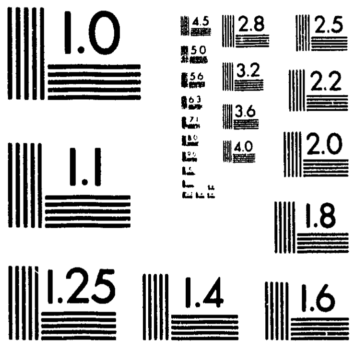

1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

1 of 1

LA-12536-MS

UC-361
Issued: August 1993

Implementation of the Turn Function Method in KIVA-F90

Peter J. O'Rourke
Margaret S. Fairfield

Los Alamos
NATIONAL LABORATORY
Los Alamos, New Mexico 87545

[illegible]

IMPLEMENTATION OF THE TURN FUNCTION METHOD IN KIVA-F90

by

Peter J. O'Rourke and Margaret S. Fairfield

ABSTRACT

We document the implementation of the Turn Function Method (TFM) in KIVA-F90, a version of the KIVA computer program written in the FORTRAN 90 programming language that is used on some massively parallel computers. TFM solves both linear momentum and vorticity equations in numerical calculations of compressible fluid flow. Solving a vorticity equation allows vorticity to be both conserved and transported more accurately than is possible in traditional methods for computing compressible flow. To calculate the convective transport of vorticity more accurately, we have also implemented an improved method for the rezone phase of the KIVA computational cycle.

This first implementation of TFM in a three-dimensional hydrodynamics code involved some modification of the original method and some novel numerical difference approximations. In particular, we used a penalty method to keep the divergence of the computed vorticity field close to zero. Also, difference operators are defined in such a way that the finite difference analog of $\nabla \cdot (\nabla \times \mathbf{u}) = 0$ is exactly satisfied.

Three example problems presented in this report show the greater accuracy that can be gained when TFM is used for calculations of flows with rotational motion, as well as the added computational costs incurred. Use of the method can increase by 60% the computational times of the Euler equation solver in KIVA-F90. Although TFM sometimes does degrade accuracy in calculations that have large vorticity gradients, generally the calculations show beneficial effects of TFM on accuracy. We will continue to evaluate the costs and benefits of TFM in future KIVA-F90 calculations.

I. INTRODUCTION

We were motivated to conduct this study because of the need to calculate rotational motion more accurately in numerical computations of fluid flow. In computer simulations of fluid flow, numerical errors usually cause significant nonconservation of angular momentum and vorticity and numerical dissipation of the kinetic energy of rotational motion. For this reason, most currently used numerical methods do a poor job of calculating rotational motion. This can affect the solution's accuracy in many ways. For example, inaccuracies in the calculation of angular momentum give rise to inaccurate predictions of convective transport by the mean flow field. In addition, since numerically dissipated mean flow energy is no longer available for augmenting turbulent kinetic energy, the prediction of the effects of turbulence on the mean flow is inaccurate.

There are many applications that could benefit from more accurate prediction of rotational motion. For example, in direct simulations or large-eddy simulations of turbulent flows mean flow kinetic energy first breaks down into small-scale vortices before it is dissipated into heat. Our specific interest is in internal combustion engine flows, in which rotational motion is often intentionally introduced to enhance the combustion process. Two types of rotational motion are used: swirl, which has its axis of rotation aligned with the engine cylinder axis, and tumble, which has its axis of rotation orthogonal to the cylinder axis. Combustion efficiency is improved because the kinetic energy of rotational motion breaks down into fluid turbulent energy when the piston approaches the cylinder head. This turbulent energy, in turn, enhances the mixing of the fuel, the intake air, and the hot combustion products in the cylinder, speeding up the combustion process. Other applications involving significant fluid rotation include gas turbine engines, liquid propellant rocket engines, and geophysical fluid flows.

Because of the importance of rotational motion in improving combustion efficiency, it has been observed[1] that it is probably more important to conserve angular momentum than linear momentum in numerical calculations of internal combustion engine flows. Very early in the development of the KIVA codes[2,3] for the numerical calculation of internal combustion engine flows, we recognized the problem of numerical decay of angular momentum and incorporated special logic that conserved angular momentum of the flow about the axis of the engine cylinder. KIVA is based on the ALE (Arbitrary Lagrangian-Eulerian) method,[4] in which the updating of the transient solution through one timestep is accomplished in a two-stage calculation. In the first stage, the fluid properties are updated in a Lagrangian calculation in which the computational mesh moves with the fluid. In the second stage, the flow field obtained by the Lagrangian calculation is mapped onto an adjusted computational mesh with the same logical structure as the original mesh but with regularized coordinates that remove Lagrangian mesh distortion. The second-stage calculation involves the convection of flow properties across cell boundaries and conserves mass, linear momentum, and energy. Test calculations showed that almost all angular momentum nonconservation occurred in this convection stage. Accordingly, we devised special logic for the conservative calculation of convection of the component of angular momentum about the engine cylinder axis.[2] After the convection calculation a simple explicit adjustment of the velocities made them consistent with the computed angular momentum field. The advantages of this angular momentum conservation logic are its simplicity and its computational efficiency. Its disadvantages are that it only conserves one component of angular momentum, the swirl component, and that the velocity adjustment destroys linear momentum conservation.

An alternative to this angular momentum conservation logic is the Turn Function Method (TFM).[5] This is a numerical method that conserves both linear momentum and vorticity and that treats all coordinate directions in the same manner. TFM has previously been used very little because of its large computational cost: in three space dimensions, four implicit equations must be solved for each timestep. Because of the greater computer power available on the new massively parallel computers, however, we were encouraged to implement TFM in our version of KIVA, which runs on the Connection Machine CM-2, in order to assess whether the increased accuracy obtainable with the method was worth its computational cost.

This report documents our implementation of TFM in KIVA-F90. We first give the equations solved by TFM and the modifications that must be used when the fluid equations are being solved in three dimensions. Then, we describe the numerical implementation of the method in KIVA-F90. Finally, we compare the results of three test problems run both with and without TFM. Although use of the method can increase by 60% the computational times of the Euler equation solver in KIVA-F90, we believe the increased cost is warranted by the increased computational accuracy that we obtained in the test calculations.

II. THE TURN FUNCTION METHOD

A. The Equations

In TFM one numerically solves transport equations for both linear momentum and vorticity in compressible flow calculations. The numerical solutions are consistent in the sense that the numerical curl of the computed velocity field is approximately equal to the computed vorticity field. Use of TFM is not associated with a particular spatial or temporal difference scheme. The method has been implemented in a two-dimensional Eulerian code with cell-face velocities,[5] and it is currently used in the three-dimensional ALE-method code KIVA-F90, which has velocities located at the vertices of hexahedrons.

Using TFM improves the calculation of rotational motion in two ways. First, the vorticity transport equation can be formulated and differenced in a conservative fashion. Second, because the transport terms explicitly appear in the TFM equations, more accurate calculation of vorticity transport is possible using higher-order difference schemes.

The linear momentum and vorticity equations solved by TFM each contain an additional conservative transport term that is not normally found in the fluid flow equations:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) + \nabla p = \nabla \cdot \boldsymbol{\sigma} + \nabla \times \boldsymbol{\tau} \tag{1}$$

and

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \cdot (\mathbf{u}\boldsymbol{\omega}) + \nabla \times (\tfrac{1}{\rho}\nabla p) = \nabla \cdot (\boldsymbol{\omega}\mathbf{u}) + \nabla \times \left(\tfrac{1}{\rho}\nabla \cdot \boldsymbol{\sigma}\right) + \nabla \phi \tag{2}$$

where ρ is the mass density, $\mathbf{u}$ is the velocity, p is the thermodynamic pressure, $\boldsymbol{\sigma}$ is the fluid stress tensor, and $\boldsymbol{\omega}$ is the vorticity. Standard forms of the mass and energy equations are solved.

The vector field $\boldsymbol{\tau}$ in Eq. (1) is called the Turn Vector and is given by

$$\boldsymbol{\tau} = \mu_\tau (\boldsymbol{\omega} - \nabla \times \mathbf{u}) \ . \tag{3}$$

Thus $\boldsymbol{\tau}$ is proportional to the difference between the computed vorticity field and the curl of the computed velocity field. The proportionality factor μ_τ, which has dimensions of a viscosity, will be specified later. The exact solution to Eqs. (1) and (2) is $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ and $\boldsymbol{\tau} = \mathbf{0}$, but the curl of the computed velocity field and the computed vorticity field can diverge from each other due to numerical errors in the approximations to Eqs. (1) and (2). The role of the Turn Vector in Eq. (1) is to apply a local torque to the fluid to keep the computed velocity and vorticity fields consistent. The boundary condition on walls is $\boldsymbol{\tau} = \mathbf{0}$, so that linear momentum is not lost at walls as a result of the addition of the Turn Vector term. In two space dimensions, $\boldsymbol{\tau}$ has one component, which is normal to the plane of the flow and is called the Turn Function.

The gradient of scalar ϕ is added to the vorticity equation, Eq. (2), to keep the divergence of the computed vorticity field close to zero. This term was not in the equations of the original Turn Function paper[5] but is necessary in three-dimensional implementations. The function ϕ is given by

$$\phi = v_\tau \nabla \cdot \boldsymbol{\omega} \ , \tag{4}$$

where $v_\tau = \mu_\tau/\rho$. By Eq. (4), ϕ is proportional to the divergence of the computed vorticity. The exact solution to Eq. (2) has $\nabla \cdot \boldsymbol{\omega}$ equal to $\mathbf{0}$, but because of numerical errors the computed divergence of $\boldsymbol{\omega}$ can drift from zero, and the gradient of ϕ in Eq. (2) provides a restoring force that cancels this effect. The boundary condition on walls is $\phi = 0$, so that vorticity is not lost at walls. Use of ϕ here is closely analogous to the use of an artificial bulk viscosity in penalty methods for incompressible flow to keep the divergence of the velocity field close to zero[6] and for MHD flows to keep the divergence of the magnetic field close to zero.[7]

B. The Discrepancy Diffusivity ν_T

We now discuss our choice of the parameter $\nu_T = \mu_T/\rho$, which we believe is better than that used in earlier work.[5] Parameter ν_T serves as a diffusion coefficient for the discrepancies $\mathbf{d}_1 = \boldsymbol{\omega} - \nabla \times \mathbf{u}$ and $d_2 = \nabla \cdot \boldsymbol{\omega}$, as can be seen when transport equations for $\mathbf{d}_1$ and d_2 are derived. The $\mathbf{d}_1$ equation is given in Ref. 5. The equation for d_2 is

$$\frac{\partial d_2}{\partial t} = \nabla \cdot \mathbf{T} + \nabla^2 (v_T d_2) \ , \tag{5}$$

where $\mathbf{T}$ represents the truncation errors of the discretized vorticity equation. In their transport equations, $\mathbf{d}_1$and d_2 are produced by numerical truncation errors and diffused with diffusion coefficient ν_T. The discrepancies are destroyed at computational boundaries because of the boundary conditions on $\boldsymbol{\tau}$ and ϕ. If we take $\nu_T = +\infty$ (or some very large number) in our calculations, the computed approximations to $\mathbf{d}_1$ and d_2 would be equal to zero since $\boldsymbol{\tau}$ and ϕ remain finite. While this is an appealing reason for choosing $\nu_T = +\infty$, there are two reasons for taking a finite value of ν_T. First, we must choose ν_T finite in order to limit the upstream propagation of TFM-generated disturbances in the computed velocity and vorticity fields.[5] These disturbances propagate with speed ν_T/L, where L is the disturbance wavelength—typically a gradient length of the flow field. We wish to limit this disturbance speed to being less than a characteristic flow speed. If this is not done, disturbances can propagate upstream of a region of discrepancy production. Thus, we must choose ν_T so that

$$\nu_T \lesssim U_0 L \ , \tag{6}$$

where U_0 is a characteristic flow velocity.

