SANDIA REPORT

SAND2009-5154
Unlimited Release
Printed September 2009

A Comparison of Lagrangian/Eulerian Approaches for Tracking the Kinematics of High Deformation Solid Motion

Allen C. Robinson, David I. Ketcheson, Thomas L. Ames, Grant V. Farnsworth

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2009-5154
Unlimited Release
Printed September 2009

A Comparison of Lagrangian/Eulerian Approaches for Tracking the Kinematics of High Deformation Solid Motion

Allen C. Robinson, David I. Ketcheson,
Thomas L. Ames, and Grant V. Farnsworth
Computational Shock & Multiphysics Department
Sandia National Laboratories
MS-0378, P.O. Box 5800
Albuquerque, NM 87185-0378

Abstract

The modeling of solids is most naturally placed within a Lagrangian framework because it requires constitutive models which depend on knowledge of the original material orientations and subsequent deformations. Detailed kinematic information is needed to ensure material frame indifference which is captured through the deformation gradient **F**. Such information can be tracked easily in a Lagrangian code. Unfortunately, not all problems can be easily modeled using Lagrangian concepts due to severe distortions in the underlying motion. Either a Lagrangian/Eulerian or a pure Eulerian modeling framework must be introduced. We discuss and contrast several Lagrangian/Eulerian approaches for keeping track of the details of material kinematics.

Acknowledgments

The authors acknowledge helpful discussions, interactions and communications with our colleagues including Rebecca Brannon, Kevin Brown, Christopher Garasi, Mark Christon, Greg Miller, Sharon Petney, Duane Labreche, William Scherzinger and Michael Wong. Ed Love pointed out the utility and form of the rotation preserving midstep interpolation formulae and kindly reviewed a draft of this report.

The student intern program at Sandia provided opportunities for authors Ames, Ketcheson and Farnsworth to provide contributions to this work.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Contents

List of Figures

List of Tables

This page intentially left blank.

1 Introduction

A Lagrangian framework is ideal for modeling the response of solids to external loads because such modeling typically requires constitutive models which depend on detailed knowledge of the material motion. This relationship is captured through the deformation gradient $\mathbf{F}$. However, some motions cannot be described on a Lagrangian grid and Eulerian concepts must be introduced. Arbitrary Lagrangian-Eulerian (ALE) codes accomplish this by taking a Lagrangian step followed by an optional remap step.[20] The Lagrangian step solves the equations of motion in the frame of reference of the moving material. The Eulerian or remap step moves the mesh coordinates back to their original configuration or to some other convenient position determined by the remesh step. Some approaches remap to nearby meshes using conservative interpolation concepts from hyperbolic equations, and for this reason, the remap step can be referred to as an advection step. Other technologies implement a remap step that is more general.

A pure Eulerian framework for solid dynamic modeling is also possible. The general approach is to write the equations in conservation form, analyze their relevant mathematical properties and then solve numerically. Equations for elastic flow in Eulerian form are discussed by Plohr and Sharp.[22] This work was subsequently extended to include rate-dependent and rate-independent plasticity.[23] One-dimensional numerical solution techniques for this type of model were later presented.[37] The 20-component equation system for rate-dependent metals includes the inverse deformation gradient (9), momentum (3) and energy conservation (1), plastic strain (6) and a work hardening equation. (If the assumption of an initial uniform density is relaxed then one more equation would be required for a total of 21 variables.) Application of these equations to shear band modeling has also been presented.[10] One of the difficulties of solid dynamics modeling is that the most effective and universal modeling framework and associated constitutive forms have not been completely agreed upon. Careful and thoughtful reasoning based on physical principles and experimental data is required.[21,36] Multi-dimensional modeling results from this group are relatively recent and are associated with a front tracking numerical method.[34,35] The issue of the use of stable, properly posed equations for the inverse deformation gradient is discussed briefly by Walter, et al.[35] in relationship to an interface equation and in detail by Miller and Colella.[18,19]

Trangenstein and Collella introduced a general framework for modeling finite deformation in solids.[30] Their paper details many of the difficulties of modeling finite deformation in solids in an Eulerian framework. Additional papers by Trangenstein and coauthors have also appeared.[29,31,32] More recently, work by Miller and Collella discusses the stability requirements for the inverse deformation gradient equations in Eulerian form and presents both elastic and elastic-plastic results.[18,19] The model presented is very similar but involves 24 independent variables because the plastic deformation gradient is solved for in place of the symmetric plastic strain and an additional mass density equation is used.

Lin and Ballmann develop a conservative Godunov formulation based on the solution of a Riemann problem for one-dimensional motion of a thin-walled tube with longitudinal and torsional stress wave propagation.[16] The system maintains longitudinal and circumferential velocities and strains as independent degrees-of-freedom. A Godunov approach is also followed by Udaykumar, et. al.[33] They solve a two-dimensional conservation law system for mass, momentum, energy, equivalent plastic strain and deviatoric stress. This system only keeps advective terms for the deviatoric stress in the conservative flux leaving the elastic contributions to be discretized as source terms.

Fundamentally, solid dynamic modeling must include first, a description of the kinematics of material motion; and second, an optional set of state variables which may evolve with the motion and track local material state properties. Linear elastic or hyperelastic models provide the stress given information about the current deformation. However, for large elasto-plastic deformations, the stress in general depends on the history of the deformation. Hypoelastic models, which relate invariant stress rate measures to deformation rates, have long been popular despite concerns about the thermodynamic validity of commonly used simple hypoelastic forms.[28] This report touches on an ALE formulation developed from a hypoelastic model point of view in terms of the Green-Naghdi rate. All constitutive models are cast in the unrotated configuration and the rotation tensor is used to transform the stress from the unrotated to the current configuration. The fundamental kinematic variables related to the deformation gradient are evaluated at element centers. This is equivalent to the average value in the element to second order in the element size. A conservative interpolation method is used to update element centered average values across a remap step. Thus, we effectively solve

$$\phi_t + \nabla \cdot (\phi \mathbf{v}_m) = 0$$

where ϕ is a scalar value and $\mathbf{v}_m$ is the effective remesh velocity. In this document components of tensors are remapped individually. In the remap operations, both the kinematic descriptors and state descriptors may be moved off any constraint surface where they are supposed to lie. For example, the determinant of a deformation gradient must be positive in order to create a physically relevant one-to-one mapping between each material point and its current location. Furthermore, curls of gradients should vanish identically. A given remapping method may not preserve these properties, and the results may not be desirable. As another example, it may be difficult to preserve a stress state that is constrained to lie on a yield surface in the remap step. Issues associated with remap of material state information are difficult, and tradeoffs may be necessary.[24]

Any Eulerian or ALE scheme for tracking the kinematics of solid deformation is bound to eventually break down given enough deformation. Solid dynamics intrinsically depends on the one-to-one mapping between a reference set of coordinates and the current coordinates. Because it is always possible to generate a flow which is distorted enough that this mapping is difficult or impossible to achieve, some sort of implicit or explicit regularization or correction is required. Thus, it is essential to determine which techniques are the most accurate and robust in the face of a variety of difficult material motions.

In this report we are concerned with understanding issues associated with the choice of algorithms for modeling the kinematics of solid motion as well as the consequences of these choices with respect to speed and accuracy. We will lay the state descriptor issue aside and concentrate on approaches for maintaining the kinematic information in the form of the deformation gradient or its inverse. We shall investigate a traditional approach for updating components of the polar decomposition, a more compact form based on quaternions and an approach based on a direct computation of the inverse deformation gradient. We propose a few simple test problems with exact solutions and make comparisons of the methods in Lagrangian and ALE frameworks.

2 Stretch and Rotation Update Method (VR)

We define the Lagrangian motion $\mathbf{x}(\mathbf{a}, t)$, where $\mathbf{x}$ represents the current coordinates as a function of the initial Lagrangian positions $\mathbf{a}$ and time t, and the corresponding velocity, $\mathbf{v} = \dot{\mathbf{x}}(\mathbf{a}, t) = \partial\mathbf{x}/\partial t$. Vector differential operators ∇, $\nabla\times$ and $\nabla\cdot$ are always with respect to the specified spatial coordinates, $\mathbf{x}$. Also, $\mathbf{w}$ is the axial vector corresponding to the skew-symmetric tensor $\mathbf{W}$ (i.e. $\mathbf{W}\mathbf{u} = \mathbf{w} \times \mathbf{u}$ for every vector $\mathbf{u}$). We assume a time step Δt and mesh spacing Δx.

The kinematics of modeling continuum solids is described via a motion, $\mathbf{x}(\mathbf{a}, t)$, where $\mathbf{x}$ is the current position as a function of the initial Lagrangian coordinates $\mathbf{a}$ and time t. Of particular interest is the deformation gradient tensor $\partial\mathbf{x}/\partial\mathbf{a}$ which is needed in order to properly compute material stress states through the constitutive assumptions.

One common approach is to decompose the deformation gradient at the initial time,

$$\mathbf{F} = \frac{\partial\mathbf{x}}{\partial\mathbf{a}} = \mathbf{V}\mathbf{R} \tag{2.1}$$

into a symmetric positive definite left stretch tensor $\mathbf{V}$ and an orthonormal rotation tensor $\mathbf{R}$. The $\mathbf{V}$ and $\mathbf{R}$ matrices are then updated separately in a Lagrangian sense using a rate equation based on the mid-point velocity gradient tensor $\mathbf{L}$.[5,8,14] That is,

$$\mathbf{L}_{n+1/2} = \nabla_{n+1/2}\mathbf{v}_{n+1/2} = (\nabla\mathbf{v})_{n+1/2} \tag{2.2}$$

and

$$\mathbf{D}_{n+1/2} = \frac{1}{2}(\mathbf{L}_{n+1/2} + \mathbf{L}^T_{n+1/2}) \tag{2.3}$$

$$\mathbf{W}_{n+1/2} = \frac{1}{2}(\mathbf{L}_{n+1/2} - \mathbf{L}^T_{n+1/2}). \tag{2.4}$$

An angular velocity (or spin) tensor $\boldsymbol{\Omega}$ is calculated from $\mathbf{D}_{n+1/2}$ and $\mathbf{V}_n$ using

$$\boldsymbol{\omega}_{n+1/2} = \mathbf{w}_{n+1/2} + [tr(\mathbf{V}_n)\mathbf{I} - \mathbf{V}_n]^{-1}\mathbf{z}_{n+1/2}, \tag{2.5}$$

where $\mathbf{z}_{n+1/2}$, $\mathbf{w}_{n+1/2}$ and $\boldsymbol{\omega}_{n+1/2}$ are the axial vectors corresponding to the skew-symmetric tensors $\mathbf{Z}_{n+1/2} = \mathbf{D}_{n+1/2}\mathbf{V}_n - \mathbf{V}_n\mathbf{D}_{n+1/2}$, $\mathbf{W}_{n+1/2}$ and

$\mathbf{\Omega}_{n+1/2}$, respectively. A common first-order in time algorithm for $\mathbf{V}_n$ and $\mathbf{R}_n$,[5,8] is

$$\begin{aligned}
\mathbf{V}_{n+1} &= \mathbf{V}_n + \Delta t(\mathbf{L}_{n+1/2}\mathbf{V}_n - \mathbf{V}_n\mathbf{\Omega}_{n+1/2}) && (2.6)\\
\mathbf{R}_{n+1} &= [\mathbf{I} - \frac{1}{2}\Delta t\mathbf{\Omega}_{n+1/2}]^{-1}[\mathbf{I} + \frac{1}{2}\Delta t\mathbf{\Omega}_{n+1/2}]\mathbf{R}_n. && (2.7)
\end{aligned}$$

Note that the right hand side of Equation 2.7 is an approximation to the matrix exponential solution

$$\mathbf{R}_{n+1} = \exp[\mathbf{\Omega}_{n+1/2}\Delta t]\mathbf{R}_n.$$

which is exact if Ω is constant. The overall algorithm is first-order accurate because the derivative is computed using $\mathbf{V}_n$ instead of $\mathbf{V}_{n+1/2}$ in Equation 2.5.

