JOURNAL OF COMPUTATIONAL PHYSICS 112, 316–333 (1994)

Exact Projections and the Lagrange–Galerkin Method: A Realistic Alternative to Quadrature*

A. PRIESTLEY

Institute of Computational Fluid Dynamics, Department of Mathematics, University of Reading, Whiteknights, Reading, United Kingdom

Received April 13, 1993; revised October 18, 1993

The Lagrange–Galerkin method has proved a very successful method in computational fluid dynamics (CFD), in practice, but it has suffered from question marks about its stability when the integrals are approximated by quadratures. In this paper we will prove instability for wider ranges of quadrature formulae than previously considered. We will also introduce an approximate integration technique on triangular grids that cheaply recovers the unconditional stability of the Lagrange-Galerkin method with no loss in accuracy. The same technique can also be used for the exact projection of data from one arbitrary triangular grid to another, different, arbitrary triangular grid. This could be used in grid adaptation algorithms or in triangular multigrid procedures.

1. INTRODUCTION

The Lagrange–Galerkin method is a scheme that combines the method of characteristics with a standard finite element procedure; see Benqué *et al.* [2], Bercovier *et al.* [3, 4], Douglas and Russell [12], Ewing *et al.* [18], Hasbani *et al.* [26], Lesaint [38], Pironneau [44], Russell [54], and Süli [61], for example. There are two versions of the scheme available, depending upon whether one travels forwards in time along the particle trajectory or backwards. In Morton and Priestley [41] these two schemes were labelled the weak Lagrange–Galerkin method and the direct Lagrange–Galerkin method. In this paper we shall only concern ourselves with the direct Lagrange–Galerkin method, although everything proved here can be equally well applied to either version of the scheme.

Below we shall briefly describe the derivation of the Lagrange Galerkin scheme. In Section 2 new results about the instability of quadrature will be proved. These show that even wider classes of interpolatory quadrature formulae than previously realized are unstable when used in conjunction with the Lagrange–Galerkin method. This makes the search for a stable implementation of the method on triangles if the scheme is to make full use of the advantages of the finite element procedure even more important. In Section 3 a new integration technique for the Lagrange–Galerkin method on triangles will be introduced that is really just an extension of the area-weighting method introduced by Priestley [45] and Morton *et al.* [42]. This involves performing an exact projection from one arbitrary triangular grid onto a different arbitrary triangular grid, and as such it has applications outside of the Lagrange–Galerkin method. To demonstrate its effectiveness a simple test problem is solved in Section 4 and preliminary results are also given for a standard test problem involving the Navier–Stokes equations.

Consider the Cauchy problem for the scalar, linear advection equation for $u(\mathbf{x}, t)$:

$$u_t + \mathbf{a}(\mathbf{x}, t) \cdot \nabla u = 0, \qquad \mathbf{x} \in \mathbb{R}^d, \qquad t > 0, \tag{1.1}$$

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \tag{1.2}$$

where u_0 belongs to $L^2(\mathbb{R}^d)$, d being the number of dimensions. For the purpose of proving certain theoretical results the velocity field $\mathbf{a}(\mathbf{x}, t)$ is assumed to be incompressible, i.e., $\nabla \cdot \mathbf{a} = 0 \; \forall \mathbf{x}, t$ but this is not a restriction in practice. We can define characteristics paths or trajectories, $\mathbf{X}(\mathbf{x}, s; t)$, in two ways, either as the solution to an ordinary differential equation,

$$\mathbf{X}(\mathbf{x}, s; s) = \mathbf{x}, \tag{1.3}$$

$$\frac{d\mathbf{X}(\mathbf{x}, s; t)}{dt} = \mathbf{a}(\mathbf{X}(\mathbf{x}, s; t), t), \tag{1.4}$$

or, if desired, as the solution of the integral equation,

$$\mathbf{X}(\mathbf{x}, s; t) = \mathbf{x} + \int_s^t \mathbf{a}(\mathbf{X}(\mathbf{x}, s; \tau), \tau)\, d\tau.$$

In order to simplify the notation we will denote the foot, or departure point, of the characteristic path at time t^n by $\mathbf{x}$

* The work reported here forms part of the research programme of the Reading/Oxford Institute for Computational Fluid Dynamics and was funded by the SERC.

0021-9991/94 $6.00

and its arrival point at time t^{n+1} by $\mathbf{y}$. In terms of the more general notation above, these are $\mathbf{x} = \mathbf{X}(\mathbf{y}, t^{n+1}; t^n)$ and $\mathbf{y} = \mathbf{X}(\mathbf{x}, t^n; t^{n+1})$. A unique (absolutely continuous) solution to Eqs. (1.3), (1.4) can be guaranteed if it is assumed that $\mathbf{a}(\mathbf{x}, t)$ belongs to the Bochner space $L^1(0, T; (W^{1,\infty})^d)$; see Mizohata [40], for example. The solution to the original partial differential equations (1.1), (1.2) are now given by the relation

$$u(\mathbf{X}(\cdot, t, t+\tau), t+\tau) = u(\cdot, t). \tag{1.5}$$

For an approximation, $U^n = \sum_j U_j^n \phi_j$, at time t^n, expanded in terms of finite element basis functions ϕ_j, the direct Lagrange–Galerkin method uses Eq. (1.5) to seek U^{n+1} in $L^2(\mathbb{R}^d)$, satisfying

$$\langle U^{n+1}, \phi_i \rangle = \int U^n(\mathbf{x})\, \phi_i(\mathbf{y})\, d\mathbf{y} \qquad \forall i. \tag{1.6}$$

Equation (1.6) is obtained by taking the weak form of (1.5), i.e., multiplying (1.5) by $\phi_i(\mathbf{y})$ and integrating over the whole domain with respect to $\mathbf{y}$, the L^2 inner product over $\mathbb{R}^d$ being denoted by $\langle \cdot, \cdot \rangle$. This is the same approach as that used by Bercovier and Pironneau [3], Douglas and Russell [12], and Pironneau [44], for example. More recently this approach has been recast and generalized, as a streamline diffusion scheme by Johnson [33] and Hansbo [24, 25]. There is also a closely linked finite difference scheme. A review of the semi-Lagrangian scheme as applied to meteorological problems can be found in Staniforth and Côté [59] and some examples of its use for non-meteorological problems can be found in Holly and Preissmann [30]. Schohl and Holly [56], Priestley [46], and Garcia-Navarro and Priestley [19]. The relationship between the finite difference scheme and the finite element method has been discussed in Bermejo [5].

A second, alternative, formulation has been proposed by Benqué *et al.* [2] and this is referred to as the weak formulation, or the weak Lagrange–Galerkin method, by Morton and Priestley [41] because the adjoint of the differential operator in Eq. (1.1) is applied to a test function. The advantages of this approach have been rediscovered in the use of the locally adjoint method (LAM) and the Eulerian–Lagrange locally adjoint method (ELLAM); see Herrera *et al.* [27–29] and Celia *et al.* [8, 9], for example. The results in Section 2 and the new method introduced in Section 3 are equally applicable to this formulation.

The convergence and unconditional stability of the Lagrange–Galerkin method applied to the solution of (1.1) is addressed in Morton *et al.* [42]. Also in this paper certain special cases are mentioned where the stated results can be improved upon. E.g., for one-dimensional constant linear advection the Lagrange–Galerkin method with piecewise linear elements on a uniform grid becomes third-order accurate. When $\mathbf{a}(\mathbf{x}, t)$ is a smooth function the method is second-order accurate on a non-uniform mesh. Other theoretical results can be found in Lesaint [38], Süli [61], Süli and Ware [63], and in Priestley [47], which contains a short review of some of the properties of the Lagrange–Galerkin method.

Apart from certain trivial problems, the integral on the right-hand side of (1.6) has to be approximated in some way. There appear to be three ways of accomplishing this in a stable manner, all of which have problems. Priestley [45] and Morton *et al.* [42] introduced the area-weighted Lagrange–Galerkin method which, in contrast to normal quadrature, approximates the velocity field in such a way that the integrals can then be carried out exactly. This approach is stable and converges and gives reasonable results. However, there is, formally, a severe loss of accuracy compared with the exactly integrated method and the high order quadrature methods. It was also considered difficult to generalise area-weighting to triangular grids. The EPIC algorithm of Eastwood, see Eastwood and Arter [17], for example, is essentially the direct Lagrange–Galerkin method. It overcomes the conditions of the stability theorem of Morton *et al.* [42] by using a quadrature that does not integrate quadratics exactly, namely a compound trapezium rule. This same rule (or one with fewer sub-intervals) must also be used to calculate the integrals that give the elements of the mass matrix. Since the Lagrange–Galerkin method loses accuracy rapidly if the full mass matrix is not used, this means that many sub-intervals need to be taken. This becomes very expensive in higher dimensions if a product rule is used. The spectral Lagrange–Galerkin method of Süli and Ware [63] uses globally smooth basis functions. Convergence and unconditional stability are proven for Fourier polynomials even under quadrature, as long as the relevant Gaussian quadrature is used. The problems with this method are its cost, its propensity to give oscillations, and, in common with spectral methods in general, the difficulty in applying it to arbitrary domains.

In practice, then, because of the difficulties with the above implementations, quadrature formulae have been generally used. Although Lesaint [38] gives theoretical results involving quadrature, little work had been done on this aspect of the method. In Morton and Priestley [41] some elementary results were given regarding the effects of quadrature. Possibly the most interesting result is the stability theorem given by Priestley [45] and Morton *et al.* [42], later extended to include diffusion by Süli [62], which may be achieved by a simple Fourier analysis of the constant coefficient advection equation. The consequences of this theorem are quite wide ranging. Gauss–Legendre quadratures are unconditionally unstable, whilst Newton–Cotes formulae, various Radau formulae, and Gauss–Lobatto quadrature formulae (of which vertex and Simpson's rule

are the lowest order versions) are only conditionally stable. Even though Gauss–Lobatto is one of the best quadratures as regards stability, in the case of Simpson's rule the region of stability is still only $[0, \frac{1}{3}]$, a considerable restriction compared with the unconditional stability of the exactly integrated scheme.

Up to a point these results are largely academic in that, for the schemes using the higher order quadratures it can be very hard to generate signs of instability, unless the CFL number is kept in a very specific range, because the regions of instability decrease in size and the amplification factor in these unstable regions becomes closer to unity. In calculations involving the Navier-Stokes equations with all the non-convective terms being treated implicitly, detecting instabilities becomes even more difficult. Using bilinear elements on rectangles for the rotating cone problem Morton *et al.* [42] were unable to make a 4×4 Gauss–Legendre quadrature go unstable, although Priestley [45] showed that for a one-dimensional constant coefficient example, where it was possible to maintain the CFL number in the unstable range, the scheme did eventually become unstable. The danger, in practical calculations with the Navier–Stokes equations, for example, is that near boundaries or around stagnation points the scheme will be exposed to fairly constant CFL numbers in the unstable region. Once the unstable modes in the solution have been excited they can then grow very quickly even with the more accurate integration schemes. However, we know of no examples where, in a physical situation, the quadrature instability has caused any problems. Priestley [47, 48], though, has suggested a version of the Lagrange–Galerkin method for use in solving the advection problems associated with multi-dimensional wave models; see Roe [52], for example. Without diffusion the problems caused by the evaluation of the integrals could become more apparent.

2. UNSTABLE IMPLEMENTATIONS OF THE LAGRANGE–GALERKIN METHOD

In Priestley [45] and Morton *et al.* [42] it was shown that when the right-hand side of (1.6) is evaluated using the most common types of polynomial interpolating quadrature, Gauss–Legendre and Gauss–Lobatto, the method becomes unconditionally unstable or conditionally stable, respectively, a severe reduction on the unconditional stability obtained with exact integration. We can talk of "the" Lagrange–Galerkin method here since Morton and Priestley [41] showed the equivalence of the two approaches, even when quadrature was used, for the constant coefficient advection equation used in the Fourier analysis. Here we present some new results on quadrature that again show wide classes of quadrature to be unstable for implementing the Lagrange–Galerkin method. Undoubtedly the exact projection method to be presented in the next section is far harder to implement than quadrature, but the following results provide a strong incentive for doing so. First, we recall the main result on quadrature from Priestley [45] and Morton *et al.* [42].