Second, we want to choose ν_T finite because computational efficiency improves with smaller values of ν_T . This is because we are, in effect, solving discrepancy transport equations with diffusion terms whose diffusion coefficient is ν_T. The values that we use for ν_T are large enough that it is necessary to use implicit differencing of the diffusion terms, and an iterative solution procedure will be required. In KIVA-F90 we use a conjugate residual method.[8] In solving implicitly discretized diffusion equations, the convergence rates of most iterative methods are improved when smaller values of the diffusion coefficients are used.[9]

Balancing these two reasons for taking ν_T small is the requirement that we take ν_T large enough that the discrepancies remain tolerably small. We can estimate the magnitudes of the discrepancies by examining their transport equations. We now do this for the d_2-equation, Eq. (5). Examination of the $\mathbf{d}_1$-equation gives a similar result. In Eq. (5), the diffusion terms approximately balance the truncation error terms that produce the discrepancy. The discrepancy diffusion terms have order of magnitude

$$\frac{\nu_T D}{L^2} , \tag{7}$$

where D is a characteristic value of the discrepancy d_2 . As we will see later, the truncation errors that concern us are associated with difference approximations to the convection terms of the linear momentum and vorticity equations. The spatial truncation errors of the vorticity convection scheme in KIVA-F90 (see Appendix A) give rise to production terms in the d_2-equation that have order of magnitude

$$\frac{U_0^2}{L^3}\left(\frac{\delta x}{L}\right)^n , \tag{8}$$

where δx is the computational cell size and n is the order of accuracy. On a uniform mesh, $n = 2$ when gradients in the computed quantities are well resolved. However, because a gradient-limiting procedure is used,[3] n can be unity in regions of steep gradients. The lowest order temporal truncation errors of the vorticity convection terms are second order. These errors give rise to discrepancy production terms with approximate magnitude

$$\frac{U_0^2}{L^3}\left(\frac{U_0\delta t}{L}\right)^2 = \frac{U_0^2}{L^3}\left(\frac{\delta x}{L}\right)^2\left(\frac{U_0\delta t}{\delta x}\right)^2 \tag{9}$$

where δt, the timestep associated with the subcycled convection calculation,[3] is such that $U_o\delta t < \delta x$. Equating the magnitudes of the diffusion terms (7) and of the largest of the truncation errors (8) and (9) gives the magnitude of the discrepancy d_2 :

$$D = \frac{U_0^2}{\nu_T}\frac{\delta x}{L^2} . \tag{10}$$

We want the magnitude of d_2 to be less than some small number ϵ times U_0/L^2. From Eq. (10), this is equivalent to requiring that

$$\nu_T \gtrsim \frac{1}{\epsilon}U_0\delta x . \tag{11}$$

Requirements (6) and (11) on ν_T are satisfied if

$$\frac{\delta x}{L} \lesssim \epsilon \tag{12}$$

and if

$$\nu_T = \frac{1}{\epsilon} U_0 \delta x \ . \tag{13}$$

Inequality (12) is a requirement that gradients in the flow be well resolved. In our calculations, we have used ν_T given by Eq. (13) and taken $\epsilon = 0.1$.

It can now be seen why it is necessary to finite difference the Turn Vector terms implicitly if we use a timestep that is equal to or larger than the convective timestep of KIVA-F90, which satisfies $U_0 \delta t \approx 0.5\, \delta x$. In order to use explicit differencing with timestep δt, it would be necessary to satisfy the diffusional stability condition for a three-dimensional mesh:

$$\nu_T \delta t \leq \frac{1}{6} \delta x^2 \ . \tag{14}$$

By substituting the value of ν_T of Eq. (13) with $\epsilon = 0.1$, we see that δt would be required to satisfy

$$U_0 \delta t \leq 0.017\, \delta x \ . \tag{15}$$

This constrains δt to be more than an order of magnitude smaller than the convective timestep we currently use.

III. NUMERICAL IMPLEMENTATION IN KIVA-F90

We now describe the numerical implementation of TFM in KIVA-F90. First we give the locations of the new cell variables $\boldsymbol{\tau}$, $\boldsymbol{\omega}$, and ϕ and the spatial difference approximations to the derivatives $\nabla \times \boldsymbol{\tau}$, $\nabla \phi$, $\nabla \times \mathbf{u}$, and $\nabla \cdot \boldsymbol{\omega}$. We show that the finite difference analog of the differential identity $\nabla \cdot (\nabla \times \mathbf{u}) = 0$ is satisfied. Then we give the finite difference approximation to the vorticity transport equation and our method of solving for the function ϕ in the vorticity equation and the Turn Vector term in the linear momentum equation. Finally, we compare TFM with a related method developed by Dukowicz and Meltz.[10]

In KIVA velocities are located at the vertices of computational cells and thermodynamic variables are located at cell centers.[2,3] The control volumes surrounding vertices are called momentum cells, and those surrounding cell-centered quantities are called regular cells. In general, spatial derivatives of cell-centered quantities are located at vertices, and spatial derivatives of vertex quantities are cell-centered. Thus, for example, the ordinary fluid stress tensor, which depends on the rate of strain, is located at cell centers. It seems natural, therefore, to locate the vorticity $\boldsymbol{\omega}$ and Turn Vector $\boldsymbol{\tau}$ at cell centers. Since the vorticity is located at cell centers, its divergence is located at vertices, as is the function ϕ. The locations of these new cell variables are indicated in Fig. 1.

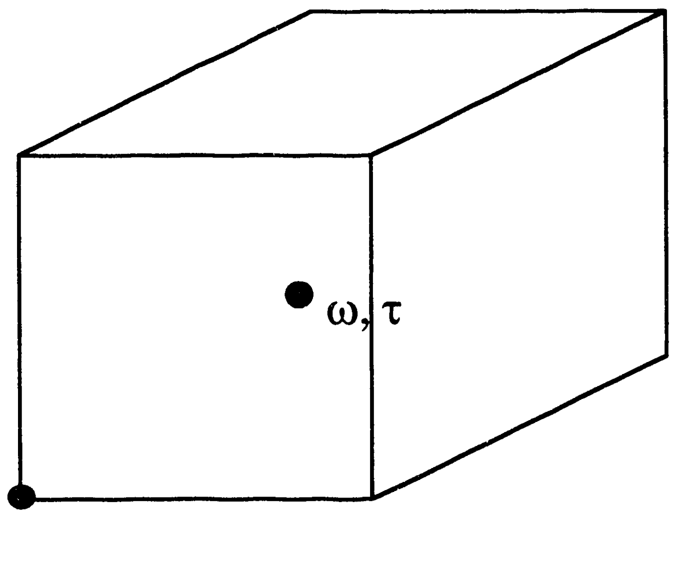


Fig. 1. Locations of new cell variables $\boldsymbol{\omega}$, $\boldsymbol{\tau}$, and ϕ.

Finite difference approximations are needed for the curl of the cell-centered quantity $\boldsymbol{\tau}$ and the divergence of the cell-centered quantity $\boldsymbol{\omega}$. For these approximations, given below, we use the same discretized form of the gradient operator $\mathbf{D}$ that is used to approximate the derivatives of the fluid stress terms in the linear momentum equation in KIVA:

$$FD(\nabla \times \boldsymbol{\tau}) = \mathbf{D}_v \times \boldsymbol{\tau} = \frac{1}{V_v} \sum_{\beta} (\mathbf{A}'_{\beta} \times \boldsymbol{\tau}_{\beta}) \qquad (16)$$

and

$$FD(\nabla \cdot \boldsymbol{\omega}) = \mathbf{D}_v \cdot \boldsymbol{\omega} = \frac{1}{V_v} \sum_{\beta} (\mathbf{A}'_{\beta} \cdot \boldsymbol{\omega}_{\beta}) \ , \qquad (17)$$

where V_v is the volume of momentum cell v, β is an index that runs over the faces of the momentum cell, $\mathbf{A}'_{\beta}$ is the outward area projection vector of face β , and $\boldsymbol{\tau}_{\beta}$ (resp. $\boldsymbol{\omega}_{\beta}$) is the value of $\boldsymbol{\tau}$ (resp. $\boldsymbol{\omega}$) in the regular cell in which face β is located. The momentum cell volumes and their faces are defined in the KIVA documentation.[2,3]

The approximation to the gradient of vertex quantity ϕ is given by

$$FD(\nabla\phi) = \mathbf{D}_c\phi = \frac{1}{V_c}\sum_{\alpha} \mathbf{A}_\alpha \phi_\alpha \ , \tag{18}$$

where V_c is the volume of the regular control cell, α is an index that runs over all faces of the regular cell, $\mathbf{A}_\alpha$ is the outward area projection vector of face α, and ϕ_α is the average of the values of ϕ at the four vertices that define face α. Note that Eqs. (16) and (18), which approximate terms in the linear momentum and vorticity equations, respectively, conserve both momentum and vorticity since the cell-face area projection vectors change sign when "viewed" from neighboring cells.

The above difference approximations to spatial derivatives are obtained by integrating the derivatives over their appropriate control volumes, converting volume integrals to surface integrals using the divergence theorem, and approximating the surface integrals assuming that integrands are uniform within the cells in which the faces are located. In defining the approximation to the curl of the velocity field, however, we depart from this procedure. This is because we want to define $FD(\nabla \times \mathbf{u}) = \mathbf{G}_c \times \mathbf{u}$ in such a way that the difference analog to the differential identity $\nabla \cdot (\nabla \times \mathbf{u}) = 0$ is satisfied. In the limit of $\mu_\tau = +\infty$ this is certainly necessary, because we are constraining $\mathbf{d}_2 = \boldsymbol{\omega} - (\mathbf{G}_c \times \mathbf{u}) = 0$ and $d_1 = \mathbf{D}_v \cdot \boldsymbol{\omega} = 0$, and hence

$$\mathbf{D}_v \cdot (\mathbf{G}_c \times \mathbf{u}) = 0 \ . \tag{19}$$

It seems like a good idea to devise $\mathbf{G}_c$ so that Eq. (19) is generally satisfied.

To define $\mathbf{G}_c$ we first need to define cell-face circulations C_α. As are area projection vectors, C_α is defined relative to both a cell face and a cell on either side of that face. Consider the cell face shown in Fig. 2, with area projection vector $\mathbf{A}_\alpha$ pointing outward from the regular cell relative to which C_α is defined. We number the vertices bounding face α in ascending order counterclockwise when the face is viewed from the side of its area projection vector. Then we have

$$\begin{aligned} C_\alpha = \frac{1}{2}\Big[&(\mathbf{u}_1 + \mathbf{u}_2)\cdot(\mathbf{x}_2 - \mathbf{x}_1) + (\mathbf{u}_2 + \mathbf{u}_3)\cdot(\mathbf{x}_3 - \mathbf{x}_2) + \\ &(\mathbf{u}_3 + \mathbf{u}_4)\cdot(\mathbf{x}_4 - \mathbf{x}_3) + (\mathbf{u}_4 + \mathbf{u}_1)\cdot(\mathbf{x}_4 - \mathbf{x}_1)\Big] \ . \end{aligned} \tag{20}$$

Obviously, for the same cell face, C_α changes sign when defined relative to the cell on the opposite side of that face.

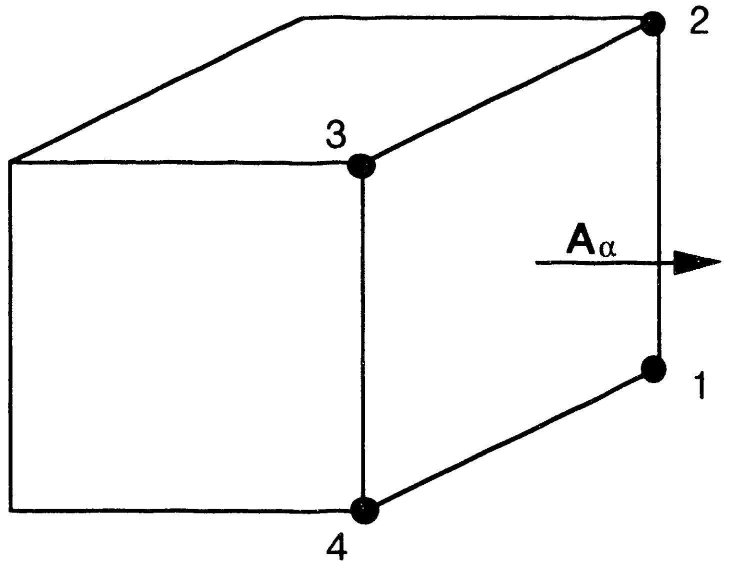


Fig. 2. Vertex numbering used in defining cell-face circulations.