For staggered space-time discretizations, the velocity gradient $\mathbf{L}$ is known only at the mid-point time, but the $\mathbf{V}$ and $\mathbf{R}$ tensors are known at the time step. After obtaining $\mathbf{\Omega}_n$, we can compute an estimate of $\mathbf{V}$ at the mid-point time using

$$\mathbf{V}_{n+1/2} = \mathbf{V}_n + \frac{1}{2}\Delta t(\mathbf{L}_{n+1/2}\mathbf{V}_n - \mathbf{V}_n\mathbf{\Omega}_n) \tag{2.8}$$

and then use this estimate of $\mathbf{V}_{n+1/2}$ to compute $\mathbf{\Omega}_{n+1/2}$ as in Equation 2.5.

$$\boldsymbol{\omega}_{n+1/2} = \mathbf{w}_{n+1/2} + [tr(\mathbf{V}_{n+1/2})\mathbf{I} - \mathbf{V}_{n+1/2}]^{-1}\mathbf{z}_{n+1/2}, \tag{2.9}$$

with $\mathbf{Z}_{n+1/2} = \mathbf{D}_{n+1/2}\mathbf{V}_{n+1/2} - \mathbf{V}_{n+1/2}\mathbf{D}_{n+1/2}$. The final update then becomes

$$\begin{aligned}
\mathbf{V}_{n+1} &= \mathbf{V}_n + \Delta t(\mathbf{L}_{n+1/2}\mathbf{V}_{n+1/2} - \mathbf{V}_{n+1/2}\mathbf{\Omega}_{n+1/2}), && (2.10)\\
\mathbf{R}_{n+1} &= [\mathbf{I} - \frac{1}{2}\Delta t\mathbf{\Omega}_{n+1/2}]^{-1}[\mathbf{I} + \frac{1}{2}\Delta t\mathbf{\Omega}_{n+1/2}]\mathbf{R}_n. && (2.11)
\end{aligned}$$

Equations 2.8 through 2.10 amount to a mid-point Runge-Kutta integration and thus lead to a second-order accurate time integration algorithm. Additional computational cost involves little more than one extra tensor inversion per cycle. Equation 2.11 is directly derivable from a second order time discretization of the rotation differential equation $\dot{\mathbf{R}} = \Omega\mathbf{R}$ and, as can easily be shown, provides an orthonormal rotation update.[14,27] Both the first-order accuracy of the original algorithm and the second-order accuracy of the algorithm described in this section have been verified numerically. We refer to these algorithms as the Hughes-Winget (HW) algorithms based on the intial motivation by Thomas Hughes and James Winget.

The principal reason for calculating $\mathbf{R}$ is for use in satisfying the principle of material frame indifference in material models. The Cauchy stress , $\mathbf{T}$, is required to integrate the equations of motion however what is stored is the unrotated or reference stress configuration, $\mathbf{\Sigma}$. They are related by

$$\mathbf{\Sigma}_n = \mathbf{R}_n^T \mathbf{T}_n \mathbf{R}_n. \tag{2.12}$$

For example, a hypo-elastic material model may take the form

$$\begin{aligned} \mathbf{d}_{n+1/2} &= \mathbf{R}_{n+1/2}^T \mathbf{D}_{n+1/2} \mathbf{R}_{n+1/2} && (2.13) \\ \mathbf{\Sigma}_{n+1} &= \mathbf{\Sigma}_n + \Delta t \mathbf{f}(\mathbf{d}_{n+1/2}, \mathbf{\Sigma}_n). && (2.14) \end{aligned}$$

where we see that the rate-of-strain tensor, $\mathbf{D}_{n+1/2}$, is rotated into the material frame as $\mathbf{d}_{n+1/2}$ for use by the constitutive equations. $\mathbf{R}_{n+1}$ may also be used instead of $\mathbf{R}_{n+1/2}$ but this reduces the overall order of the stress update equations. Whenever $\mathbf{T}$ is needed, it is extracted using

$$\mathbf{T}_{n+1} = \mathbf{R}_{n+1} \mathbf{\Sigma}_{n+1} \mathbf{R}_{n+1}^T. \tag{2.15}$$

We point out that Equation 2.11 is only one of several forms for updating rotation tensors in terms of $\mathbf{\Omega}_{n+1/2}$. Various forms of the Euler-Rodrigues formula or exponential map are possible. The exponential map transforms skew-symmetric matrices into orthogonal matrices according to the expression

$$\exp[\mathbf{\Omega}] = \sum_{n=0}^{\infty} \frac{1}{n!} [\mathbf{\Omega}]^n. \tag{2.16}$$

Rodrigues is attributed with showing that this has the closed form solution

$$\exp[\mathbf{\Omega}] = \mathbf{I} + \frac{\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|} \mathbf{\Omega} + \frac{1}{2} \left[\frac{\sin(\frac{\|\boldsymbol{\omega}\|}{2})}{\frac{\|\boldsymbol{\omega}\|}{2}} \right]^2 \mathbf{\Omega}^2. \tag{2.17}$$

where $\boldsymbol{\omega}$ is the axial vector corresponding to $\mathbf{\Omega}$. This can then be used to update $\mathbf{R}_n$ using the expression

$$\mathbf{R}_{n+1} = \exp[\Delta t \mathbf{\Omega}_{n+1/2}] \mathbf{R}_n \tag{2.18}$$

This algorithmn has the advantage of being an exact solution for constant spins, and is second-order accurate.[4,27]

Lagrangian motion algorithms and associated material models require one or both of $\mathbf{R}_{n+1/2}$ (e.g. for stress integration algorithms) and $\mathbf{R}_{n+1}$ for rotation to the current frame as in Equation 2.15. Due to the Lie algebra generated by skew symmetric tensors, very tight and efficient closed form equations can be obtained for these quantities. $\mathbf{R}_{n+1/2} = \hat{\mathbf{Q}}^{1/2}\mathbf{R}_n$ has been proposed by Trangenstein.[30] For the exponential map form, this is equivalent to

$$\mathbf{R}_{n+1/2} = \exp[\frac{\Delta t}{2}\mathbf{\Omega}_{n+1/2}]\mathbf{R}_n = \exp[-\frac{\Delta t}{2}\mathbf{\Omega}_{n+1/2}]\mathbf{R}_{n+1} = \hat{\mathbf{Q}}^{1/2}\mathbf{R}_n = \hat{\mathbf{Q}}^{-1/2}\mathbf{R}_{n+1} \tag{2.19}$$

Thus, simple efficient expressions are possible for $\mathbf{R}_{n+1/2}$ in terms of $\mathbf{\Omega}_{n+1/2}$, $\hat{\mathbf{Q}}^{1/2}$ or $\hat{\mathbf{Q}}^{-1/2}$. Alternatively, if storing the rotations at n and $n+1$ is more desirable than storing the midpoint rotations in some representation, we can compute the midpoint rotation from $\mathbf{R}_{n+1}$ and $\mathbf{R}_n$ using the following formula derived from Equation 2.17. Define $\mathbf{R}_{n+1}\mathbf{R}_n^T = \exp[\mathbf{\Omega}]$. By taking the trace of Equation 2.17

$$\cos(\|\boldsymbol{\omega}\|) = \frac{Tr(\exp[\mathbf{\Omega}]) - 1}{2} \tag{2.20}$$

which allows the computation of $\|\boldsymbol{\omega}\|$. In addition using Equation 2.17 we find that

$$\exp[\mathbf{\Omega}] - \exp[\mathbf{\Omega}]^T = \frac{2\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|}\mathbf{\Omega} \tag{2.21}$$

giving $\mathbf{\Omega}$ which thus allows for the straightforward computation of $\mathbf{R}_{n+1/2}$ as

$$\mathbf{R}_{n+1/2} = \exp[\mathbf{\Omega}/2]\mathbf{R}_n. \tag{2.22}$$

2.1 VR Remap

One approach for accomplishing the remap step is to advect the stretch and rotation matrices componentwise using a volume-based element remap algorithm that preserves the integral of the components.[20] This means that 9 components are remapped in 2D (4 from the stretch tensor and 5 from the rotation tensor), and 15 components are remapped in 3D (6 from stretch and 9 from rotation). (Note that in 2D we have included the $\mathbf{V}_{33}$ and $\mathbf{R}_{33}$ components as these are advected even if they are equal to one.) Because the VR update algorithm is local in nature, only materials in the problem that need to track the rotation tensor do so. In our implementation, the rotation and stretch are associated only with those materials needing them for material modeling purposes.

After the componentwise remap operation is completed we have new stretch and rotation matrices $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{R}}$. This algorithm does not violate the symmetry of $\mathbf{V}$, but it may not preserve the orthonormality of $\mathbf{R}$. For large deformations, successive remaps may cause the stretch tensor to lose the property of positive definiteness. A stretch that is sufficiently large or small will cause numerical round-off issues. Additionally, it is possible that very large stretch rates may be introduced relative to the chosen time step and there appears to be no reason why loss of positive definiteness may not occur directly in the Lagrangian time integration step. These issues represent fundamental errors relative to the modeling assumption that $\det(\mathbf{F}) > 0$ and must be corrected or at least managed in some way.

A negative eigenvalue in the stretch tensor indicates a breakdown in the kinematic description as a result of finite resolution and remap errors. For sufficiently large deformations, these errors are inevitable. The errors are often spatially localized and it may be desirable to minimize the impact of these errors on the rest of the grid while pushing the computation through to completion. To do so, it is important to detect when possible fatal errors are developing and take measures to reduce them.

We propose that the eigenvalues of the stretch tensor be required to lie in a user selectable interval (λ_s, λ_l). The eigenvalues of $\tilde{\mathbf{V}}$ will lie in this range if $\tilde{\mathbf{V}} - \lambda_s \mathbf{I}$ and $\lambda_l \mathbf{I} - \tilde{\mathbf{V}}$ are both positive definite. A useful test for positive definiteness is to verify that all of the principal determinants of the tensors are positive.[9] If this test fails, we propose a fast spectral decomposition solver to examine the eigenvalues of the stretch tensor at each time step and force them to remain positive. The eigenvalue solver used for our work was provided by Scherzinger.[25] Let

$$\tilde{\mathbf{V}}\tilde{\mathbf{Q}} = \tilde{\mathbf{Q}}\boldsymbol{\Lambda} \tag{2.23}$$

represent an eigenvalue/eigenvector diagonalization of $\tilde{\mathbf{V}}$ and let λ_s be a preselected floor value and $1/\lambda_s$ be a the preselected ceiling value. If $\lambda_1 \leq$

$\lambda_2 \leq \lambda_3$ are the ordered eigenvalues of $\mathbf{V}$, then we modify the elements of $\mathbf{\Lambda}$ according to

$$\begin{aligned} \hat{\lambda_1} &= \min(1/\lambda_s, \max(\lambda_1, \lambda_s)) && (2.24) \\ \hat{\lambda_2} &= \min(1/\lambda_s, \max(\lambda_2, \lambda_s)) && (2.25) \\ \hat{\lambda_3} &= \min(1/\lambda_s, \max(\lambda_3, \lambda_s)) && (2.26) \end{aligned}$$

and compute the "corrected" $\mathbf{V}$ using

$$\mathbf{V} = \tilde{\mathbf{Q}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{Q}}^T. \tag{2.27}$$

Although this correction does not remove the underlying problem associated with finite grid resolution and remapping, it can lessen the impact on the rest of the calculation.

Figures 1, 2, and 3 compare stretch tensor eigenvalues for a discontinuous rotation profile to be described later in our detailed tests. It demonstrates the eigenvalue reset algorithm with various limits on the eigenvalues. In Figure 1, the eigenvalues are not reset because they always lie within the bounds of $\lambda_s = 10^{-2}$ and $\lambda_l = 10^2$. In Figure 2, a few eigenvalues are limited to demonstrate the algorithm. In Figure 3, values of limiting eigenvalues are selected to exaggerate the effect of the algorithm.

After the components of the rotation tensor are remapped, the remapped rotation tensor $\tilde{\mathbf{R}}$ will deviate from the orthonormality property required by the kinematic theory. The obvious solution to this problem is to project $\mathbf{R}$ into the space of orthonormal tensors at the end of the remap step by performing performing a polar decomposition as in Equation 2.1 and setting $\mathbf{R}$ to the rotation tensor portion of this decomposition. This essentially throws away the stretch part or the decomposition which we consider remap error.

