and therefore they support, in addition to modes which tend, for mesh size going to zero, to the modes of the differential problem, other 'unphysical' modes which have no correspondent in the original problem. Hopefully, these additional modes should be well behaved and fade away under the effect of suitably chosen boundary conditions, but if a higher-order scheme which would *a priori* seem workable suddenly turns out numerically unstable in a case where a lower-order scheme is stable, the cause may often be identified in an unphysical mode, either intrinsically unstable or made so by an unfortunate choice of additional boundary conditions. At least, this identification is possible in a linear context. The need for additional boundary conditions and the lack of insight about what effects unphysical modes may cause in non-linear problems have contributed to creating the slight sense of mistrust that some numerical analysts feel for higher-order difference methods.

The birth of unphysical modes and the consequent necessity of additional boundary conditions when higher-order difference schemes are employed may easily be verified in the context of a one-dimensional linear problem, for instance the second-order advection–diffusion equation

$$f_{xx} - uf_x = g, \tag{1}$$

where $u(x)$ and $g(x)$ are known functions and f is the unknown. If a standard finite difference

approximation using, say, a fourth-order polynomial is adopted for f_{xx} and f_x, (1) is transformed into a difference equation of the form

$$a_{-2,n}f_{n-2} + a_{-1,n}f_{n-1} + a_{0,n}f_n + a_{1,n}f_{n+1} + a_{2,n}f_{n+2} = g_n, \tag{2}$$

where, in the case just mentioned and for a uniform discretization with spacing h, $a_{-2,n} = -1/12h^2 + u_n/12h$, $a_{-1,n} = 4/3h^2 - 2u_n/3h$, $a_{0,n} = -5/2h^2$, $a_{1,n} = 4/3h^2 + 2u_n/3h$ and $a_{2,n} = -1/12h^2 - u_n/12h$.

Equation (2) is a difference equation coupling five adjacent points, and as such is expected to have four independent homogeneous solutions. These may be found as Fourier modes for constant or frozen coefficients or, more generally, as the solutions that can be obtained by arbitrarily assigning four adjacent values of f in four linearly independent ways and then calculating all the other values in a sequence through (2) by marching forwards and backwards from there. In particular, on assuming frozen coefficients (i.e. $u_n = \bar{u}$, independent of n) and $f_n = \lambda^n$, Fourier analysis reduces to solving the eigenvalue equation

$$\bar{u}h - 1 + (16 - 8\bar{u}h)\lambda - 30\lambda^2 + (16 + 8\bar{u}h)\lambda^3 - (1 + \bar{u}h)\lambda^4 = 0, \tag{3}$$

the four roots of which correspond to the four modes. In the continuum limit ($h \to 0$) one would expect the roots of (3) to tend to those of the continuum mode equation, i.e. the Fourier transform of the LHS of (1), namely

$$-\beta^2 + i\beta\bar{u} = 0, \tag{4}$$

under the substitution $\lambda = e^{i\beta h}$. However, it is immediately apparent that (3) has four roots and (4) has two, so that a one-to-one correspondence is impossible. In fact, it may easily be computed that two roots of (3) tend to unity in the limit for $h \to 0$, and in a higher approximation may be seen to approach the values of λ which correspond to the two roots of (4), but the other two have as limiting values $7 \pm \sqrt{48}$. It is thus seen that two unphysical modes exist in the difference equation which do not have any correspondent in the differential problem. Therefore two additional boundary conditions must be imposed which basically have the purpose of selecting a solution from which these modes are absent.

Is there a way to get rid of unphysical modes and additional boundary conditions? An answer is contained both in the Runge–Kutta method for ordinary differential equations and in finite element methods for partial differential equations. In both cases one and the same higher-order polynomial (either in one or more dimensions) is employed over the whole range of mesh points that were used to define it, rather than switching to a new polynomial for every mesh point. (In the case of the Runge–Kutta method there are additional complications necessary in order to make the calculation explicit, but the relevant concept is that a number of evaluations of the differential equation are co-operatively used to create a higher-order approximation of a new value of the unknown and then the calculation starts afresh from this value without keeping track of the intermediate results.)

A polynomial interpolation scheme suitable for boundary value problems involving second-order differential equations, which has the property of permitting higher-order formulations without introducing unphysical modes and additional boundary conditions, is shown, in one dimension and in the case of a fourth-degree polynomial, in Figure 1. We call this scheme a 'block polynomial'.

As may be seen in the figure, in the standard finite difference scheme a new fourth-order polynomial is fitted to every quintuple of points and used only for the central point of the quintuple. As a result, every difference equation is coupled to four others, rather than to two as in the minimal second-order scheme, and two unphysical modes may arise.