The curl of the velocity in a regular cell is defined by requiring that $\mathbf{G}_c \times \mathbf{u}$ satisfy the following difference approximation to Stokes' Theorem for each vertex v of the regular cell:

$$(\mathbf{G}_c \times \mathbf{u}) \cdot \sum_{\substack{\alpha \\ \text{touching} \\ v}} \mathbf{A}_\alpha = \sum_{\substack{\alpha \\ \text{touching} \\ v}} C_\alpha \,. \tag{21}$$

In Eq. (21) the sum is over the three cell faces of the regular cell that touch vertex v. At first glance it appears that Eq. (21) gives eight constraints for the three components of $\mathbf{G}_c \times \mathbf{u}$, one for each vertex. One can show, however, using the identities

$$\sum_\alpha \mathbf{A}_\alpha = 0 \tag{22}$$

and

$$\sum_\alpha C_\alpha = 0 \,, \tag{23}$$

where each sum is over *all* faces of the cell in which $\mathbf{G}_c \times \mathbf{u}$ is defined, that Eq. (21) gives only three independent constraints.

We now show that $\mathbf{D}_v \cdot (\mathbf{G}_c \times \mathbf{u}) = 0$. Using the definition of the vertex operator $\mathbf{D}_v$, Eq. (17), we see that

$$V_v \mathbf{D}_v \cdot (\mathbf{G}_c \times \mathbf{u}) = \sum_{\beta} \mathbf{A}'_\beta \cdot (\mathbf{G}_c \times \mathbf{u})_\beta \ . \tag{24}$$

Now the sum over momentum cell faces β in Eq. (24) can be split into two sums: one over the eight regular cells γ touching vertex v and the other over the three momentum cell faces inside each regular cell γ :

$$V_v \mathbf{D}_v \cdot (\mathbf{G}_c \times \mathbf{u}) = \sum_{\gamma} \left(\sum_{\substack{\beta \\ \text{in} \\ \gamma}} \mathbf{A}'_\beta \right) \cdot (\mathbf{G}_c \times \mathbf{u})_\gamma \ . \tag{25}$$

Using the same trick that is used to difference the stress terms in KIVA,[2,3] we now express the sum over momentum cell faces in cell γ in terms of the sum of the regular cell faces of cell γ that touch vertex v:

$$V_v \mathbf{D}_v \cdot (\mathbf{G}_c \times \mathbf{u}) = \sum_{\gamma} \left(-\frac{1}{4} \sum_{\substack{\alpha \\ \text{in } \gamma \\ \text{touching } v}} \mathbf{A}_\alpha \right) \cdot (\mathbf{G}_c \times \mathbf{u})_\gamma \ . \tag{26}$$

Using the definition of $\mathbf{G}_c \times \mathbf{u}$, Eq. (21), we see that

$$V_v \mathbf{D}_v \cdot (\mathbf{G}_c \times \mathbf{u}) = -\frac{1}{4} \sum_{\gamma} \left(\sum_{\substack{\alpha \\ \text{in } \gamma \\ \text{touching } v}} C_\alpha \right) \ . \tag{27}$$

Since each C_α occurs twice in Eq. (27), once for each cell with face α, and these occurrences have opposite signs, the sum in Eq. (27) is zero. Hence we have proven Eq. (19).

We now give the finite difference approximation to the vorticity transport equation. For the present we assume, as we did for the angular momentum conservation logic,[2,3] that most errors in vorticity transport arise in the rezone or convection phase of KIVA. Therefore, for the change in vorticity in the Lagrangian phase, we take

$$\boldsymbol{\omega}^B - \boldsymbol{\omega}^n = (\mathbf{G}_c^B \times \mathbf{u}^B) - (\mathbf{G}_c^n \times \mathbf{u}^n) \ , \tag{28}$$

where superscript n (resp. B) denotes values at the beginning of the computational cycle (resp. end of the Lagrangian phase). In the example calculations we have performed to date this has been a very good approximation.

During the rezone phase of the calculation, the three components of $\boldsymbol{\omega}$ are convected just like any other cell-centered extrinsic quantity. Details of this convection calculation are given in Appendix A.

The advanced-time values of $\boldsymbol{\omega}$ are obtained by solving the following equations for $\boldsymbol{\omega}^{n+1}$ and ϕ^{n+1}:

$$V_c^{n+1}\left(\boldsymbol{\omega}^{n+1} - \tilde{\boldsymbol{\omega}}\right) = \Delta t V_c^{n+1}\left(\mathbf{D}_c^{n+1}\phi^{n+1}\right) \tag{29}$$

and

$$\phi^{n+1} = \frac{\mu_T}{\rho}\mathbf{D}_v^{n+1} \cdot \boldsymbol{\omega}^{n+1} . \tag{30}$$

The quantity $\tilde{\boldsymbol{\omega}}$ is the value of the vorticity that results from the convection calculation, and Δt is the main computational timestep. These equations are solved by eliminating $\boldsymbol{\omega}^{n+1}$ in Eq. (30) using Eq. (29), and solving the resulting equation implicitly for ϕ^{n+1}. The conjugate residual procedure used to solve for ϕ^{n+1} is described in Appendix B.

After the advanced-time value of the vorticity has been obtained, the advanced-time values of the velocity and Turn Vector are obtained by solving

$$M_v^{n+1}\left(\mathbf{u}^{n+1} - \tilde{\mathbf{u}}\right) = \Delta t V_v^{n+1}\left(\mathbf{D}_v^{n+1} \times \boldsymbol{\tau}^{n+1}\right) \tag{31}$$

and

$$\boldsymbol{\tau}^{n+1} = \mu_T\left[\boldsymbol{\omega}^{n+1} - \left(\mathbf{G}_c^{n+1} \times \mathbf{u}^{n+1}\right)\right] \tag{32}$$

for $\mathbf{u}^{n+1}$ and $\boldsymbol{\tau}^{n+1}$. Once again $\tilde{\mathbf{u}}$ represents the value of the velocity after the convection calculation. These equations are solved by eliminating $\mathbf{u}^{n+1}$ from Eq. (32) using Eq. (31), and then solving the resulting equation implicitly for $\boldsymbol{\tau}^{n+1}$. Appendix B describes the implicit solution for $\boldsymbol{\tau}^{n+1}$.

Note that we are applying the Turn Vector and ϕ corrections to the velocity and vorticity fields once each computational cycle. An alternative procedure that we have not tried is to apply the corrections after each convective subcycle. This would be more accurate, but much more costly in calculations with many convective subcycles. Another possibility that we have not explored and that would save computational time is to apply the corrections only once every n computational cycles, where $n > 1$.

We can now compare TFM with a method recently published by Dukowicz and Meltz[10] that bears a close resemblance, but is inferior in two respects, to TFM. Dukowicz and Meltz are concerned with reducing numerical errors that arise because of mesh distortion in Lagrangian hydrodynamics calculations. Spurious mesh distortion is attributed to inaccuracies in the calculation of vorticity. To correct these errors the authors propose solving a vorticity transport equation and correcting the computed vertex velocity field so that its curl agrees with the computed vorticity. The close resemblance of this method to TFM can be seen if we compare the large μ_τ formulation of Eqs. (31) and (32) with Dukowicz and Meltz's Eq. (4.12) of Ref. 10. To show the resemblance, we express the difference approximations to the Turn Function equations in the notation of Dukowicz and Meltz. When μ_τ is large Eq. (32) becomes the constraint that the curl of the computed velocity field equals the computed vorticity field:

$$\boldsymbol{\omega}^{n+1} = \mathbf{G}_c^{n+1} \times \mathbf{u}^{n+1} \ . \tag{33}$$

If we define $\Delta\mathbf{v} = \tilde{\mathbf{u}} - \mathbf{u}^{n+1}$, $\Delta\boldsymbol{\omega} = (\mathbf{G}_c^{n+1} \times \tilde{\mathbf{u}}) - \boldsymbol{\omega}^{n+1}$, and $\mathbf{A} = -\Delta t \boldsymbol{\tau}^{n+1}$, Eq. (31) becomes

$$\Delta\mathbf{v} = \frac{V_v^{n+1}}{M_v^{n+1}} \mathbf{D}_v^{n+1} \times \mathbf{A} \ , \tag{34}$$

and Eq. (33) can be combined with Eq. (34) to give

$$\Delta\boldsymbol{\omega} = \mathbf{G}_c^{n+1} \times \left[\frac{V_v^{n+1}}{M_v^{n+1}} \left(\mathbf{D}_v^{n+1} \times \mathbf{A} \right) \right] \ . \tag{35}$$

Neglecting the factor V_v^{n+1}/M_v^{n+1}, the right-hand side of Eq. (35) is a difference approximation to

$$\boldsymbol{\nabla} \times (\boldsymbol{\nabla} \times \mathbf{A}) = -\nabla^2 \mathbf{A} + \boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \mathbf{A}) \ . \tag{36}$$

When μ_τ is constant, Eqs. (32), (19), and (30) show that

$$\mathbf{D}_v \cdot \boldsymbol{\tau}^{n+1} = \rho \phi^{n+1} \tag{37}$$

or, in the notation of Dukowicz and Meltz,

$$\mathbf{D}_v \cdot \mathbf{A} = -\rho \Delta t \phi^{n+1} \ . \tag{38}$$

The boundary condition on $\boldsymbol{\tau}$ can be written as

$$\mathbf{A} = 0 \ . \tag{39}$$

With the exception of the right-hand side of Eq. (38), which is zero in the Dukowicz and Meltz method, and the factor V_v^{n+1}/M_v^{n+1} appearing in Eqs. (34) and (35), Eqs. (34), (35), (38), and (39) are exactly the same as Eq. (4.12) of Ref. 10.

There are two important differences between the method of Dukowicz and Meltz and TFM. One of these differences is related to the appearance of the factor V_v^{n+1}/M_v^{n+1} in Eqs. (34) and (35). This factor arises because TFM corrects the velocity field in a way that conserves linear momentum. In the method of Dukowicz and Meltz, velocities are fundamentally located at cell centers, and vertex velocities are interpolated from these. The method does not attempt to correct the cell-centered velocity field to bring about agreement with the computed vorticity field. Only the vertex velocities are corrected, and conservation of momentum is not a concern in performing this correction because momentum is carried by the cell-centered velocities. In the absence of a correction to the cell-centered velocity field, however, its curl can drift away from the computed vorticity field. Therefore the vertex velocity field can differ considerably from the cell-centered velocity field. This undesirable state of affairs could be remedied by using TFM; the vorticity correction would then appear in both the cell-centered and vertex velocities.

The second difference between the Dukowicz and Meltz method and TFM arises in problems with shocks. We are not concerned with the calculation of shocks in KIVA-F90, but this was an issue in the original Turn Function paper,[5] and we proposed "turning off" TFM in the vicinity of shocks, and not solving a vorticity transport equation, to avoid calculation of incorrect jump conditions of velocity and vorticity across computed shocks.[5] In contrast, Dukowicz and Meltz propose solving the vorticity transport equation everywhere in the flow field. Because of errors in the computed vorticity jump across shocks, the method of Dukowicz and Meltz can again have vertex velocities that differ considerably from the cell-centered velocities.

IV. EXAMPLE CALCULATIONS

In this section we compare calculations performed both with and without TFM. Three problems were calculated: The first is two-dimensional with an initially uniform vorticity field and a nonuniform mesh, the second is two-dimensional with a uniform mesh and nonuniform vorticity field, and the third is a three-dimensional calculation of the interaction of two vortices. All of the calculations, except one noted below, use an extension of QSOU (Quasi-Second-Order Upwind) differencing,[3] called QSOU2, for the convection terms. This extension, which is documented in Appendix A, has been developed to improve the accuracy of the convection calculation of vorticity, as well as other quantities. We compare the results and run times of the calculations with and without TFM. The two-dimensional problems we also compare with known analytic solutions.

