UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Numerical Solution of the Incompressible Navier-Stokes Equations in Infinite Domains

Christopher R. Anderson

January 2000
CAM Report 00-02

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

Numerical Solution of the Incompressible Navier-Stokes Equations in Infinite Domains

C.R. Anderson [*]

January 7, 2000

Abstract

We present a pressure-velocity based finite-difference method for solving the incompressible Navier-Stokes equation in infinite domains. An implementation of the method based upon a MAC grid discretization is discussed. Computational results include a comparison of the computed solution with an analytic solution for a translating cylindrical vortex in $\mathcal{R}^2$.

1 Introduction

In this paper we consider the task of computing solutions to the two-dimensional incompressible Navier-Stokes equations for flows in unbounded domains. Our aim is the construction of pressure-velocity based methods that approximate the infinite domain solution in a finite computational domain.

One immediate concern is the feasibility of constructing such methods. It seems extraordinary that one should be able create a computational procedure that operates in a finite domain, and yet captures infinite domain behavior. However, the fact that incompressible flow is completely determined by the dynamics of its vorticity allows such methods to be constructed. Specifically, even if a flow is in an unbounded region, the vorticity of the flow is often contained in a bounded region (or is at least negligible outside the bounded region). Thus, it is not too surprising to find that finite domain procedures based on the vorticity stream-function formulation exist. Vortex methods [5] (see [10] for a survey and list of references) are routinely used to create approximate solutions for flows in unbounded domains and finite difference methods [2] have also been used to create such solutions. The possibility that one can create vorticity stream-function based methods for infinite domain problems indicates that it should be possible to create pressure-velocity based methods. We show that this possibility can be realized; for an incompressible flow in an unbounded region where the vorticity is confined to a finite region, we describe a pressure-velocity based finite difference method that can be used to compute its evolution.

The construction of pressure-velocity based schemes for infinite domain problems is of interest because of the simplicity with which no-slip boundary conditions can be specified at solid boundaries. Additionally, if one is constructing finite difference schemes to simulate flows where the

*Research supported in part by AFOSR Grant F-49620-96-1-0327.

underlying vorticity is highly concentrated (e.g. in high Reynolds number flows with vortex shedding) one has fewer numerical problems evolving the velocity on a grid than evolving the vorticity on a grid.

The method we describe is a projection scheme [4]; a method where at each step the velocity field is advanced without regard to the incompressibility constraint and then the result is projected onto a velocity field that satisfies the constraint. There are two specific difficulties introduced by the infinite nature of the domain. One difficulty concerns the implementation of the projection operator in an infinite domain. The other difficulty concerns the specification of boundary conditions for the velocity at the outer edges of the computational domain. Our solution to the projection operator problem is to implement the projection using a vorticity stream-function approach. This translates the problem into one of computing the solution to Laplace's equation in an infinite domain — a problem to which Greens' function based techniques can be applied. With regard to the velocity boundary conditions, we show that as long as the computational domain completely contains the vorticity of the fluid, then the specification of the velocity at the computational boundary can be more or less arbitrary. Any errors in the velocity introduced by the boundary conditions are annihilated when the projection of the velocity field is computed.

In the second section we present the Navier-Stokes equations and review the principle components of projection methods for computing their solution. We also describe an implementation of the projection operator that uses a vorticity/stream-function construction and how an infinite domain can be accommodated in a projection based scheme. This description is given in general terms; discrete in time but continuous in space. In section three we present the details of a complete numerical method that uses MAC grid spatial discretizations.

One of the benefits of being able to compute solutions in infinite domains is that it provides greater opportunity to compare computational results with analytical results. In particular, we take advantage of this opportunity and compare the results of our numerical method to the analytic solution for the inviscid motion of a translating two dimensional cylindrical vortex in $\mathcal{R}^2$ [8].

2 A stream-function based projection method

The two dimensional incompressible Navier-Stokes equations that describe flow in a domain Ω are

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\nabla P + \nu \Delta \vec{u} \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

$$\begin{array}{ll} \vec{u}(\vec{x},0) = \vec{u}_0(\vec{x}) & \vec{x} \in \Omega \\ \vec{u}(\vec{x},t) = 0 & \vec{x} \in \partial\Omega \end{array}$$

Here $\vec{u} = (u, v)$ is the velocity, P is the pressure, and ν is the viscosity.

