An Edge-Centered Tensor Artificial Viscosity

MultiMat 2015

7th International Conference on Numerical Methods for Multi-Material Fluid Flow

Sept 7, 2015

Doug Miller, Mike Owen

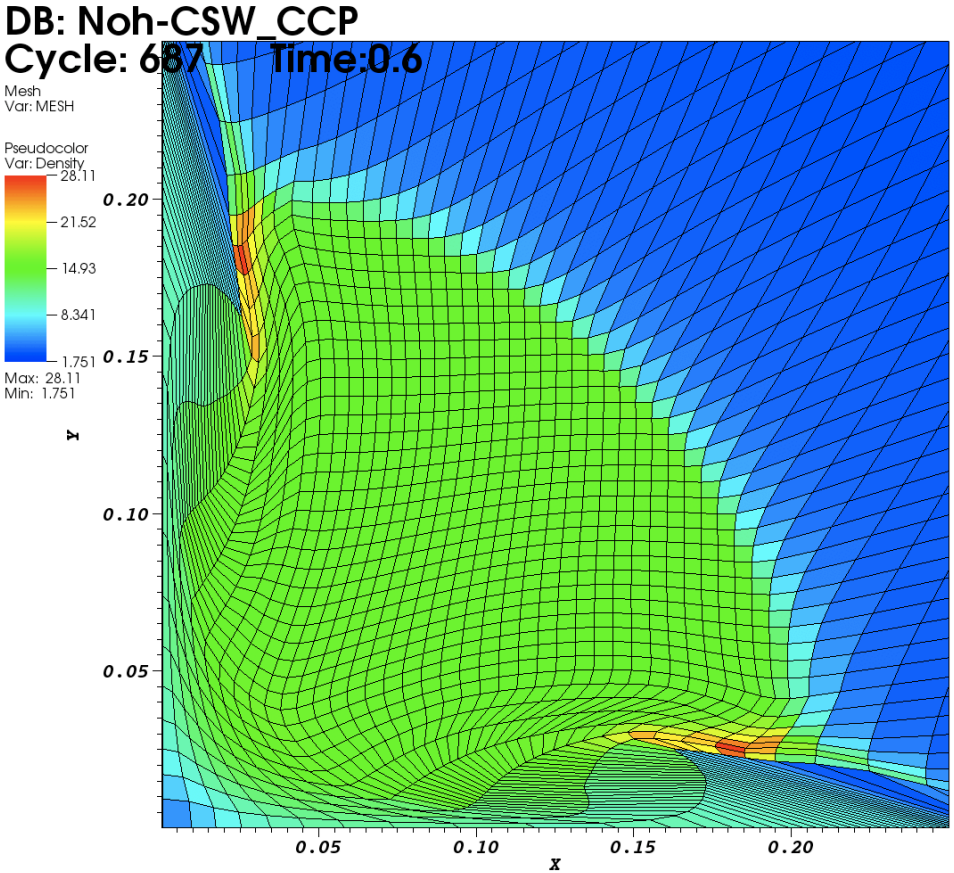
Lawrence Livermore National Laboratory

LLNL-PRES-676809

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

“Q…Q is seductive….you always think you can make it better.” --Ed Caramana

- We are looking for improvements over the Caramana-Shashkov-Whalen[1] edge-centered Q, which KULL has used for several years as part of its staggered-grid hydro.
- This is still a work in progress.



Noh problem on Cartesian grid, CSW edge Q shows issues

We want the following properties:

- Galilean invariant
- Less sensitive to grid topology
- Robust for high aspect ratio zoning
- Low velocity diffusion (don’t push in the non-shock direction)
- Dissipativity (only decrease KE, only increase IE)
- Vanish for rotation, uniform compression, shear
- Vanish smoothly as compression vanishes

Artificial viscosity enters the Euler equations like an additional stress term.

$$\frac{\partial \rho}{\partial t} = -\rho \nabla \bullet \mathbf{u}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho}\left(\nabla P + \nabla \bullet \mathbf{Q}\right)$$

$$\frac{\partial \varepsilon}{\partial t} = -\frac{1}{\rho}\left(P \nabla \bullet \mathbf{u} + \mathbf{Q} : \nabla \mathbf{u}\right)$$

A typical Q has a linear and quadratic part (in $\Delta\mathbf{u}$), and often a multiplying limiter (Ψ).