THEOREM 1 (Priestley [45]). *If the right-hand side of the Lagrange–Galerkin method, using piecewise linear elements on a uniform mesh, is approximated by a quadrature of the form*

$$\int_0^1 f(x)\,dx \approx w_0 f(0) + \sum_1^m w_k f(x_k) + w_{m+1} f(1), \quad (2.1)$$

where the weights $w_0, \ldots, w_{m+1}$ and the quadrature points $0 < x_1 < \cdots < x_m < 1$ are free to be chosen, except that we assume that the quadrature evaluates the integrals of quadratic polynomials exactly, then the method is unstable for CFL numbers $\nu \in (2w_0, x_1)$ if

$$2w_0 < x_1. \quad (2.2)$$

From Eq. (2.2) it is then trivial to show that Gauss–Legendre quadrature, for which $w_0 = 0$ leads to unconditional instability. A little more work gives conditional stability results for Gauss–Lobatto and Newton–Cotes integration formulae. We now prove some new results for quadrature-based schemes.

THEOREM 2. *There are no unconditionally stable implementations of the Lagrange–Galerkin method with piecewise linear basis functions using symmetric four-point quadratures of the form* (2.1) *that integrate quadratics exactly.*

Remark. By the Lagrange–Galerkin method it is meant the method using the full mass matrix. Mass lumped, full or partial, variants of the scheme are not included in this theorem. Symmetry does not seem to be an overwhelming constraint on the quadrature since all the integration schemes used in practice are of this form. In one dimension, for constant coefficient linear advection, the idea of using a Gauss–Radau type quadrature, changing the fixed end according to the flow direction, to circumvent Theorem 1, seems quite attractive. Problems are evident in two-dimensions though, particularly with the extension to triangular elements.

Proof. Consider quadratures of the form (2.1) with $m = 2$. From Theorem 1 it is clear that we must have quadrature points at the ends of the interval, i.e., $x_0 = 0$ and $x_3 = 1$. Symmetry imposes the further conditions that $w_0 = w_3$, $w_1 = w_2$, and $x_1 = (1 - x_2)$. Writing down the equations that enforce the quadratures exact integration of quadratic polynomials and then eliminating all the variables except x_1, say, then leads to the condition for stability,

$$x_1(1 - x_1)^2 \geqslant \tfrac{1}{6}. \quad (2.3)$$

The maximum, for $x_1 \in (0, 1)$, of the function on the left-hand side of (2.3) occurs at $x_1 = \frac{1}{3}$ and since $\frac{4}{27} \not\geq \frac{1}{6}$ we deduce that there is no value of x_1 that gives a stable scheme, hence the result. ∎

Conjecture 3. There are no unconditionally stable implementations of the Lagrange–Galerkin method with piecewise linear basis functions using symmetric five-point quadratures, of the form (2.1), that integrate quadratic exactly.

Assuming $x_p < v < x_{p+1}$, then we can write down the four terms contributing to the right-hand side of the Lagrange–Galerkin method as

$$\sum_{k=0}^{p} w_k x_k \{(v - x_k)\, U_{i-2}^n + (1 - v + x_k)\, U_{i-1}^n\},$$

$$\sum_{k=p+1}^{m+1} w_k x_k \{(1 + v - x_k)\, U_{i-1}^n + (x_k - v)\, U_i^n\},$$

$$\sum_{k=0}^{p} w_k (1 - x_k) \{(v - x_k)\, U_{i-1}^n + (1 - v + x_k)\, U_i^n\}$$

$$\sum_{k=p+1}^{m+1} w_k (1 - x_k) \{(1 + v - x_k)\, U_i^n + (x_k - v)\, U_{i+1}^n\}.$$

Summing these four parts and adding, and subtracting, summations over $k = 0$ to $k = p$ in order to complete the summations from $k = p + 1$ to $k = m + 1$, we then obtain the following expression for the right-hand side of the Lagrange–Galerkin method:

$$\begin{aligned}
&\sum_{k=0}^{m+1} w_k (U_{i-1}^n \{x_k + x_k v - x_k^2\} \\
&\quad + U_i^n \{1 - 2x_k + v - 2x_k v + 2x_k^2\} \\
&\quad + U_{i+1}^n \{-v + x_k + x_k v - x_k^2\}) \\
&\quad + \sum_{k=0}^{p} w_k (U_{i-2}^n \{x_k v - x_k^2\} \\
&\quad + U_{i-1}^n \{v - x_k - 3x_k v + 3x_k^2\} \\
&\quad + U_i^n \{-2v + 2x_k + 3x_k v - 3x_k^2\} \\
&\quad + U_{i+1}^n \{-x_k + v + x_k^2 - x_k v\}).
\end{aligned}$$

Since the quadrature evaluates quadratics exactly the sum from $k = 0$ to $k = m + 1$ can be replaced with the values of the integrals. Now, replacing terms by their Fourier transforms and denoting the sine and cosine of the half angles by s and c respectively, we have

$$\begin{aligned}
&1 - \tfrac{2}{3}s^2 + \sum_{k=0}^{p} w_k (v - x_k)(-4s^2 + x_k(8s^4 - 4sc)) \\
&\quad + i\left(-2vsc + 8s^3 c \sum_{k=0}^{p} w_k (v - x_k)\, x_k\right) \qquad (2.4)
\end{aligned}$$

which, for $p = 0$, reduces to the expression used in the proof of Theorem 1. Unfortunately, the expression seems to be too unwieldy to be of much use outside of this case.

For five-point symmetric quadrature, conditions can again be written down but now they only lead to restrictions on the values of the weights and the positioning of the one free abscissa. It is then possible to test a quadrature satisfying those conditions with the formula (2.4) to investigate its stability. Some 100 million five-point symmetric quadratures that integrate quadratics exactly and are stable for $v \in [0, x_1]$ were tried and all were found to be unstable for some $v \in (x_1, \frac{1}{2})$.

The main cost, as we will discuss more fully in the next section, of the Lagrange–Galerkin method is not in the inversion of the mass matrix, which can be done very cheaply using preconditioned conjugate gradient methods (Wathen [64]), nor is it in the cost of calculating the integrals per se. The main cost is in calculating the trajectory, i.e., the value of the departure point **x** and, even more importantly, which element this point is in. Once this information has been calculated the function evaluation is quite trivial. It would make sense, therefore, if as much information as possible were extracted from any one departure point. That is, instead of just evaluating the function at the departure point we also make use of its derivatives, again quite trivial to calculate, and we use a quadrature by differentiation formula; see Lanczos [37], Krylov [35], Ghizzetti and Ossicini [21], Davis and Rabinowitz [10, p. 105–106], Lambert and Mitchell [36], Hammer and Wicke [23], and Struble [60], for example. These methods do not seem as common as their counterparts that just use function values, but they do have benefits in certain application areas; see Squire [58] for a discussion of some of these.

Despite the attraction of having to perform fewer trajectory calculations and fewer searches to achieve a given order of accuracy for the integration, we shall see, in the following two theorems, that no useful Lagrange–Galerkin schemes arise from using quadrature by differentiation formulae. First we consider quadratures given by Lanczos [37]. These are of the form

$$\int_0^h f(y)\, dy \approx \frac{1}{C_0^n} \sum_{k=0}^{n-1} C_{k+1}^n h^{k+1} (f^k(0) + (-1)^k f^k(h)), \qquad (2.5)$$

where

$$C_k^n = \frac{(2n-k)!}{(n-k)!\, k!}. \qquad (2.6)$$

THEOREM 4. *The direct (and weak) Lagrange–Galerkin methods with piecewise linear basis functions lead to unconditionally unstable schemes when the integrals are evaluated using the Lanczos quadrature by differentiation formulae.*

Proof. In (2.5) we put $f = U^n(x)\,\phi_j(y)$, $f^1 = U'^n(x)\,\phi_j(y) + U^n(x)\,\phi_j'(y)$, $f^2 = 2U'^n(x)\,\phi_j'(y)$, and $f^k = 0$ $\forall k \geqslant 3$. The higher derivatives vanish because, at a point, the integrand is at most quadratic. However, the weights of the remaining terms change, leading to a different approximation of the integral, as the integrand is not quadratic over the entire interval, but piecewise quadratic. The formulae for f^1 and f^2 are not more complicated because for constant coefficient linear advection $x = y - a\,\Delta t$ and hence $\partial x/\partial y = 1$.

Dropping the subscript j from the test function $\phi_j(y)$ and the superscript n from U^n and integrating over $[y_{j-1}, y_{j+1}]$, we use the following quantities:

$$\begin{aligned}
\phi(y_{j-1}) &= 0, \qquad \phi'(y_{j-1}) = 1/h,\\
\phi(y_j) &= 1, \qquad \phi'(y_j) = 1/h \qquad \text{or} \qquad -1/h.\\
\phi(y_{j+1}) &= 0, \qquad \phi'(y_{j+1}) = -1/h,\\
U(x(y_{j-1})) &= vU_{j-2} + (1-v)\,U_{j-1},\\
U'(x(y_{j-1})) &= \frac{U_{j-1} - U_{j-2}}{h},\\
U(x(y_j)) &= vU_{j-1} + (1-v)\,U_j,\\
U'(x(y_j)) &= \frac{U_j - U_{j-1}}{h},\\
U(x(y_{j+1})) &= vU_j + (1-v)\,U_{j+1},\\
U'(x(y_{j+1})) &= \frac{U_{j+1} - U_j}{h}.
\end{aligned}$$

The derivatives of ϕ are, at best, ambiguous at the ends of the intervals, but we choose the most obvious values. We replace y_j by $y_j^{\pm}$ to distinguish the values from the intervals $[y_{j-1}, y_j]$ and $[y_j, y_{j+1}]$. Our approximation to (2.5) is now given by

$$\frac{1}{C_0^n}\sum_{k=0}^{n-1} C_{k+1}^n h^{k+1}(f^k(y_{j-1}) + (-1)^k\, f^k(y_j^-) + f^k(y_j^+) + (-1)^k\, f^k(y_{j+1})). \tag{2.7}$$

Expanding the terms in (2.7) with the expressions given above and simplifying we then obtain the following expression for the right-hand side of the Lagrange–Galerkin method:

$$\begin{aligned}
\frac{1}{C_0^n}\{&2C_1^n(vU_{j-1} + (1-v)\,U_j) + C_2^n(vU_{j-2} + (1-v)\,U_{j-1}\\
&- 2vU_{j+1} - 2(1-v)\,U_j + vU_j + (1-v)\,U_{j+1})\\
&+ C_3^n(-U_{j-2} + U_{j+1} + U_j - U_{j+1})\}.
\end{aligned}$$

Replacing the U's with their Fourier transforms, where again $s = \sin(\theta/2)$ and $c = \cos(\theta/2)$, we then obtain that the transform of the right-hand side is

$$\begin{aligned}
\frac{1}{C_0^n}\{&2C_1^n(1 - 2v(s^2 + isc)) + C_2^n(-4s^2 + 8v(s^4 - isc + is^3c))\\
&+ C_3^n(8s^2(1 - s^2 - isc))\}.
\end{aligned}$$

Before calculating the amplification factor, λ, the algebra can be simplified somewhat by assuming $v = 0$. The equation for the modulus of the amplification factor then turns out to be

$$\begin{aligned}
\left\{1 - \frac{4s^2}{3} + \frac{4s^4}{9}\right\}|\lambda|^2 &= \frac{1}{C_0^n}\{-16s^2C_1^nC_2^n + 32s^2C_1^nC_3^n + 64s^6C_2^nC_3^n\\
&\quad - 64s^4C_2^nC_3^n + 4C_1^{n^2} + 16s^4C_2^{n^2} + 64s^4C_3^{n^2}\\
&\quad - 64s^6C_3^{n^2} - 32s^4C_1^nC_3^n\}.
\end{aligned} \tag{2.8}$$

Using the fact that $C_1^n/C_0^n = \frac{1}{2}\,\forall n$, (2.8) can be simplified to give the following condition for the avoidance of unconditional instability,

$$C_1^n + 6C_3^n - 3C_2^n \leqslant 0. \tag{2.9}$$

For the improved trapezium rule we have $C_3^2 = 0$, $C_2^2 = 1$, $C_1^2 = 6$, and $C_0^2 = 12$. Substituting into (2.9) we immediately see that this is an unconditionally unstable scheme. For $n \geqslant 3$ we use the values from Eq. (2.6) and it is then quite easy to deduce that in order to avoid unconditional instability we need $2n - 1 \leqslant 0$. This is clearly untrue and hence the result. Mass lumping does not lead to any useful schemes. Nor does trying to alter the ambiguous values of the derivatives of ϕ that we have to assume are at the ends of the intervals. ∎

Remarks. It is very disconcerting that this result is achieved by just considering the case $v = 0$. Even Gauss–Legendre quadrature managed to be stable for $v = 0$. Although both types of quadrature have similar error estimates involving $f^{2n}(\psi)$, where ψ is some unknown point, the Gauss–Legendre quadrature ψ is strictly within the domain of integration. With the Lanczos formulae this is not so, see [37], and singularities outside the domain of integration can seriously affect the convergence of the quadrature; see Squire [58], for example. Since the piecewise linear functions we are dealing with do not possess continuity of derivatives across elements, this explains the instability even at $v = 0$.