In two dimensions we use an explicit method provided by Brannon.[4] This is

$$\begin{aligned} R_{11} &= (\tilde{R}_{11} + \tilde{R}_{22})/a && (2.28) \\ R_{21} &= (\tilde{R}_{21} - \tilde{R}_{12})/a && (2.29) \\ R_{12} &= (\tilde{R}_{12} - \tilde{R}_{21})/a && (2.30) \\ R_{22} &= (\tilde{R}_{11} + \tilde{R}_{22})/a && (2.31) \end{aligned}$$

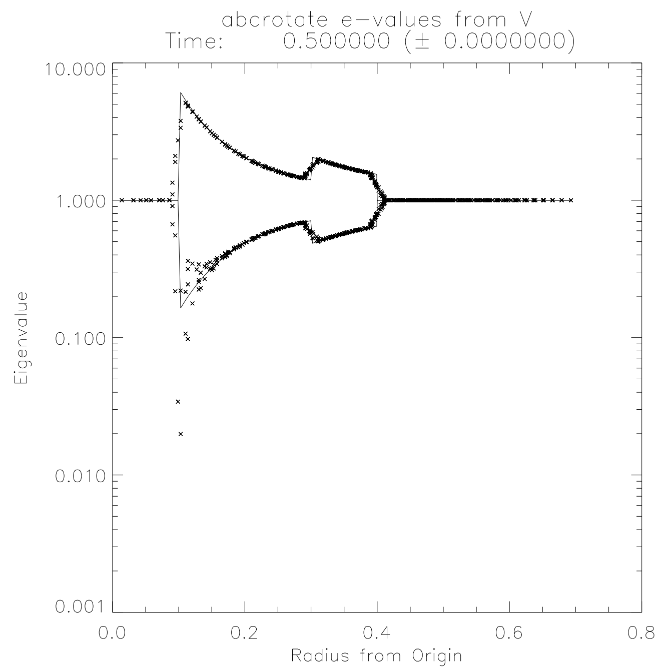

Figure 1: Stretch eigenvalues for a rotation problem with a discontinuous vorticity field. Plotted is the eigenvalue vs. radius from origin with a cut off of $\lambda_s = 10^{-2}$. The exact solution is plotted as a solid line.

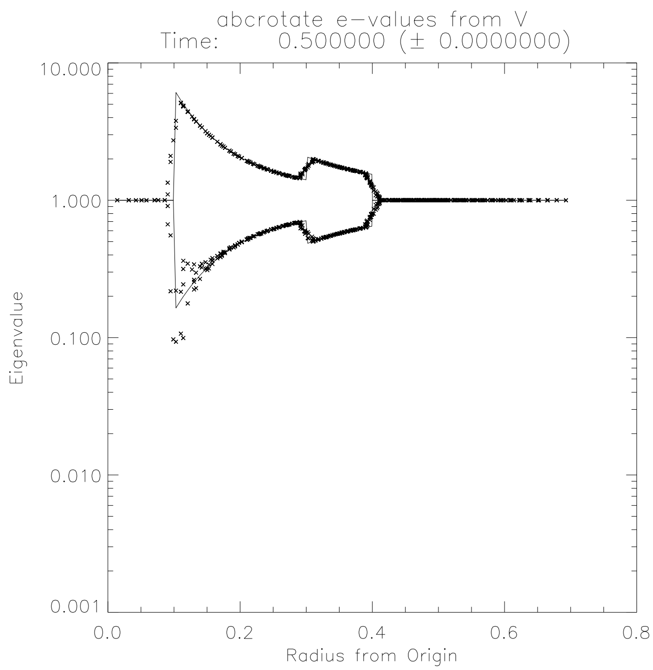

Figure 2: Stretch eigenvalues for a rotation problem with a discontinuous vorticity field. Plotted is the eigenvalue vs. radius from origin with a cut off of $\lambda_s = 10^{-1}$. The exact solution is plotted as a solid line.

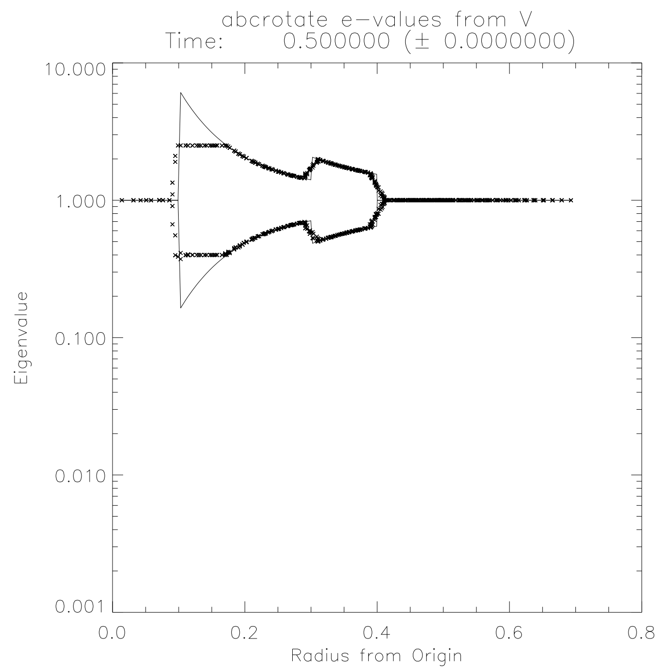

Figure 3: Stretch eigenvalues for a rotation problem with a discontinuous vorticity field. Plotted is the eigenvalue vs. radius from origin with a cut off of $\lambda_s = 4 \cdot 10^{-1}$. The exact solution is plotted as a solid line.

where

$$a = \sqrt{(\tilde{R}_{11} + \tilde{R}_{22})^2 + (\tilde{R}_{21} - \tilde{R}_{12})^2} \tag{2.32}$$

Expanding the terms inside the radical shows that

$$a = \sqrt{\tilde{R}_{11}^2 + \tilde{R}_{12}^2 + \tilde{R}_{21}^2 + \tilde{R}_{22}^2 + 2\det\tilde{\mathbf{R}}} \tag{2.33}$$

and since the remapped rotation matrices should be close to proper orthogonal rotation matrices over one remap step it is clear that a should not approach zero and the algorithm should be well defined and robust.

In three dimensions we have used both a direct method provided by Scherzinger[25] and an iterative method provided by Brannon.[4] Fast methods for polar decompositions of non-singular matrices have been studied in detail.[12,13,15] Brannon's algorithm extracts the rotation part of $\tilde{\mathbf{R}}$ via a fixed point iteration, which converges if the maximumal eigenvalue of $\tilde{\mathbf{R}}$ is less than $\sqrt{3}$. Since the orthonormal part of a tensor is invariant to scalar multiples, first rescale $\mathbf{R}$ according to

$$\mathbf{R}^0 = \sqrt{\frac{3}{tr(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}})}}\tilde{\mathbf{R}}. \tag{2.34}$$

If $\lambda_1^2 \leq \lambda_2^2 \leq \lambda_3^2$ are the non-zero ordered eigenvalues of $(\mathbf{R}^0)^T\mathbf{R}^0$, then by construction $\lambda_1^2 + \lambda_2^2 + \lambda_3^2 = 3$ and $\lambda_3^2 < 3$. Thus, iterating

$$\mathbf{R}^{m+1} = \frac{1}{2}\mathbf{R}^m[3\mathbf{I} - (\mathbf{R}^m)^T\mathbf{R}^m]. \tag{2.35}$$

is guaranteed to converge to the nearest orthonormal tensor. The iteration stops for some M where $||(\mathbf{R}^M)^T(\mathbf{R}^M) - \mathbf{I}||^2 < \epsilon^2$ and ϵ is an small number close to machine precision and then $\mathbf{R}$ is set to $\mathbf{R}^M$. The componentwise remap generally produces updated rotation tensors, $\tilde{\mathbf{R}}$ that are "nearly" orthonormal, and the iterative projection method usually converges to machine precision in just a few iterations to produce an orthonormal $\mathbf{R}$.

3 Quaternion Representation VR Method (QVR)

A more compact representation for a rotation is available in terms of quaterions.[1,38] Quaternions form an algebra called a division ring in which the elements of the algebra are not commutative under multiplication.[11] A quaternion can be represented as $q = (s, \mathbf{v})$ where s is real and $\mathbf{v}$ is a vector. The quaternion product is given by

$$q_1 q_2 = (s_1, \mathbf{v}_1)(s_2, \mathbf{v}_2) = (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2).$$

The quaternion inverse is $q^{-1} = \bar{q}/(q\bar{q})$ with $\bar{q} = (s, -\mathbf{v})$ and the quaternion norm is $|q| = \sqrt{q\bar{q}}$.

The space of quaterions with unit magnitude represents rotations through the correspondence

$$q = (\cos(\theta/2), \sin(\theta/2)\mathbf{u}) = \exp((0, \boldsymbol{\omega} t/2))$$

where $\theta = |\boldsymbol{\omega} t|$ is the rotation angle and $\mathbf{u} = \boldsymbol{\omega}/|\boldsymbol{\omega}|$ is unit spin axis. The products of unit quaterions are also unit quaternions. The mapping to a rotation tensor is given by

$$\mathbf{R} = (2s^2 - 1)\mathbf{I} + 2\mathbf{v} \otimes \mathbf{v} + 2s\hat{\mathbf{v}} \tag{3.36}$$

where $\hat{\mathbf{v}}$ represents the skew-symmetric tensor such that $\hat{\mathbf{v}}\mathbf{h} = \mathbf{v} \times \mathbf{h}$ for all $\mathbf{h}$, and $\mathbf{v} \otimes \mathbf{v} = \mathbf{v}_i \mathbf{v}_j$. Thus the rotation in three dimensions can be thought of as being given by a real 4-vector with unit scaling. In two dimensions only s and v_z are non-zero so the rotation can be represented by a vector of length two. In both cases we use one more degree of freedom to represent the rotation than strictly required. One way to think of this is that instead of storing the rotation angle we are storing the sine and the cosine of the angle. Thus when it comes time to obtain a rotation matrix only algebraic operations are required and no-transcendental functions are needed. We are still far ahead in term of compactness since 9 and 4 storage locations are required in 3D and 2D respectively in the VR case discussed in the previous section.

The quaternion rotation differential equation is

$$\dot{q} = (0, \boldsymbol{\omega}/2)q$$

which can be solved exactly for constant axis spin vector $\boldsymbol{\omega}$ as an exponential in the quaternion algebra.

$$q(t) = \exp((0, \boldsymbol{\omega}/2)t)q(0) = (\cos(|\omega t|/2), \sin(|\omega t|/2)\frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|})q(0).$$

Thus any constant spin rotation update can be computed exactly with two transcendental function evaluations and some minor algebra. The major advantage of the quaternion representation of a rotation update is that a subsequent rotation can be represented simply as a quaternion product as shown above.

The Lagrangian update algorithm for $\mathbf{V}$ is unaffected. The natural Lagrangian rotation update algorithm for the quaternion representation is the exponential map,

$$\mathbf{q}_{n+1} = \exp(0, \boldsymbol{\omega}_{n+\frac{1}{2}}\Delta t/2)\mathbf{q}_n = \exp(0, \boldsymbol{\alpha})\mathbf{q}_n. \tag{3.37}$$

We term this Lagrangian algorithm the LQVR algorithm.

It may be useful, in a similar manner to the VR algorithm, to generate the midstep rotation from the values at the end points. Recall that

$$\mathbf{q}_{n+1} = (\cos(|\boldsymbol{\alpha}|), \sin(|\boldsymbol{\alpha}|)\frac{\boldsymbol{\alpha}}{|\boldsymbol{\alpha}|})\mathbf{q}_n \tag{3.38}$$

In order to compute $|\boldsymbol{\alpha}|$ it is easily seen that

$$\frac{\mathbf{q}_{n+1}\mathbf{q}_n^{-1} + \overline{\mathbf{q}_{n+1}\mathbf{q}_n^{-1}}}{2} = s_{n+1}s_n + \mathbf{v}_{n+1} \cdot \mathbf{v}_n = \cos(|\boldsymbol{\alpha}|). \tag{3.39}$$

To compute the rotation at the half step some algebra shows that

$$\exp(0, \boldsymbol{\alpha}/2) = (\cos(|\boldsymbol{\alpha}/2|), \sin(|\boldsymbol{\alpha}/2|)\frac{\boldsymbol{\alpha}}{|\boldsymbol{\alpha}|}) = \frac{1}{2\cos(|\boldsymbol{\alpha}/2|)}((1, \mathbf{0}) + \exp(0, \boldsymbol{\alpha})) \tag{3.40}$$

and therefore

$$\mathbf{q}_{n+1/2} = \exp(0, \boldsymbol{\alpha}/2)\mathbf{q}_n = \frac{1}{2\cos(|\boldsymbol{\alpha}/2|)}(\mathbf{q}_{n+1} + \mathbf{q}_n) \tag{3.41}$$

3.1 QVR Remap

In this representation we remap the components of V exactly as in the VR case. However, the rotation treatment is vastly different. The conservative remap algorithm is applied to each component of $\mathbf{q}$ so that only 2 components in 2D and 4 components in 3D are remapped. Since the remap routine is a major computational cost we expect significant savings over the VR case.