A. Calculations of Solid-Body Rotation on a Nonuniform Mesh

In this problem an incompressible fluid is initially in solid-body rotation about the axis of a cylinder. Thus, the initial vorticity field is uniform with one nonzero component aligned with the axis of the cylinder, the z-coordinate direction. Because there is no wall friction, the exact solution is a steady flow varying only in the radial direction. The calculated solutions exhibit unsteadiness and two-dimensionality, however, because of the computational mesh and numerical truncation errors.

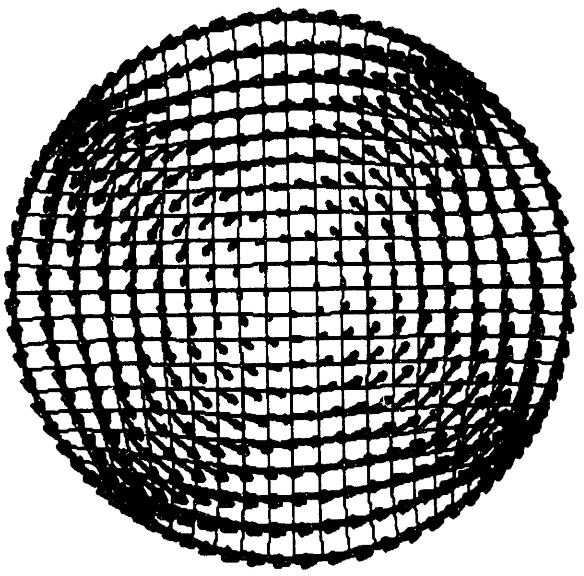
Figure 3 shows the initial velocity field superimposed on the computational mesh in a plane that is perpendicular to the z-axis. The mesh was generated by deforming a square mesh of cells into a circular cross section. This was accomplished by placing neighboring boundary points equidistant from each other and then requiring that interior mesh points lie at the average location of their four nearest neighbors. This type of mesh is now popular in calculations of in-cylinder flows in internal combustion engines.[11] There is no singularity at the axis as in a mesh based on cylindrical coordinates, but this is replaced by four singularities at the four mesh locations of the corners of the square. Three mesh resolutions were used, with 10, 20, and 40 cells in each mesh direction. Although a three-dimensional mesh was used, neither the mesh nor the computed solution varied in the z-direction.

The initial timestep was such that $\omega_z \Delta t = 0.01675$, where ω_z was the initial axial component of vorticity. On subsequent computational cycles, the timestep was allowed to grow by two percent per cycle until KIVA's acceleration timestep limit[3] was reached. Each problem was run for one rotation time based on the initial vorticity value.

Figure 4 shows plots of the integrated z-component of vorticity as a function of time for calculations 1) with TFM, 2) without TFM, and 3) without TFM but with donor-cell or upwind differencing for convection. The last is presented to show the large vorticity conservation errors that are present in codes that use donor-cell differencing. As Fig. 4 shows, the calculation with TFM nearly perfectly conserves vorticity, whereas the calculations without TFM lose 9% and 35% of their initial vorticity. These results support our assumption that most vorticity transport errors in KIVA result from the calculation of convection.

Figure 5 shows the effect of varying mesh resolution on the total vorticity versus time. In the calculation using TFM on the 10^3 mesh, vorticity is nearly perfectly conserved. Calculations without TFM show that vorticity is conserved better as mesh resolution is improved, but the 40^3 mesh still loses 5% of its initial vorticity.

The major reason for the loss of vorticity in the calculations performed without TFM can be seen in Fig. 6, which gives a contour plot of the z-component of vorticity in the



```
20**3 cylinder mesh: swirl
velocity vector plot
time:  0.000000E+00
cycle #     0
on plane k =    11
maximum u:  7.037168E+02
maximum v:  7.037168E+02
```

Fig. 3. Velocity vectors and computational mesh for solid-body rotation problems.

20^3 mesh at the end of the calculation. The initial vorticity value is maintained to within 5% in a large, centrally-located region, but the vorticity has dropped to 16% of its initial value at the locations of the corners of the logical mesh. Apparently, this is due to large numerical errors associated with the mesh distortion at these corners. Correcting these errors by using a higher-order method rather than TFM would be extremely difficult on this mesh and on the distorted meshes that must be used to model complex geometries.

The computational times with and without TFM are given in Table 1. Using TFM increases times by 60% to 70%.

TABLE 1. Computational times (s) for solid-body rotation problem.

Mesh size	with TFM	without TFM
10^3	38	23
20^3	108	63
40^3	696	434

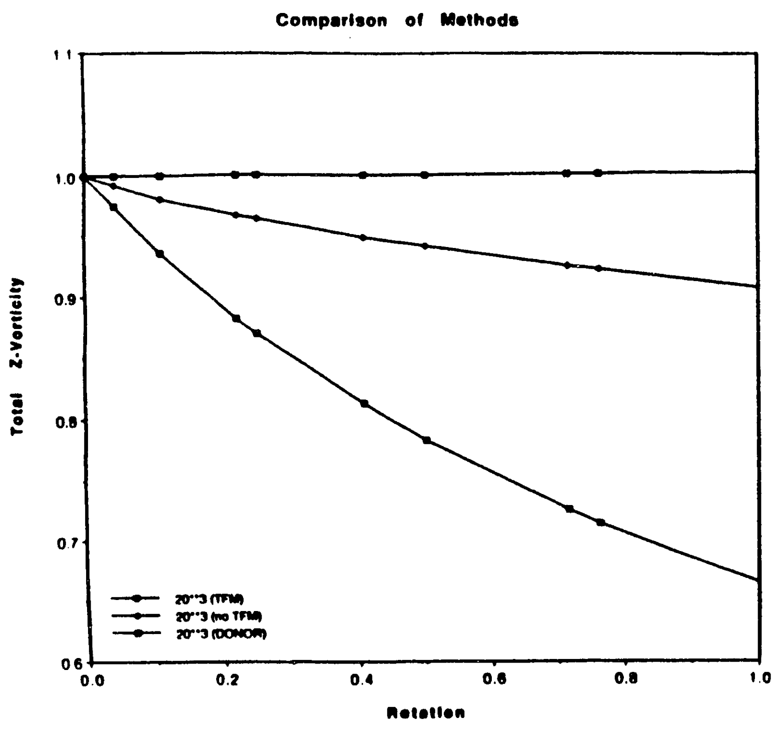


Fig. 4. Total z-vorticity versus time with TFM, without TFM, and with donor-cell differencing.

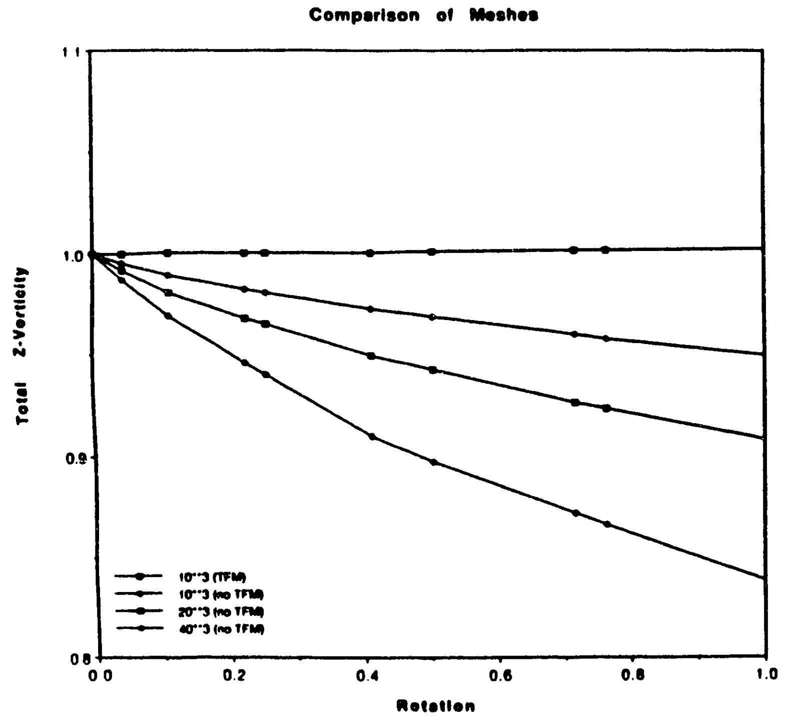
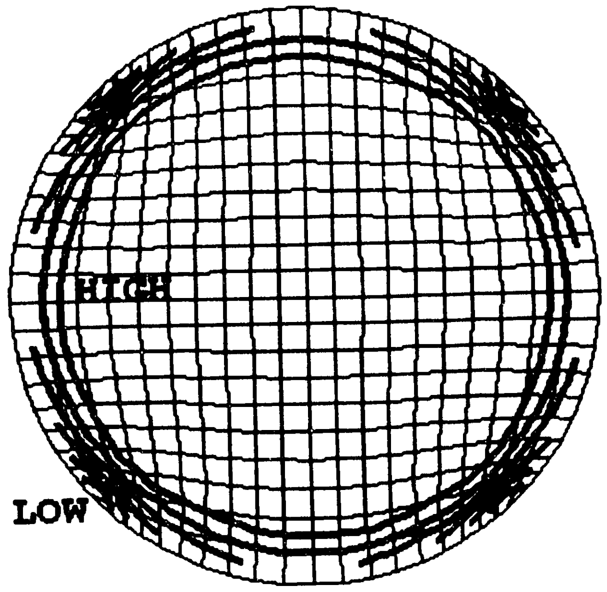


Fig. 5. Total z-vorticity versus time for TFM on a coarse mesh and without TFM using three different mesh resolutions.



```
21x21x21 cylinder: swirl, no TFM
   omega-z contour plot
   time:   3.829099E-02
   cycle #    141
   on plane k =      11
   maximum value:   1.049369E+00
   minimum value:   1.651571E-01
```

Fig. 6. Contour plot of z-vorticity without TFM, showing locations of largest errors.

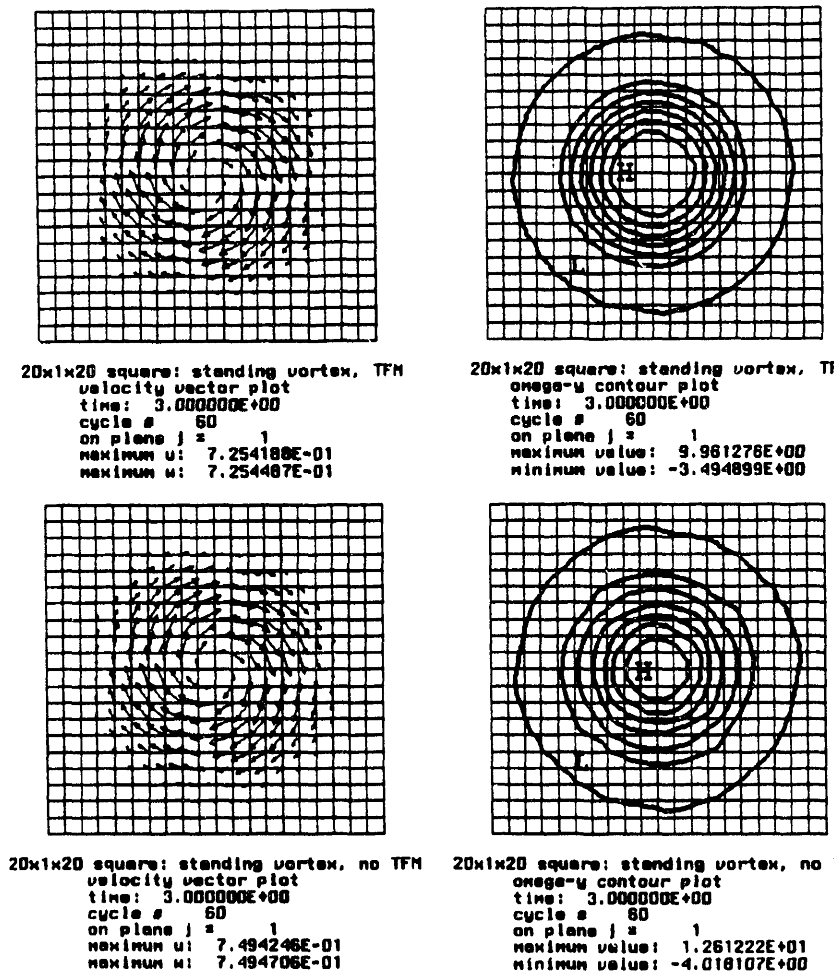
B. Calculations of a Standing Vortex on a Uniform Mesh

This problem was originally suggested by Gresho[12] and has recently been used as a test calculation for a finite element method by Tezuyar et al.[13] The initial condition is an axisymmetric, incompressible vortex contained in a 1×1 box. The circumferential velocity is given by

$$U_\theta = \begin{cases} 5r & 0 < r < .2 \\ 2 - 5r & .2 < r < .4 \\ 0 & .4 < r \end{cases} .$$

As in the first test problem, the initial condition is the steady-state solution, but the calculations exhibit unsteadiness because of numerical errors. The box is resolved with 20 uniform cells in each mesh direction and the computational timestep is equal to 0.05, resulting in a maximum Courant number of 1.0. The problem is run to a time of 3.0, which is about 4.8 rotation times of the initial vortex core. Following Tezuyar et al.,[13] we use the time history of the total kinetic energy in the mesh as a measure of the accuracy of a numerical method, although the L^2-norm of the difference between the exact and computed solutions would be a better measure.