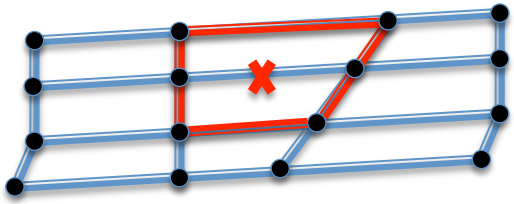
$$\mathbf{Q} = \rho\left(C_{quad}\left(\Delta\mathbf{u}\right)^2 - C_s C_{lin} \Delta\mathbf{u}\right)\Psi$$

In 2011, we presented a Q that we thought had a bright future!

$\mathbf{Q}_e$ is an edge-centered tensor with a scalar length-scale (L), using a symmetrized velocity gradient.

$$\mathbf{Q}_e = \rho\left(C_q L^2 \mathbf{G} \bullet \mathbf{G} - C_s C_l L \mathbf{G}\right)$$

$$\nabla \mathbf{u}_e = \frac{\sum_n^e \mathbf{u}_n \otimes \mathbf{A}_{es}}{V_e}$$



G is the symmetrized velocity gradient, with expanding components removed, used as the velocity jump tensor for **Q**[2].

$$\mathbf{S} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right), \mathbf{A} = \frac{1}{2}\left(\nabla \mathbf{u} - \nabla \mathbf{u}^T\right)$$

$$\mathbf{R} = \begin{pmatrix} \hat{\mathbf{e}}_1 & \hat{\mathbf{e}}_2 \end{pmatrix}, \text{ eigen vectors of } \mathbf{S}$$

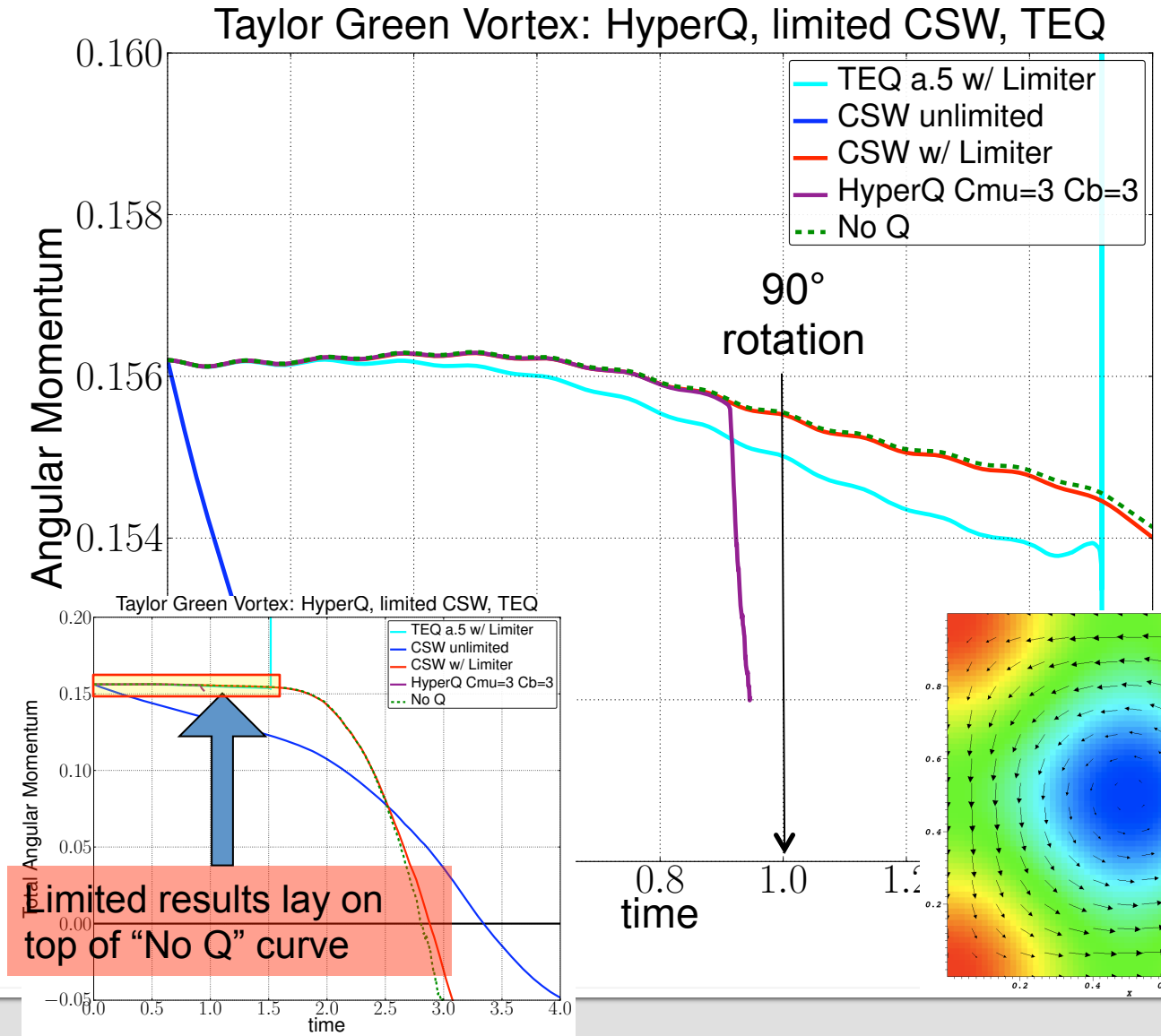
$$\mathbf{D} = \mathbf{R}^{\mathrm{T}}\mathbf{S}\mathbf{R}, \text{ diagonalized form of } \mathbf{S}$$