In addition, the algorithm for keeping the quaternions representation in the proper space is much simpler when compared to the complex rotation tensor projection methodologies outlined in the previous section. The standard procedure for dealing with floating point roundoff in rotation quaternion updates is to rescale the quaternion to unit magnitude. In the remap case, truncation error will enter into the quaternion representation of the remapped field and we will obtain a q_r which does not exactly represent a rotation. We can however still use the same simple scaling methodology to ensure that the quaternion does indeed represent a rotation. That is, $q = q_r/\sqrt{q_r \bar{q}_r}$.

We tested this algorithm with and without renormalization, and found that for particularly severe, discontinuous deformations, renormalization led to much more accurate results. For smooth problems it made little difference. Because the renormalization is inexpensive, we have used it in all subsequent tests.

In Figure 4, we compare the results of two two-dimensional rotation problems using the RVR and QVR algorithms. The comparison is based on a percent L^2 error norm in R calculated with the equation

$$\sqrt{\frac{\sum\sum\left|R_{ij_{calc}} - R_{ij_{exact}}\right|^2}{\sum\sum\left|R_{ij_{exact}}\right|^2}} \tag{3.42}$$

Note that all differences here, up to roundoff errors, are due to the differences between remapping the four-component tensor and the two-component quaternion and to the differences in renormalization. Time integration does not play a factor in the differences seen because the RVR and QVR algorithms both use the exponential map instead of the Cayley transformation. Both test problems show that the QVR representation yields equivalent results or better. This seems reasonable because in the quaternion case, the remap step has fewer degrees of freedom.

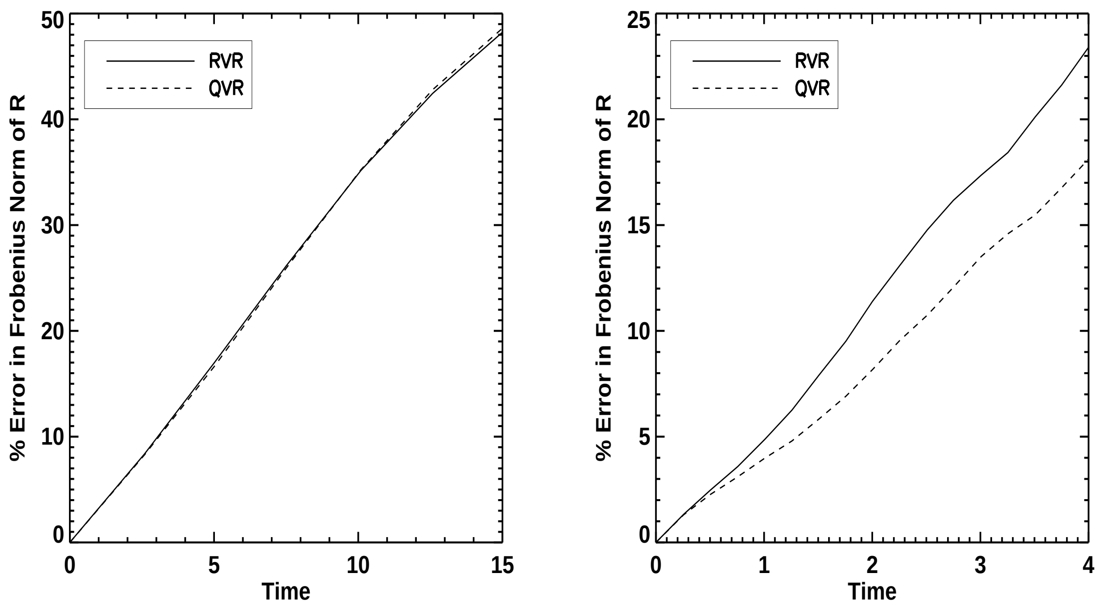

Figure 4: Comparison of RVR and QVR representations for R on two 2D rotational problems. The left figure corresponds to a smooth velocity field (Exponential Vortex), while the right is a discontinuous velocity field (ABC Vortical Flow).

4 Inverse Deformation Gradient Method (IDG)

In a Lagrangian code it is easy to compute the inverse deformation gradient with respect to a given set of spatial coordinates at any time level. Only a fast polar decomposition algorithm is required to compute $\mathbf{R}$. We simply keep track of initial Lagrangian coordinates at the mesh nodes and take a gradient with respect spatial coordinates whenever the inverse deformation gradient is needed. For example,

$$\mathbf{F}_{n+1/2}^{-1} = \mathbf{G}_{n+1/2} = \partial \mathbf{a}/\partial \mathbf{x}_{n+1/2} \tag{4.43}$$

$$\mathbf{F}_{n+1}^{-1} = \mathbf{G}_{n+1} = \partial \mathbf{a}/\partial \mathbf{x}_{n+1}. \tag{4.44}$$

The rotation tensor can then be computed by a fast right polar decomposition algorithm

$$\mathbf{F}^{-1} = \mathbf{G} = \mathbf{R}^T \mathbf{V}^{-1} \tag{4.45}$$

to compute $\mathbf{R}$ and $\mathbf{V}$ at any given time. Robust methods based on eigenvalue decompositions or iterative methods are available.[25,4]

We call this method the inverse deformation gradient (IDG) method. Note that neither the rotation nor the stretch tensor needs to be stored explicitly. For practical software reasons we have not eliminated storage of the rotation tensor in our implementation.

4.1 DG Constrained Transport Remap

One might propose that Lagrangian coordinates $\mathbf{a}$ be remapped using the standard node-centered remapping routines found in an ALE framework. This is simple and for some problems might be very effective. However, it is apparent from the the classical numerics of ideal magnetohydrodynamics (MHD) that this will be flawed. What we really care about is not the coordinates themselves but the gradients of these coordinates given by the inverse deformation gradient. To the extent that preservation of gradient monotonicity through successive remaps steps is important, advection of the coordinates themselves is a problem. In MHD, non-preservation of monotonicity results in highly undesirable and unphysical current reversals. Non-monotonicity of the inverse deformation gradient may not be as critical for solid dynamic applications but may still lead to undesirable solution characteristics or numerical breakdown. Fortunately, aspects of this remapping problem have already been addressed in the MHD community.

The constrained transport (CT) algorithm introduced by Evans and Hawley on structured meshes provides a mechanism for advection of magnetic flux density, $\mathbf{B}$, which also exactly preserves a discrete $\nabla \cdot \mathbf{B} = 0$ property.[6] The constrained transport algorithm is applicable to a staggered field representation where the magnetic flux is represented on cell faces and the electric fields on cell edges. In the finite element context the appropriate generalization of the CT staggered magnetic field representation is given by edge and face finite element bases. These bases form a deRham complex for nodes, edges, faces and volumes connected by the operators ∇, $\nabla\times$ and $\nabla\cdot$, respectively in the sense that the gradient of the nodes yields edges, the curl of the edges yields faces and the divergence of the faces yields volume. It is then natural to try to extend the CT algorithms to unstructured meshes since the finite element formalism of the deRham complex matches precisely the geometry required for computing high order upwind fluxes in the structured grid CT algorithm. A solenoidal field $\mathbf{B}$ may be generated from a vector potential $\mathbf{A}$ through the relationship

$$\mathbf{B} = \nabla \times \mathbf{A}.$$

$\mathbf{A}$ is naturally represented by edge elements with circulations on edges as degrees of freedom, and $\mathbf{B}$ is given by the solenoidal subspace of face elements with fluxes on faces. The degrees of freedom of $\mathbf{A}$ and $\mathbf{B}$ are related by simple algebraic relations in which the fluxes are represented as sums of circulations. Londrillo and Del Zanna have noted that the constrained transport algorithm is applicable to advection of vector potentials provided that the vector potential is properly represented on edges.[17] Vector potential formulations have been considered to be deficient representations for advection of magnetic fields due to errors which accumulate during the remap process. This is strictly true only if one insists on remapping nodal vector potential values using some standard reconstruction and limiting procedure for nodal quantities. If the limiting is carried out in the flux density space, good algorithms can be obtained. If vector potential components are represented on edges, it is natural to compute edge centered updates in terms of a high order magnetic flux representation. In this case there is no essential difference between a vector potential formulation and a flux centered approach because the updates in the first case are added to the vector potential while the curl of the updates is added to the face centered fluxes in the second case. Constrained transport succeeds because advection is based on reconstruction and limiting of the underlying field instead of the vector potential. This ensures that the gradients of the vector potential have monotonic properties. Advecting the vector potential directly can be disastrous because underlying gradients will not be monotonic and current reversals can be introduced.

Returning now to the kinematics of solids, the basic constrained transport ideas for magnetic flux remapping are readily applicable to constrained transport for Lagrangian positions or associated gradients. Lagrangian coordinates can be considered to be the "potentials" of the inverse deformation gradient fields. The difference now is that we are trying to stay within a "curl free" space so we expect to either update potentials on nodes or update circulations on edges as gradients of nodal increments. The idea is to ensure that the deformation gradient representation always lives in the curl free subspace of edge elements. We do this by representing each row $\mathbf{g}$ of $\mathbf{G}$ as an edge element. The degrees of freedom on each edge are the circulations of the coordinate gradients which are initialized with the point wise signed difference of the corresponding initial Lagrangian coordinates. During the Lagrangian step these circulation values must be invariant. During the remap we define a high order representation of this field by extending the edge element description to include edge circulation gradients. Given this new representation, we compute an upwind nodal flux contribution at each node and then take the gradient to update the edge circulations. This ensures that the inverse deformation gradients stays within the space of gradients.

We describe now a constrained transport remapping algorithm for hexahedral (quadrilateral) grids for a row of the inverse deformation gradient $\mathbf{g}$ where $\mathbf{g}$ is represented by a low order edge element. The algorithm consists of several parts:

1. Compute the second-order limited reconstruction of the inverse deformation gradient field by computing a circulation slope centered on each edge.

2. Determine the upwind element for each node.

3. Perform the line integral update associated with each node.

4. Update the circulation on each edge.

We shall find that each of these operations has a natural algorithm within the context of the edge element representation.

We assume that the inverse deformation gradient is represented using edge elements embedded in a finite element deRham complex associated with linear isoparametric hexahedral elements. This representation has been described previously.[2] In particular there exists an exact sequence of finite element spaces $\mathcal{W}^i(K)$ such that

$$\mathcal{W}^0(K) \quad \stackrel{\nabla}{\longmapsto} \quad \mathcal{W}^1(K) \quad \stackrel{\nabla\times}{\longmapsto} \quad \mathcal{W}^2(K) \quad \stackrel{\nabla\cdot}{\longmapsto} \quad \mathcal{W}^3(K). \tag{4.46}$$

An explicit representation for the basis functions on the reference element $\hat{K}$ in terms of the reference coordinates, $-1 \leq \xi_i \leq 1$, for $i, j, k = 1, 2, 3$ are:

$$\begin{aligned}
\hat{W}^{\alpha\beta\gamma}_{ijk} &= \frac{1}{8}(1+\alpha\xi_i)(1+\beta\xi_j)(1+\gamma\xi_k), i \neq j \neq k \\
\hat{W}^{\alpha\beta}_{ij} &= \frac{1}{8\det J_F}(1+\alpha\xi_i)(1+\beta\xi_j)(V_i \times V_j), i \neq j \\
\hat{W}^{\alpha}_{i} &= \frac{1}{8\det J_F}(1+\alpha\xi_i)V_i \\
\hat{W} &= \frac{1}{8\det J_F}
\end{aligned}$$

where we define the Jacobian matrix by $J_F = (V_1, V_2, V_3)$. The columns vectors of this matrix are defined as $V_i = (\partial F_1/\partial\xi_i, \partial F_2/\partial\xi_i, \partial F_3/\partial\xi_i)^T$, and α, β, and γ take on values of ± 1. The isoparametric mapping for the hexahedral element is given by

$$F_K(\boldsymbol{\xi}) = \sum_{\alpha\beta\gamma=\pm 1} \mathbf{x}^{\alpha\beta\gamma}\hat{W}^{\alpha\beta\gamma}_{ijk}(\boldsymbol{\xi}). \tag{4.47}$$

where $\mathbf{x}^{\alpha\beta\gamma}$ represents the physical space coordinates of the element vertices. F_K is the representation of the mapping from reference to physical coordinates. The rows of the inverse deformation gradient are represented by

$$\mathbf{g}(\xi_1, \xi_2, \xi_3) = \sum_{i \neq j, \alpha, \beta} \Gamma_{ij}^{\alpha\beta} \hat{W}_{ij}^{\alpha\beta} \tag{4.48}$$

where $\Gamma_{ij}^{\alpha\beta}$ is the total circulation along each edge. This circulation exactly equals the discrete difference of the initial Lagrangian coordinate. Elements which share an edge also share the properly signed circulation value.