Surprisingly, the calculation without TFM preserved total kinetic energy better than the calculation with TFM. In the former, 83% of the initial kinetic energy was still present at time 3.0, while only 73% was present with TFM. Tezuyar et al. reported preserving between 80% and 91% of the total kinetic energy for the finite element methods they tested. Figure 7 gives plots of the velocity and vorticity fields of the KIVA-F90 calculations at time 3.0. The calculation without TFM preserved slightly better the maximum value 1.0 of the velocity; but the calculation with TFM preserved the maximum vorticity value much better, keeping within 1% of the initial maximum value of 10.0. Both calculations conserved total vorticity and angular momentum within 0.5%.



```
20x1x20 square: standing vortex, TFM
      velocity vector plot
      time:  3.000000E+00
      cycle #    60
      on plane j =     1
      maximum u:  7.254188E-01
      maximum w:  7.254487E-01
```

```
20x1x20 square: standing vortex, TFM
      omega-y contour plot
      time:  3.000000E+00
      cycle #    60
      on plane j =     1
      maximum value:  9.961276E+00
      minimum value: -3.494899E+00
```

```
20x1x20 square: standing vortex, no TFM
      velocity vector plot
      time:  3.000000E+00
      cycle #    60
      on plane j =     1
      maximum u:  7.494246E-01
      maximum w:  7.494706E-01
```

```
20x1x20 square: standing vortex, no TFM
      omega-y contour plot
      time:  3.000000E+00
      cycle #    60
      on plane j =     1
      maximum value:  1.261222E+01
      minimum value: -4.018107E+00
```

Fig. 7. Velocity vectors and contours of z-vorticity for standing vortex problems, with and without TFM.

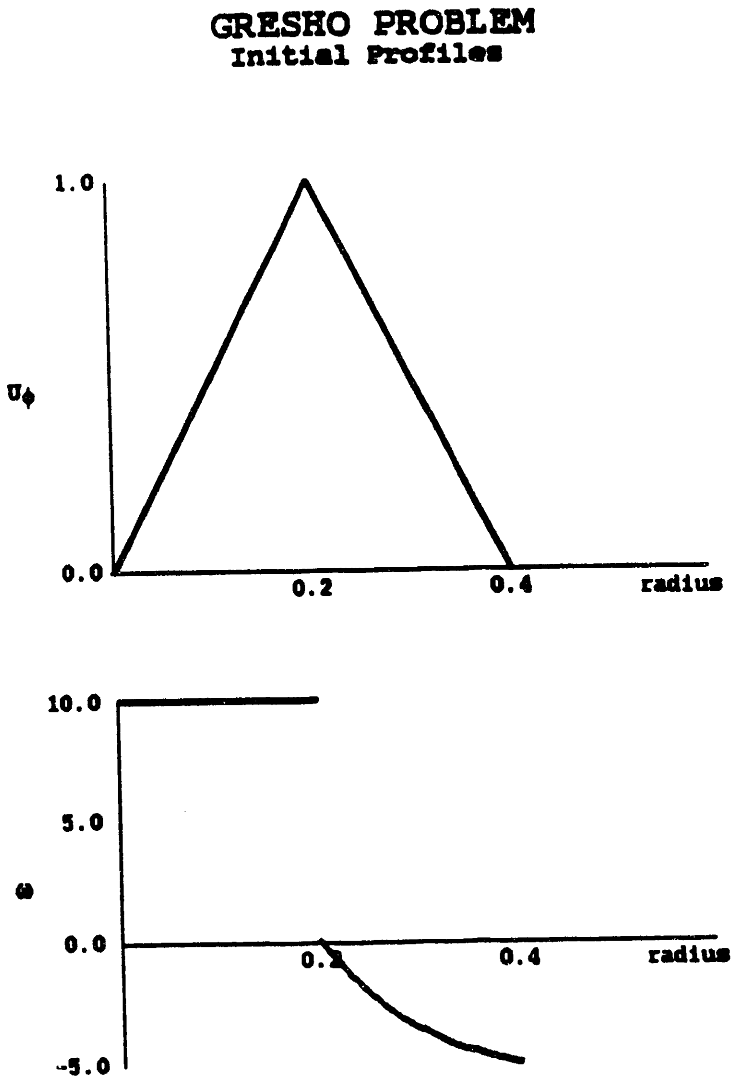


Fig. 8. Initial velocity (top) and vorticity (bottom) profiles in the Gresho problem.

The reason TFM is more dissipative in this problem is clear when we consider the initial velocity and vorticity profiles in Fig. 8. The velocity field is continuous, increasing linearly to one in the core of the vortex and decreasing linearly to zero outside of the core. In contrast, the vorticity field has two discontinuities, one at the outer edge of the core and another at $r = 0.4$, where the velocity again becomes zero. In the calculation without TFM, the continuous velocity field is numerically convected through the mesh. In the calculation with TFM, the vorticity field is convected through the mesh, and a velocity field that is consistent with this computed vorticity field is found. The reason the TFM calculation dissipates more kinetic energy is that it is more difficult for the QSOU2 convection scheme to convect accurately the discontinuous vorticity field than it is to convect the continuous velocity field. This is because the QSOU2 convection scheme reverts to upwind differencing in regions where the derivative of a convected quantity is discontinuous (see Appendix A).

This supplies numerical smoothing to the variation of that quantity so that its computed profile can be resolved by the computational mesh. The more discontinuous a quantity is, the more smoothing is supplied.

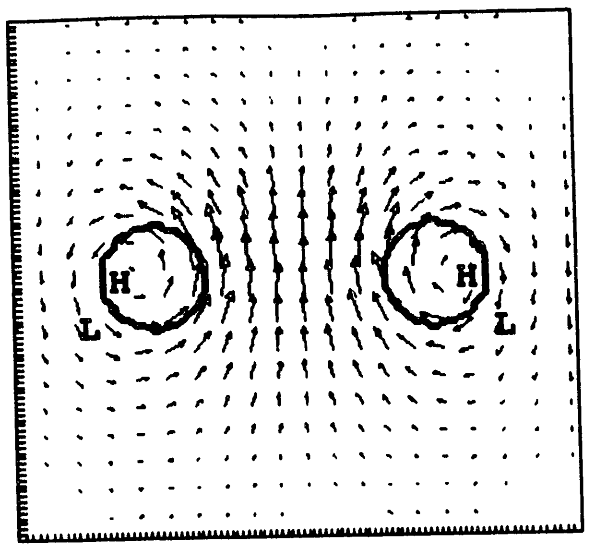
Calculational results that used the original QSOU method of KIVA-II[3] were worse, showing more numerical dissipation of kinetic energy and more diffusion of vorticity. This is what motivated us to develop the improved QSOU2 convection algorithm discussed in Appendix A.

One possible explanation for the lackluster performance of KIVA-F90 relative to that of the finite element methods is the KIVA-F90 gradient-limiter (see Appendix A), which introduces numerical dissipation in order to maintain monotonicity and positivity. The finite element methods do not use a gradient-limiter. To test this hypothesis we performed two calculations with and without TFM, both using the convection scheme of Appendix A without the gradient limiter. These calculations preserved 86% (no TFM) and 81% (TFM) of the initial kinetic energy. Despite the increased kinetic energy error caused by the gradient-limiter in this problem, maintaining monotonicity is an over-riding consideration for us; and therefore use of the gradient-limiter is retained in KIVA-F90.

C. Calculations of a Rotating Ring Vortex

The purpose of these calculations was to determine whether rotation affects the evolution of a vortex ring placed in a cylindrical vessel. The initial conditions are shown in Fig. 9. The azimuthal component of the vorticity is specified to be a constant value inside a doughnut-shaped region whose axis coincides with the axis of the vessel. Outside this doughnut-shaped region, the azimuthal component of the vorticity is zero. The radial component of vorticity is identically zero. The resulting velocity field in the r-z plane is also shown in Fig. 9. Note that the direction of rotation in the ring vortex is such that the flow is upward along the axis of the vessel. Two calculations were performed. In the first, there was no axial component of vorticity. In the second, a solid-body rotation was superimposed on the initial velocity field of Fig. 9. The Rossby number, the ratio of the ring-vortex velocity at the edge of the doughnut to the swirl velocity at the center of the vortex ring, was one. Both calculations were performed with and without TFM.

All calculations were performed with 95 cells in each of the mesh coordinate directions, resulting in a total of nearly a million computational cells. As in the first set of example calculations, the mesh was formed by deforming a logically-rectangular block of cells into a cylindrical shape and was not derived from a cylindrical coordinate transformation. Thus, although the initial and boundary conditions for this problem are axisymmetric, the numerical solution could develop asymmetries about the cylinder axis because of numerical



Velocity Vectors and Omega-θ Vorticity Contour

```
time: 0.000000E+00
cycle #    0
on plane j=   48
maximum u: 6.400065E+01
maximum w: 1.109866E+02
```

Fig. 9. Initial conditions for ring vortex problems, superimposed velocity vectors and omega-theta vorticity contours.

errors or possible hydrodynamic instabilities. Another purpose of the calculations was to see if the solutions developed asymmetries about the axis.

The calculations using TFM remained axisymmetric. Shown in Fig. 10 are contours of the azimuthal component of vorticity in the TFM calculation without rotation. The time between each plot is approximately one-half a rotation time, based on the initial rotation rate in the vortex ring. In this solution, the ring drifts upward in the direction of the fluid flow along the cylinder axis, in agreement with the solution for the direction of motion of an isolated vortex ring.[14] As it approaches the top of the cylinder, the ring spreads out radially but retains its doughnut shape. In contrast, in the TFM calculation with rotation the ring spawns counter-rotating vortex rings, as shown in Fig. 11. An explanation for these counter-rotating rings, given by Von Karman,[15] is that they result from the interaction of pressure and angular momentum gradients.

The calculations without TFM quickly developed large velocity shears and asymmetries about the cylinder axis. Because of this, the computational timesteps in the calculations without TFM were smaller, and the computational times of approximately 6.3 hours were actually longer than those in the calculations with TFM, which were approximately 6.0 hours.