In the standard projection method [4] the solution of (1) is advanced one Euler time step by advancing the velocity without regard to the divergence condition (2) and then projecting this velocity onto its divergence free component. Specifically, one first computes the candidate velocity field $\vec{u}^*$ using the equation

$$\vec{u}^* = \vec{u}^n + dt * (-\vec{u}^n \cdot \nabla \vec{u}^n + \nu \Delta \vec{u}^n) \tag{3}$$

and then computes $\vec{u}^{n+1}$so that the relation

$$\vec{u}^* = \vec{u}^{n+1} + \nabla\Phi \tag{4}$$
$$\nabla \cdot \vec{u}^{n+1} = 0$$

is satisfied (e.g. $\vec{u}^{n+1}$is the projection of $\vec{u}^*$ onto it's divergence free component).

The most common technique for determining $\vec{u}^{n+1}$in (4) is to apply the divergence ($\nabla\cdot$) operator to (4) to obtain a Poisson equation for Φ,

$$\Delta\Phi = \nabla \cdot \vec{u}^*$$

Boundary conditions are obtained by taking the inner product of (4) with the normal vector $\vec{n}$ yielding

$$\frac{\partial\Phi}{\partial\vec{n}} = 0 \quad (x,y) \in \partial\Omega.$$

$\vec{u}^{n+1}$is then obtained by subtraction : $\vec{u}^{n+1} = \vec{u}^* - \nabla\Phi$. As will be discussed in section three, one can create higher order (in time) projection methods by combining Euler step solutions.

There is an alternative method for implementing the projection of $\vec{u}^*$, one that incorporates the use of a stream function Ψ. Specifically, if $\nabla \cdot \vec{u}^{n+1} = 0$, then there exists a stream function Ψ so that

$$\vec{u}^{n+1} = (\frac{\partial\Psi}{\partial y}, -\frac{\partial\Psi}{\partial x}) \tag{5}$$

An equation for Ψ can be obtained by applying the curl ($\nabla\times$) operator to (5). This results in the Poisson equation,

$$\Delta\Psi = -\nabla \times \vec{u}^{n+1} \tag{6}$$

The right hand side of (6) can be expressed in terms of $\vec{u}^*$ by applying the curl operator to equation (4); we find $\nabla \times \vec{u}^{n+1} = \nabla \times \vec{u}^*$. Dirichlet boundary conditions for Ψ are obtained by taking the inner product of (5) with the boundary normal. This inner product yields the expression $\frac{\partial\Psi}{\partial\vec{\tau}} = 0$ which implies that Ψ has a constant value c_i on each separate component of $\partial\Omega$.

Therefore, given $\vec{u}^*$, an alternate method for implementing the projection consists of first computing the stream function as a solution of

$$\Delta\Psi = -\nabla \times \vec{u}^* \tag{7}$$
$$\Psi = c_i \quad (x,y) \in \partial\Omega_i$$

where $\partial\Omega_i$ is the boundary of the ith object in the domain. $\vec{u}^{n+1}$ is determined through the relation (5). If the domain is simply connected then the constant in the boundary condition, c_0, can be arbitrary. For non-simply connected domains, the values of c_i are chosen so that the circulation about each object in the domain vanishes. If one desires the function Φ in (4) (the pressure $p = \frac{\Phi}{dt}$), then it can be obtained (up to a constant) by integrating (4) along any path in the domain from a fixed location. The fact that $\nabla \times \vec{u}^* - \nabla \times \vec{u}^{n+1} = 0$ ensures that this construction of Φ is path independent.

The use of (3) with (5) and (7) comprise what we refer to as a "stream function based" projection method.

2.1 Infinite domain considerations

If the domain is infinite, then the challenge is to create a computational procedure in a finite region (typically a rectangular region) that can be used to capture the infinite domain solution in that region. As mentioned in the introduction, it is possible to create such methods if the vorticity ($\omega = \nabla \times \vec{u}$) is confined to the computational region; we will also make this assumption.

In a stream function based projection method, there are two computational issues that must be addressed when the domain is infinite. One issue is the choice of boundary conditions for $\vec{u}$ at the edge of the computational boundary. These boundary conditions must be specified in order to compute $\vec{u}^*$. The other issue concerns computing the stream function used to implement the projection operator.