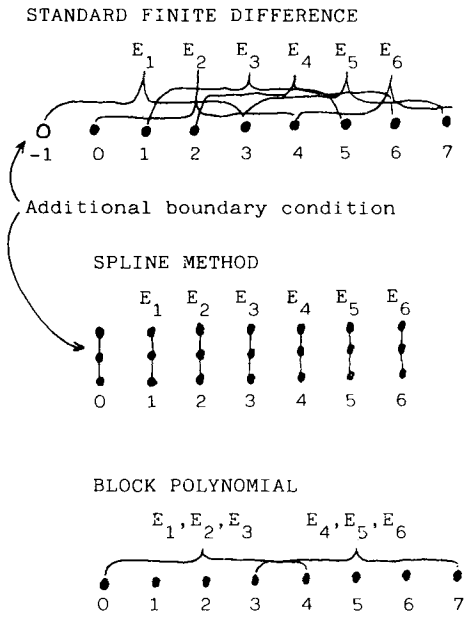

Figure 1. Mesh points encompassed by successive difference equations $E_1, E_2, \ldots, E_6$ in three differencing schemes

In the spline methods additional spline variables are stored at every mesh point and used to define a different polynomial in every interval between two points; the differential equations are applied only to the spline variables (which represent the first and second derivatives of the unknown) at a single point while difference equations coupling two or three adjacent points are derived from a variety of derivative continuity requirements. (Incidentally, it is perhaps unfair to compare the spline with the other methods on the basis of an equal mesh size as was done in Reference 1; it is a defendable position that they should be compared on the basis of an equal number of stored quantities and therefore with a threefold mesh size.) The need for additional boundary conditions arises from the missing continuity links at the edges of the calculation interval.

In the block-polynomial scheme a, say, fourth-degree polynomial is fitted to a block of five points and used to evaluate derivatives appearing in the differential equation at the three interior points of the block. In this way three difference equations in five unknowns are obtained.

In order to see how this scheme gets rid of unphysical modes, we may think of eliminating the central unknown of each block (for the purpose of the demonstration only, there is no need to actually perform these steps in the computer programme) through the corresponding equation, say unknown 2 through equation E_2 with reference to Figure 1, and re-express the remaining equations as a vector difference equation of the form

$$\mathbf{A}_{-1,n}\mathbf{f}_{n-1} + \mathbf{A}_{0,n}\mathbf{f}_n = \mathbf{g}_n, \tag{5}$$

where $\mathbf{A}_{-1,n}$ and $\mathbf{A}_{0,n}$ are 2×2 matrices and the vectors $\mathbf{f}_n$ are formed by the pairs of overlap variables (in Figure 1: f_0 and f_1; f_3 and f_4; f_6 and f_7; etc.). The transformation to (5) is possible for approximations of any order and not only for the fourth-order polynomial chosen as an example; it is in fact always possible in the block-polynomial scheme to eliminate the internal variables and re-express the difference equations in terms of the overlap variables only.

Equation (5), being a two-vector difference equation that couples adjacent vectors only, may readily be seen to admit exactly two propagation modes. These may be obtained, just as before, by arbitrarily choosing an initial vector in two linearly independent ways and working forwards and backwards from there, or in the case of constant or frozen coefficients by Fourier analysis. In the latter case one must set $\mathbf{f}_n = \lambda^n \mathbf{F}$; inserting this position into the homogeneous part of (5) and assuming that $\mathbf{A}_{-1}$ and $\mathbf{A}_0$ are independent of n gives the following linear homogeneous system in the vector $\mathbf{F}$:

$$(\mathbf{A}_{-1} + \lambda \mathbf{A}_0)\mathbf{F} = 0. \tag{6}$$

The compatibility condition of this system is a second-degree equation in λ which gives the two eigenvalues; the solution of (6) then gives the two eigenvectors $\mathbf{F}$. It is thus seen that the number of modes is, for approximations of any order, the same as that of the differential problem, and no unphysical modes arise.

Corresponding to the presence of only two modes, the block-polynomial scheme requires only two boundary conditions to determine a unique solution.

It will have been noticed that the block-polynomial scheme looks more or less like a one-dimensional version of a finite element scheme, except for the unusual feature of block overlap, whose equations are derived by point collocation (an extreme case of weighted integrals). In two or more dimensions, however, rather than resorting to general multidimensional polynomials, the block-polynomial scheme allows substantially identical formulae to be applied independently along each co-ordinate line to evaluate derivatives. Moreover, since the same interpolating polynomial is to be used for calculating more than one quantity at more than one point, it is reasonably run-time-effective to use a general procedure that calculates Lagrange's interpolation polynomial through Newton's representation[7] to obtain the polynomial's coefficients and then evaluate the required derivatives from the latter, rather than analytically calculating the expressions of the derivatives in terms of the function's values as is usually done in finite difference codes. Doing so considerably simplifies the programming and, as an added benefit, allows any-order approximations and uniform as well as variable meshes to be handled by a single programme.

As will be seen from the examples, the practical application of the block-polynomial approximation has been made convenient by the DCMG algorithm, which affords a converged solution in about the same time for the high-order as for the low-order discretization. The same algorithm can also solve the equations obtained from standard higher-order finite difference or spline approximations with suitable additional boundary conditions. The block-polynomial scheme, however, although sometimes it does not afford the highest order of approximation that the symmetry of the equations allows, retains the advantage of being applicable to any-order approximations on uniform as well as variable meshes with greater ease of programming.