It is necessary to generate second-order accurate point values of $\mathbf{g}$. These values will be used to compute the corresponding edge-wise circulation gradient. Obtaining these values at the element nodes is a non-trivial question because the edge element representation is discontinuous at element nodes. A projection operator must be defined to obtain high order accurate estimates of the field at the nodes. A patch recovery operator is suggested.

The first action in the reconstruction is to extend the definition of the edge element coefficient to contain a linear term proportional to each edge-wise reference coordinate. This term will integrate to zero along the edge thus contributing nothing to the total circulation. However, the term will contribute to the update integrals associated with each node as described later. The edge element representation for each element is now

$$\mathbf{g}(\xi_1, \xi_2, \xi_3) = \sum_{i \neq j \neq k, \alpha, \beta} \Gamma_{ij}^{\alpha\beta}(\xi_k) \hat{W}_{ij}^{\alpha\beta} \tag{4.49}$$

where, for example,

$$\Gamma_{ij}^{\alpha\beta}(\xi_k) = \bar{\Gamma}_{ij}^{\alpha\beta} + s_{ij}^{\alpha\beta}\xi_k \tag{4.50}$$

The edge circulation slope values are obtained by limiting based on the nodal reconstructed values in the edge direction. To do so, compute a node centered circulation value by dotting the reconstructed $\mathbf{g}$ values with the corresponding V_k of the attached edge. For example,

$$g(\alpha, \beta, \pm 1) \cdot V_3(\alpha, \beta, \pm 1) = \Gamma_{1,2}^{\alpha\beta}(\pm 1)/2 \tag{4.51}$$

where

$$V_3(\alpha, \beta, \pm 1) = \frac{\mathbf{x}(\alpha, \beta, +1) - \mathbf{x}(\alpha, \beta, -1)}{2} \tag{4.52}$$

This gives two slopes for each edge.

$$s_{ij}^{\alpha\beta}(\pm 1) = \pm(\Gamma_{1,2}^{\alpha\beta}(\pm 1) - \bar{\Gamma}_{1,2}^{\alpha\beta}) \tag{4.53}$$

and we chose a single slope

$$s_{ij}^{\alpha\beta} = \begin{cases} 0 & \text{if } s_{ij}^{\alpha\beta}(+1)s_{ij}^{\alpha\beta}(-1) < 0 \\ \min(s_{ij}^{\alpha\beta}(+1), s_{ij}^{\alpha\beta}(-1)) & \text{if } s_{ij}^{\alpha\beta}(+1) \text{ and } s_{ij}^{\alpha\beta}(-1) > 0 \\ \max(s_{ij}^{\alpha\beta}(+1), s_{ij}^{\alpha\beta}(-1)) & \text{if } s_{ij}^{\alpha\beta}(+1) \text{ and } s_{ij}^{\alpha\beta}(-1) < 0 \end{cases} \tag{4.54}$$

A basic requirement for computation of the circulation associated with each node is a knowledge of the upwind element associated with each edge. This is accomplished by computing a node centered position vector offset

$$-\mathbf{v}\Delta t = \delta \mathbf{x}^{nc} \tag{4.55}$$

and determining if the direction cosines of the associated elements are positive with respect to the outward edges. If they are all positive, an offset is present in that element.

We now need to compute an approximate circulation contribution at each node. The line integral at each node is conveniently given in terms of differentials in the terms of reference element coordinates ξ_i.

$$\int_\Gamma \mathbf{g} \cdot d\mathbf{s} = \int_\Gamma \mathbf{g} \cdot (\frac{\partial \mathbf{x}}{\partial \xi_1} d\xi_1 + \frac{\partial \mathbf{x}}{\partial \xi_2} d\xi_2 + \frac{\partial \mathbf{x}}{\partial \xi_3} d\xi_3) \tag{4.56}$$

Since the representation for $\mathbf{g}$ is in terms of reference element coordinates additional major simplifications will be possible. An appropriate integration method and a domain must be chosen. We chose a one point quadrature rule located at the center of the node midpoint offset vector. We define

$$\hat{\xi}_i = \frac{\delta \xi_i}{2} + \xi_i^{nc} \tag{4.57}$$

The reference element differentials for the one-point quadrature are $d\xi_1 = \delta\xi_1, d\xi_2 = \delta\xi_2$ and $d\xi_3 = \delta\xi_3$. Then

$$\int_\Gamma \mathbf{g} \cdot d\mathbf{s} \approx \mathbf{g}(\hat{\xi}) \cdot (\frac{\partial \mathbf{x}}{\partial \xi_1}(\hat{\xi})\delta\xi_1 + \frac{\partial \mathbf{x}}{\partial \xi_2}(\hat{\xi})\delta\xi_2 + \frac{\partial \mathbf{x}}{\partial \xi_3}(\hat{\xi})\delta\xi_3) \tag{4.58}$$

But the reconstructed field is given by Equation 4.49 which leads to

$$\int_\Gamma \mathbf{g} \cdot d\mathbf{s} \approx \sum_{i \neq j \neq k, \alpha, \beta} \Gamma_{ij}^{\alpha\beta}(\hat{\xi}_k) \mathbf{W}_{ij}^{\alpha\beta}(\hat{\xi}) \frac{\partial \mathbf{x}}{\partial \xi_k}(\hat{\xi}) \delta\xi_k \tag{4.59}$$

which can be simplified to

$$\int_{\Gamma} \mathbf{g} \cdot d\mathbf{s} \approx \sum_{i \neq j \neq k, \alpha, \beta} \Gamma_{ij}^{\alpha\beta}(\hat{\xi}_k)(1 + \alpha\hat{\xi}_i)(1 + \beta\hat{\xi}_j)\delta\xi_k/8. \tag{4.60}$$

The update contributions are now available on each node as circulations. These can be added directly to the coordinate (potential) representation on node or equivalently the gradient of the updates values on each edge can be taken to update the edge circulations.

By construction $\mathbf{G}$ must stay within a proper discrete curl free space and will be acceptable as long as $\det \mathbf{G} > 0$. This will be true at the beginning of the calculation but may fail later in the calculation. It is quite unclear what a rational fix might be because the degrees of freedom are differences of Lagrangian coordinates on edges. It is possible that an optimization based remap algorithm similar to the one advocated by Shashkov and Bochev for divergence free remapping might work.[3] We also note that, unlike the VR and QVR update algorithms, the IDG algorithm may not be as easily reduced to an algorithm acting only on regions of space (materials) that require stretch and rotation information. We shall see that these issues along with the cost of the current implementation prohibits recommendation of this algorithm at this time for multi-material codes.

5 Results for Given 2D Motions

We compare the methods we have described by observing each method's accuracy and cost. An accurate method converges on smooth solutions at the expected rate as Δx is decreased at a fixed Courant number. We expect that some loss of accuracy will occur for motions that are sufficiently stressful. Cost is represented by the time to solution.

We propose that much can be learned from specific test problems. The greatest difficulties are generated by flows with large strains and rotations, and we focus on how fast the stretch, rotation, and/or deformation gradient tensors lose consistency and/or accuracy in various situations. For any given test, the motion $\mathbf{x}(\mathbf{a}, t)$ can be specified directly. The deformation gradient is computed directly as a function of the Lagrangian coordinates. To find the value of the deformation gradient at a spatial point, the motion must be inverted before evaluating the deformation gradient. If desired, an alternative computational method for calculating exact solutions is including in Appendix A.

In each of the following 2D test problems, stretch tensor eigenvalues and rotation tensor matrices for both the Lagrangian and ALE cases are computed using the following algorithms:

- VR method using the Hughes-Winget algorithm (HWVR)
- VR method using the Rodrigues algorithm (RVR)
- Quaternion representation of the VR method (QVR)
- Inverse Deformation Gradient Method (IDG)

Lagrangian solutions and results are denoted by prepending the letter L to these acronymns. Computations are repeated on five different square meshes, and the convergence rates are calculated. Additionally, the cost of each method is compared using the finest grid.

5.1 Constant Velocity

We expect all methods to be able to model a constant velocity profile and deal sensibly with inflow boundary conditions from finite grids. The motion is

$$\mathbf{x} = \mathbf{a} + \hat{\mathbf{v}}t \tag{5.61}$$

with velocity

$$\mathbf{v} = \hat{\mathbf{v}} \tag{5.62}$$

Thus, the exact solution of the deformation gradient is

$$\mathbf{F} = \mathbf{I} \tag{5.63}$$

As expected, all methods yield exact results given these conditions. However, this problem is interesting because it allows us to see a cost comparison of the simplest of all cases. In Table 1, only slight cost differences are visible between the Lagrangian algorithms. On the other hand, in Table 2, the direct decomposition algorithm has a visible disadvantage.

Method	CPU Time	Rel. CPU Time
LHWVR	21.24	1.000
LRVR	22.17	1.044
LQVR	21.50	1.012
LIDG	20.86	0.982

Table 1: Relative and actual CPU time for Lagrangian Constant Velocity problem

Method	CPU Time	Rel. CPU Time
HWVR	69.85	1.000
RVR	69.89	1.001
QVR	69.00	0.988
IDG	79.24	1.134

Table 2: Relative and actual CPU time for ALE Constant Velocity problem

5.2 Sinusoidal Shear

The second variant uses a sinusoidal velocity field in one coordinate direction representing a pure shear flow.

$$\mathbf{v} = (0, \alpha sin(2\pi x)) \tag{5.64}$$

with motion

$$\mathbf{x} = \mathbf{a} + (0, \alpha t sin(2\pi a_x)) \tag{5.65}$$

The exact solution for $\mathbf{F}$ is

$$F = \begin{pmatrix} 1 & 0 \\ 2\pi\alpha t cos(2\pi x) & 1 \end{pmatrix} \tag{5.66}$$

Tables 3 through 6 show stretch tensor eigenvalue and rotation matrix convergence rates. All results indicate that second order convergence is acheived as desired. Furthermore, in the Lagrangian case we see a significant cost savings using the LIDG method. However, in the ALE case, the IDG method again displays an increased cost while the QVR method shows an advantage.

1/h	LHWVR		LRVR		LQVR		LIDG	
	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	1.07E-01	–	1.07E-01	–	1.07E-01	–	1.03E-01	–
32	2.70E-02	1.99	2.70E-02	1.99	2.70E-02	1.99	2.57E-02	2.00
64	6.77E-03	2.00	6.77E-03	2.00	6.77E-03	2.00	6.43E-03	2.00
96	2.98E-03	2.02	2.98E-03	2.02	2.98E-03	2.02	2.86E-03	2.00
128	1.67E-03	2.01	1.67E-03	2.01	1.67E-03	2.01	1.61E-03	2.00

Table 3: Stretch tensor eigenvalue convergence rates for Lagrangian Sinusoidal Shear problem.

1/h	LHWVR		LRVR		LQVR		LIDG	
	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	2.09E-03	–	2.00E-03	–	2.00E-03	–	1.81E-03	–
32	5.99E-04	1.80	5.61E-04	1.83	5.61E-04	1.83	4.94E-04	1.87
64	1.51E-04	1.98	1.38E-04	2.02	1.38E-04	2.02	1.16E-04	2.10
96	6.36E-05	2.14	5.87E-05	2.11	5.87E-05	2.11	5.03E-05	2.05
128	3.56E-05	2.02	3.28E-05	2.02	3.28E-05	2.02	2.82E-05	2.01

Table 4: Rotation tensor matrix convergence rates for Lagrangian Sinusoidal Shear problem.