We now consider quadrature by differentiation formulae with nodes purely internal to the region of integration. It is

possible to consider formulae with nodes at the extremities, as in [42], but this results in much more complicated algebra and does not seem to exclude any other common rules. The main purpose of the following theorem is to exclude some particularly accurate rules quoted by Ghizzetti and Ossicini [21]. The least accurate of these (formula (4.13.10) of [21]) uses only two abscissae to give the approximation

$$\int_{-1}^{1} f(x)\,dx = f(-\alpha)+f(\alpha) + 0.09629177[f'(-\alpha)+f'(\alpha)] + 0.02930120[f''(-\alpha)+f''(\alpha)] + R(f),$$

where

$$\alpha = 0.6292111 \quad \text{and} \quad R(f)=0 \quad \text{if } f(x) \text{ is a polynomial of degree } \leqslant 7.$$

THEOREM 5. *The direct (or weak) Lagrange–Galerkin method evaluated using a quadrature of the form*

$$\int_0^h f(x)\,dx \approx \sum_{j=0}^{d}\sum_{k=1}^{m} w_{k,j} h^{j+1} f^j(x_k) \qquad (2.10)$$

that integrates quadratics exactly and has abscissae $0<x_1<x_2<\cdots<x_m<1$ *leads to an unconditionally unstable scheme.*

Proof. The powers in h can immediately be disregarded as they cancel with terms from the definition of the derivatives, with the remaining power cancelling with a similar term from the mass matrix. As discussed in Theorem 4 we need only consider $d\leqslant 2$ and we will also only consider $v\leqslant x_1$. The integral on the right-hand side of the Lagrange-Galerkin method is then approximated by to parts. From element $[j-1, j]$ we obtain

$$\sum_{k=1}^{m} w_{k,0}\{U_{j-1}x_k(1-x_k+v)+U_j x_k(x_k-v)\} + \sum_{k=1}^{m} w_{k,1}\{U_{j-1}(1-2x_k+v)+U_j(2x_k-v)\} + \sum_{k=1}^{m} w_{k,2}\{2(U_j-U_{j-1})\}$$

and from element $[j, j+1]$ we obtain

$$\sum_{k=1}^{m} w_{k,0}\{U_j(1-x_k)(1-x_k+v)+U_{j+1}(1-x_k)(x_k-v)\} + \sum_{k=1}^{m} w_{k,1}\{U_j(-2+2x_k-v)+U_{j+1}(1-2x_k-v)\} + \sum_{k=1}^{m} w_{k,2}\{2(U_j-U_{j+1})\}.$$

Combining these two parts we see that we have an approximation of the form (2.10) to the integral

$$\int_0^1 (U_{j-1}(x-x^2+xv)+U_j(1-2x+2x^2+v-2xv) + U_{j+1}(x-v-x^2+xv))\,dx$$

and, since the quadrature rule integrates quadratics exactly, we can simply replace both summations by $\frac{1}{6}U_{j-1}+\frac{2}{3}U_j+\frac{1}{6}U_{j+1}+(v/2)(U_{j-1}-U_{j+1})$. This has a Fourier transform of $1-2s^2/3-2ivsc$ and it is then a trivial matter to show that for stability we require

$$v^2s^2(1-s^2)\leqslant 0.$$

Except in the case that $v=0$, this is clearly untrue and hence the result. ∎

THEOREM 6. *No quadrature of the form*

$$\int_{-a}^{a} f(x)\,dx \approx \alpha f(-a)+2(a-\alpha)f(0)+\alpha f(a), \qquad (2.11)$$

with $\alpha\geqslant 0$, *leads to an unconditionally stable scheme when used to approximate the integrals arising from the Lagrange–Galerkin method* (1.6).

Remark. Although this result appears to be weaker than the others proved here and in [42], it is a surprisingly general result. The previous work has tended to imply that there is some stumbling block in using a quadrature rule that integrates quadratic (and higher order) polynomials exactly. Since the vast majority of quadrature rules, with a weighting function of one, increase their accuracy by integrating ever higher degree polynomials in x exactly, it is tempting to ask what happens if we attempt to integrate other functions of x exactly instead. This can be achieved using Eq. (2.11) and choosing different values of α. For example, with

$$\alpha = \frac{\sinh(a)-a}{\cosh(a)-1}$$

the quadrature integrates the functions 1, $\sinh(x)$, and $\cosh(x)$ exactly. With

$$\alpha = \frac{a-\sin(a)}{1-\cos(a)}$$

the quadrature integrates the functions 1, $\sin(x)$, and $\cos(x)$ exactly.

Proof. Transforming the quadrature to the range $[0, 1]$ we have

$$\int_0^1 f(y)\,dy \approx \frac{\alpha}{2}f(0)+(1-\alpha)f\left(\frac{1}{2}\right)+\frac{\alpha}{2}f(1).$$

With our integrand $f(y) = U(x)\,\phi_j(y)$ we obtain, for the right-hand side of the jth Equation (assuming that $v \leqslant \frac{1}{2}$),

$$\frac{1}{4}(U_{j-1} + 2U_j + U_{j+1}) - \frac{\alpha}{4}(U_{j-1} - 2U_j + U_{j+1})$$

$$+ \frac{v\alpha}{2}(U_{j-1} - 2U_j + U_{j+1}) + \frac{v}{2}(U_{j-1} - U_{j+1}). \quad (2.12)$$

The Fourier transform of (2.12) is $1 - s^2 + \alpha s^2 - 2v\alpha s^2 - 2ivsc$ and hence for stability we require, after simplification,

$$-\tfrac{2}{3} + 2\alpha - 4v\alpha + 4v^2 \leqslant 0. \quad (2.13)$$

It is now an obvious condition ($v = 0$) that we require

$$\alpha \leqslant \tfrac{1}{3}. \quad (2.14)$$

The maximum value of (2.13) occurs at $v = \frac{1}{2}$ in which case the condition (2.13) becomes $\frac{1}{3} \leqslant 0$, independently of α, and hence the result. Cases with $\alpha \leqslant 0$ can be dealt with in an analogous manner. ∎

COROLLARY 1. *Simpson's rule has the largest stability region of quadrature rules of the form* (2.11).

Proof. It is a simple task to calculate the maximum value of v allowed by (2.13) for a given α. Not surprisingly we find that the maximum value (over α) of these maximum values occurs at the limit of (2.14), i.e., $\alpha = \frac{1}{3}$. This is then Simpson's rule. ∎

3. A STABLE IMPLEMENTATION ON TRIANGULAR ELEMENTS

3.1. *Description of the Method*

In the preceeding section it was shown that the results given in Morton *et al.* [42] can be extended to even wider classes of integration rules, including quadrature by differentiation formulae which would have been particularly beneficial to the Lagrange–Galerkin method because of the reduction in time spent calculating the positions of the departure points. It was also shown that it is not just those formulae that interpolate polynomials exactly that cause the method to be unstable.

In other application areas, namely grid adaption and triangular multigrid, it is necessary to project from one arbitrary triangular grid onto a second arbitrary triangular grid, just as it is in the Lagrange–Galerkin method. Although Gauss–Legendre quadrature can still lead to an unconditionally unstable scheme even when diffusion is present, Süli [62], the author is not aware of any published results for the Navier–Stokes equations, say, where the instabilities inherent in the quadrature approximated Lagrange–Galerkin method have caused any noticeable problems. However, in these other related application areas they have been observed; see Peraire *et al.* [43].

The stable implementation proposed here is an extension of the area-weighting technique of [45, 42]. The situation we are faced with is shown in Fig. 1, where the arbitrary background grid is depicted as the regular grid (purely for demonstration purposes only, it must be stressed) and the triangle we wish to integrate over as the irregular triangle, with nodes 1, 6, and 9 as depicted in the figure. The integrand of (1.6) is now formed by the product of the linear function over the triangle (1, 6, 9) with the piecewise linear functions defined on the background grid. The integrand is then made up of piecewise quadratic functions, with the pieces being formed by the intersection of the background grid with the triangle in question.

The shapes of the regions caused by the intersections can be triangles, quadrilaterals, irregular pentagons, or irregular hexagons; i.e., they can have 3, 4, 5, or 6 nodes and sides. Given the positions of these nodes, and the values of the two linear functions at these points, we can then quite simply calculate the integral exactly, for example, by splitting the region into triangles for which an analytic formula can be written down quite simply in terms of the nodal values. We shall now concentrate on the more difficult question of determining the regions.

This problem is greatly simplified if extra information about the grid is stored, apart from the usual connectivity table. The most important new information to be stored is that of which two elements are either side of a given side and which sides go up to make a given element. Less importantly it is also useful to store node–node connections and which two nodes form a given side. It is not strictly necessary to store any additional information as it could all be calculated as needed, directly from the connectivity table, but by storing the first two arrays mentioned, in particular, the computational time will be substantially reduced. The

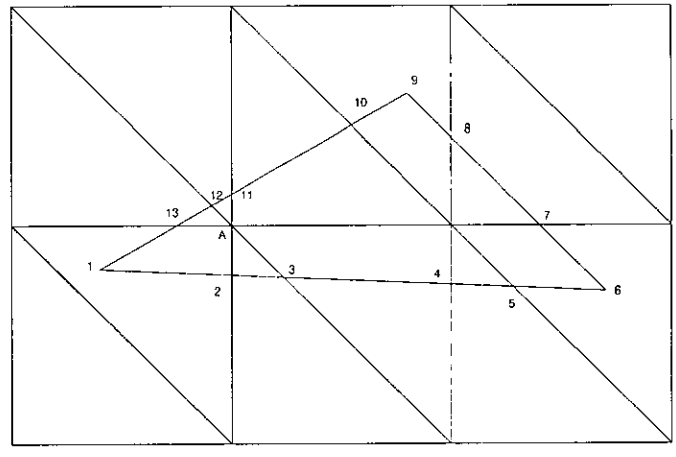

FIG. 1. The initial "front" for determining the regions of integration.

second pair of arrays are much more readily calculated "on the fly" and so these should be dropped first if storage is a problem. However, all of these arrays are integer arrays and their storage is not likely to be a problem in practice.

The logic and data structure in the region identification procedure is perhaps most analogous to that used in moving front grid generators (see Cavendish [7], Jin and Wiberg [32], Lo [39], and Sadek [55], for example), but is much simplified as there are no nodes or topology to generate.