4. APPLICATION TO THE NAVIER–STOKES EQUATIONS

The block-polynomial scheme has been applied to the two-dimensional, steady, incompressible Navier–Stokes equations in ψ–ζ form:

$$\psi_{xx} + \psi_{yy} - \zeta = 0, \tag{7}$$

$$\zeta_{xx} + \zeta_{yy} - Re(\psi_y \zeta_x - \psi_x \zeta_y) = 0, \tag{8}$$

where ψ is the streamfunction, ζ the vorticity and Re the Reynolds number. Both a non-conservative and a conservative formulation have been considered. The non-conservative formu-

lation simply involves evaluating the derivatives that appear in (7) and (8) from suitable block-polynomial interpolations and then calculating the residuals of the two equations for use in the DCMG algorithm. In order to give a conservative formulation, in which a vorticity flux appears that is exactly conserved in the discrete equations, (8) was integrated over a four-mesh rectangle (not necessarily a square because allowance is made for co-ordinate stretching) surrounding the given mesh point of co-ordinates x_i, y_j and, after transforming the integral by Gauss's formula, divided by the area of this rectangle, thus becoming

$$\left(\int_{y_{j-1}}^{y_{j+1}} (\zeta_x - Re\psi_y\zeta)_{x_{i+1}}\,dy + \int_{x_{i-1}}^{x_{i+1}} (\zeta_y + Re\psi_x\zeta)_{y_{j+1}}\,dx - \int_{y_{j-1}}^{y_{j+1}} (\zeta_x - Re\psi_y\zeta)_{x_{i-1}}\,dy - \int_{x_{i-1}}^{x_{i+1}} (\zeta_y + Re\psi_x\zeta)_{y_{j-1}}\,dx\right) \Big/ [(x_{i+1}-x_{i-1})(y_{j+1}-y_{j-1})] = 0. \tag{9}$$

The line integrals of (9) were evaluated by fitting a new block polynomial, through the Lagrange–Newton procedure, to each of the integrand quantities and integrating the polynomial between appropriate bounds. In the limit for mesh spacing going to zero, of course, (9) reduces to (8). For finite mesh size, however, they are not exactly equal, and in the conservative discrete formulation the residual of (9) was driven to zero rather than that of (8). Exact discrete conservation stems from the same value of each line integral being used in the equation corresponding to either of the points facing each other across the given line.

In order to complete the description of the block-polynomial method for the Navier–Stokes equations we must also discuss the treatment of boundary conditions. The most frequently encountered boundary conditions for this problem are velocity conditions, which specify that the streamfunction and its normal derivative should take on given values on the boundary. The condition on the value of the streamfunction is explicit and gives no problems. The normal derivative condition can be imposed to the required order of accuracy in the block-polynomial approach by simply equating the wall derivative of the polynomial which interpolates ψ in the block next to the wall to the required wall velocity u_w. Doing so gives an implicit condition on the values of the streamfunction used to calculate the interpolation polynomial which properly closes the set of difference equations.

Although the above approach to boundary conditions does work out in practice, it was found that a certain improvement in speed of the DCMG algorithm could be obtained by giving a slightly different, though analytically equivalent, normal derivative boundary condition. In fact, the main assumption underlying the DCMG algorithm is that the high-order difference equations whose residuals are to be driven to zero and the low-order equations which are used in the multigrid iterative cycle to achieve this aim are different-order discrete formulations of the same differential equations and boundary conditions.

Now, the boundary condition used inside the low-order formulation is the standard Thom equation[8]

$$(\psi_{nw} - \psi_w)/h + (h/2)(\psi_{wtt} - \zeta_w) = u_w, \tag{10}$$

where ψ_w and ψ_{nw} are the values of the streamfunction at the wall and next-to-wall points, separated by a distance h, ψ_{wtt} is the second tangent derivative of the streamfunction at the wall, which may be obtained from the streamfunction boundary condition, ζ_w is the unknown wall value of vorticity and u_w is the value of wall velocity to be imposed. Equation (10) gives an approximation of the wall derivative from a second-order Taylor expansion of the streamfunction at the wall in which the second derivative is eliminated by using the equation $\Delta_2\psi = \zeta$. In this way it lets ζ_w enter the calculation, which is fundamental in the explicit relaxation cycle.

Equation (10) more closely represents a linear combination of the boundary condition $\partial\psi/\partial n = u_w$ and the equation $\Delta_2\psi = \zeta$ rather than the boundary condition alone. Of course, such a combination is allowed in the differential problem, at least on smooth boundaries, since both the boundary condition and the equation must be satisfied anyhow, but again in the discrete problem there may be a slight difference. We found that a somewhat faster convergence of the DCMG algorithm is obtained if a similar combination is adopted in the high-order as well as in the low-order formulation, thus requiring that

$$P'_{\psi w} + (h/2)(P''_{\psi w} + \psi_{wtt} - \zeta_w) = u_w, \tag{11}$$

where $P'_{\psi w}$ and $P''_{\psi w}$ are the first and second derivatives at the wall of the polynomial approximating ψ in the block next to the wall. Equation (11) reduces to (10) for a second-order polynomial and represents for a polynomial of any order a comparable-order approximation of a linear combination of the two equations $\partial\psi/\partial n = u_w$ and $\Delta_2\psi = \zeta$.

5. RESULTS FOR THE DRIVEN CAVITY PROBLEM

A programme written according to the above discretization scheme was tried on the well-known driven cavity problem, which consists of resolving the steady flow of an incompressible fluid in a square cavity, one of whose walls slides tangentially along at constant speed and drives the fluid into motion.