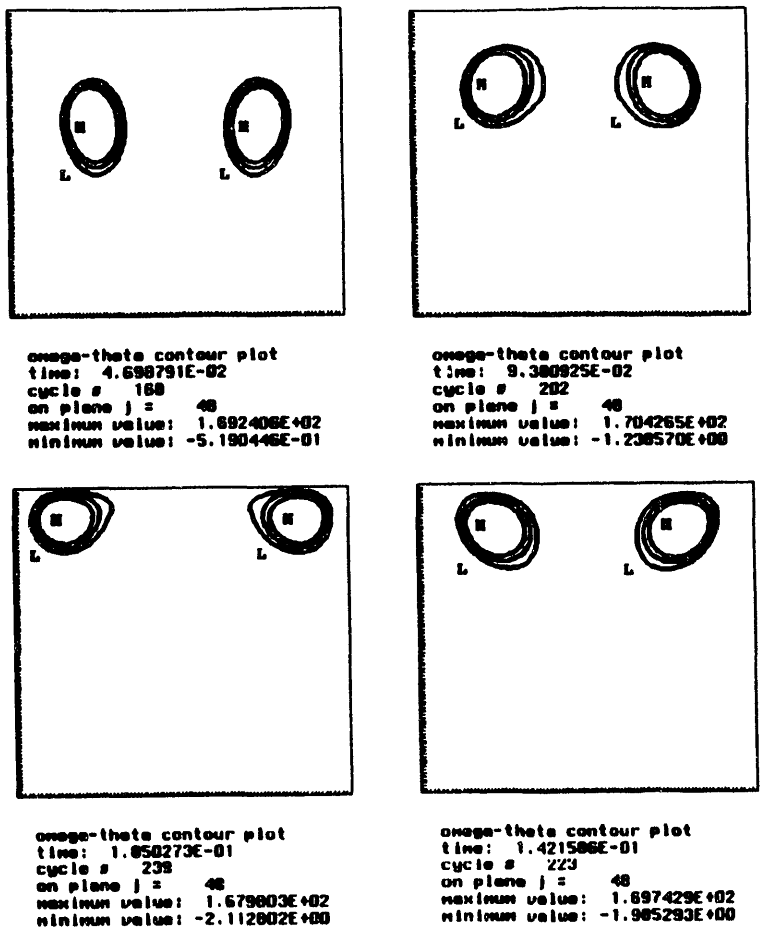
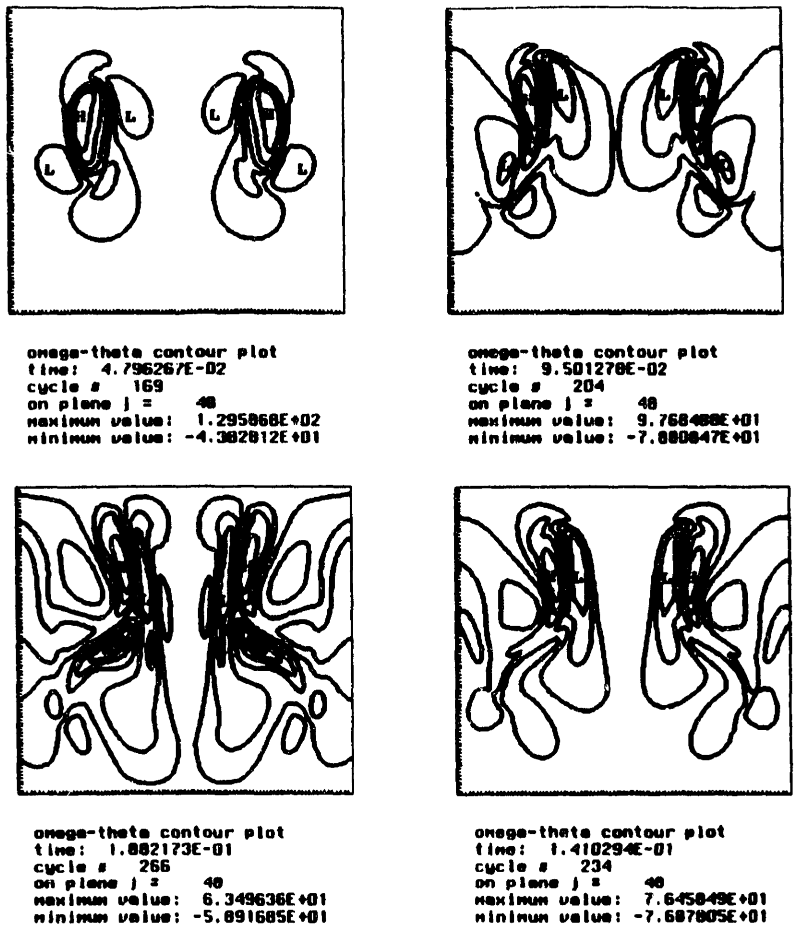


Fig. 10. Contour plots of omega-theta vorticity in nonrotating ring vortex, showing evolution through time.

V. CONCLUSIONS

We have implemented the Turn Function Method (TFM) for conserving both linear momentum and vorticity in the compressible-fluid hydrodynamics code KIVA-F90. This first implementation in a three-dimensional hydrodynamics code involved some modification of the original method and some novel difference approximations. In addition, we have derived and used a better value for the discrepancy diffusivity than was reported in the original method. The discrepancy diffusivity controls the difference between the computed vorticity and the curl of the computed velocity.

Test problems show that TFM corrects vorticity conservation errors that can occur when using highly distorted meshes in KIVA-F90 calculations. TFM can also degrade the accuracy of calculations with large vorticity gradients by introducing numerical diffusion of vorticity. Generally, TFM has a smoothing effect on computed velocity fields.



```
omega-theta contour plot
time:  4.796267E-02
cycle #   169
on plane j =     48
maximum value:  1.295868E+02
minimum value: -4.382812E+01
```

```
omega-theta contour plot
time:  9.501278E-02
cycle #   204
on plane j =     48
maximum value:  9.768408E+01
minimum value: -7.880847E+01
```

```
omega-theta contour plot
time:  1.882173E-01
cycle #   266
on plane j =     48
maximum value:  6.349636E+01
minimum value: -5.891685E+01
```

```
omega-theta contour plot
time:  1.410294E-01
cycle #   234
on plane j =     48
maximum value:  7.645849E+01
minimum value: -7.687805E+01
```

Fig. 11. Contour plots of omega-theta vorticity in rotating ring vortex, showing evolution through time.

Use of TFM can increase by 60% the computational times of the Euler equation solver in KIVA-F90, although run times have decreased in some calculations. It is expected that when additional physics submodels are installed in KIVA-F90, the fraction of time spent solving the TFM equations will be reduced. We will continue to evaluate the costs and benefits of using TFM, but we are highly encouraged by the accuracy improvements of the initial calculations reported here.

ACKNOWLEDGMENTS

The authors wish to thank T. D. Butler, J. K. Dukowicz, and M. S. Sahota for many helpful discussions.

APPENDIX A
QSOU2 CONVECTION ALGORITHM

In this appendix we document the QSOU2 convection algorithm, which replaces the QSOU scheme of KIVA-II. QSOU2 improves upon QSOU in two respects. First, to find the values of fluxed quantities on cell faces, interpolation is performed along material paths instead of along grid coordinate directions. Second, a less diffusive gradient-limiter is used. Before describing QSOU2, we briefly review the QSOU convection algorithm. For a more detailed description of QSOU, the reader is referred to Appendix M of the KIVA-II report.[3]

In KIVA's convection phase, the flux of a cell-centered quantity across a cell boundary is the product of a volume flux across that boundary and an interpolated density ρ, where ρ represents a mass or energy density per unit volume or a component of the vorticity. Momentum fluxes, which will be described later, are the products of interpolated velocities and the mass fluxes across each momentum cell face. QSOU is a method for calculating the interpolated values in such a way that the convection scheme is strongly monotone.[3] The interpolated density for fluxing a cell-centered quantity is obtained in two steps. First, we calculate the derivative of ρ with respect to distance in the logical coordinate direction in which fluxing is occurring. For example, if fluxing is in the i-coordinate direction, we calculate $\frac{\partial\rho}{\partial s}\big|_i$, the derivative of ρ with respect to distance in the i-direction, according to

$$\left.\frac{\partial\rho}{\partial s}\right|_i = \frac{\operatorname{sign}\left(\frac{\partial\rho}{\partial s_R}\right) + \operatorname{sign}\left(\frac{\partial\rho}{\partial s_L}\right)}{2} \min\left(\left|\frac{\partial\rho}{\partial s_R}\right| , \left|\frac{\partial\rho}{\partial s_L}\right|\right) . \qquad \text{(A-1)}$$

The quantities $\frac{\partial\rho}{\partial s_R}$ and $\frac{\partial\rho}{\partial s_L}$ are right- and left-face gradients that are obtained as follows:

$$\frac{\partial\rho}{\partial s_R} = \frac{\rho_R - \rho_c}{|\mathbf{x}_R - \mathbf{x}_c|}$$

and (A-2)

$$\frac{\partial\rho}{\partial s_L} = \frac{\rho_c - \rho_L}{|\mathbf{x}_c - \mathbf{x}_L|} .$$

In these formulas, ρ_c =value of ρ in cell (i,j,k),

ρ_R = value of ρ in cell $(i+1,j,k)$,

ρ_L = value of ρ in cell $(i-1,j,k)$,

$\mathbf{x}_c$ = location of center of cell (i,j,k),

$\mathbf{x}_R$ = location of center of cell $(i+1,j,k)$, and

$\mathbf{x}_L$ = location of center of cell $(i-1,j,k)$.

Thus QSOU selects the gradient of minimum magnitude from the two gradients on either side of cell (i,j,k), if these gradients have the same sign. Otherwise $\frac{\partial\rho}{\partial s}\big|_i$ is set to zero.

In the second step of the QSOU calculation, the boundary density is obtained by interpolation in the cell that is donating material. For example, for the left face of cell (i, j, k),

$$\rho_L^f = \begin{cases} \rho_L + \frac{\partial \rho}{\partial s}\Big|_{i-1} \left|\mathbf{x}_L^f - \mathbf{x}_L\right| \left(1 - \frac{FXL}{V_L}\right) & \text{if } FXL > 0 \\ \rho_c - \frac{\partial \rho}{\partial s}\Big|_{i} \left|\mathbf{x}_L^f - \mathbf{x}_c\right| \left(1 - \frac{FXL}{V_c}\right) & \text{if } FXL < 0\,. \end{cases} \tag{A-3}$$

ρ_L^f and $\mathbf{x}_L^f$ are the interpolated density and location of the center of the left face of cell (i, j, k). V_L and V_c are the volumes of cells $(i-1, j, k)$ and (i, j, k). FXL is the flux volume through the left face and is positive if cell $(i-1, j, k)$ is donating material. FXL is calculated as it is in KIVA-II.[3] From Eqs. (A-1), (A-2), and (A-3), it is seen that the interpolated density depends only on gradients in the grid coordinate direction in which fluxing is being calculated and is independent of gradients in other coordinate directions. In regions where the solution is smooth, QSOU approaches second-order accuracy in space, but because the interpolation of Eq. (A-3) is only in the fluxing coordinate direction, the QSOU scheme is first-order accurate in time.[15]

In the QSOU2 scheme the interpolated density depends, in general, on the gradients in all coordinate directions, and the interpolation is in the direction of the relative velocity vector, which is the difference between the fluid and grid velocities. Because of this, QSOU2 is second-order accurate in time. A second difference between QSOU2 and QSOU is that QSOU2 uses VanLeer gradient limiting,[3,14] which is weakly monotone rather than strongly monotone and introduces less numerical diffusion than does QSOU differencing.

The QSOU2 calculation has three steps. First, in each cell we calculate the density gradients in all logical coordinate directions. Similar formulas are used for each direction, so we just give those for the i-coordinate :

$$\frac{\partial \rho}{\partial s}\Big|_i = \frac{\operatorname{sign}\left(\frac{\partial \rho}{\partial s_R}\right) + \operatorname{sign}\left(\frac{\partial \rho}{\partial s_L}\right)}{2} \min\left\{ \frac{1}{2}\left|\frac{\partial \rho}{\partial s_R} + \frac{\partial \rho}{\partial s_L}\right|, \frac{|\rho_R - \rho_c|}{\left|\mathbf{x}_R^f - \mathbf{x}_c\right|}, \frac{|\rho_L - \rho_c|}{\left|\mathbf{x}_L^f - \mathbf{x}_c\right|} \right\}. \tag{A-4}$$

The right- and left-face gradients of Eq. (A-4) differ slightly from those of the QSOU scheme:

$$\frac{\partial \rho}{\partial s_R} = \frac{\rho_R - \rho_c}{\left|\mathbf{x}_R - \mathbf{x}_R^f\right| + \left|\mathbf{x}_R^f - \mathbf{x}_c\right|}$$

and (A-5)

$$\frac{\partial \rho}{\partial s_L} = \frac{\rho_c - \rho_L}{\left|\mathbf{x}_c - \mathbf{x}_L^f\right| + \left|\mathbf{x}_L^f - \mathbf{x}_L\right|} ,$$

where $\mathbf{x}_R^f$ equals the location of the center of the right face of cell (i, j, k), and $\mathbf{x}_L^f$ equals the location of the center of the left face of cell (i, j, k). From (A-1) and (A-4), it can be verified that, unlike with QSOU, the quantity $\frac{\partial \rho}{\partial s}\big|_i$ can exceed either the right- or the left-face gradient in magnitude, but, in compliance with VanLeer limiting, the right-face density $\rho_c + \frac{\partial \rho}{\partial s}\big|_i \left|\mathbf{x}_R^f - \mathbf{x}_c\right|$ lies between ρ_c and ρ_R and the left-face density $\rho_c - \frac{\partial \rho}{\partial s}\big|_i \left|\mathbf{x}_L^f - \mathbf{x}_c\right|$ lies between ρ_c and ρ_L.