The vertices of the triangle are first transported by the velocity field, there being no advantage with arbitrary triangular grids in transporting the entire triangle by its centroid value as in [42]. Indeed by transporting the vertices, rather than the centroids, we formally retain the accuracy of the method and maintain conservation because we are again mapping the whole domain onto the whole domain. This is not the case with the area-weighting implementation in [45, 42]. The existence of this approximate, but well-defined, transformation $\mathbf{x} \to \mathbf{y}$ will hopefully make the theoretical development of the method much easier than would otherwise be the case. This transportation gives us the positions labelled 1, 6, and 9 in Fig. 1. The interior of the triangle is then assumed to have moved in a linear manner, governed by the movement of the nodes and, hence, the area we wish to integrate over remains a triangle. That is, we make the approximation

$$\begin{aligned}\mathbf{X}((1-p-q)\,\mathbf{y}_1 + p\mathbf{y}_2 + q\mathbf{y}_3, t^{n+1}; t^n) \\ \approx (1-p-q)\,\mathbf{X}(\mathbf{y}_1, t^{n+1}; t^n) + p\mathbf{X}(\mathbf{y}_2, t^{n+1}; t^n) \\ + q\mathbf{X}(\mathbf{y}_3, t^{n+1}; t^n) \qquad (3.1)\end{aligned}$$

to the trajectories over an element parameterised by (p, q) and with nodes $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$. For the Lagrange–Galerkin method this indeed represents an approximation since, aside from uniform translations and uniform rotations, triangles are not mapped onto triangles by a more general velocity field. With the extra grid information that is now stored about the sides of the background mesh it is a fairly easy task to determine the points where the transported triangle intersects the background mesh. These points are labelled in an anti-clockwise fashion. The initial front is then given by the edges $\{(1, 2), (2, 3), \ldots, (13, 1)\}$. Choosing an edge from the active front, (1, 2) say, we then proceed to identify a region for integration, by always moving in the most anti-clockwise sense possible. From node 2 there are three possible points we could move to, namely, node 3, the node marked A, and the unmarked node below A on the background mesh. This latter node can be immediately discounted. Of the two remaining possibilities it is clear the edge $(2, A)$ is the one we require. From A, knowing that we have come from 2, we readily identify the next edge as $(A, 13)$ and from there we use the existing edge (13, 1) to complete the region. The nodes of the new region are, by construction, already labelled in the anti-clockwise sense which makes the integration over the region that much more simple. Having identified this first region it only remains to delete the edges (13, 1) and (1, 2) from the front and insert the two new edges, giving a new front $\{(2, 3), (3, 4), \ldots, (13, A), (A, 2)\}$. We can then continue to identify all the other regions in exactly the same manner. Complications arise, as in the moving front grid generation codes, when a front splits. Once recognised as having happened, each separate front can be treated as described before.

This procedure is obviously entirely general and in particular copes with the trivial case of the element we are integrating over lying completely within a triangle on the background mesh and with the case where entire elements of the background mesh are totally enclosed within the element over which we are performing the integration. That is, we are not dependent on an element of one mesh intersecting with elements of the other mesh.

As we have said before the problem of transferring information from one grid to another has applications apart from its use with the Lagrange–Galerkin method. In one of these other application areas, namely grid rezoning, very similar appearing problems have been solved; see Ramshaw [50, 51] and Dukowicz *et al.* [14–16], for example. This method involves replacing the integrand, $q(\mathbf{x})$ say, by $\nabla \cdot \mathbf{F}$, where $F = \mathbf{F}(\mathbf{x})$ is some flux to be determined from $q(\mathbf{x})$. The purpose of this is to then apply the Gauss divergence theorem to simplify the integrals considerably. However, due to the Galerkin projection in (1.6), our integrand, unlike those considered in the grid rezoning problem, depends upon values from both meshes. In particular this means that the vector functions $\mathbf{F}(\mathbf{x})$ cannot be constructed in a single pass over the old mesh with the integrals then being evaluated in a single pass over the new mesh. Hence, for the exact projection problem, it seems that the direct approach described here is more appropriate.

Clearly the code for doing this procedure in the general situation is significantly longer and potentially more time consuming than the linear interpolation needed for interpolating at quadrature points. In the next section the

TABLE I

Time (in seconds) for the Rotating Cone Problem on a SUN SPARC 2 Workstation for Various Implementations of the Lagrange–Galerkin Method

	7pt. Gauss; no searching	7pt. Gauss	Exact	Exact; no searching
10×10	12	52	39	24
20×20	53	755	231	102
40×40	264	9947	1994	435
80×80	1465	101851	25469	1814

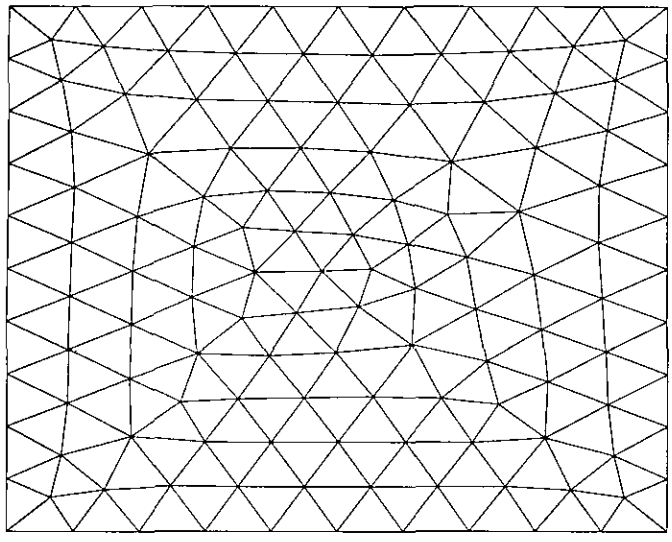

FIG. 2. 10 × 10 grid; 135 nodes; number of elements is 228.

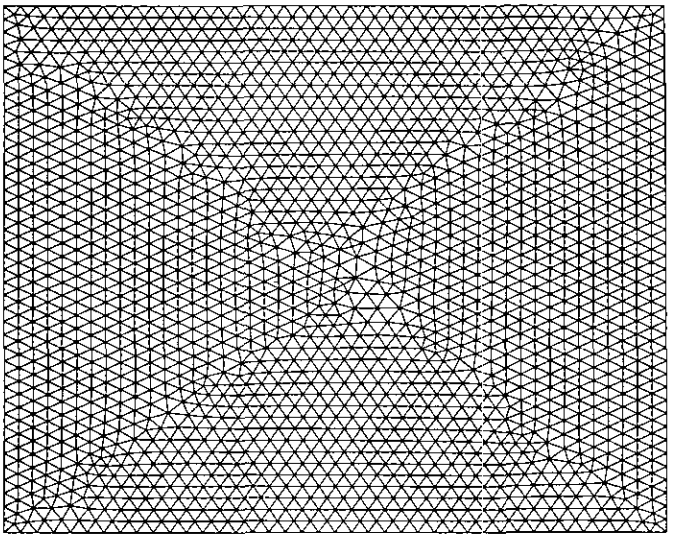

FIG. 4. 40 × 40 grid; 1911 nodes; number of elements is 3660.

rotating cone problem is used as a test case. For the present purpose, though, we just use it as a means of extracting and comparing run times for performing the projections with the standard seven-point Gaussian quadrature and by the exact projection method presented here. The meshes used are shown in Figs. 2–5. The times for 40 time-steps of the rotating cone problem are shown in columns 2 and 3 of Table I. Rather surprisingly we see that the exact projection is substantially quicker than the seven-point quadrature. There are two reasons for this. The least important is that much of the new code uses operations that only require integer logic and hence are very fast. More important is the fact that the integration part of the code is a minor consideration compared to the calculation of the trajectories and, in particular, determining from which element of the background grid a departure point came. Assuming an unstructured mesh in the exact projection method, we have to do one global search for each node, which means we have to de nele/2 global searches (nele being the number of elements and assuming that there are roughly twice as many elements as nodes). For the quadrature case it is possible to do only nele global searches if one searches only for the departure point of the first quadrature point from the element and then does local searches to find the elements containing the departure points of the other six abscissae. The searching used in Table I was very simple and, as a result, on the two finer meshes anyway, it caused us to have to perform approximately 2(nele) global searches. Even allowing for this factor, though, we see that the exact method would still be faster.

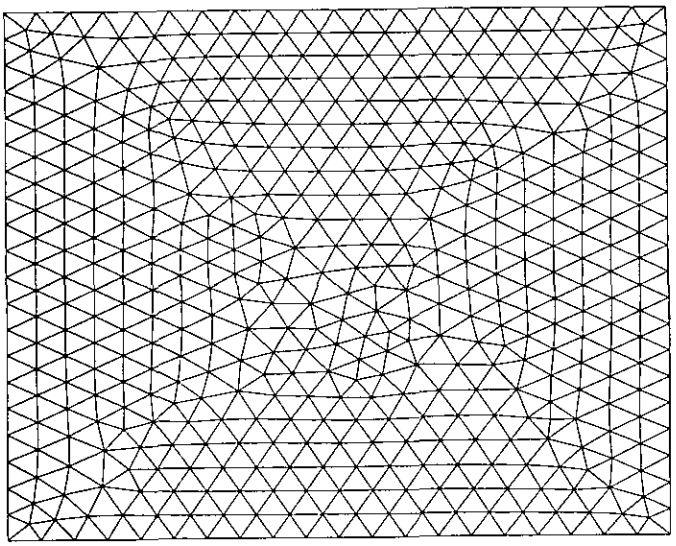

FIG. 3. 20 × 20 grid; 496 nodes; number of elements is 910.

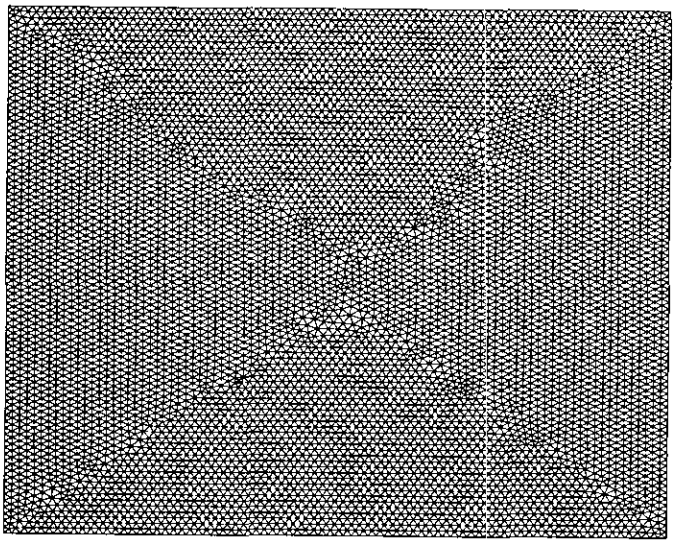

FIG. 5. 80 × 80 grid; 7521 nodes; number of elements is 14720.

Moreover, recall that for the exact projection case it was found advisable to store information about sides. This information can also be used in the trajectory problem to keep track of where the departure point lands, removing any need to perform any global searches. This can significantly reduce the cost of the exact projection method, column 4 of Table I. Of course if we use the same information for the trajectory problem in the quadrature case then the benefit is even greater and the quadrature implementation is at last faster than the exact projection method, column 1 of Table I. Note, though, that it appears that on a fine enough mesh the exact projection method would still be faster.

3.2. *Theoretical Results*

Like the area-weighting algorithm of [42], to which this method is very closely related, the procedure here reduces to exact integration in the case of constant coefficient advection and, hence, unconditional stability, so a more general velocity field must be considered. Fortunately, much of the theory for area-weighting in [42] is directly applicable to the case here. All that is required is to replace Lemma 3.1 of [42] with

LEMMA 1. *If the velocity field* $\mathbf{a}(\mathbf{x}, t)$ *belongs to* $L^\infty(0, T; (W^{2,\infty})^d)$, *then the distance between the true foot of the trajectory and the approximate foot given by* (3.1) *is of the order* $h^2\,\Delta t$, *where* h *is a measure of the mesh size and* d *is the number of spatial dimensions* ($d = 2$ *here*).

The stability of the method, with the restriction on $\mathbf{a}(\mathbf{x}, t)$ given above, can now be obtained immediately by using Lemma 1 in Theorem 3.4 of [42]. Actually, for the stability result, we can weaken the condition on $\mathbf{a}$ to $\mathbf{a}(\mathbf{x}, t) \in L^\infty(0, T; (W^{1,\infty})^d)$, since an error $h\,\Delta t$ in the foot of the trajectory is sufficient to prove stability.