With regard to the boundary conditions on $\vec{u}$, we observe that $\vec{u}^*$ only influences $\vec{u}^{n+1}$ through its curl; so in one time step the errors introduced by any erroneous boundary conditions are only propagated as errors in $\nabla \times \vec{u}^*$. Moreover, with a sufficiently small time step, any the errors induced in $\nabla \times \vec{u}^*$will be confined to a narrow region at the outer edge of the computational boundary. Under our assumptions on the vorticity, the term $\nabla \times \vec{u}^*$ occurring in (7) will also be confined to the computational region and so should vanish near the computational boundary. Any erroneous values induced by incorrect boundary conditions can therefore be corrected by merely setting the value of $\nabla \times \vec{u}^*$ to zero. Thus, the boundary conditions for the velocity at the outer edge of the computational region can be specified more or less arbitrarily; errors are corrected in the projection step by setting the value of the $\nabla \times \vec{u}^*$ to zero near the edges of the computational domain.

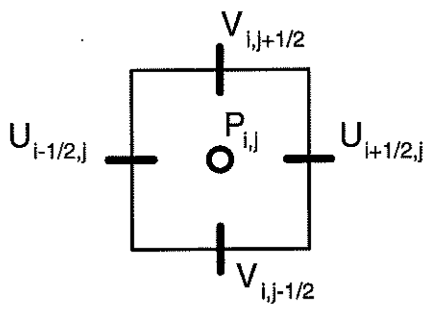
The other challenge to implementing a stream function projection method in infinite domains is the solution of (7) for the stream function. As mentioned above, our assumption on the vorticity allows us to conclude that $\nabla \times \vec{u}^*$ is confined to the computational region. If one has the Greens' function, $G(\vec{x}, \vec{y})$, for the Laplace operator in the domain, then a solution of (7) can be obtained by computing an integral over a finite region. Specifically, the integral over the infinite domain Ω becomes an integral over the finite computational domain $\mathcal{D}$;

$$\Psi(\vec{x}) = \iint\limits_{\Omega} G(\vec{x}, \vec{y}) * (\nabla \times \vec{u}^*)\, d\vec{y} = \iint\limits_{\mathcal{D}} G(\vec{x}, \vec{y}) * (\nabla \times \vec{u}^*)\, d\vec{y}$$

Thus, the computation of the solution to (7) can be computed in a finite region through the use of a Greens' function.

3 A numerical method for problems in $\mathcal{R}^2$

As an example of a method based upon the previous discussion, we consider the problem of computing a solution to the initial value problem (1) with $\Omega = \mathcal{R}^2$. Our computational domain $\mathcal{D}$ will be a rectangular region, and we assume for inviscid problems that $\nabla \times \vec{u} \subseteq \mathcal{D}$ for $t \in [0, T]$. In the case of viscous problems where $\nabla \times \vec{u}$ will be non-zero in all of $\mathcal{R}^2$, we assume that $\nabla \times \vec{u}$ is negligible outside of $\mathcal{D}$. For our spatial discretization we use a MAC grid. The location of the computational variables and their indexing is given in Figure 1. One reason for using a MAC grid is that when second order centered differences are used to approximate the velocity derivatives in (5) and in the curl and divergence operators, then the appropriate approximation of the Laplace operator used in the projection step (7) is the standard five point discrete Laplace operator. When the domain is rectangular, this allows one the luxury of using standard fast Poisson solvers to compute the stream function.



Staggered grid unit cell

Figure 1

Our discretization in time is a 3rd order Runge-Kutta method. This higher order Runge-Kutta method is used because of its greater temporal accuracy and its stability characteristics. In particular, its region of absolute stability contains a portion of the imaginary axis and can thus be used as a stable integrator for finite difference methods that use centered spatial derivative approximations. Implementing the Runge-Kutta scheme requires the combination of three velocity fields where each velocity field can be viewed as the result of a single Euler time step. We therefore describe the numerical procedure for advancing the discrete solution one Euler time step and then indicate how these are combined to obtain the complete solution.