In the second step of the QSOU2 convection calculation, the gradient of ρ is found by solving the three simultaneous equations:

$$\begin{aligned}
\nabla \rho \cdot \left(\mathbf{x}_R^f - \mathbf{x}_L^f\right) &= \frac{\partial \rho}{\partial s}\bigg|_i \left|\mathbf{x}_R^f - \mathbf{x}_L^f\right| \\
\nabla \rho \cdot \left(\mathbf{x}_D^f - \mathbf{x}_F^f\right) &= \frac{\partial \rho}{\partial s}\bigg|_j \left|\mathbf{x}_D^f - \mathbf{x}_F^f\right| \qquad \text{(A-6)} \\
\nabla \rho \cdot \left(\mathbf{x}_T^f - \mathbf{x}_B^f\right) &= \frac{\partial \rho}{\partial s}\bigg|_k \left|\mathbf{x}_T^f - \mathbf{x}_B^f\right| .
\end{aligned}$$

The subscripts F, D, B, and T denote the front, back, bottom, and top faces of cell (i, j, k).

The interpolated densities are obtained in the third step of the QSOU2 calculation by using the density gradients of Eq. (A-6), the relative velocities between the fluid and grid, and the signs of the flux volumes. The relative velocities, which are defined at cell vertices, are given by

$$\mathbf{u}_r = \mathbf{u}^B - \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} , \qquad \text{(A-7)}$$

where $\mathbf{u}^B$ is the fluid velocity after Phase B of the computational cycle, $\mathbf{x}^n$ and $\mathbf{x}^{n+1}$ are the vertex locations at the beginning and end of the computational cycle, and Δt is the main computational timestep. The interpolated density on the left face of cell (i,j,k) is calculated according to

$$\rho_L^f = \begin{cases} \rho_L + (\nabla\rho)_L \cdot \left(\mathbf{x}_L^f - \mathbf{x}_L - \frac{\bar{\mathbf{u}}_r \delta t}{2}\right) & \text{if } FXL > 0 \\ \rho_c + (\nabla\rho)_c \cdot \left(\mathbf{x}_L^f - \mathbf{x}_c - \frac{\bar{\mathbf{u}}_r \delta t}{2}\right) & \text{if } FXL < 0 \,, \end{cases} \tag{A-8}$$

where δt is the convection timestep and $\bar{\mathbf{u}}_r$ is the average value of the relative velocity at the four vertices bounding the left face. The density gradients $(\nabla\rho)_L$ and $(\nabla\rho)_c$ are located at the centers of cells $(i-1,j,k)$ and (i,j,k), respectively.

QSOU2 momentum fluxing is a three-step calculation that closely parallels the new fluxing for cell-centered quantities. Letting q denote any of the three velocity components u, v, or w, in the first step, we calculate the derivatives of q with respect to distance in each of the logical coordinate directions. For the i-direction these are given by

$$\left.\frac{\partial q}{\partial s}\right|_i = \frac{\text{sign}\left(\frac{\partial q}{\partial s_R}\right) + \text{sign}\left(\frac{\partial q}{\partial s_L}\right)}{2} \min\left\{\frac{1}{2}\left|\frac{\partial q}{\partial s_R} + \frac{\partial q}{\partial s_L}\right| , 2\left|\frac{\partial q}{\partial s_R}\right| , 2\left|\frac{\partial q}{\partial s_L}\right|\right\} , \tag{A-9}$$

where $\frac{\partial q}{\partial s_L}$ and $\frac{\partial q}{\partial s_R}$ are left and right gradients given by

$$\frac{\partial q}{\partial s_R} = \frac{q_{i+1,j,k} - q_{ijk}}{|\mathbf{x}_{i+1,j,k} - \mathbf{x}_{ijk}|}$$

and (A-10)

$$\frac{\partial q}{\partial s_L} = \frac{q_{ijk} - q_{i-1,j,k}}{|\mathbf{x}_{ijk} - \mathbf{x}_{i-1,j,k}|} \,.$$

In the second step, we calculate the gradient of q at each vertex by solving the three simultaneous equations

$$(\nabla q)_{ijk} \cdot (\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i-1,j,k}) = \left.\frac{\partial q}{\partial s}\right|_i |\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i-1,j,k}| \,,$$

$$(\nabla q)_{ijk} \cdot (\mathbf{x}_{i,j+1,k} - \mathbf{x}_{i,j-1,k}) = \left.\frac{\partial q}{\partial s}\right|_j |\mathbf{x}_{i,j+1,k} - \mathbf{x}_{i,j-1,k}| \,, \tag{A-11}$$

and

$$(\nabla q)_{ijk} \cdot (\mathbf{x}_{i,j,k+1} - \mathbf{x}_{i,j,k-1}) = \left.\frac{\partial q}{\partial s}\right|_k |\mathbf{x}_{i,j,k+1} - \mathbf{x}_{i,j,k-1}| \,.$$

The interpolated velocities for each momentum cell face are obtained in the third step. As an example, the interpolated velocity on the face between momentum cells (i,j,k) and $(i-1,j,k)$ is calculated from

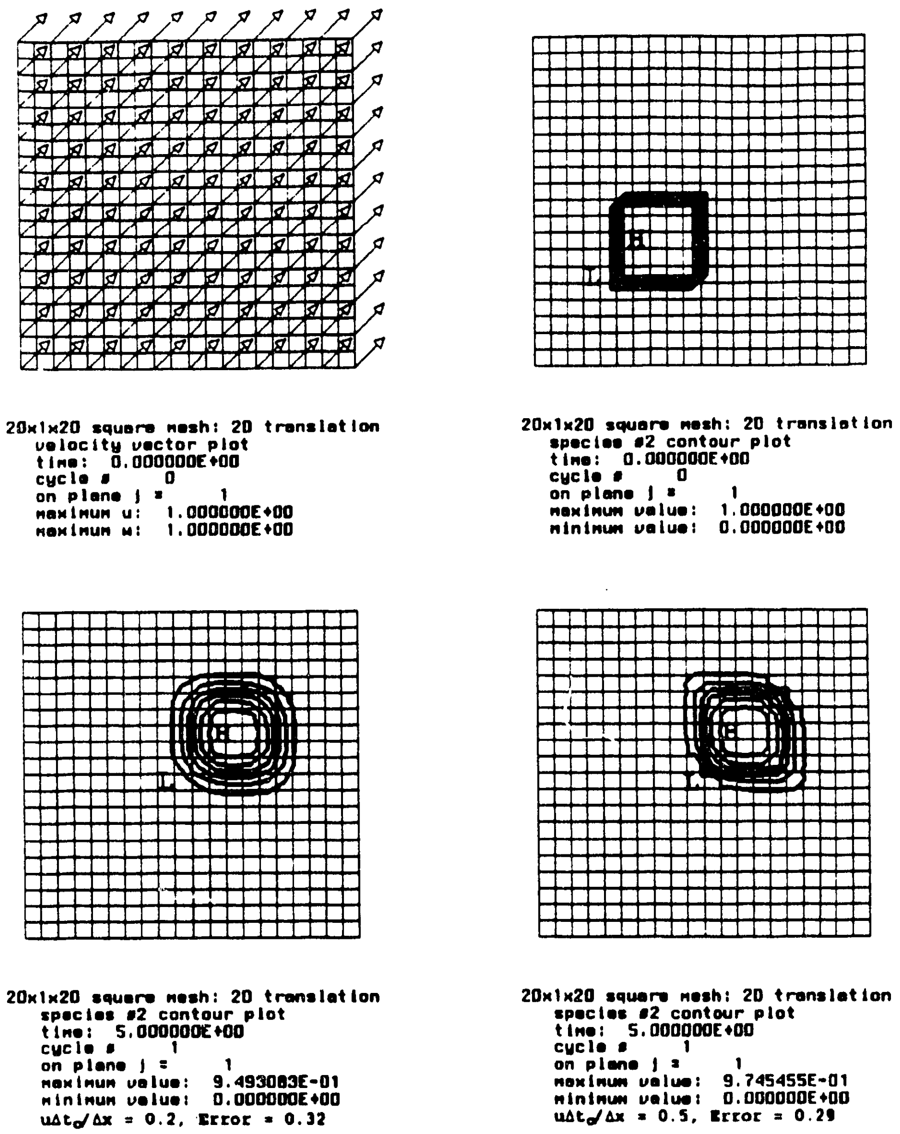
$$q^f = \begin{cases} q_{i-1,j,k} + (\nabla q)_{i-1,j,k} \cdot \left(\frac{\mathbf{x}_{ijk} - \mathbf{x}_{i-1,j,k} - \bar{\mathbf{u}}_r \delta t}{2}\right) & \text{if } FXM > 0 \\ q_{ijk} + (\nabla q)_{ijk} \cdot \left(\frac{\mathbf{x}_{i-1,j,k} - \mathbf{x}_{ijk} - \bar{\mathbf{u}}_r \delta t}{2}\right) & \text{if } FXM < 0 \,, \end{cases} \tag{A-12}$$

where FXM, the mass flux across this cell face, is positive if cell $(i-1, j, k)$ is donating mass to cell (i, j, k). FXM is obtained in the cell-centered fluxing portion of the calculation, as is described in the KIVA-II documentation.[3] In Eq. (A-12), the quantity $\bar{\mathbf{u}}_r$ is the average value of the relative velocities at vertices (i, j, k) and $(i-1, j, k)$.

To illustrate the increased accuracy of the QSOU2 convection calculation, we repeated the example calculation of the KIVA-II report[3] in which a scalar field is convected with a uniform velocity directed at a 45° angle to the mesh directions of a two-dimensional mesh of square cells. The scalar field is initially unity on a square that has five cells to a side and is zero otherwise. This initial field, along with the results of two calculations using QSOU2 with $u\,\delta t/\delta x = 0.2$ and $u\,\delta t/\delta x = 0.5$, is shown in the contour plots of Fig. A-1. The calculations were run to a time T such that $u\,T/\delta x = 5.0$, at which time the exact solution was a uniform translation of the square five cells in each direction. Also given in Fig. A-1 are the maximum and minimum computed values of the scalar at time T and the root-mean-square error between the computed and exact solutions, averaged over the twenty-five cells for which the scalar field is unity in the exact solution. Because of poor resolution, this example problem is a severe test of convection schemes and exaggerates many of their shortcomings.

The square shape of the initial profile is more nearly preserved in the calculation with $u\,\delta t/\delta x = 0.2$, but the maximum scalar value is closer to the initial value, and the error is smaller, in the calculation with $u\,\delta t/\delta x = 0.5$. In both calculations, the computed minima are 0.0, evidence of the monotonicity of the QSOU2 scheme. As reported in the KIVA-II documentation,[3] when we use QSOU with $u\,\delta t/\delta x = 0.2$ in the same calculation, the computed maximum scalar value and error are 0.87 and 0.36, respectively. Both of these values are worse than those obtained in the QSOU2 calculations.