Convergence is proved using Theorem 3.6 of [42]. Here we can make extra use of $\mathbf{a}(\mathbf{x}, t) \in L^\infty(0, T; (W^{2,\infty})^d)$ to obtain an error of order h^2. For the area-weighting technique of [42] only a first-order error estimate was obtained. This represented a loss of accuracy over the exactly integrated method which is second order, although in experiments on highly non-linear problems (Priestley [45]), the first-order estimate for the area-weighting technique was shown to be pessimistic. For the exact projection method, which as we have said is not exact for general velocity fields in the context of the Lagrange–Galerkin method, the error caused by the approximation of the feet of the trajectories does not alter the error estimate of the exactly integrated method and hence our earlier comment about the adequacy of the trajectory approximation.

4. NUMERICAL RESULTS

4.1. *Rotating Cone Problem*

This commonly used two-dimensional test problem looks at the advection of a cone in a fixed, rotating velocity field governed by the equation,

$$u_t + 2\pi(-y, x) \cdot \nabla u = 0$$

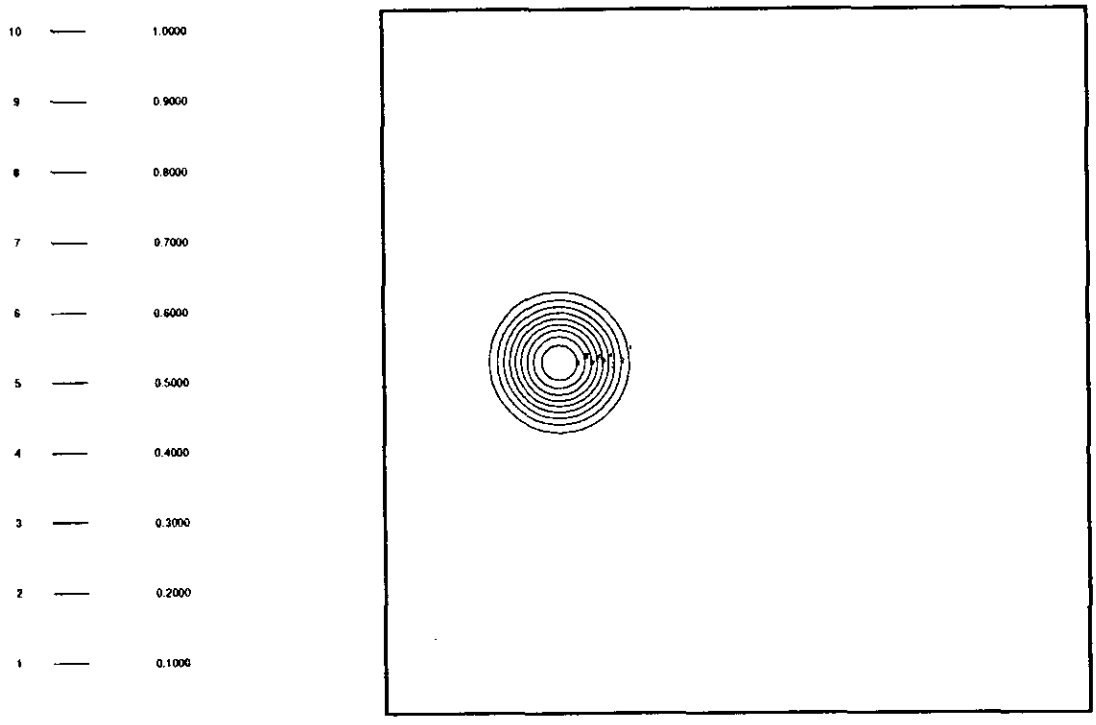

FIG. 6. Initial data for the rotating cone problem.

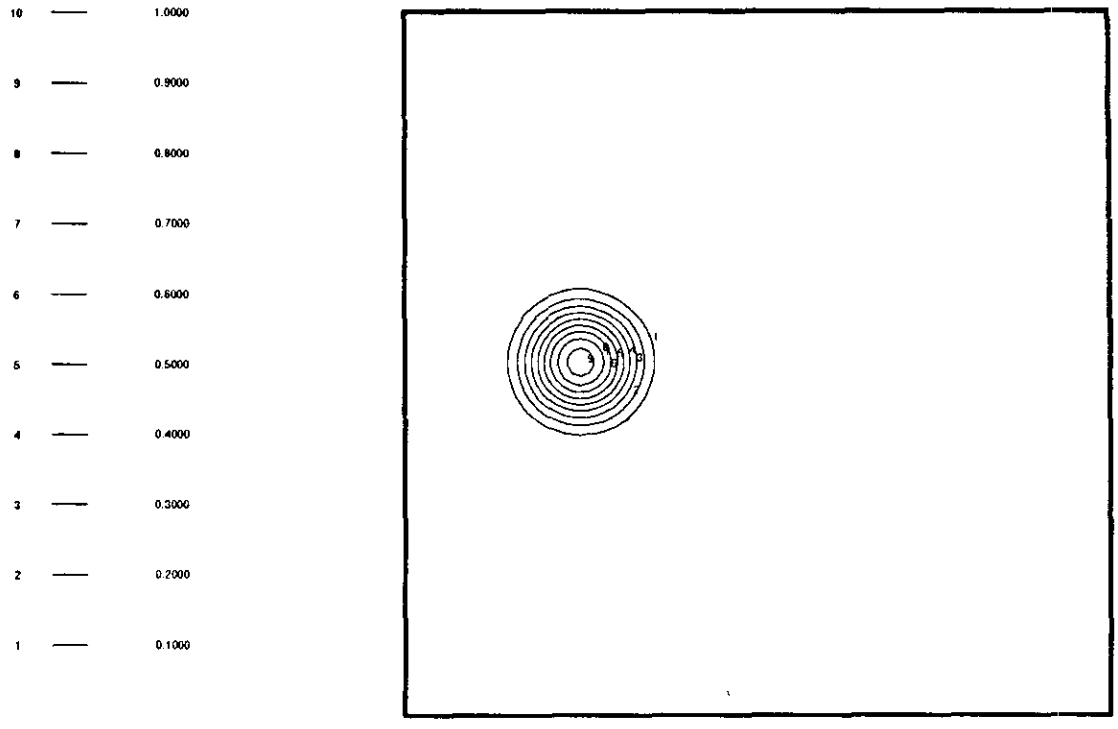

FIG. 7. Solution on the 80×80 mesh after 100 revolutions.

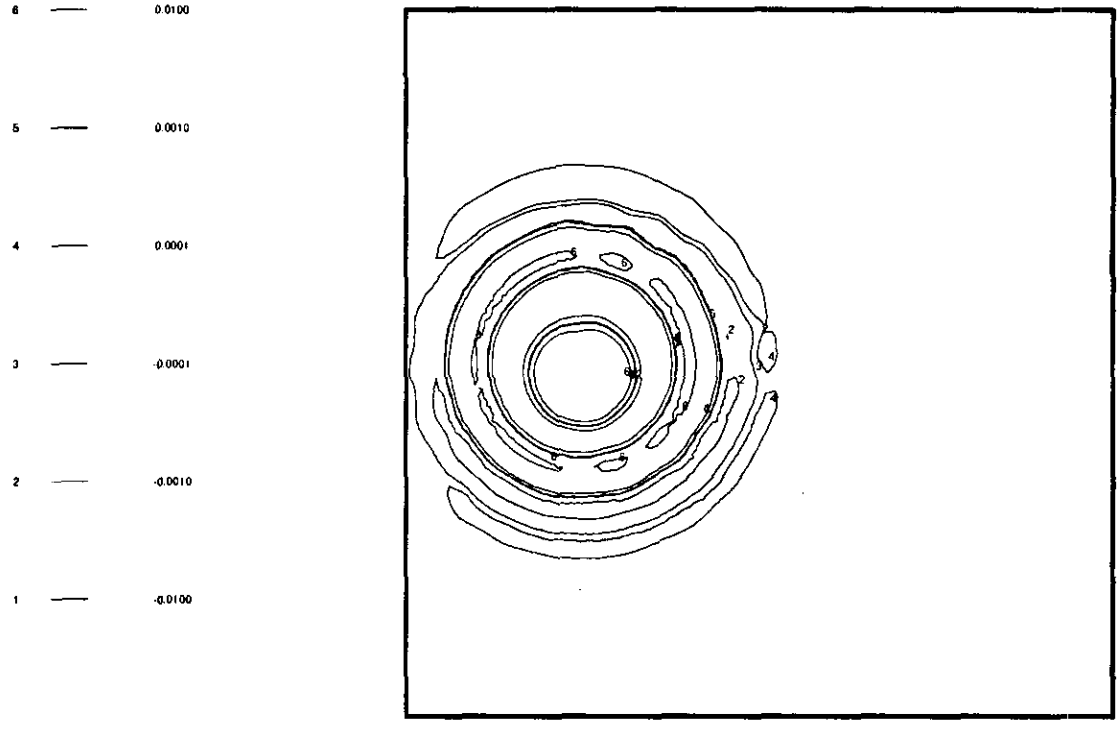

FIG. 8. Plot of (exact solution—approximate solution) after 100 revolutions.

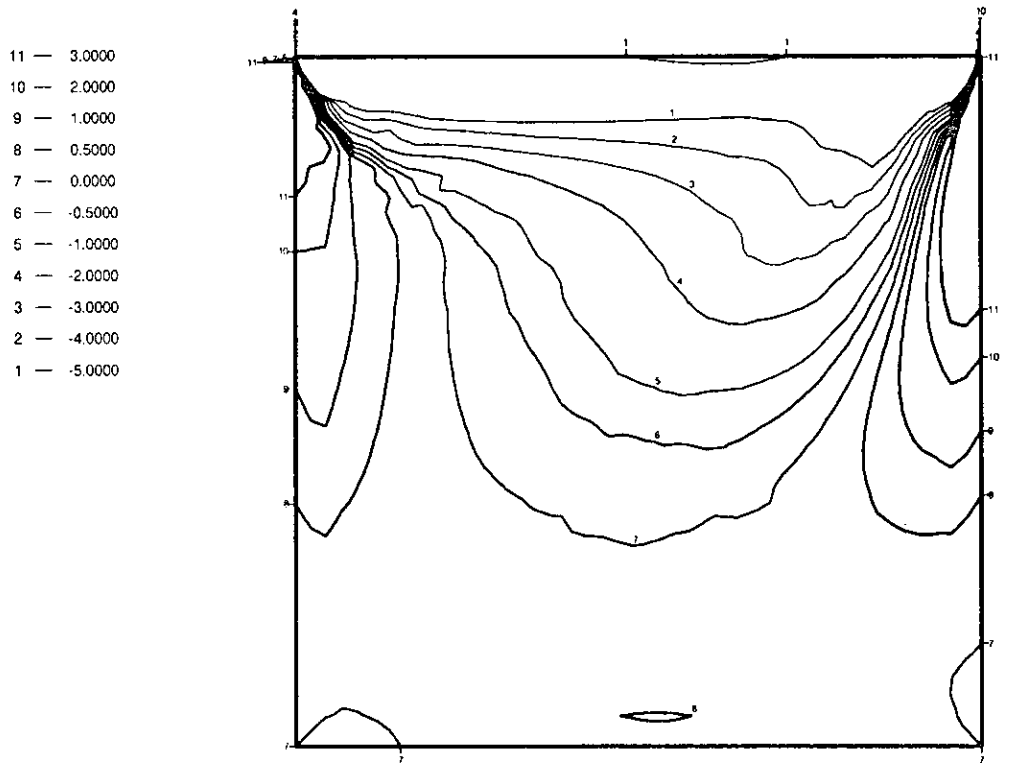

FIG. 9. Vorticity for the Re = 100 case on 40 × 40 mesh.