The first task in computing one Euler step is to compute approximations to $\vec{u}^*$, defined by

$$\vec{u}^* = \vec{u}^n + dt * (-\vec{u}^n \cdot \nabla \vec{u}^n + \nu \Delta \vec{u}^n) \tag{8}$$

or, when expressed in conservation form,

$$\vec{u}^* = \vec{u}^n + dt * (-\nabla \cdot (\vec{u}^n (\vec{u}^n)^t) + \nu \Delta \vec{u}^n). \tag{9}$$

Any of a large number of methods can be used to compute approximations to the spatial derivatives occurring in (8) or (9). For simplicity, we've used the second order spatial discretization given in [9] of the conservation form (9),

$$u^*_{i+1/2,j} = u^n_{i+1/2,j} + dt * (-\nabla^h_x(u^2)^n_{i+1/2,j} - \nabla^h_y(uv)^n_{i+1/2,j} + \nu\Delta^h u^n_{i+1/2,j}) \qquad (10)$$
$$v^*_{i,j+1/2} = v^n_{i,j+1/2} + dt * (-\nabla^h_x(uv)^n_{i,j+1/2} - \nabla^h_y(v^2)^n_{i,j+1/2} + \nu\Delta^h v^n_{i,j+1/2})$$

where ∇^h_x, ∇^h_y are standard forward divided differences (which are actually centered about half-index variables), Δ^h the standard five point discrete Laplace operator, and

$$(u^2)_{i,j} = \tfrac{1}{4}(u_{i+1/2,j} + u_{i-1/2,j})^2$$
$$(uv)_{i,j} = \tfrac{1}{4}(u_{i+1/2,j} + u_{i-1/2,j})(v_{i,j+1/2} + v_{i,j-1/2})$$
$$(v^2)_{i,j} = \tfrac{1}{4}(v_{i,j+1/2} + v_{i,j-1/2})^2$$

For problems where the solution is not smooth, or large gradients are expected, then more advanced discretization procedures can be used. (In the computation of flow about a flat plate we've obtained good results using third order ENO [11][6] discretizations of (8)). In order to close the difference approximations (10) boundary conditions at the edge of the computational domain must be specified. Since these can be specified more or less arbitrarily, we specified $\vec{u}^n = 0$.

After the computation of $\vec{u}^*$, the next step is the computation of its projection. In the projection step we used centered second order discretizations for the derivatives occurring in the curl, the divergence, and the derivatives in (5), specifically,

$$u_{i+1/2,j} = \frac{(\Psi_{i+1/2,j+1/2} - \Psi_{i+1/2,j-1/2})}{dy} \qquad (11)$$
$$v_{i,j+1/2} = -\frac{(\Psi_{i+1/2,j+1/2} - \Psi_{i-1/2,j+1/2})}{dx}$$

$$(\nabla \times \vec{u})_{i+1/2,j+1/2} = \frac{(u_{i+1/2,j+1} - u_{i+1/2,j})}{dy} - \frac{(v_{i+1,j+1/2} - v_{i,j+1/2})}{dx} \qquad (12)$$

$$(\nabla \cdot \vec{u})_{i,j} = \frac{(u_{i+1/2,j} - u_{i-1/2,j})}{dx} + \frac{(v_{i,j+1/2} - v_{i,j-1/2})}{dy} \qquad (13)$$

With this choice of difference operators the approximation of the Laplace operator in (7) that yields a discrete velocity field so that (4) will be satisfied is the standard five point Laplacian. This can be verified by applying the discrete curl operator (12) to (11). (Alternately, one can assume that the five point discrete Laplace operator is used and then verify that a velocity field defined by (11) will be discretely divergence free when the divergence is computed with (13).)

The first step in the calculation of the projection is the computation of $\nabla \times \vec{u}^*$. This is done using (12). Since the velocity boundary values $\vec{u}^n = 0$ will create erroneous values of $\nabla \times \vec{u}^*$ near the edge of the computational boundary, the values of $\nabla \times \vec{u}^*$ are set to zero at grid nodes on the boundary and one mesh width away from the boundary. Only points one mesh width from the boundary require correction because the stencil width of the spatial approximations is one mesh width. For discretizations with wider stencil widths, all values within its stencil width from the computational boundary will need to be corrected.