Tables I–IV report the values of three parameters (streamfunction and vorticity at the cavity centre, ψ_{CC} and ζ_{CC}, and vorticity at the moving wall centre, ζ_{MWC}) calculated with the standard second-order central difference scheme, CD2, and with the third-order block-polynomial scheme, BP4 (i.e. the one using a fourth-degree polynomial, which is expected to yield a third-order approximation of the second derivatives), using conservative and non-conservative formulations at various Reynolds numbers and for several different grid sizes. The last row in each table gives the value obtained by extrapolation from the last two results of the BP4 method (in the most precise formulation in the case of $Re = 100$ where several are available), which can be used as a substitute of the exact solution for the purpose of estimating errors.

In particular, Table I contains a comparison of the non-conservative schemes at $Re = 100$. The BP4 scheme turns out better, especially on the value of ζ_{MWC}, which in all tests appeared to be the most prone to errors, but the advantage is lost if the non-conservative BP4 scheme is compared with the conservative CD2 scheme (whose results are given in Table II). This outcome is consistent with the conclusions of other authors[2] who found that the spline-based non-conservative fourth-order schemes are not competitive with the conservative central difference scheme for

Table I. Driven cavity results for $Re = 100$ (non-conservative schemes)

	ψ_{CC}		ζ_{CC}		ζ_{MWC}	
No. of points	CD2	BP4	CD2	BP4	CD2	BP4
17×17	−0·05248	−0·06858	0·7758	1·2124	8·461	6·670
29×29	−0·06166	−0·06735	1·0295	1·1944	7·131	6·370
41×41	−0·06415	−0·06699	1·1025	1·1858	6·822	6·516
65×65	−0·065512	−0·066675	1·14381	1·17775	6·6457	6·5485
Extrapolated values:	−0·066545	−0·066524	1·17416	1·17421	6·5636	6·5638

CD2, second-order central difference scheme.
BP4, fourth-degree block-polynomial scheme.

Table II. Driven cavity results for $Re = 100$ (conservative schemes)

	ψ_{CC}		ζ_{CC}		ζ_{MWC}	
No. of points	CD2	BP4	CD2	BP4	CD2	BP4
11 × 11	−0·05896	−0·05084	0·8827	0·6533	7·641	8·744
17 × 17	−0·06204	−0·06094	1·0054	1·0102	7·372	7·484
23 × 23	−0·06393	−0·06610	1·0766	1·1535	7·017	6·503
29 × 29	−0·06491	−0·066817	1·1127	1·1853	6·830	6·389
41 × 41	−0·065670	−0·066870	1·1449	1·18566	6·688	6·4939
65 × 65	−0·066219	−0·066652	1·16314	1·17824	6·6093	6·5488
89 × 89	−0·066344	−0·066513	1·16759	1·17411	6·5836	6·5515
11 × 11s	−0·05971	−0·04237	0·8641	0·4765	7·216	8·348
17 × 17s	−0·06241	−0·06425	1·0267	1·0962	6·851	6·045
23 × 23s	−0·06493	−0·06677	1·0918	1·1639	6·713	6·403
29 × 29s	−0·065147	−0·066331	1·1226	1·1732	6·654	6·511
41 × 41s	−0·065860	−0·066438	1·1487	1·17526	6·6075	6·5479
65 × 65s	−0·066280	−0·066492	1·16426	1·17475	6·5806	6·5606
89 × 89s	−0·066405	−0·066512	1·16892	1·17442	6·5726	6·5626
Extrapolated values:	−0·066545	−0·066524	1·17416	1·17421	6.5636	6·5638

s, stretched co-ordinates.

Table III. Driven cavity results for $Re = 10$ (conservative schemes)

	ψ_{CC}		ζ_{CC}		ζ_{MWC}	
No. of points	CD2	BP4	CD2	BP4	CD2	BP4
11 × 11	−0·05888	−0·05964	0·7853	0·7901	5·969	5·948
17 × 17	−0·05902	−0·05917	0·7847	0·7855	5·896	5·815
23 × 23	−0·05900	−0·05902	0·7842	0·7838	5·878	5·847
29 × 29	−0·058998	−0·058976	0·78397	0·78349	5·8740	5·8588
41 × 41	−0·058989	−0·058959	0·78386	0·78346	5·8717	5·8653
Extrapolated values:	−0·058980	−0·058950	0·78375	0·78344	5·8695	5·8634

Table IV. Driven cavity results for $Re = 1000$ (conservative schemes)

	ψ_{CC}		ζ_{CC}		ζ_{MWC}	
No. of points	CD2	BP4	CD2	BP4	CD2	BP4
89 × 89	−0·11351	−0·11640	2·0201	2·0675	15·429	14·067
41 × 41s	−0·11092	−0·11678	1·9840	2·0791	15·979	13·547
65 × 65s	−0·11439	−0·11640	2·0335	2·0658	15·178	14·418
89 × 89s	−0·11537	−0·11652	2·0491	2·0671	14·966	14·638
Extrapolated values:	−0·11647	−0·11660	2·0666	2·0679	14·728	14·775

mesh sizes in a similar range. If, however, the conservative BP4 scheme is compared with the conservative CD2 scheme, as is done, again for $Re = 100$, in Table II, the advantage reappears, in connection with both uniform and stretched grids (the results marked 's' refer to a grid obtained by applying independently to the x- and y-co-ordinates a 'stretching' transformation of the type

$x=[1+\tanh(2\xi-1)/\tanh(1)]/2$ and choosing a uniform spacing in the ξ-co-ordinate, similar to what was done in Reference 2 and by many other authors).