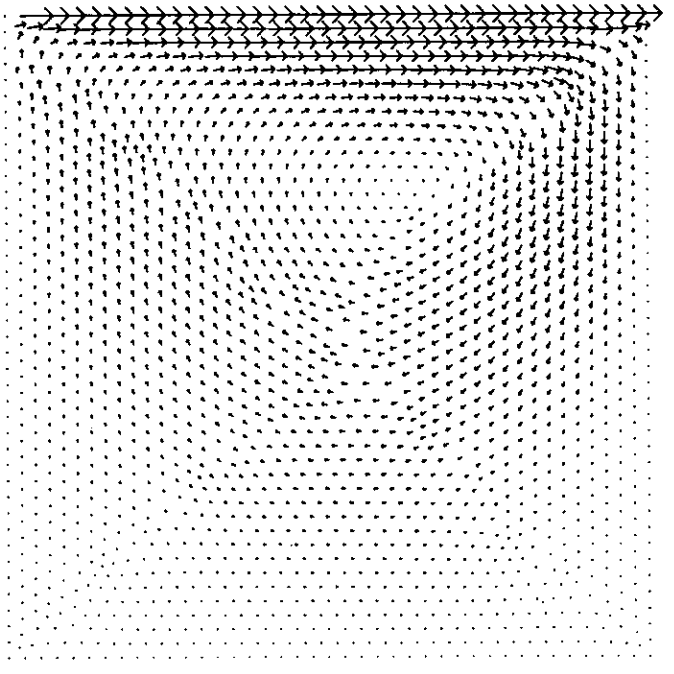

FIG. 10. Velocity arrows for the Re = 100 case on 40 × 40 mesh.

on the domain $\Omega = (-1, 1) \times (-1, 1)$, with periodic boundary conditions. The initial data consists of a $\cos^2$ cone centred at $(-\frac{1}{2}, 0)$ and of radius $\frac{1}{4}$. If $r^2 = (x + \frac{1}{2})^2 + y^2$ then

$$u = \begin{cases} \cos^2 2\pi r & \text{for } r \leqslant \frac{1}{4} \\ 0, & \text{otherwise.} \end{cases}$$

The initial data is interpolated rather than L^2-fitted and is shown in Fig. 6. The four meshes used are shown in Figs. 2–5. Although these meshes are fairly smoothly varying (not that this is particularly necessary for the exact projection method to work), it is important to note that the meshes are totally irregular (as opposed to the explanatory diagram of Fig. 1) and have no preferred directions. This is an important advantage over the triangle based upwind finite difference schemes, see Deconinck *et al.* [11] and Roe *et al.* [53], for example, where the results seem to be extremely dependent on the triangles being arranged in a way sympathetic to the problem being considered. The time-step is chosen to be $\Delta t = 0.025$ and 40 time-steps are performed; i.e., one complete revolution is done.

The results for this problem are summarized in Table II for the integrals evaluated with seven-point Gaussian quad-

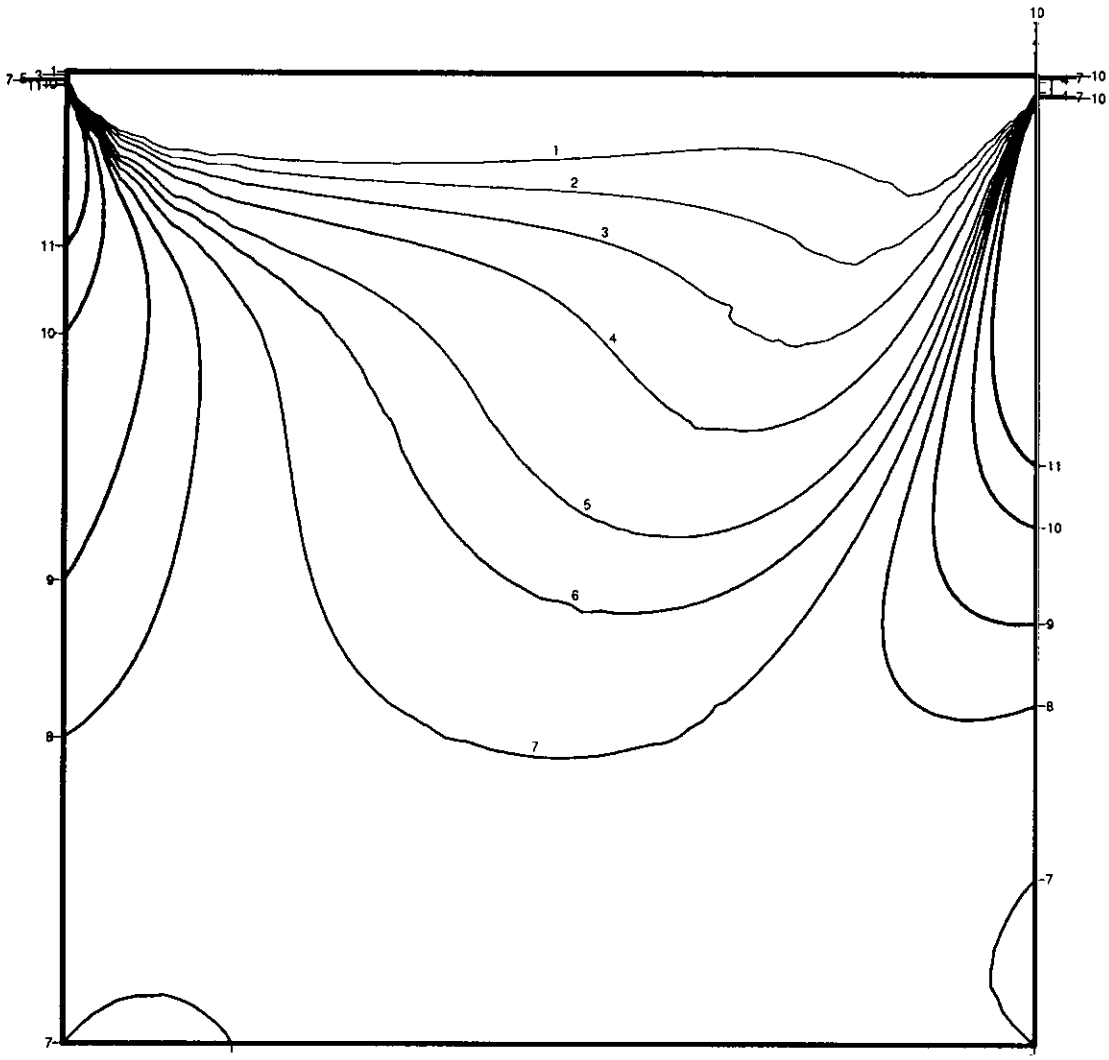

FIG. 11. Vorticity for the Re = 100 case on 80 × 80 mesh. Contours as in Fig. 9.

rature and in Table III for the integrals evaluated by the exact projection method. As we might expect there are no significant differences between the two methods for this problem, the quadrature giving a very accurate representation of the integrals. Conservation, though, is better with the exact projection method. Since for the rotating cone problem it is actually an exact implementation of the integrals in (1.6), we might expect it to be exactly conservative, but rounding errors and, more importantly, errors in inverting the mass matrix mean that this is difficult to

TABLE II

Results after One Revolution for the Rotating Cone Problem with the Integrals Evaluated Using Seven-Point Gaussian Quadrature

	Maximum	Minimum	Percentage of conservation	l_2 error
10 × 10	0.30446	-5.82863×10^{-2}	91.66835	0.134232
20 × 20	0.77387	-2.79735×10^{-2}	99.83719	4.89047×10^{-2}
40 × 40	0.97005	-1.06939×10^{-2}	100.16463	8.33986×10^{-3}
80 × 80	0.995	-2.81922×10^{-3}	100.13378	1.63186×10^{-3}

TABLE III

Results after One Revolution for the Rotating Cone Problem with the Integrals Evaluated Using the Exact Projection Method

	Maximum	Minimum	Percentage of conservation	l_2 error
10 × 10	0.317	-4.74561×10^{-2}	99.998	0.133472
20 × 20	0.78536	-2.75947×10^{-2}	99.99996	4.7934×10^{-2}
40 × 40	0.96876	-9.35397×10^{-3}	100.00000	8.47666×10^{-3}
80 × 80	0.9939	-2.48703×10^{-3}	100.00000	1.78774×10^{-3}

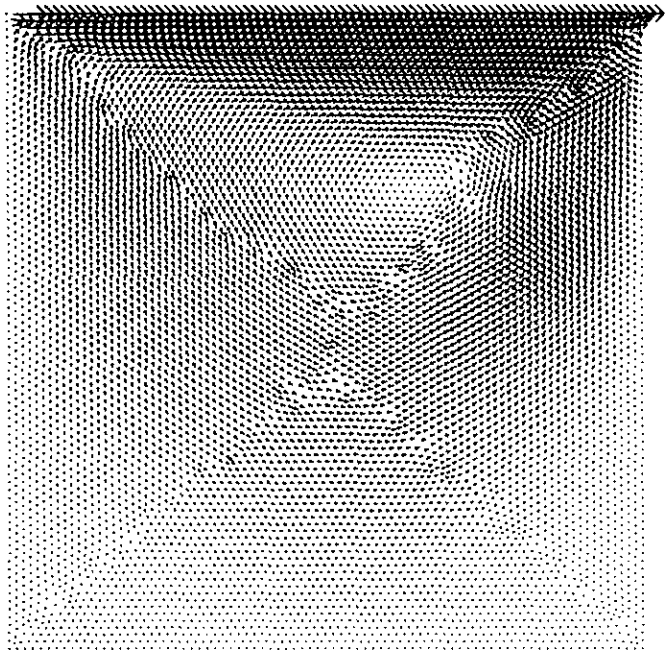

FIG. 12. Velocity arrows for the Re = 100 case on 80×80 mesh.

achieve in practice. In Fig. 7 we see the solution on the 80×80 mesh after 100 revolutions. As can be seen this is very similar to the initial data and so in Fig. 8 the exact solution minus the approximate solution has been plotted. This function has a minimum of about -4×10^{-2} and a maximum of about 5×10^{-2}.

4.2. *Lid-Driven Cavity Problem*

This problem, with results, can be found in Ghia *et al.* [20] and Gresho *et al.* [22], for example. The results presented here will be most directly comparable to those in [20]. We solve the incompressible Navier–Stokes equations in the primitive variables, i.e.,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \nu \nabla^2 \mathbf{u} \tag{4.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{4.2}$$

where $\mathbf{u} = (u, v)$ is the velocity and p is the pressure deviation from hydrostatic pressure. The kinematic viscosity is