The next step in the projection is the calculation of the solution of

$$\Delta^h \Psi = -\nabla \times \vec{u}^* \tag{14}$$

where the domain is all of $\mathcal{R}^2$. As indicated in section two, in the continuous case this can be computed using the Greens' function for the domain. The same holds true for the discrete case. In particular we can compute the solution of (14) as

$$\Psi_{i+1/2,\, j+1/2} = \sum_{p,q} -(\nabla \times \vec{u}^*)_{p+1/2,q+1/2} G^h(p+1/2, q+1/2, i+1/2,\, j+1/2) \tag{15}$$

where G^h is the Greens' function for the discrete Laplacian given in [3]. Since $-(\nabla \times \vec{u}^*)_{p+1/2,q+1/2}$ vanishes outside the computational boundary, the sum occurring in (15) is a finite sum over grid points in the computational domain. The calculation of (15) as the formula is written requires $\mathrm{O}(N^2)$ operations if the number of grid points in the domain is N. This complexity can be reduced by computing Ψ as a sum of two calculations, one calculation is the result of a standard Dirichlet problem in the computational domain, and the other a discrete harmonic function added to correct this result so that the sum solves the infinite domain problem. With such a construction the complexity associated with evaluating (15) becomes $\mathrm{O}(N\log(N)) + \mathrm{O}(\sqrt{n})$. (See [7] for a more detailed description of this latter procedure.)

After the solution to (14) is computed, the projected velocity field is evaluated using the approximations given in (11).

The extension of the single Euler step to a Runge-Kutta step is straight forward. We follow the standard Runge-Kutta procedure where each stage is an Euler step, and just apply the projection projection operator to each stage. Specifically, if $\mathcal{F}$ is used to denote the application of the convection diffusion operator and $\mathcal{P}$ is the projection of a velocity field onto it's divergence free component, then the time stepping formula becomes

$$\begin{aligned}
\vec{k}_1^* &= \vec{u}^n + dt * \mathcal{F}(\vec{u}^n) \\
\vec{k}_1 &= \mathcal{P}(\vec{k}_1^*) \\
\vec{k}_2^* &= \vec{u}^n + \frac{dt}{3} * \mathcal{F}(\vec{k}_1) \\
\vec{k}_2 &= \mathcal{P}(\vec{k}_2^*) \\
\vec{k}_3^* &= \vec{u}^n + \frac{2dt}{3} * \mathcal{F}(\vec{k}_2) \\
\vec{k}_3 &= \mathcal{P}(\vec{k}_3^*) \\
\vec{u}^{n+1} &= \vec{u}^n + dt * (\frac{1}{4}\vec{k}_1 + \frac{3}{4}\vec{k}_3)
\end{aligned}$$

Note that $\vec{u}^{n+1}$ need not be projected since it is the linear combination of velocity fields that have zero divergence.

4 Computational Results

As a test of the accuracy of the method described above we computed the solution to the inviscid Navier-Stokes equations corresponding to a translating cylindrical vortex [8]. The velocity field of the solution is given by

$$u(x,y,t) = \begin{cases} U_s + (\frac{C}{\bar{r}})\cos(2\bar{\theta})J_1(k\bar{r}) + Ck\sin^2(\bar{\theta})J_0(k\bar{r}) & \bar{r} \le a \\ U_s(\frac{a^2}{\bar{r}^2})\cos(2\bar{\theta}) & \bar{r} > a \end{cases}$$

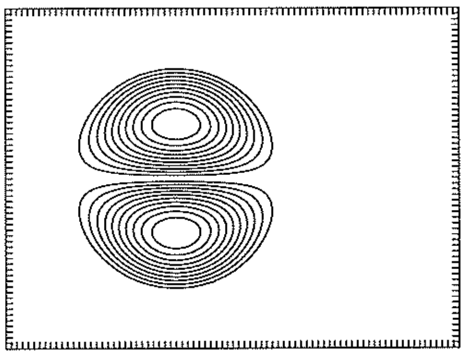
$$v(x,y,t) = \begin{cases} (\frac{C}{\bar{r}})\sin(2\bar{\theta})J_1(k\bar{r}) - Ck\sin(\bar{\theta})\cos(\bar{\theta})J_0(k\bar{r}) & \bar{r} \le a \\ U_s(\frac{a^2}{\bar{r}^2})\sin(2\bar{\theta}) & \bar{r} > a \end{cases} \tag{16}$$