The superiority of the conservative BP4 scheme is confirmed by Tables III and IV, which report similar data for $Re=10$ and 1000 respectively. It will be noticed that at $Re=10$ BP4 passes CD2 slightly earlier than at $Re=100$, whereas at $Re=1000$ an advantage only appears for rather high values of the number of points in the mesh. (Indeed, results for $Re=1000$ with less than 40 meshes per side are not even reported because BP4 needs at least this number to give any result at all.) This trend is, in our opinion, related to the general decrease in precision for a given number of points, or increase in number of points necessary for a given precision, which stems from the appearance of smaller-and-smaller-scale details in the flow field with increasing Reynolds number; the gradual loss of ground of the higher-order with respect to the lower-order schemes may be ascribed to this general loss of precision, since it is at very precise calculation that higher-order methods are better.

In order to show that the gain of precision of BP4 is obtained with no substantial cost in computation time, Figures 2–4 report the convergence histories of three typical runs, $Re=100$ on stretched and unstretched (65×65)-point grids and $Re=1000$ on a stretched (65×65)-point grid, obtained with the CD2 and BP4 formulations starting from all zero initial values and ending when machine-limited accuracy is reached. (The plots are timed in 'work units', as is usual for multigrid algorithms, one work unit corresponding to one Gauss–Seidel relaxation cycle on the finest grid.) In all cases the convergence of BP4 turns out just marginally slower than that of CD2. To test the ability of the DCMG algorithm of working also in connection with the standard higher-order discrete formulation, we have also run one case using central fourth-order differences at all but the next-to-wall points and a non-centred interpolation using a fifth-degree polynomial there, so as to obtain a uniformly fourth-order approximation. The convergence history of this run, reported in Figure 5, shows no significant difference from the previous cases.

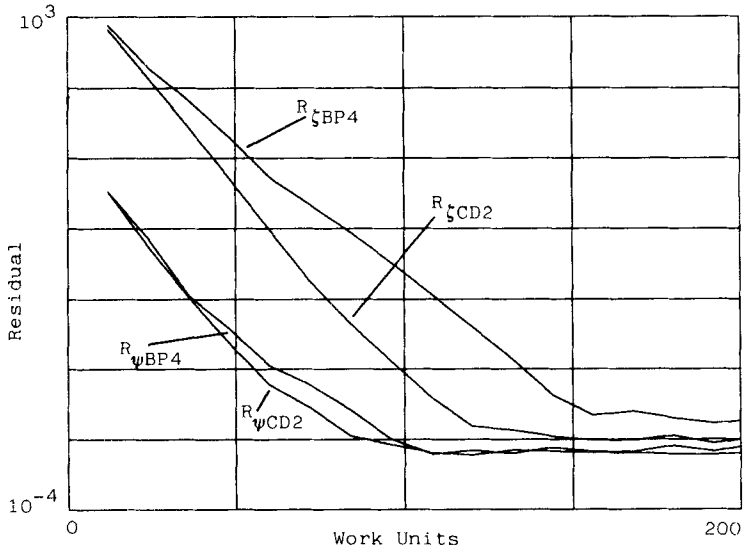

Figure 2. Convergence history of the conservative BP4 and CD2 algorithms using a uniform (65×65)-point grid at $Re=100$. Shown is the maximum absolute value of the residual of the streamfunction and vorticity equations as a function of time in work units

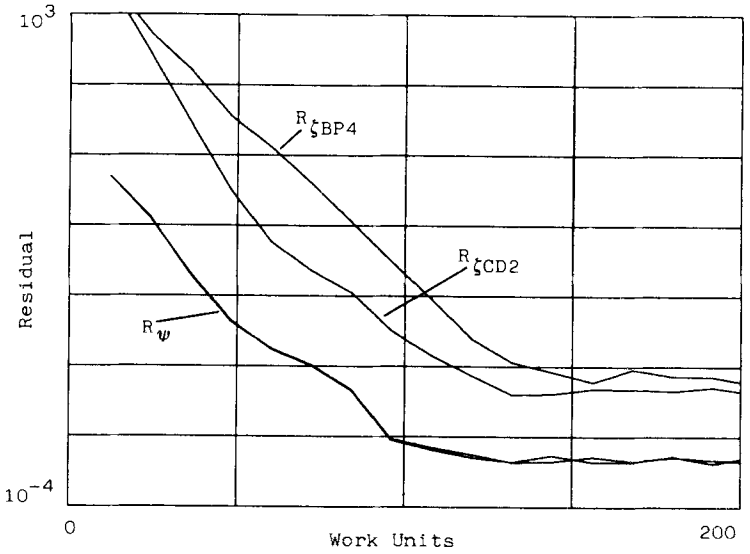

Figure 3. Convergence history of the conservative BP4 and CD2 algorithms using a stretched (65 × 65)-point grid at $Re = 100$