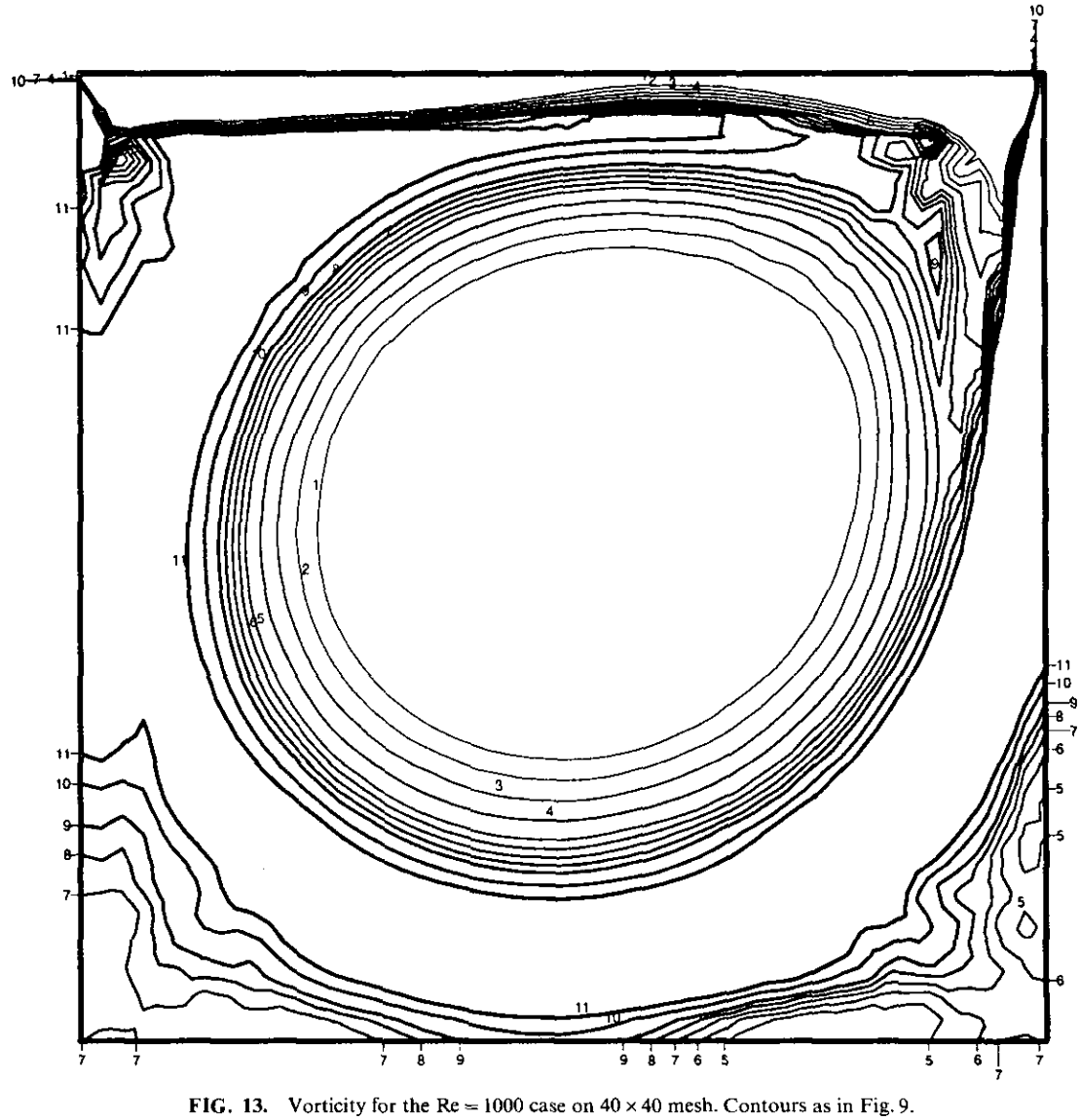

FIG. 13. Vorticity for the Re = 1000 case on 40×40 mesh. Contours as in Fig. 9.

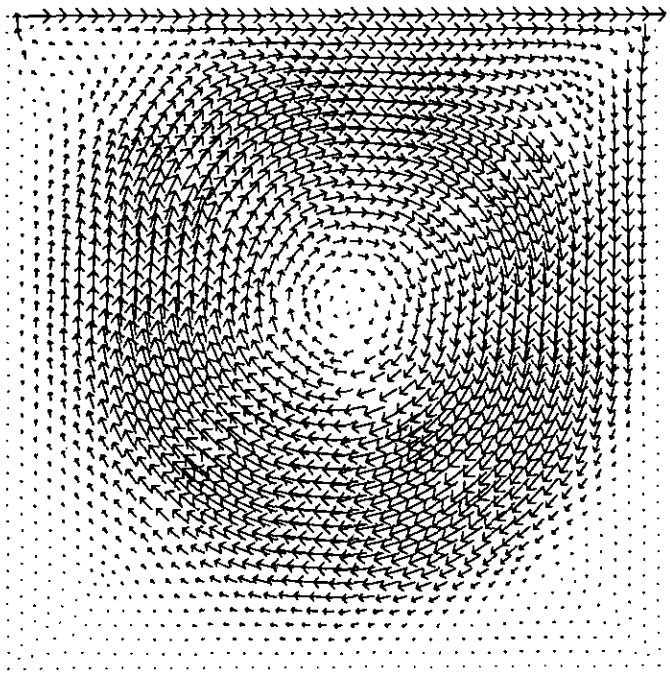

FIG. 14. Velocity arrows for the Re = 1000 case on 40 × 40 mesh.

denoted by v and the Reynolds number is defined to be $\mathrm{Re} = UL/v$, where U is a reference velocity and L is a reference length. In the lid-driven cavity problem $U = L = 1$. The domain of the problem is $[0, 1] \times [0, 1]$ and the boundary conditions are given by

$$v = 0, \quad \text{everywhere}$$
$$u = 1, \quad y = 1, \quad 0 < x < 1,$$
$$u = 0, \quad \text{otherwise.}$$

To apply the Lagrange–Galerkin method to Eq. (4.1) we replace the convective terms by the Lagrangian derivative D/Dt to obtain

$$\frac{D\mathbf{u}}{Dt} + \nabla p - v\nabla^2\mathbf{u} = 0. \tag{4.3}$$

The finite element method can then be applied to Eqs. (4.3), (4.2) in the usual way, with the Lagrange–Galerkin method being used to treat the convective terms and all the other

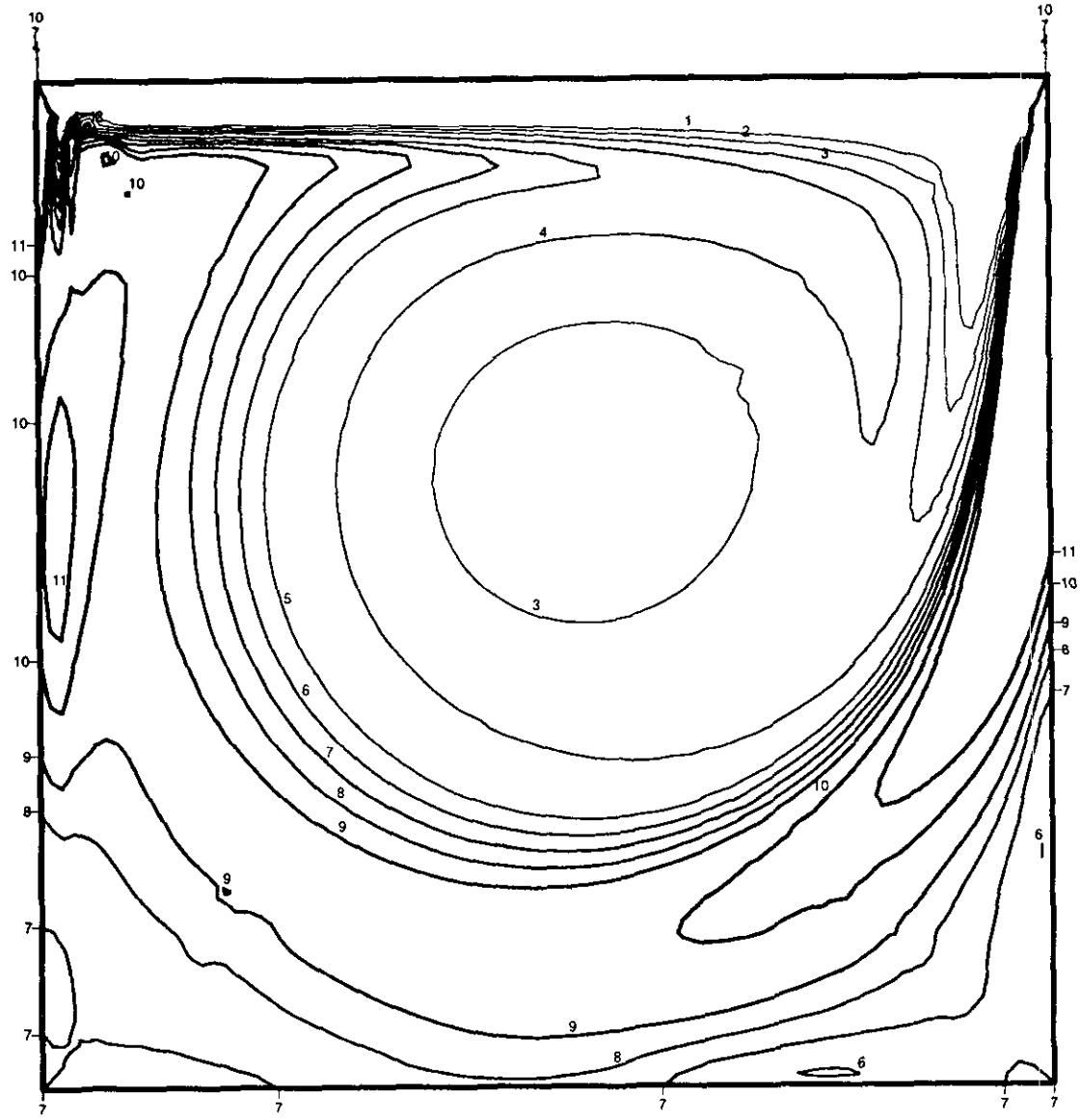

FIG. 15. Vorticity for the Re = 1000 case on 80 × 80 mesh. Contours as in Fig. 9.

terms being treated implicitly. This results in a matrix system of the form

$$\begin{pmatrix} M+K & B^{\mathrm{T}} \\ B & 0 \end{pmatrix}\begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}=\begin{pmatrix} E(\mathbf{u}) \\ 0 \end{pmatrix}, \tag{4.4}$$

where M and K are the symmetric mass and stiffness matrices, B is the gradient matrix, and $E(\mathbf{u})$ is just the right-hand side from the Lagrange–Galerkin method. An important feature of (4.4) is that the matrix equations are symmetric. This is a consequence of the use of the Lagrange–Galerkin method and means that we can use the very efficient, and simple, conjugate gradient based methods to solve the equations; see Ramage and Wathen [49] for further details.

The simplest choices of basis functions for us to use to test the exact projection implementation of the Lagrange–Galerkin method are piecewise linear functions for the velocities and piecewise constant functions (on the same mesh) for the pressure. This choice allows us to use the previously generated meshes, particularly the 40×40 and 80×80 meshes and makes it quite simple for us to extend this new implementation to the Navier–Stokes equations, but unfortunately this choice violates the Babuška–Brezzi stability condition; see [1, 6]. However, we can use a stabilization technique such as the method of Hughes and Franca [31]. Although this method has been criticized [13], it is preferred here over the extension in [13], because the symmetric nature of the Stokes problem, represented by (4.4), is maintained. The method, which has been further analysed in Kechkar and Silvester [34], consists of replacing the finite element discretization of Eq. (4.2) by

$$(q, \nabla\cdot\mathbf{u}_h)+\beta\sum_{e\in\Gamma_h}\int_e h_e[p_h]_e\,[q]_e\,ds=0 \qquad \forall q\in P_h, \tag{4.5}$$

where P_h is the piecewise constant pressure subspace, $\mathbf{u}_h$ and p_h are the discrete approximations to $\mathbf{u}$ and p, Γ_h is the set of all interelement boundaries, h_e is the length of edge e, and $[\cdot]_e$ is the jump operator across the boundary e.

One criticism of this procedure is that Eq. (4.5) cannot be assembled in the usual element-by-element manner because of the interelement jump operator. However, due to our storage of side information for the efficient implementation of the exact projection method, it is very simple for us to assemble these terms. The other main criticism is due to the appearance of a parameter $\beta>0$ that needs to be chosen. Although some insight has been gained in how to choose this constant for the Stokes problem, Wathen and Silvester [65] and Silvester [57], we have found the choice fairly forgiving in that there seems to be quite a large range where the value stabilizes the discretization without having any undue effect upon the solution quality. If we were concentrating on the accuracy of the solutions here, rather than just demonstrating the implementation of the exact projection method, the precise choice might be more crucial. This will obviously be a point for further investigation.

The finest mesh used here has less than half the nodes of the coarsest mesh used in Ghia *et al.* [20]. Although Gresho *et al.* [22] did use coarser meshes, they were using higher-order basis functions and had made some attempt to adapt the grids to the problems. The vorticity plots given here are directly comparable to the reference solutions given in [20]. It must be stressed that the solutions presented here are to demonstrate the implementation of the exact projection method rather than the Lagrange–Galerkin method itself. More realistic calculations are clearly a high priority for the future. By using grids adapted to the flow, particularly to resolve the boundary layer, we would clearly hope to obtain far superior solutions to the ones presented here.