$$\tilde{\mathbf{D}} = \min(0, \mathbf{D}) \text{ expansion components removed}$$

$$\mathbf{G} = \mathbf{R}\tilde{\mathbf{D}}\mathbf{R}^{\mathrm{T}} + \mathbf{A}, \text{ rotate back to lab frame}$$

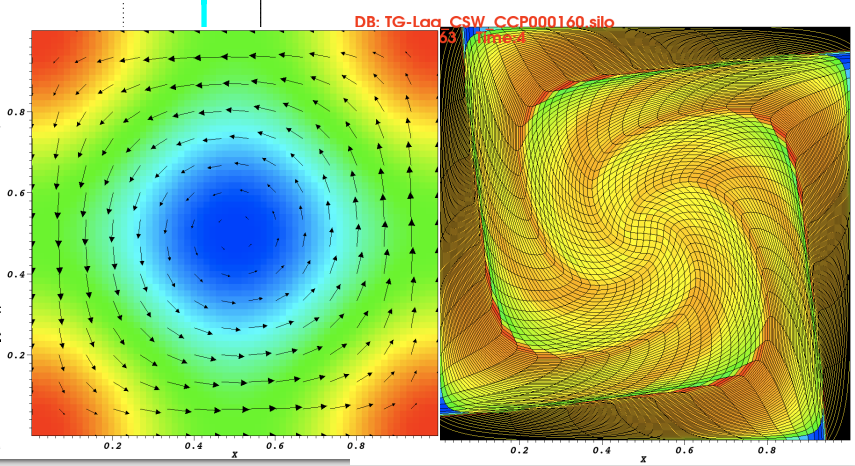
Expanding velocity components are removed via eigenvalue analysis.

Taylor Green Vortex shows the CSW limiter[1] is adequate with tensor edge **Q**.