Because of the increased temporal accuracy of the QSOU2 scheme, we are able to use larger convective subcycle timesteps in Phase C of KIVA-F90. Currently, the convective timestep in KIVA-F90 is limited by the Courant condition $u\,\delta t/\delta x \leq 0.5$, whereas in KIVA-II this timestep is limited by $u\,\delta t/\delta x \leq 0.2$. Thus the number of convective subcycles in KIVA-F90 calculations will be reduced by approximately 40%, which saves computational time in calculations with large numbers of convective subcycles.



```
20x1x20 square mesh: 2D translation
   velocity vector plot
   time:   0.000000E+00
   cycle #      0
   on plane j =       1
   maximum u:   1.000000E+00
   maximum w:   1.000000E+00
```

```
20x1x20 square mesh: 2D translation
   species #2 contour plot
   time:   0.000000E+00
   cycle #      0
   on plane j =       1
   maximum value:   1.000000E+00
   minimum value:   0.000000E+00
```

```
20x1x20 square mesh: 2D translation
   species #2 contour plot
   time:   5.000000E+00
   cycle #      1
   on plane j =       1
   maximum value:   9.493083E-01
   minimum value:   0.000000E+00
   uΔto/Δx = 0.2, Error = 0.32
```

```
20x1x20 square mesh: 2D translation
   species #2 contour plot
   time:   5.000000E+00
   cycle #      1
   on plane j =       1
   maximum value:   9.745455E-01
   minimum value:   0.000000E+00
   uΔto/Δx = 0.5, Error = 0.29
```

Fig. A-1. Velocity field (upper left) and initial scalar profile (upper right) used in convection problem with $u\,\delta t/\delta x = 0.2$ (lower left) and $u\,\delta t/\delta x = 0.5$ (lower right).

APPENDIX B
THE CONJUGATE RESIDUAL METHOD

In this appendix we give some details of how the conjugate residual (CR) method is used to solve Eqs. (29) through (32) of this report. For details of the CR method, the reader should consult Ref. 8, which provides a proof of convergence. Here we give just an overview of the method and derive the preconditioning matrices that are used in conjunction with solution of the TFM equations.

It is convenient to think of the CR method as a special type of Gram-Schmidt orthogonalization procedure.[16] A sequence of basis vectors is constructed by making use of the sequence of residuals $\mathbf{r}_v = \mathbf{y} - \mathbf{A}\mathbf{x}_v$ in the solution of linear equation $\mathbf{y} = \mathbf{A}\mathbf{x}$. The basis vectors are $\mathbf{q}_v = \mathbf{P}\mathbf{r}_v$, where $\mathbf{P}$ is an approximation to $\mathbf{A}^{-1}$ and is called the preconditioning matrix. As in the Gram-Schmidt procedure, an orthonormal basis is constructed by subtracting from $\mathbf{q}_v$ all the components of $\mathbf{q}_\mu$ for $\mu < v$. Unlike the Gram-Schmidt procedure, however, one need only subtract from $\mathbf{q}_v$ its component in the direction of the previously constructed basis vector; orthogonality with respect to the other basis vectors is ensured because a special inner product is used in the CR method. This inner product is defined by $[\mathbf{x}_1, \mathbf{x}_2] = (\mathbf{P}\mathbf{A}\mathbf{x}_1, \mathbf{A}\mathbf{x}_2)$, where $(,)$ denotes the standard Euclidean inner product. The computational economy of the CR method results from this need to orthogonalize only with respect to the previously constructed basis vector. Once the new basis vector is obtained, the component of the solution $\mathbf{x}$ in its direction is found and is added to $\mathbf{x}_v$ to obtain the new approximation $\mathbf{x}_{v+1}$.

For the preconditioning matrix, we use what we call Jacobi preconditioning. $\mathbf{P}$ is taken to be the matrix whose main diagonal elements are the reciprocals of the main diagonal elements of $\mathbf{A}$ and whose off-diagonal elements are zero.

For solution of Eqs. (29) and (30), the residual equation is

$$r_v = \phi_v - \frac{\mu_T}{\rho}\mathbf{D}_v^{n+1} \cdot \left[\tilde{\omega} + \Delta t \mathbf{D}_c^{n+1}\phi\right] , \tag{B-1}$$

where v is the vertex where ϕ_v is located. Since $\frac{\partial r}{\partial \phi} = -\mathbf{A}$, the diagonal elements of $\mathbf{A}$ can be found by taking the derivative of (B-1) with respect to ϕ_v; and since $\frac{\partial}{\partial \phi_v}$ commutes with the operators $\mathbf{D}_v^{n+1}$ and $\mathbf{D}_c^{n+1}$, we obtain

$$\frac{\partial r_v}{\partial \phi_v} = 1 - \frac{\mu_T}{\rho}\Delta t \mathbf{D}_v^{n+1} \cdot \left(\mathbf{D}_c^{n+1}\delta_v\right) , \tag{B-2}$$

where δ_v is a mesh function that is unity at vertex v and zero at other vertices. Using the definitions (17) and (18) of $\mathbf{D}_v$ and $\mathbf{D}_c$ and expressing momentum cell-face areas in terms of regular cell-face areas, it can be seen that

$$\frac{\partial r_v}{\partial \phi_v} = 1 + \frac{1}{16}\frac{\mu_r}{\rho}\frac{\Delta t}{V_v}\sum_r \left(\sum_{\substack{\alpha \text{ in } r \\ \text{touching } v}} \mathbf{A}_\alpha \right)^2 \Big/ V_r \,, \tag{B-3}$$

where $\sum_r$ is the sum over regular cells touching vertex v, and $\sum_{\substack{\alpha \text{ in } r \\ \text{touching } v}}$ is the sum over faces α of regular cell r that touch vertex v. Equation (B-3) is used to calculate the diagonal elements of the preconditioning matrix when solving for ϕ^{n+1}.

The components of the Turn Vector $\boldsymbol{\tau}$ are solved for simultaneously by CR iteration. We now derive the preconditioning matrix used in this iteration. The residual equation is obtained by combining Eqs. (31) and (32):

$$\mathbf{r} = \boldsymbol{\tau} - \mu_r \left\{ \boldsymbol{\omega}^{n+1} - \mathbf{G}_c^{n+1} \times \left[\tilde{\mathbf{u}} + \frac{\Delta t V_v^{n+1}}{M_v^{n+1}} \left(\mathbf{D}_v^{n+1} \times \boldsymbol{\tau} \right) \right] \right\} \,. \tag{B-4}$$

Taking the derivative of the i^{th} component of this equation with respect to the i^{th} component of $\boldsymbol{\tau}$ gives

$$\frac{\partial r_i}{\partial \tau_i} = 1 + \mu_r \frac{\partial}{\partial \tau_i} \left\{ \mathbf{G}_c^{n+1} \times \left[\tilde{\mathbf{u}} + \frac{\Delta t V_v^{n+1}}{M_v^{n+1}} \left(\mathbf{D}_v^{n+1} \times \boldsymbol{\tau} \right) \right] \right\} \,. \tag{B-5}$$

From the definition, Eq. (21), of the operator $\mathbf{G}_c$, we see that the derivative of the expression in braces in Eq. (B-5) is determined by the constraints

$$\frac{\partial}{\partial \tau_i} \left\{ \mathbf{G}_c^{n+1} \times \mathbf{u} \right\} \cdot \sum_{\substack{\alpha \\ \text{touching} \\ v}} \mathbf{A}_\alpha = \frac{\partial}{\partial \tau_i} \left\{ \sum_{\substack{\alpha \\ \text{touching} \\ v}} C_\alpha \right\} \,, \tag{B-6}$$

where

$$\mathbf{u}_v = \tilde{\mathbf{u}}_v + \frac{\Delta t V_v^{n+1}}{M_v^{n+1}} \left(\mathbf{D}_v^{n+1} \times \boldsymbol{\tau} \right) \,, \tag{B-7}$$

v is any vertex of the cell in which $\mathbf{G}_c$ is being evaluated, and C_α are the cell-face circulations induced by the velocity field $\mathbf{u}$. From the definition of C_α, it is easily seen that

$$\frac{\partial}{\partial \tau_i} C_\alpha(\mathbf{u}) = C_\alpha \left(\frac{\partial \mathbf{u}}{\partial \tau_i} \right) \,; \tag{B-8}$$

that is, the derivative of the cell-face circulation based on $\mathbf{u}$ is the cell-face circulation based on the derivative of $\mathbf{u}$. Taking the derivative of (B-7) with respect to τ_i gives

$$\frac{\partial \mathbf{u}_v}{\partial \tau_i} = \frac{\Delta t}{4M_v} \left[\mathbf{e}_i \times \sum_{\substack{\alpha \\ \text{touching} \\ v}} \mathbf{A}_\alpha \right] , \tag{B-9}$$

where $\mathbf{e}_i$ is the unit vector in the i^{th} coordinate direction.

In summary, the diagonal elements of the preconditioning matrix for the $\boldsymbol{\tau}$-iteration are obtained from

$$\frac{\partial r_i}{\partial \tau_i} = 1 + \mu_\tau \frac{\partial}{\partial \tau_i} \left[\mathbf{G}_c^{n+1} \times \mathbf{u} \right] , \tag{B-10}$$

where the derivative with respect to τ_i is determined by the constraints

$$\frac{\partial}{\partial \tau_i} \left[\mathbf{G}_c^{n+1} \times \mathbf{u} \right] \cdot \sum_{\substack{\alpha \\ \text{touching} \\ v}} \mathbf{A}_\alpha = \sum_{\substack{\alpha \\ \text{touching} \\ v}} \tilde{C}_\alpha , \tag{B-11}$$

where $\tilde{C}_\alpha$ are the cell-face circulations based on use of the vertex vector field Eq. (B-9). As with Eq. (21), Eq. (B-11) gives three independent constraints that determine the components of $\frac{\partial}{\partial \tau_i} \left[\mathbf{G}_c^{n+1} \times \mathbf{u} \right]$.

REFERENCES

1. F. V. Bracco, Princeton University, private communication (1991).
2. A. A. Amsden et al., "KIVA: A Computer Program for Two- and Three-Dimensional Fluid Flows with Chemical Reactions and Fuel Sprays," Los Alamos National Laboratory report LA-10245-MS (February 1985).
3. A. A. Amsden, P. J. O'Rourke, and T. D. Butler, "KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays," Los Alamos National Laboratory report LA-11560-MS (May 1989).
4. C. W. Hirt, A. A. Amsden, and J. L. Cook, *J. Comput. Phys.* **14**, 227 (1974).
5. P. J. O'Rourke, *J. Comput. Phys.* **53**, 359 (1984).
6. J. D. Ramshaw and G. L. Mesina, *Computers Fluids* **20**, 165 (1991).
7. J. U. Brackbill, "Numerical Models for High Beta Magnetohydrodynamic Flow," in *Computer Applications in Plasma Science and Engineering*, Springer-Verlag (1991).
8. P. J. O'Rourke and A. A. Amsden, "Implementation of a Conjugate Residual Iteration in the KIVA Computer Program," Los Alamos National Laboratory report LA-10849-MS (October 1986).

9. G. D. Smith, *Numerical Solution of Partial Differential Equations*, Clarendon Press (Oxford, 1978).
10. J. K. Dukowicz and B. J. A. Meltz, *J. Comput. Phys.* **102**, 336 (1992).
11. A. A. Amsden et al., "Comparisons of Computed and Measured Three-Dimensional Velocity Fields in a Motored Two-Stroke Engine," Society of Automotive Engineers Paper 920418 (February 1992).
12. P. M. Gresho and S. T. Chan, "Semi-Consistent Mass Matrix Techniques for Solving the Incompressible Navier-Stokes Equations," Lawrence Livermore National Laboratory report UCRL-99503 (1988).
13. T. E. Tezuyar et al., *Comput. Methods Appl. Mech Engrg.* **95**, 221 (1992).
14. B. VanLeer, *J. Comput. Phys.* **32**, 101 (1979).
15. J. K. Dukowicz and J. D. Ramshaw, *J. Comput. Phys.* **32**, 71 (1979)
16. P. R. Halmos, *Finite-Dimensional Vector Spaces*, D. Van Nostrand, Inc. (Princeton, New Jersey, 1958).