where $\bar{r} = \sqrt{\bar{x}^2 + \bar{y}^2}$, $\bar{x} = x - U_s t$, and $\bar{y} = y$. U_s is the translational velocity of the vortex in the x-direction, J_0 and J_1 are the Bessel functions of order zero and one respectively, a is the vortex radius, k is determined so that the product ka is the first root of J_1 (=3.831705...), and the constant $C = \frac{-2U_s}{kJ_0(ka)}$. The stream function and vorticity corresponding to this solution are given by

$$\Psi(x,y,t) = \begin{cases} CJ_1(k\bar{r})\sin(\bar{\theta}) + U_s\bar{r}\sin(\bar{\theta}) & \bar{r} \le a \\ U_s(\frac{a^2}{\bar{r}^2})\sin(\bar{\theta}) & \bar{r} > a \end{cases}$$

$$\omega(x,y,t) = \begin{cases} Ck^2 J_1(k\bar{r})\sin(\bar{\theta}) & \bar{r} \le a \\ 0 & \bar{r} > a \end{cases}$$

With the value $U_s = 1$, the solution consists of a cylindrical vortical structure translating to the right with speed 1. Figure 2 contains contour plots of the vorticity and stream function at time $t = 0.0$


Vorticty Contours -10:10 (1)

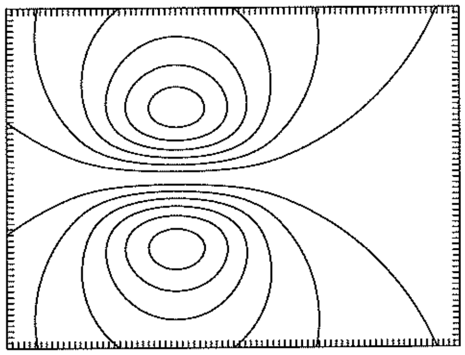

Stream Function Contours -2:2 (.2)

Figure 2

The velocity field of this solution is continuously differentiable, but the second derivatives are discontinuous (This can be deduced from the discontinuous nature of the derivatives of the vorticity distribution at the edge of the vortex). Even though this problem is not a good one for showing the benefits of high order spatial discretization, there is sufficient smoothness to test convergence of the numerical method to an analytic solution for an infinite domain flow problem.

We choose the vortex to be of unit radius ($a = 1$), and the rectangular region $[-1.5, 3.5] \times [-1.5, 1.5]$ to be the computational domain. Mesh sizes of $dx = dy = .1, .05$ and $.025$ were used. The solution was computed from $t = 0$ to $t = 2.0$, using a time step $dt = dx/4$ (so that the CFL restriction is satisfied). Over the computational time interval, the center of the vortex translates from the origin to the point $(2, 0)$.

To assess the spatial accuracy we computed the errors in the solution obtained with different mesh sizes. The errors were measured in the discrete L^2 norm; for a scalar function $||f - f^h||_2 = \sqrt{\sum_{ij} f_{i,j}^2\, dxdy}$ and for vector functions $||\vec{g} - \vec{g}^h||_2 = ||g_1 - g_1^h||_2 + ||g_2 - g_2^h||_2$. The results are presented in Table 1.

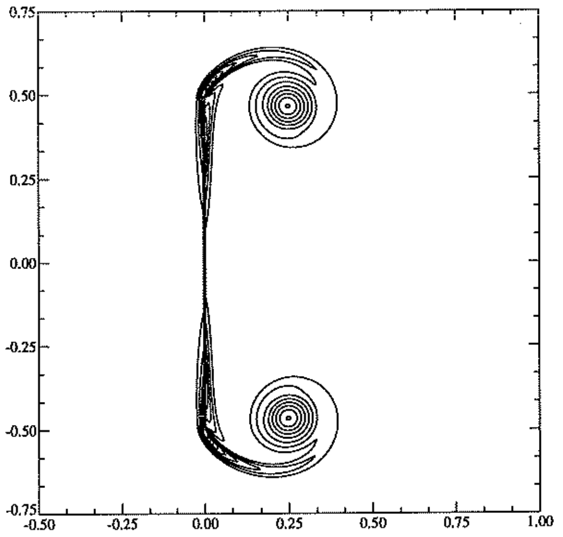
Mesh Size	$\lvert\lvert\omega - \omega^h\rvert\rvert_2$	$\lvert\lvert\vec{u} - \vec{u}^h\rvert\rvert_2$	$\lvert\lvert\Psi - \Psi^h\rvert\rvert_2$
.1	0.984982	0.188648	0.038352
.05	0.352738	0.048957	0.010492
.025	0.155330	0.014167	0.002525

Table 1
Errors in the vorticity, ω, the velocity, $\vec{u}$, and the stream function,Ψ, at time $t = 2.0$.