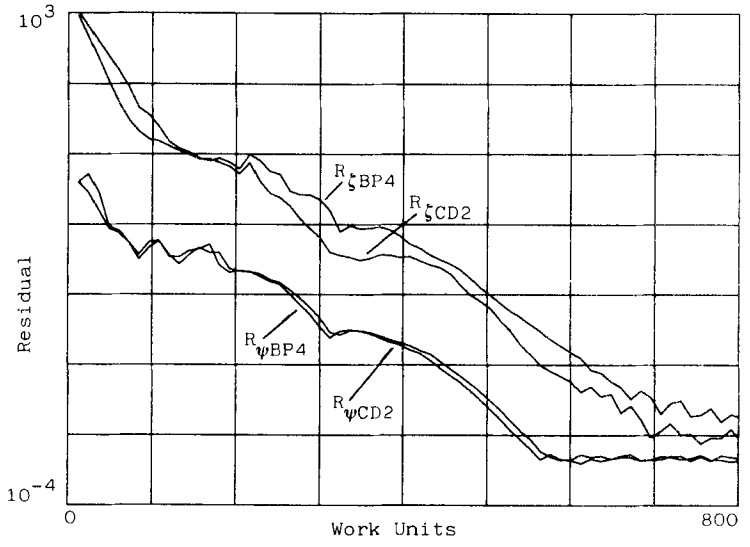

Figure 4. Convergence history of the conservative BP4 and CD2 algorithms using a streched (65 × 65)-point grid at $Re = 1000$

The final values obtained, $\psi_{CC} = -0{\cdot}066612$, $\zeta_{CC} = 1{\cdot}17727$ and $\zeta_{MWC} = 6{\cdot}5501$, appear to be comparable with those given, for the same number of meshes, by BP4.

It may be concluded that whereas the advantage of conservative over non-conservative schemes is considerable, and sufficient to make a second-order scheme comparable in accuracy to

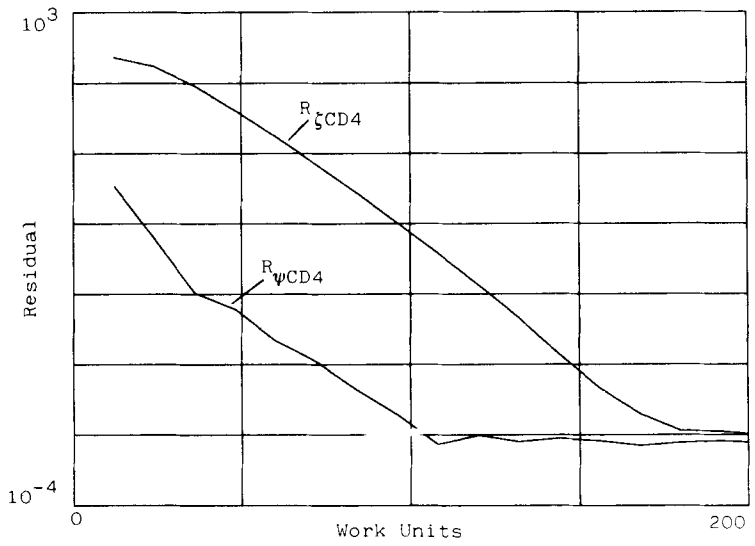

Figure 5. Convergence history of the fourth-order centred finite difference algorithm using a uniform (65 × 65)-point grid at $Re = 100$

a non-conservative third- or fourth-order one, conservative higher-order methods restore the balance, and conservative higher-order methods of the block-polynomial type *are* definitely to be preferred to the standard central difference method whenever precise (say, better than a few per cent) results are wanted. This statement may be considered to be valid for any Reynolds number provided it is remembered that a similar precision requires a much larger number of mesh points at a high than it does at a low Reynolds number.

6. THE CORNER SINGULARITY

The driven cavity problem has become a standard test case for numerical methods of solving the Navier–Stokes equations, because its square geometry fits well a square calculation grid without the added difficulties of dealing with a curved boundary, because the only characterizing parameter is the Reynolds number defined with the cavity side, the wall velocity and the fluid viscosity, because a non-trivial range of different behaviours is obtained with varying Reynolds number, but even more because of the great number of data that have been accumulated in time (see e.g. Reference 9). There is, however, one defect in this problem as a test case: the two corners where the moving wall meets the other, steady, walls give rise to analytical singularities in the solution, and since polynomial approximations are certainly unsuitable in the proximity of these singular points, a doubt always lurks around that the solution method under test may perform less brilliantly than it would if applied to other, non-singular, problems. This doubt is particularly well justified in the case of comparing higher- with lower-order difference approximations, since the former may be affected more heavily by singularities than the latter, and in fact the presence of singularities was one of the causes conjectured by Giannattasio and Napolitano[2] for the unexpectedly low performance of their fourth-order spline method. The results presented in this paper too might rightly be thought to be biases because of the presence of these singularities.