1/h	HWVR		RVR		QVR		IDG	
	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	1.13E-01	–	1.13E-01	–	1.13E-01	–	1.03E-01	–
32	2.86E-02	1.98	2.86E-02	1.98	2.86E-02	1.98	2.57E-02	2.00
64	7.17E-03	1.99	7.17E-03	1.99	7.17E-03	1.99	6.43E-03	2.00
96	3.19E-03	2.00	3.19E-03	2.00	3.19E-03	2.00	2.86E-03	2.00
128	1.75E-03	2.09	1.75E-03	2.09	1.75E-03	2.09	1.61E-03	2.00

Table 5: Stretch tensor eigenvalue convergence rates for ALE Sinusoidal Shear problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	2.44E-03	–	2.25E-03	–	2.25E-03	–	1.81E-03	–
32	7.07E-04	1.79	6.36E-04	1.82	6.36E-04	1.82	4.94E-04	1.87
64	1.82E-04	1.96	1.59E-04	2.00	1.59E-04	2.00	1.16E-04	2.10
96	8.20E-05	1.97	7.11E-05	1.99	7.11E-05	1.99	5.02E-05	2.05
128	4.10E-05	2.41	3.66E-05	2.31	3.66E-05	2.31	2.82E-05	2.01

Table 6: Rotation tensor matrix convergence rates for ALE Sinusoidal Shear problem.

Method	CPU Time	Rel. CPU Time
LHWVR	1641.68	1.000
LRVR	1648.11	1.004
LQVR	1621.37	0.988
LIDG	1448.29	0.882

Table 7: Relative and actual CPU time for Lagrangian Sinusoidal Shear problem

Method	CPU Time	Rel. CPU Time
HWVR	462.90	1.000
RVR	460.32	0.994
QVR	459.66	0.993
IDG	556.25	1.202

Table 8: Relative and actual CPU time for ALE Sinusoidal Shear problem

5.3 Exponential Vortex

This simple smooth vortical flow asymptotes to an irrotational $1/r$ angular velocity profile

$$\mathbf{v}_\theta = \frac{\Gamma}{2\pi r}(1 - e^{-r^2/2}) \tag{5.67}$$

where Γ is the total circulation at infinity. Since the flow is parameterized by the radius we can compute the motion and thus the deformation gradient exactly. In particular, in polar coordinates

$$\begin{aligned} r &= r_0 \\ \theta &= \frac{\Gamma t}{2\pi r^2}(1 - e^{-r^2/2}) + \theta_0 \end{aligned} \tag{5.68}$$

and via the chain rule one can compute the deformation gradient in Cartesian coordinates. The vorticity profile is

$$\omega = \frac{1}{r}\frac{\partial}{\partial r}(r\mathbf{v}_\theta) = \frac{\Gamma}{2\pi}e^{-r^2/2} \tag{5.69}$$

showing that a small cylindrical region of vorticity generates the flow.

As with the Sinusoidal Shear problem, the stretch tensor eigenvalue and rotation matrix converges at second-order. The LIDG algorithm again shows some cost benefits in the Lagrangian case, and the QVR method has a slight advantage in the ALE case.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	3.93E-02	–	3.93E-02	–	3.93E-02	–	3.95E-02	–
32	1.03E-02	1.94	1.03E-02	1.94	1.03E-02	1.94	1.03E-02	1.94
64	2.60E-03	1.98	2.60E-03	1.98	2.60E-03	1.98	2.61E-03	1.98
96	1.16E-03	1.99	1.16E-03	1.99	1.16E-03	1.99	1.17E-03	1.99
128	6.52E-04	2.00	6.52E-04	2.00	6.52E-04	2.00	6.56E-04	2.00

Table 9: Stretch tensor eigenvalue convergence rates for Lagrangian Exponential Vortex problem.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	9.99E-02	–	9.96E-02	–	9.96E-02	–	9.99E-02	–
32	2.72E-02	1.87	2.72E-02	1.87	2.72E-02	1.87	2.72E-02	1.87
64	6.95E-03	1.97	6.93E-03	1.97	6.93E-03	1.97	6.96E-03	1.97
96	3.10E-03	1.99	3.09E-03	1.99	3.09E-03	1.99	3.10E-03	1.99
128	1.75E-03	2.00	1.74E-03	2.00	1.74E-03	2.00	1.75E-03	2.00

Table 10: Rotation tensor matrix convergence rates for Lagrangian Exponential Vortex problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	5.54E-02	–	5.54E-02	–	5.54E-02	–	1.50E-01	–
32	1.07E-02	2.37	1.07E-02	2.37	1.07E-02	2.37	5.73E-02	1.38
64	2.01E-03	2.42	2.01E-03	2.42	2.01E-03	2.42	1.67E-02	1.78
96	7.93E-04	2.30	7.93E-04	2.30	7.93E-04	2.30	7.57E-03	1.95
128	4.10E-04	2.29	4.10E-04	2.29	4.10E-04	2.29	4.33E-03	1.94

Table 11: Stretch tensor eigenvalue convergence rates for ALE Exponential Vortex problem.

Method	CPU Time	Rel. CPU Time
LHWVR	116.91	1.000
LRVR	116.98	1.001
LQVR	116.98	1.001
LIDG	103.93	0.889

Table 12: Relative and actual CPU time for Lagrangian Exponential Vortex problem

Method	CPU Time	Rel. CPU Time
HWVR	184.57	1.000
RVR	182.77	0.990
QVR	179.06	0.970
IDG	187.55	1.016

Table 13: Relative and actual CPU time for ALE Exponential Vortex problem

5.4 Diverging Flow

It is equally important to consider flows with non-trivial volume change. We consider flows with constant non-zero divergence in both cylindrical ($d = 2$) and spherical ($d = 3$) geometry .

$$\mathbf{v}_r = \alpha r/d \tag{5.70}$$

where α is the divergence which will be recalled represents the logarithm derivative of density. The flow density will change exponentially in time. The general solution is

$$\rho(r, t) = \rho(re^{-\alpha t/d}, 0)e^{-\alpha t} \tag{5.71}$$

where $\rho(r, 0)$ is the initial density. In addition, $\mathbf{x} = \mathbf{a}e^{\alpha t/d}$ so that $\mathbf{F} = e^{\alpha t/d}\mathbf{I}$ and $\det \mathbf{F} = e^{\alpha t}$. This test case is able to discriminate issues related to divergent flow.

Stretch tensor convergence rates are shown in Tables 14 and 15. Rotation tensor convegence rates are not included for this test problem because exact results were found using every method. In addition, from the tables it can be seen that the IDG and LIDG methods yield exact results in the stretch tensor while the other methods demonstrate second-order convergence. This is expected due to the linear nature of the velocity profile. As for cost, there seems to be no significant difference between the different methods in the Lagrangian case, but IDG continues to exhibit a disadvantage in the ALE case.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	7.89E-04	–	7.89E-04	–	7.89E-04	–	4.44E-16	–
32	2.31E-04	1.77	2.31E-04	1.77	2.31E-04	1.77	1.11E-15	NA
64	6.29E-05	1.87	6.29E-05	1.87	6.29E-05	1.87	2.66E-15	NA
96	2.86E-05	1.94	2.86E-05	1.94	2.86E-05	1.94	1.56E-14	NA
128	1.62E-05	1.97	1.62E-05	1.97	1.62E-05	1.97	5.77E-15	NA

Table 14: Stretch tensor eigenvalue convergence rates for Lagrangian Diverging Flow problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	2.57E-04	–	2.57E-04	–	2.57E-04	–	2.74E-15	–
32	7.44E-05	1.79	7.44E-05	1.79	7.44E-05	1.79	3.95E-15	NA
64	2.00E-05	1.89	2.00E-05	1.89	2.00E-05	1.89	7.28E-15	NA
96	8.58E-06	2.09	8.58E-06	2.09	8.58E-06	2.09	5.22E-14	NA
128	4.77E-06	2.04	4.77E-06	2.04	4.77E-06	2.04	1.18E-14	NA

Table 15: Stretch tensor eigenvalue convergence rates for ALE Diverging Flow problem.

Method	CPU Time	Rel. CPU Time
LHWVR	133.04	1.000
LRVR	133.66	1.005
LQVR	131.84	0.991
LIDG	135.61	1.019

Table 16: Relative and actual CPU time for Lagrangian Diverging Flow problem

Method	CPU Time	Rel. CPU Time
HWVR	808.47	1.000
RVR	812.77	1.005
QVR	790.02	0.977
IDG	912.10	1.128

Table 17: Relative and actual CPU time for ALE Diverging Flow problem

5.5 ABC Vortical Flow

We now consider a flow field with discontinuities in the vorticity field (i.e. discontinuous velocity gradients). Consider a simple incompressible flow in which tangential velocity discontinuities are allowed. A cylindrical block rotates at constant angular velocity, surrounded by an irrotational circular flow region with constant circulation, surrounded finally by a flow with negative vorticity until the flow drops to a zero velocity. The equations for the flow are

$$\mathbf{v}_\theta = \omega_0 r \ \ 0 < r < a \tag{5.72}$$

$$\mathbf{v}_\theta = \frac{\omega_0 a^2}{r} \ \ a < r < b \tag{5.73}$$

$$\mathbf{v}_\theta = \frac{\omega_0 a^2}{r}\left(\frac{c^2 - r^2}{c^2 - b^2}\right) \ \ b < r < c \tag{5.74}$$

$$\mathbf{v}_\theta = 0 \ \ c < r \tag{5.75}$$

This problem provides a region of pure local rotation, a region of pure local stretch, and a mixed region with both stretch and rotation, as well as a quiescent region. As with the exponential vortex, the solution can be obtained explicitly in polar coordinates and the motion and deformation gradient calculated explicitly.

Convergence rates for the Lagrangian methods are shown in Tables 18 and 19. In this case we get approximately first-order accuracy as might be expected.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	2.39E-01	–	2.39E-01	–	2.39E-01	–	2.39E-01	–
32	1.37E-01	0.80	1.37E-01	0.80	1.37E-01	0.80	1.37E-01	0.80
64	6.03E-02	1.18	6.03E-02	1.18	6.03E-02	1.18	6.03E-02	1.18
96	3.58E-02	1.29	3.58E-02	1.29	3.58E-02	1.29	3.58E-02	1.29
128	2.38E-02	1.42	2.38E-02	1.42	2.38E-02	1.42	2.38E-02	1.42

Table 18: Stretch tensor eigenvalue convergence rates for Lagrangian ABC Vortical Flow problem.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	3.93E-01	–	3.93E-01	–	3.93E-01	–	3.93E-01	–
32	2.80E-01	0.49	2.80E-01	0.49	2.80E-01	0.49	2.80E-01	0.49
64	1.62E-01	0.79	1.62E-01	0.79	1.62E-01	0.79	1.62E-01	0.79
96	1.02E-01	1.13	1.02E-01	1.13	1.02E-01	1.13	1.02E-01	1.13
128	8.17E-02	0.78	8.17E-02	0.78	8.17E-02	0.78	8.17E-02	0.78

Table 19: Rotation tensor matrix convergence rates for Lagrangian ABC Vortical Flow problem.

Convergence rates for the ALE methods are shown in Tables 20 and 21. It appears that in this ALE case that none of the algorithms are converging. It seems that the gradient jumps in the solution are not well-approximated and pollute the solution.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	1.77E-01	–	1.77E-01	–	1.77E-01	–	2.58E-01	–
32	1.24E-01	0.52	1.24E-01	0.52	1.24E-01	0.52	1.81E-01	0.51
64	7.05E-02	0.81	7.05E-02	0.81	7.05E-02	0.81	1.16E-01	0.64
96	5.01E-02	0.84	5.01E-02	0.84	5.01E-02	0.84	8.73E-02	0.70
128	3.83E-02	0.94	3.83E-02	0.94	3.83E-02	0.94	7.14E-02	0.70

Table 20: Stretch tensor eigenvalue convergence rates for ALE ABC Vortical Flow problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
16	3.76E-01	–	3.76E-01	–	3.70E-01	–	4.46E-01	–
32	1.49E-01	1.34	1.49E-01	1.34	1.42E-01	1.39	2.29E-01	0.96
64	9.97E-02	0.58	9.96E-02	0.58	9.32E-02	0.60	1.58E-01	0.53
96	8.66E-02	0.35	8.66E-02	0.35	8.19E-02	0.32	1.36E-01	0.37
128	8.43E-02	0.09	8.43E-02	0.09	8.05E-02	0.06	1.32E-01	0.10

Table 21: Rotation tensor matrix convergence rates for ALE ABC Vortical Flow problem.