The first observation that can be made is that the numerical solution is converging to the infinite domain solution. Thus, even though the computational domain is finite and remains fixed as the mesh size is decreased, the infinite domain solution is accurately predicted. The errors in the stream function diminish by a factor of approximately four as the mesh size is reduced by a factor of two, indicating a second order rate of convergence. With the finer meshes, the convergence rate of the velocity is a bit less, ~1.78 and the convergence rate of vorticity is slightly more than first order ~1.18. These reduced rates of convergence are expected, since the velocity has discontinuous second derivatives and the vorticity has discontinuous first derivatives.

To assess the accuracy of the procedure on a smooth solution, we computed the solution to the viscous Navier-Stokes equation using (16) at $t = 0$ as an initial condition and a value of viscosity $\nu = 0.025$. The rates of convergence were estimated at each point using Aitken extrapolation [1]. An average rate of convergence was obtained by averaging the pointwise rates that were less than three in magnitude (this discounts spurious rate values). The average rate of convergence for the stream function was 2.11835, for the average U velocity 2.09488, and for the vorticity 1.96398. For the inviscid problem the rates as estimated by Aitken extrapolation are 2.0016 for the stream function, 1.93854 for the average U velocity and 1.67921 for the vorticity. As expected, we find that the convergence rates improve with the smoothness of the solution.

The method described for $\mathcal{R}^2$ can be extended to other infinite domains as well. As an example of this, in Figure 3 we present the vorticity associated with the solution of the viscous Navier-Stokes equations for flow about a flat plate.



Vorticity Contours -100:100 (10)

Figure 2
Vorticity contours for flow past a flat plate at time $t = 0.5$.
Reynolds number = 800, dx=dy=.0025, dt = .00167

Due to the presence of large velocity gradients in this flow, a third order weighted ENO scheme [6] was used for the spatial discretization. The solution of Poisson's equation for the flow exterior to the plate was also solved using a discrete Greens' function based method

References

[1] Atkinson, K.E., *An Introduction to Numerical Analysis* 2nd. Ed., John Wiley and Sons, NY, (1989).

[2] C.R. Anderson, M.B. Reider, *A High Order Explicit Method for the Computation of Flow About a Circular Cylinder.* Journal of Computational Physics, 125, 207-224, (1996).

[3] Buneman, O., *Analyitic Inversion of the Five-Point Poisson Operator, Journal of Computational Physics*, 8, 500-505, (1971).

[4] Chorin, A.J. *Numerical Solution of the Navier-Stokes Equations*, Math. Comp. 22, 745 (1968).

[5] A. J. Chorin, and P. S. Bernard, *Discretization of a vortex sheet with a example of roll-up.* Journal of Computational Physics, 13, 423-429 (1973).

[6] Jiang, G.-S. and Peng, D., *Weighted ENO Schemes for Hamilton Jacobi Equations*, UCLA CAM Report 97-29, June 1997, SIAM J. Num. Analysis (to appear).

[7] James, R.A., *The Solution of Poisson's Equation for Isolated Source Distributions*, Journal of Computational Physics, 25, 71-93 (1977).

[8] Lamb, H., *Hydrodynamics*, Dover Publications, NY, (1945).

[9] Peyret, R. and Taylor, T.D., *Computational Methods for Fluid Flow*, Springer-Verlag, NY, (1983).

[10] Sarpkaya, T., *Computational Methods With Vortices — The 1988 Freeman Schloar Lecture, Journal of Fluids Engineering*, 111, 5-52, (1989)

[11] Shu, C.W. and Osher, S., *Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes*, Journal of Computational Physics, 88,.439-471, (1988).

[12] Shu, C.W. and Osher, S., *Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes II (two)*, Journal of Computational Physics, 83, 32-78, (1989).