The first case is that with a Reynolds number of 100. We give plots of vorticity, $\omega = v_x - u_y$, and velocity arrows. In Figs. 9 and 10 the results from the 40×40 mesh are given and in Figs. 11 and 12 the results from the 80×80 mesh. A similar set of results is then given, Figs. 13–16, for the case $\mathrm{Re}=1000$. The results for $\mathrm{Re}=100$ are very good but even at the still fairly low Reynolds number of 1000 the lack of resolution of the meshes is beginning to show. Comparing the solutions with those given elsewhere, in [20] for example, we see that the main vortex, in both cases, is of the correct strength and in the correct position. However, upon closer inspection we see that the smaller, and weaker, features in the corners are not well resolved, for example, in

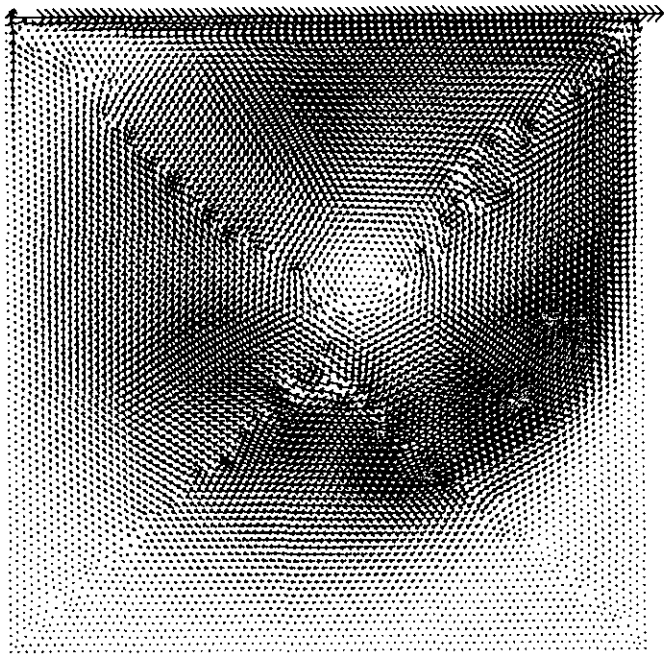

FIG. 16. Velocity arrows for the $\mathrm{Re}=1000$ case on 80×80 mesh.

the top left corner of Figs. 14 and 16 (or 13 or 15). While the finer grid solution still bears a very strong resemblance to the solution given in [20] it is equally clearly not resolving the flow near this corner. The method copes as the Reynolds number increases, but the lack of resolution becomes steadily more acute. We are very confident that this is caused by using an inappropriate mesh and not by a failing of the solution procedure described here. Time-steps ranged from $\Delta t = 0.005$ to $\Delta t = 1$, with the accuracy for these steady-state type problems being governed more by the accuracy of the trajectory solver rather than the time-step as such. Clearly for $\Delta t = 1$ much more care is required in solving the trajectory equations.

5. CONCLUSIONS

The main results of the paper are of two kinds. First in projecting from one arbitrary triangular grid to another it has long been known that for the Lagrange–Galerkin method [42] certain families of quadrature formulae give stability problems. These same problems have more recently been observed in situations not caused by the use of the Lagrange–Galerkin method, [43]. In this paper we have shown that much wider classes of quadrature than previously thought suffer from stability problems.

Second, because of the aforementioned problems with quadrature, an efficient method of exact projection from one arbitrary triangular grid to another has been introduced. This provides an unconditionally stable approximation for the Lagrange–Galerkin method with the same error as if the method had been evaluated exactly. Moreover, it completely solves the problem in the cases of grid adaption and of triangular multigrid. Early numerical results for the Lagrange–Galerkin method using this technique applied to the Navier–Stokes equations are presented and appear quite promising, although this is the main area for further work.

ACKNOWLEDGMENTS

I thank Dr. Mike Baines for many useful and interesting discussions during the course of this work and Dr. Andrew Malcolm for help in generating the grids.

REFERENCES

1. I. Babuška, *Numer. Math.* **16**, 322 (1971).
2. J. P. Benqué, G. Labadie, and J. Ronat, in *Proceedings, 4th Int. Symp. on Finite Element Methods in Flow Problems*, edited by T. Kawai (North-Holland, Amsterdam/Oxford/New York, 1982), p. 295.
3. M. Bercovier and O. Pironneau, in *Proceedings, 4th Int. Symp. on Finite Element Methods in Flow Problems*, edited by T. Kawai (North-Holland, Amsterdam/Oxford/New York, 1982), p. 67.
4. M. Bercovier, O. Pironneau, and V. Sastri, *Appl. Math. Modelling* **7**, 89 (1983).
5. R. Bermejo, *Mon. Weather Rev.* **118**, 979 (1990).
6. F. Brezzi, *RAIRO Ser. Rouge Anal. Numér. R-2* (Dunod, Paris, 1979), p. 129.
7. J. C. Cavendish, *Int. J. Numer. Methods Eng.* **8**, 679 (1974).
8. M. A. Celia, I. Herrera, E. Bouloutas, and J. S. Kindred, *Numer. Methods Partial Diff. Eqs.* **5**, 203 (1989).
9. M. A. Celia, T. F. Russell, I. Herrera, and R. E. Ewing, *Adv. Water Res.* **13**, 187 (1990).
10. P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration* (Academic Press, New York/London, 1975).
11. H. Deconinck, R. J. Struijs, and P. L. Roe, in *Computational Fluid Dynamics*, VKI Lecture Series Notes, 1990-03, Brussels, March 1990.
12. J. Douglas, Jr. and T. F. Russell, *SIAM J. Numer. Anal.* **19**, 871 (1982).
13. J. Douglas, Jr. and J. Wang, *Math. Comput.* **52**, 495 (1989).
14. J. K. Dukowicz, *J. Comput. Phys.* **54**, 411 (1984).
15. J. K. Dukowicz, M. C. Cline, and F. L. Addessio, *J. Comput. Phys.* **82**, 29 (1989).
16. J. K. Dukowicz and J. W. Kodis, *SIAM J. Sci. Stat. Comput.* **8**, 305 (1987).
17. J. W. Eastwood and W. Arter, *Numerical Methods for Fluid Dynamics II*, edited by K. W. Morton and M. J. Baines (Oxford Univ. Press, Oxford, 1986), p. 581.
18. R. E. Ewing, T. F. Russell, and M. F. Wheeler, *Comput. Meth. Appl. Mech. Eng.* **47**, 73 (1984).
19. P. Garcia-Navarro and A. Priestley, *Int. J. Numer. Methods Fluids* **18**, 273 (1994).
20. U. Ghia, K. N. Ghia, and C. T. Shin, *J. Comput. Phys.* **48**, 387 (1982).
21. A. Ghizzetti and A. Ossicini, *Quadrature Formulae* (Birkhäuser Verlag, Basel, 1970).
22. P. M. Gresho, S. T. Chan, R. L. Lee, and C. D. Upson, *Int. J. Numer. Methods Fluids* **4**, 619 (1984).
23. P. C. Hammer and H. W. Wicke, *Math. Comput.* **14**, 3 (1960).
24. P. Hansbo, *Comput. Methods Appl. Method Eng.* **96**, 239 (1992).
25. P. Hansbo, *Comput. Methods Appl. Method Eng.* **99**, 171 (1992).
26. Y. Hasbani, E. Livne, and M. Bercovier, *Comput. & Fluids* **11**, 71 (1983).
27. G. S. Herrera and I. Herrera, *Finite Elements in Fluids: New Trends and Applications*, edited by K. Morgan, E. Oñate, J. Periaux, J. Peraire, and O. C. Zienkiewicz (Pineridge Press, Swansea, UK, 1993), p. 1428.
28. I. Herrera, *Finite Elements in Fluids: New Trends and Applications*, edited by K. Morgan, E. Oñate, J. Periaux, J. Peraire, and O. C. Zienkiewicz (Pineridge Press, Swansea, UK, 1993), p. 1437.
29. I. Herrera, R. E. Ewing, M. A. Celia, and T. F. Russell, *Numer. Methods Partial Diff. Eqs.* **9**, 431 (1993).
30. F. M. Holly, Jr. and A. Preissmann, *J. Hydraul. Eng.* **103**, 1259 (1977).
31. T. J. R. Hughes and L. P. Franca, *Comput. Methods Appl. Mech. Eng.* **65**, 85 (1987).
32. H. Jin and N.-E. Wiberg, *Int. J. Numer. Methods Eng.* **29**, 1501 (1990).
33. C. Johnson, *Comput. Methods Appl. Method Eng.* **100**, 45 (1992).
34. N. Kechkar and D. Silvester, *Math. Comput.* **58**, 1 (1992).
35. V. I. Krylov, *Approximate Calculation of Integrals* (MacMillan, New York/London, 1962).
36. J. D. Lambert and A. R. Mitchell, *Comput. J.* **5**, 322 (1962).
37. C. Lanczos, *Applied Analysis*, Chap. VI (Prentice-Hall, Englewood Cliffs, NJ, 1956).
38. P. Lesaint, *Topics in Numerical Analysis III*, edited by J. J. H. Miller (Academic Press, London/New York/San Francisco, 1977), p. 199.
39. S. H. Lo, *Int. J. Numer. Methods Eng.* **28**, 2695 (1989).

40. S. Mizohata, *The Theory of Partial Differential Equations* (Cambridge Univ. Press, 1973).
41. K. W. Morton and A. Priestley, *Pitman Research Notes in Mathematics Series*, edited by D. F. Griffiths and G. A. Watson (Longman Scientific & Technical, Harlow, 1986), p. 157.
42. K. W. Morton, A. Priestley, and E. E. Süli, *RAIRO Modél. Math. Anal. Numér.* **22** (4), 625 (1988).
43. J. Peraire, J. Peiro, and K. Morgan, *Comput. Methods Appl. Method Eng.*, to appear.
44. O. Pironneau, *Numer. Math.* **38**, 309 (1982).
45. A. Priestley, D.Phil. thesis, Oxford University, 1986.
46. A. Priestley, *J. Comput. Phys.* **106**, 139 (1993).
47. A. Priestley, *IMA J. Numer. Anal.*, April, 1994
48. A. Priestley, *IMA J. Numer. Anal.*, submitted.
49. A. Ramage and A. J. Wathen, *Finite Elements in Fluids: New Trends and Applications*, edited by K. Morgan, E. Oñate, J. Periaux, J. Peraire, and O. C. Zienkiewicz (Pineridge Press, Swansea, UK, p. 278; *Int. J. Numer. Methods Fluids*, to appear.
50. J. D. Ramshaw, *J. Comput. Phys.* **59**, 193 (1985).
51. J. D. Ramshaw, *J. Comput. Phys.* **67**, 214 (1986).
52. P. L. Roe, *J. Comput. Phys.* **63**, 458 (1986).
53. P. L. Roe, H. Deconinck, and R. J. Struijs, in *12th Int. Conf. on Numer. Methods in Fluid Dynamics, Oxford*, Lecture Notes in Physics, vol. 371 (Springer-Verlag, New York/Berlin, 1990), p. 273.
54. T. F. Russell, *SIAM J. Numer. Anal.* **22**, 970 (1985).
55. E. A. Sadek, *Int. J. Numer. Methods Eng.* **15**, 1813 (1980).
56. G. A. Schohl and F. M. Holly, Jr., *J. Hydraul. Eng.* **117**, 248 (1991).
57. D. J. Silvester, *Comput. Methods Appl. Method Eng.*, to appear.
58. W. Squire, *J. Soc. Indust. Appl. Math.* **9**, 94 (1961).
59. A. Staniforth and J. Côté, *Mon. Weather Rev.* **119**, 2206 (1991).
60. G. Struble, *Math. Comput.* **14**, 8 (1960).
61. E. E. Süli, *Numer. Math.* **53**, 459 (1988).
62. E. E. Süli, *The Mathematics of Finite Elements and Applications VI, MAFELAP 1987*, edited by J. R. Whiteman (Academic Press, London/New York/San Francisco, 1988), p. 435.
63. E. E. Süli and A. Ware, *SIAM J. Numer. Anal.* **28**, 423 (1991).
64. A. J. Wathen, *IMA J. Numer. Anal.* **7**, 449 (1987).
65. A. J. Wathen and D. J. Silvester, *SIAM J. Numer. Anal.* **30**, 630 (1993).