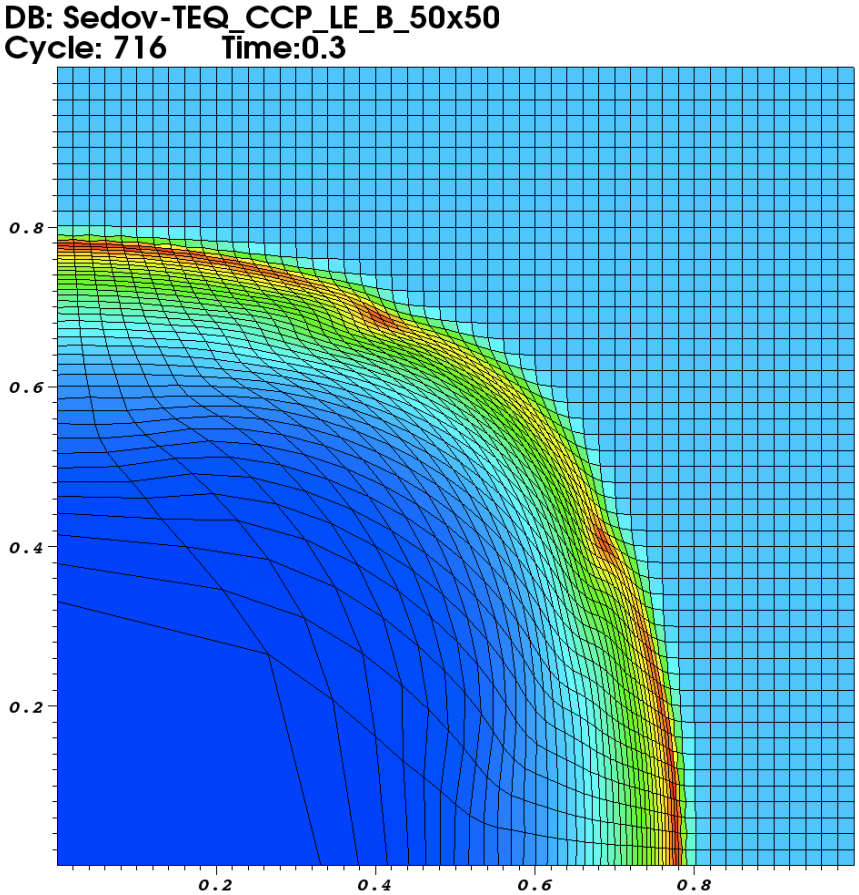
The CSW limiter detects the pure rotation mode and turns the Q off almost perfectly until mesh distortion becomes extreme.

The unlimited CSW and No Q results are shown for comparison.



But in spite of promising early results, nagging doubts remained.

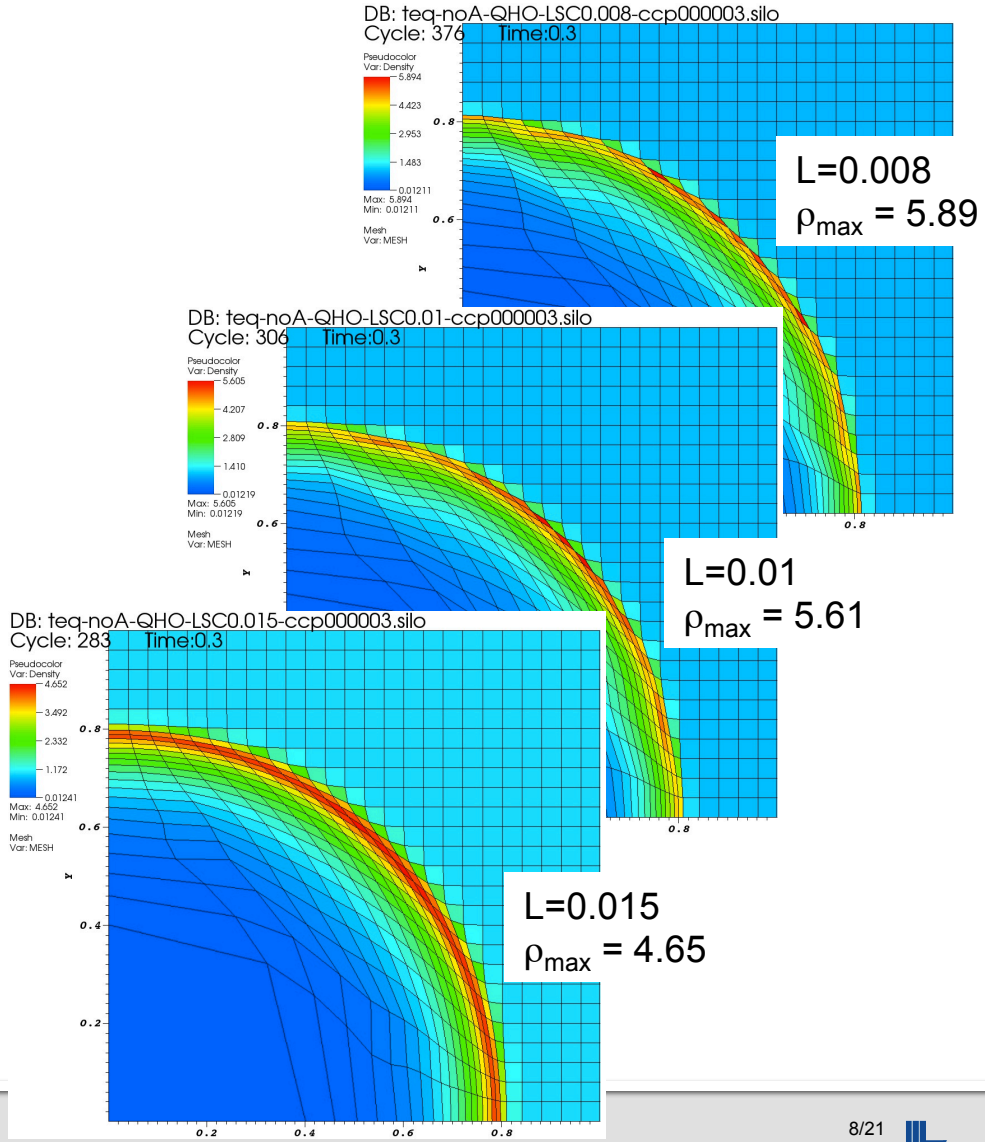
- Problems with Sedov shock shape could only be fixed by using a mixture of velocity gradient schemes[3].
- Issues with high aspect ratio zoning also arose.