The above doubts can be removed to a large extent if the local behaviour of the solution near the singularity is determined analytically and compensated for. The key to doing so is in

observing that, just because velocity gradients become infinitely large at these corners, viscous effects dominate there, and the local behaviour may therefore be obtained from the Stokes equations, i.e. the Stokes–Navier equations with convective terms neglected. The problem is thus locally reduced near each corner to the one depicted in Figure 6: solving the biharmonic equation for the streamfunction in the space enclosed between two semi-infinite straight lines at a right angle with conditions specifying that on one wall both components of velocity are zero and on the other the normal velocity is zero and the tangential velocity is unity.

This problem is an instance of a class which are easily solved by separation of variables in polar co-ordinates. The ψ–ζ Stokes equations have, as may readily be verified, a family of solutions of the form $\psi = r^p f_p(\theta)$, $\zeta = r^{p-2} g_p(\theta)$, where r, θ are polar co-ordinates and $f_p(\theta)$, $g_p(\theta)$ satisfy the equations

$$f'' + p^2 f = g, \qquad g'' + (p-2)^2 g = 0. \tag{12}$$

The boundary conditions for the problem of Figure 6 in polar co-ordinates are:

$$\psi(r, 0) = \psi(r, \pi/2) = 0, \qquad r^{-1}\psi_\theta(r, 0) = 0, \qquad r^{-1}\psi_\theta(r, \pi/2) = 1.$$

In order for the last of these conditions to be verified, the exponent p must equal unity. The conditions for f are then obtained:

$$f(0) = f(\pi/2) = 0, \qquad f'(0) = 0, \qquad f'(\pi/2) = 1. \tag{13}$$

Four independent integrals of (12) for $p = 1$ are given by $f = \sin\theta$, $\cos\theta$, $\theta\sin\theta$ and $\theta\cos\theta$. Determining a linear combination of these integrals such that (13) are verified finally gives

$$\psi = r[(\pi/2 - \theta)\sin\theta - (\pi/2)\theta\cos\theta]/(\pi^2/4 - 1), \tag{14}$$

$$\zeta = r^{-1}(\pi\sin\theta - 2\cos\theta)/(\pi^2/4 - 1). \tag{15}$$

As is seen, ζ is infinite as r^{-1} in the origin, so that its first and second spatial derivatives are of the

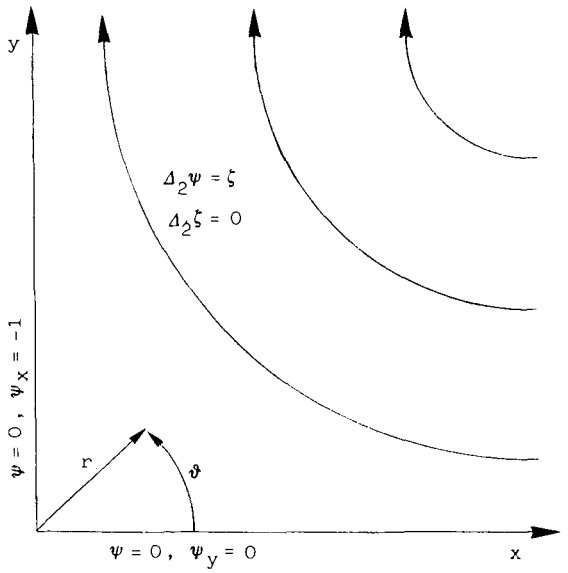

Figure 6. Local description of the corner singularity by the Stokes equations

order of r^{-2} and r^{-3} respectively and the assumption that convective terms are locally negligible with respect to viscous terms in the vorticity–transport equation is self-consistently verified. With a suitable choice of the co-ordinate axes, (14) and (15) represent the local behaviour of the solution near both moving corners of the driven cavity.

Once a local approximation is available, the polynomial interpolation procedures of the Section 3 may be applied, in the one or two blocks nearest to each corner, to the difference between the current iterate of the sought-after solution and its local approximation, rather than to the full solution. Even if the approximation given by (14) and (15) is not sufficient to ensure that this difference is analytic (i.e. admits a Taylor series expansion) at the corners, it does ensure that the difference is much smaller than the leading term and no longer goes to infinity, and therefore should strongly diminish any adverse effects that the presence of singularities may have on the overall accuracy of the solution obtained.

The results of analytically taking account of the corner singularities are shown in Figure 7 and Table V. Figure 7 shows the values of tangential velocity at the moving wall obtained from the

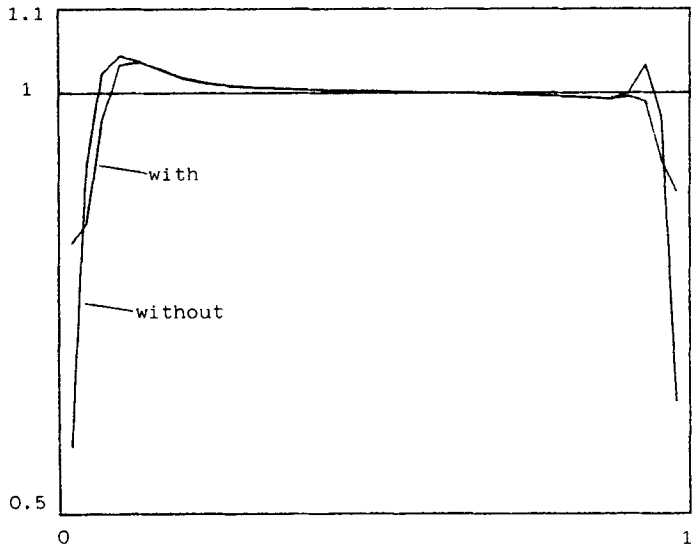

Figure 7. Wall velocity profiles obtained with and without corner singularity correction