Method	CPU Time	Rel. CPU Time
LHWVR	136.10	1.000
LRVR	136.03	1.000
LQVR	134.64	0.989
LIDG	118.90	0.874

Table 22: Relative and actual CPU time for Lagrangian ABC Vortical Flow problem

Method	CPU Time	Rel. CPU Time
HWVR	109.96	1.000
RVR	110.04	1.001
QVR	106.72	0.971
IDG	115.37	1.049

Table 23: Relative and actual CPU time for ALE ABC Vortical Flow problem

6 Results for Given 3D Motions

We now examine two of the previous described test problems in the context of a 3D simulation. The 3D meshes are Cartesian and coordinate aligned but the axis of rotation is misaligned and in the direction (0.5,0.3,0.8). We also test each problem with the axis of rotation aligned to each of the three axes, but the results are not included here. We run the problem just long enough to capture an estimate of the convergence rate.

6.1 3D Exponential Vortex

From Tables 24 to 27 it is apparent that second-order convergence is also achieved in the 3D version of the Exponential Vortex problem. Additionally, it can be seen that the LIDG algorithm has a large cost advantantage in the Lagrangian case, and the QVR algorithm is the fastest in the ALE case.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	1.58E-01	–	1.58E-01	–	1.58E-01	–	1.58E-01	–
8	8.12E-02	0.96	8.12E-02	0.96	8.12E-02	0.96	8.12E-02	0.96
16	2.55E-02	1.67	2.55E-02	1.67	2.55E-02	1.67	2.52E-02	1.69
32	6.88E-03	1.89	6.88E-03	1.89	6.88E-03	1.89	6.75E-03	1.90
64	1.75E-03	1.98	1.75E-03	1.98	1.75E-03	1.98	1.71E-03	1.98

Table 24: Stretch tensor eigenvalue convergence rates for 3D Lagrangian Exponential Vortex problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	1.82E-01	–	1.82E-01	–	1.82E-01	–	1.85E-01	–
8	1.05E-01	0.80	1.05E-01	0.80	1.05E-01	0.80	1.27E-01	0.55
16	2.93E-02	1.84	2.93E-02	1.84	2.93E-02	1.84	6.65E-02	0.93
32	6.15E-03	2.25	6.15E-03	2.25	6.15E-03	2.25	2.91E-02	1.19
64	1.24E-03	2.31	1.24E-03	2.31	1.24E-03	2.31	8.58E-03	1.76

Table 25: Stretch tensor eigenvalue convergence rates for 3D ALE Exponential Vortex problem.

.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	3.16E-01	–	3.16E-01	–	3.16E-01	–	3.17E-01	–
8	1.26E-01	1.32	1.26E-01	1.32	1.26E-01	1.33	1.28E-01	1.31
16	3.64E-02	1.79	3.64E-02	1.79	3.62E-02	1.80	3.68E-02	1.79
32	9.40E-03	1.95	9.40E-03	1.95	9.32E-03	1.96	9.51E-03	1.95
64	2.37E-03	1.99	2.37E-03	1.99	2.34E-03	1.99	2.39E-03	1.99

Table 26: Rotation tensor matrix convergence rates for 3D Lagrangian Exponential Vortex problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	3.47E-01	–	3.47E-01	–	3.48E-01	–	3.92E-01	–
8	1.76E-01	0.98	1.75E-01	0.98	1.72E-01	1.02	2.62E-01	0.58
16	5.87E-02	1.58	5.84E-02	1.59	5.33E-02	1.69	1.30E-01	1.01
32	1.36E-02	2.11	1.34E-02	2.12	1.22E-02	2.13	4.58E-02	1.51
64	2.93E-03	2.21	2.89E-03	2.22	2.71E-03	2.17	1.32E-02	1.79

Table 27: Rotation tensor matrix convergence rates for 3D ALE Exponential Vortex problem.

Method	CPU Time	Rel. CPU Time
LHWVR	1870.25	1.000
LRVR	1870.97	1.000
LQVR	1871.89	1.001
LIDG	1693.54	0.906

Table 28: Relative and actual CPU time for 3D Lagrangian Exponential Vortex problem

Method	CPU Time	Rel. CPU Time
HWVR	4220.42	1.000
RVR	4219.98	1.000
QVR	4024.52	0.954
IDG	4364.78	1.034

Table 29: Relative and actual CPU time for 3D ALE Exponential Vortex problem

6.2 3D ABC Vortical Flow

It will be seen that convergence rates for this problem appear to be approaching first-order for the Lagrangian cases. Something clearly less than first order is in evidence for the ALE results. This is consistent with the non-smooth nature of the imposed motion.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	7.76E-02	–	7.76E-02	–	7.76E-02	–	7.77E-02	–
8	5.99E-02	0.37	5.99E-02	0.37	5.99E-02	0.37	5.99E-02	0.38
16	2.90E-02	1.05	2.90E-02	1.05	2.90E-02	1.05	2.90E-02	1.05
32	1.50E-02	0.95	1.50E-02	0.95	1.50E-02	0.95	1.50E-02	0.95
64	7.19E-03	1.06	7.19E-03	1.06	7.19E-03	1.06	7.20E-03	1.06

Table 30: Stretch tensor eigenvalue convergence rates for 3D Lagrangian ABC Vortical Flow problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	7.75E-02	–	7.75E-02	–	7.75E-02	–	7.89E-02	–
8	6.11E-02	0.34	6.11E-02	0.34	6.11E-02	0.34	6.19E-02	0.35
16	3.06E-02	1.00	3.06E-02	1.00	3.06E-02	1.00	3.26E-02	0.93
32	1.68E-02	0.86	1.68E-02	0.86	1.68E-02	0.86	1.90E-02	0.78
64	8.78E-03	0.94	8.78E-03	0.94	8.78E-03	0.94	1.09E-02	0.81

Table 31: Stretch tensor eigenvalue convergence rates for 3D ALE ABC Vortical Flow problem.

	LHWVR		LRVR		LQVR		LIDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	1.39E-01	–	1.39E-01	–	1.39E-01	–	1.39E-01	–
8	6.83E-02	1.02	6.83E-02	1.02	6.83E-02	1.02	6.83E-02	1.02
16	5.19E-02	0.40	5.19E-02	0.40	5.19E-02	0.40	5.19E-02	0.40
32	3.56E-02	0.54	3.56E-02	0.54	3.56E-02	0.54	3.56E-02	0.54
64	2.46E-02	0.54	2.46E-02	0.54	2.46E-02	0.54	2.46E-02	0.54

Table 32: Rotation tensor matrix convergence rates for 3D Lagrangian ABC Vortical Flow problem.

	HWVR		RVR		QVR		IDG	
1/h	L^1	Order	L^1	Order	L^1	Order	L^1	Order
4	1.39E-01	–	1.39E-01	–	1.39E-01	–	1.41E-01	–
8	6.93E-02	1.00	6.93E-02	1.00	6.93E-02	1.00	7.18E-02	0.97
16	5.31E-02	0.39	5.31E-02	0.39	5.30E-02	0.39	5.64E-02	0.35
32	3.66E-02	0.53	3.66E-02	0.53	3.66E-02	0.53	4.05E-02	0.48
64	2.64E-02	0.47	2.64E-02	0.47	2.64E-02	0.47	3.07E-02	0.40

Table 33: Rotation tensor matrix convergence rates for 3D ALE ABC Vortical Flow problem.

Method	CPU Time	Rel. CPU Time
LHWVR	1149.51	1.000
LRVR	1143.12	0.994
LQVR	1150.23	1.001
LIDG	1049.94	0.913

Table 34: Relative and actual CPU time for 3D Lagrangian ABC Vortical Flow problem

Method	CPU Time	Rel. CPU Time
HWVR	2459.75	1.000
RVR	2460.67	1.000
QVR	2400.43	0.976
IDG	2668.27	1.085

Table 35: Relative and actual CPU time for 3D ALE ABC Vortical Flow problem

7 A Three Dimensional Impact Problem

Next we compare these various approaches on a more realistic 3D calculation involving the impact of a penetrator into a target. Due to the nature of the problem, only the ALE case is studied. Lagrangian methods do not prove to be sufficiently robust. Furthermore, we do not show the constrained transport approach (IDG) because it is not fully implemented and working for multiple material calculations and we have not resolved the issue of maintaining positivity of the inverse deformation gradient. Since the current single material implementation is significantly more expensive this algorithm has not been pursued further.

Table 36 shows time comparisons for the HWVR, RVR, and QVR approaches. In addition, first-order and second-order time stepping schemes have been implemented with each approach and the two methods that utilize a tensor rotation representation (HWVR and RVR) both show results with the iterative polar decomposition (Iterative) and the direct polar decomposition (Direct) methods for computating an updated rotation tensor after component by component remap.

Method	Rotation Projection	VR Update Time Order	Time(s)	Relative Time	Steps	μs/el/step
HWVR	Iterative	1st	166.82	1.0	446	36.528
HWVR	Iterative	2nd	170.05	1.019	456	36.418
HWVR	Direct	1st	166.78	1.0	446	36.518
HWVR	Direct	2nd	171.49	1.028	456	36.726
RVR	Iterative	1st	167.29	1.003	447	36.547
RVR	Iterative	2nd	167.28	1.003	447	36.545
RVR	Direct	1st	166.08	0.996	447	36.284
RVR	Direct	2nd	171.87	1.03	459	36.567
QVR	UQuat	1st	172.02	1.031	465	36.127
QVR	UQuat	2nd	170.76	1.024	459	36.331

Table 36: 3D impact test with varying update methods and representations for $\lambda_s = 10^{-5}$.

We observe from this table that using a quaternion representation for the rotation tensor yields a slight computational advantage in terms of computational cost per cycle. The extra cost for using the second-order time integration VR methodology is barely visible. However, for some reason in this test case we observe additional cycles indicating some sort of feedback which has lowered the time step for this test problem which leads to no real net benefit.

We can also optionally compute the midstep rotation tensor using the exponential map interpolation methodology outlined in this report for purposes of allowing hypo-elastic models to be second-order in time. The possibility was mentioned in a previous report.[7] However, we will not investigate the utility of this approach here as it is intimately involved with the time accuracy of general non-linear elastic-plastic hypo-elastic models and this subject deserves separate attention.[26]

8 Conclusions

We have characterized various approaches for tracking information associated with the kinematics of the deformation of solid materials when explicit rotation tensors are required. These approaches are applicable for use in Lagrangian/Eulerian approaches in which a full kinematic description is required. Examination of the various methods shows that the updated stretch and rotation method with a quaternion representation for the rotation appears to have the advantage at this time. Direct polar decomposition approaches with constrained transport remapping appear to be viable in principle but the current implementation is not competitive with the updated polar decomposition approaches since issues with efficiency, multi-material management and maintenance of the positivity of the inverse deformation gradient have not yet been resolved.

References

[1] David Baraff. An introduction to physically based modeling: Rigid body simulation I and II. Online Siggraph '97 Course Notes. http://www.cs.brown.edu/courses/cs224/papers/baraff97notes1.pdf, 1997.

[2] P. B. Bochev, J. J. Hu, A. C. Robinson, and R. S. Tuminaro. Towards robust 3D Z-pinch simulations: discretization and fast solvers for magnetic diffusion in heterogeneous conductors. *Electronic Transactions on Numerical Analysis*, 15:186–210, 2003.

[3] Pavel Bochev and Mikhail Shashkov. Constrained interpolation (remap) of divergence-free fields. *Computer Methods Appl. Mech. Engrg.*, 194:511–530, 2005.

[4] R. M. Brannon. Rotation: A review of useful theorems involving proper orthogonal marices referenced to three-dimensional physical space. Unpublished document available at http://www.mech.utah.edu/~brannon/gobag.html, February 2009.

[5] J. K. Dienes. On the analysis of rotation and stress rate in deforming bodies. *Acta Mechanica*, 32:217–232, 1979.

[6] Charles R. Evans and John F. Hawley. Simulation of magnetohydrodynamic flows: a constrained transport method. *The Astrophysical Journal*, 332:659–677, September 1988.