Sedov solution using only **G** from previous slide.

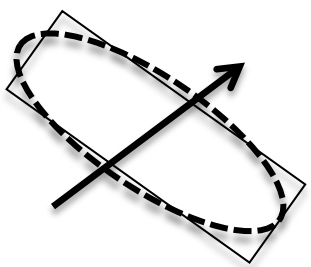
Sedov experiments made it clear our biggest problem was in the length scale.

- Up to now we had been using the mesh edge length as our scale.
- No good for real problems, but trying a constant L value fixed the Sedov shock shape. Shock width and peak density vary along with L.



We tried variations on the scalar length scale, L, which looked promising.

- Projecting the square root of the moment of inertia tensor into the shock direction to get a length is similar to to the scheme of Wilkins[4], (also used in SHALE[5])
- We tried:
  - $L=\sqrt{\mathbf{I}\bullet\hat{\mathbf{s}}\bullet\hat{\mathbf{s}}}$, $\hat{\mathbf{s}}$ is shock direction
  - $L=||\sqrt{\mathbf{I}\bullet\hat{\mathbf{s}}}||$



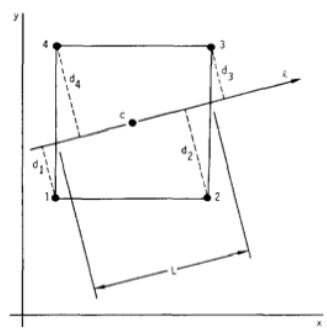
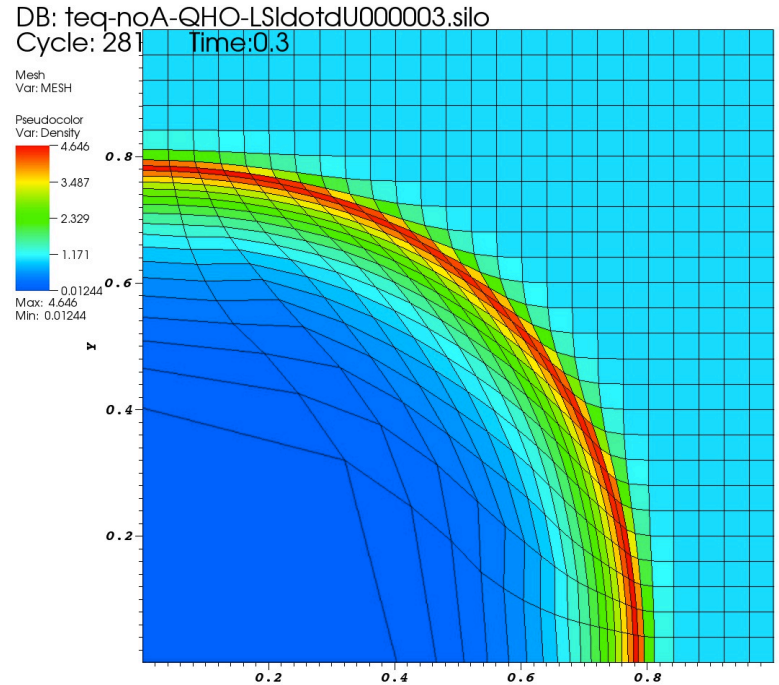