Table V. Effectiveness of corner singularity correction ($Re = 100$)

	ψ_{CC}		ζ_{CC}		ζ_{MWC}	
No. of points	With	Without	With	Without	With	Without
17 × 17	−0·06253	−0·06154	1·0418	1·0091	7·611	7·474
29 × 29	−0·06704	−0·06698	1·1850	1·1832	6·391	6·379
41 × 41	−0·06688	−0·06692	1·1825	1·1838	6·491	6·486
17 × 17s	−0·06704	−0·06547	1·1364	1·0962	6·007	6·045
29 × 29s	−0·06673	−0·06676	1·1715	1·1732	6·511	6·511
41 × 41s	−0·066143	−0·066649	1·1741	1·1753	6·5483	6·5479

polynomial approximation of the streamfunction normal derivative in the solutions obtained with and without corner singularity corrections. It will be remembered that the boundary condition of (11) does not automatically require that this derivative be unity, so that its discrepancy from unity is a measure of the local accuracy of the solution. It may be seen that very near the corners the difference is significant, meaning that analytically taking account of the singularities has brought an improvement in local accuracy; going away from the corners, however, the difference between the two solutions fades rapidly and soon becomes unnoticeable. This is confirmed by Table V, which reports the three parameters ψ_{CC}, ζ_{CC} and ζ_{MWC} calculated for various Reynolds numbers both with and without corner singularity corrections.

It may be concluded that corner singularities are not to be considered an important cause of overall inaccuracy in numerical solutions of the driven cavity problem as long as attention is not focused on flow in the proximity of the corners themselves. If an accurate calculation of just this local flow is wanted, analytical correction through (14) and (15) is an effective technique.

7. CONCLUSIONS

A novel discretization scheme has been tested which, although it employs, just as the standard finite difference schemes, one-dimensional interpolating polynomials for the purpose of calculating derivatives, shares with finite element schemes the property of presenting neither unphysical modes nor the need for additional boundary conditions.

Practical use of this scheme has been made possible by the DCMG algorithm of Reference 3, which allows a large freedom in the choice of the discretization scheme and, in particular, has turned out to be able to find a converged solution of a higher-order formulation of the ψ–ζ Navier–Stokes equations in about the same time that is needed for the standard central difference second-order discretization.

Both a conservative and a non-conservative form of the difference equations have been tested, the conservative form giving, just as in the low-order case, consistently better results. Whereas comparing the non-conservative BP4 (third-order) scheme with the conservative second-order one shows, just as found by other authors for other higher-order schemes,[2] unexciting performance for practical mesh sizes, the conservative BP4 scheme proves to be definitely worthwhile in all situations except when the mesh size is barely sufficient to yield a meaningful solution for the Reynolds number considered. It is to be emphasized that this increased performance is obtained at a negligible run-time cost from the DCMG algorithm.

Finally, an analytical study of the corner singularities which affect the driven cavity problem has given a way to describe the flow in these regions more accurately, but has also shown that the effect of these singularities is negligible as far as the accuracy of the solution far from the corners is concerned.

ACKNOWLEDGEMENT

This research was supported by the Italian Ministry of Public Education.

REFERENCES

1. S. G. Rubin and P. K. Khosla, 'Polynomial interpolation methods for viscous flow calculations', *J. Comput. Phys.*, **24**, 217–244 (1977).
2. P. Giannattasio and M. Napolitano, 'Soluzione numerica ad elevata accuratezza delle equazioni di Navier–Stokes in una "driven cavity"', *Proc. IX Conf. of AIMETA (Associazione Italiana di Meccanica Teorica e Applicata)*, Bari, 4–7 October 1988, AIMETA, Bari, 1988, pp. 453–456.

3. P. Luchini, 'A deferred-correction multigrid algorithm based on a new smoother for the Navier–Stokes equations', *J. Comput. Phys.*, in the press (1990).
4. A. Brandt, 'Multi-level adaptive solutions to boundary-value problems', *Math. Comput.*, **31,** 333–390 (1977).
5. M. Napolitano, 'Efficient ADI and spline-ADI methods for the steady-state Navier–Stokes equations', *Int. j. numer. methods fluids*, **4,** 1101–1115 (1984).
6. P. M. Gresho, S. T. Chan, R. T. Lee and C. G. Upson, 'A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. Part 1: Theory', *Int. j. numer. methods fluids*, **4,** 557–598 (1984).
7. D. E. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1969, Section 4.6.4.
8. J. P. Roache, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1976, Section III-C-2.
9. U. Ghia, K. N. Ghia and C. T. Shin , 'High-*Re* solutions for incompressible flow using the Navier–Stokes equations and a multigrid method', *J. Comput. Phys.*, **48,** 387–411 (1982).