[7] Grant V. Farnsworth and Allen C. Robinson. Improved kinematic options in ALEGRA. Technical Report SAND2003-4510, Sandia National Laboratories, Albuquerque, New Mexico, December 2003.

[8] D. P. Flanagan and L. M. Taylor. An accurate numerical algorithm for stress integration with finite rotations. *Computer Methods in Applied Mechanical Engineering*, 62:305–320, 1987.

[9] J. N. Franklin. *Matrix Theory*. Prentice-Hall, 1968.

[10] James G. Glimm, Bradley J. Plohr, and David H. Sharp. A conservative formulation for large-deformation plasticity. *Applied Mech. Rev.*, 46:519–526, 1993.

[11] I. N. Herstein. *Topics in Algebra*. John Wiley & Sons, New York, 1975.

[12] Nicholas J. Higham. Computing the polar decomposition - with applications. *SIAM Journal Scientific and Statistical Computing*, 7(4):1160–1174, October 1986.

[13] Nicholas J. Higham and Robert S. Schreiber. Fast polar decomposition of an arbitrary matrix. *SIAM Journal Scientific and Statistical Computing*, 11(4):648–655, July 1990.

[14] Thomas J. R. Hughes and James Winget. Finite rotation effects in numerical integration of rate constitutuve equations arising in large-deformation analysis. *Computer Methods in Applied Mechanical Engineering*, 15(12):1862–1867, 1980.

[15] Charles Kenney and Alan J. Laub. On scaling Newton's method for polar decompositions and the matrix sign function. *SIAM Journal Matrix Analysis*, 13(3):688–706, July 1992.

[16] X. Lin and J. Ballmann. A Riemann solver and a second-order Godunov method for elastic-plastic wave propagation in solids. *International Journal of Impact Engineering*, 13(3):463–478, 1993.

[17] P. Londrillo and L. Del Zanna. High order upwind schemes for multidimensional magnetohydrodynamics. *The Astrophysical Journal*, 530:508–524, 2000.

[18] G. H. Miller and P. Colella. A high-order Eulerian Godunov method for elastic-plastic flow in solids. *Journal of Computational Physics*, 167:131–176, 2001.

[19] G. H. Miller and P. Colella. A conservative three-dimensional Eulerian method for coupled solid-fluid shock capturing. *Journal of Computational Physics*, 183:26–82, 2002.

[20] James S. Peery and Daniel E. Carroll. Multi-material ALE methods in unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 187:591–619, 2000.

[21] Bradley J. Plohr. Mathematical modeling of plasticity in metals. *Mat. Contemp.*, 11:95–120, 1996.

[22] Bradley J. Plohr and David H. Sharp. A conservative Eulerian formulation of the equations for elastic flow. *Advances in Applied Mathematics*, 9:481–499, 1988.

[23] Bradley J. Plohr and David H. Sharp. A conservative formulation for plasticity. *Advances in Applied Mathematics*, 13:462–493, 1992.

[24] M. M. Rashid. Material state remapping in computational solid mechanics. *International Journal for Numerical Methods in Engineering*, 55:431–450, 2002.

[25] William M. Scherzinger and Clark R. Dohrmann. A robust algorithm for finding the eigenvalues and eigenvectors of 3x3 symmetric matrices. *Computer Methods in Applied Mechanics and Engineering*, 197:4007–4015, 2008.

[26] J. C. Simo and S. Govindjee. Non-linear b-stability and symmetry preserving return mapping algorithms for plasticity and viscoplasticity. *International Journal for Numerical Methods in Engineering*, 31:151–176, 1991.

[27] J. C. Simo and T. J. R. Hughes. *Computational Inelasticity.* Springer-Verlag New York, Inc., 1998.

[28] J. C. Simo and K. S. Pister. Remarks on rate constitutive equations for finite deformation problems: Computational implications. *Computer Methods in Applied Mechanics and Engineering*, 46:201–215, 1984.

[29] John A. Trangenstein. Adaptive mesh refinement for wave propagation in nonlinear solids. *SIAM Journal Scientific Computing*, 16(4):819–839, July 1995.

[30] John A. Trangenstein and Phillip Colella. A higher-order Godunov method for modeling finite deformation in elastic-plastic solids. *Communications on Pure and Applied Mathematics*, 44:41–100, 1991.

[31] John A. Trangenstein and Richard B. Pember. The Riemann problem for longitudinal motion in an elastic-plastic bar. *SIAM Journal Scientific and Statistical Computing*, 12(1):180–207, 1991.

[32] John A. Trangenstein and Richard B. Pember. Numerical algorithms for strong discontinuities in elastic-plastic solids. *Journal Computational Physics*, 103:63, 1992.

[33] H. S. Udaykumar, L. Tran, D. M. Belk, and K. J. Vanden. An Eulerian method for computation of multimaterial impact with ENO shock-capturing and sharp interfaces. *Journal of Computational Physics*, 186:136–177, 2003.

[34] John Walter, James Glimm, John Grove, Hyun-Cheol Hwang, Xiao Lin, Brakely J. Plohr, David H. Sharp, and Dahi Yu. Eulerian front tracking for solid dynamics. In K. Iyer and S. Chou, editors, *Proc. 15th U. S. Army Symposium on Solid Mechanics*, pages 343–366. Battelle Press, 1999.

[35] John Walter, Dahai Yu, Bradley J. Plohr, John Grove, and James Glimm. An algorithm for Eulerian front tracking for solid deformation. Technical Report SUNY SB-AMS-00-24, State University of New York, 2000.

[36] Feng Wang, James Glimm, and Bradley J. Plohr. A model for rate-dependent plasticity. *J. Mech. Phys. Solids*, 43(9):1497–1503, 1995.

[37] Feng Wang, James G. Glimm, John W. Grove, Bradley J. Plohr, and David H. Sharp. A conservative Eulerian numerical scheme for elasto-plasticity and application to plate impact problems. *Impact Computer. Sci. Engrg.*, 5:285–308, 1993.

[38] Eric W. Weisstein. Quaternion. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Quaternion.html, 1999 CRC Press LLC, 1999-2004 Wolfram Research, Inc.

A Alternative Method For Calculating Exact Solutions

It is possible to compute an arbitrary motion and the associated deformation gradient associated with the specification of a given velocity field, $\mathbf{v}(\mathbf{x}, t)$. This problem can be solved in general by integrating the set of ordinary differential equations (ODE)

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v}(\mathbf{x}, t) && \text{(A.76)} \\ \dot{\mathbf{F}} &= \mathbf{L}(\mathbf{x}, t)\mathbf{F} && \text{(A.77)} \\ \mathbf{x}(0) &= \mathbf{a} && \text{(A.78)} \\ \mathbf{F}(0) &= \mathbf{I} && \text{(A.79)} \end{aligned}$$

for any initial Lagrangian position $\mathbf{a}$ and $\mathbf{L} = \nabla\mathbf{v}$. If this set of ODEs can be solved exactly, $\mathbf{F}(\mathbf{x}, t)$ can be written explicitly which in general is preferable.

If the solution cannot be obtained exactly, it may be obtained numerically using an ODE solver as follows. Assume we desire the solution at a point $\hat{\mathbf{x}}$ at times $0 = T_0 < T_1 < \cdots < T_n$. We apply a Newton iteration on $\mathbf{a}$ for a given $\hat{\mathbf{x}}$ using

$$\mathbf{a}_n^{k+1} = \mathbf{a}_n^k - \mathbf{F}^{-1}(\mathbf{a}_n^k)(\mathbf{x}(\mathbf{a}_n^k) - \hat{\mathbf{x}}), \quad k = 0, 1, \cdots, K_n \qquad \text{(A.80)}$$

where $\mathbf{F}$ and $\mathbf{x}$ are obtained by numerical solution of Equations A.76 to A.79 to time T_n. The subscript refers to the time level and the superscript to the iteration level. K_n is the number of iterations to achieve convergence. The T_0 solution is, of course, $\mathbf{a}_0^k = \hat{\mathbf{x}}$, and thereafter we use as initial guesses for the Newton iteration $\mathbf{a}_n^0 = \mathbf{a}_{n-1}^{K_{n-1}}$ This procedure may be used to compute the "exact" solution corresponding to any desired spatial velocity field provided that sufficient accuracy is imposed on the ODE solver.

DISTRIBUTION:

2 Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94551
Attn: D. Bailey, MS L-016
Attn: R. N. Rieben, MS L-095

2 Los Alamos National Laboratory
Los Alamos, NM 87545
Attn: M. Shashkov, T-5, MS-B284
Attn: J. Kamm, X-3, MS-B259

1 Air Force Research Laboratory
AFRL/RDHP
3550 Aberdeen Ave SE, Bldg 322
Kirtland AFB, NM 87117-5776
Attn: D. J Amdahl

3 U.S. Army Research Laboratory
AMSRL-WT-TA
Aberdeen Proving Ground, MD, 21005-5066
Attn: R. Doney
Attn: B. Leavy
Attn: M. Love

1 R. M Brannon
Department of Mechanical Engineering
University of Utah
50 S. Central Campus Dr.
Salt Lake City, UT 84112

1 M. A. Christon
Dassult Systemes Simulia Corp.
Rising Sun Mills
166 Valley Street
Providence, RI 02909-2499

1 G. H. Miller
Applied Science
1033 Academic Surge
Davis, CA 95616

1 G. Farnsworth
800 Sherman Ave. Apt 2
Evanston, IL 60202

1 D. I. Ketcheson
Department of Applied Mathematics
University of Washington
Box 35240 Seattle, WA 98195-2420

1 MS 0316
C. C. Ober, 1433

1 MS 0370
T. G. Trucano, 1411

1 MS 0372
A. S. Gullerud, 1524

1 MS 0372
S. T. Montgomery, 1524

1 MS 0372
W. M. Scherzinger, 1524

1 MS 0372
J. E. Bishop, 1525

1 MS 0378
T. L. Ames, 1431

1 MS 0378
S. Carroll, 1431

1 MS 0378
R. R. Drake, 1431

1 MS 0378
D. M. Hensinger, 1431

1 MS 0378
M. E. Kipp, 1431

1 MS 0378
D. A. Labreche, 1431

1 MS 0378
E. Love, 1431

1 MS 0378
C. B. Luchini, 1431

1 MS 0378
S. J. Mosso, 1431

1 MS 0378
J. H. Neiderhaus, 1431

1 MS 0378
S. V. Petney, 1431

1 MS 0378
W. J. Rider, 1431

3 MS 0378
A. C. Robinson, 1431

1 MS 0378
G. Scovazzi, 1431

1 MS 0378
C. Siefert, 1431

1 MS 0378
R. M. Summers, 1431

1 MS 0378
V. G. Weirs, 1431

1 MS 0378
T. E. Voth, 1433

1 MS 0378
M. K. Wong, 1431

1 MS 0380
N. Crane, 1542

1 MS 0380
J. D. Hales, 1542

1 MS 0380
M. Heinstein, 1542

1 MS 0380
J. Jung, 1542

1 MS 0380
K. Pierson, 1542

1 MS 0380
B. W. Spencer, 1542

1 MS 0380
J. D. Thomas, 1542

1 MS 0382
M. W. Glass, 1541

1 MS 0384
P. Yarrington, 1550

1 MS 0521
K. H. Brown, 2617

1 MS 0836
S. C. Schumacher, 1516

1 MS 0836
D. A. Crawford, 1541

1 MS 0836
R. G. Schmitt, 1541

1 MS 0847
S. W. Attaway, 1534

1 MS 1189
T. A. Brunner, 1641

1 MS 1189
S. J. Chantrenne, 1641

1 MS 1189
K. R. Cochrane, 1641

1 MS 1189
D. G. Flicker, 1641

1 MS 1189
C. J. Garasi, 1641

1 MS 1189
T. A. Gardiner, 1641

1 MS 1189
T. A. Haill, 1641

1 MS 1189
H. L. Hanshaw, 1641

1 MS 1189
C. S. Kueny, 1641

1 MS 1189
R. W. Lemke, 1641

1 MS 1189
G. T. Sharp, 1641

1 MS 1189
T. K. Mattsson, 1641

1 MS 1189
E. P. Yu, 1641

1 MS 1195
J. P. Davis, 1646

1 MS 1320
P. B. Bochev, 1414

1 MS 1322
J. B. Aidun, 1435

1 MS 1322
J. H. Carpenter, 1435

1 MS 1323
S. Swan, 1431

1 MS 0899
Technical Library, 9536
(electronic copy)