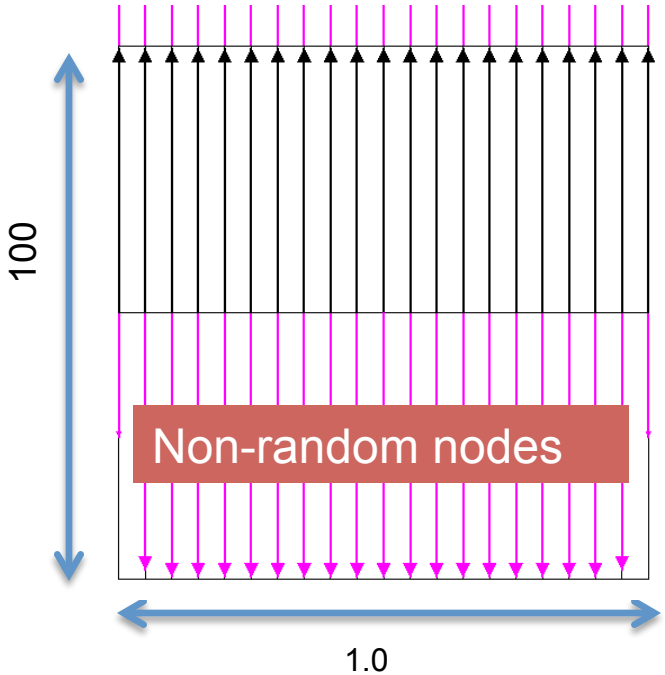
FIG. 6. Scheme for obtaining a characteristic grid length L: l = line through zone center c in direction of acceleration; d_i = perpendicular distance from point i (1, 2, 3, 4) to line l; and $L = 2A/(d_1 + d_2 + d_3 + d_4)$, where A = area of the zone.



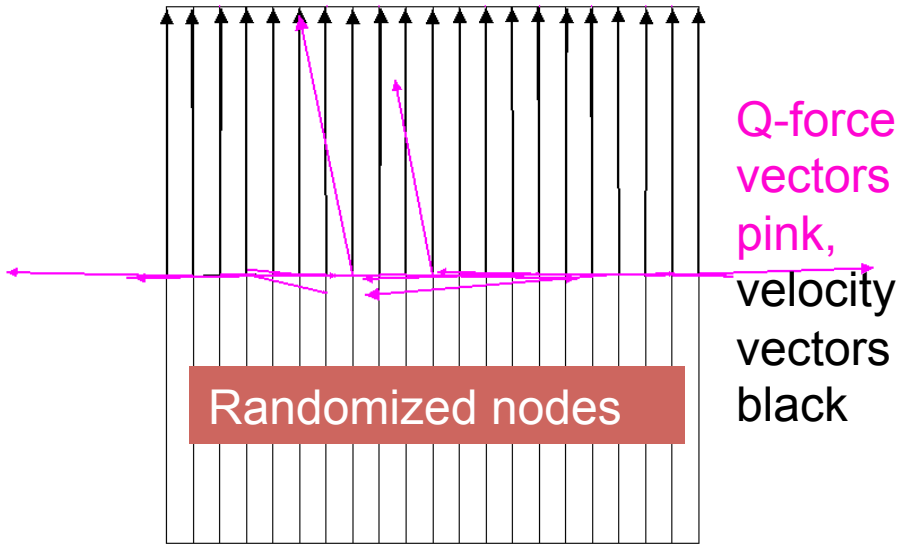
$L=||\sqrt{\mathbf{I}\bullet\hat{\mathbf{s}}}||$ variation, Sedov test

But ultimately, scalar L founders on high aspect ratio zoning.

Velocity jump = (0,1). Randomizing node positions 1% in y-direction eliminates "lucky" cancellations and produces wild changes in **Q** forces.



Q-force **magnitudes ~ 0.8** on non-random mesh



Q-force vectors are pink, velocity vectors are black

Q-force **magnitudes ~ 450** on 1% randomized mesh.

The error was always there, but it is only apparent at high aspect ratio.

Scalar L combined with a full velocity gradient tensor will always have errors in force direction.

$$\mathbf{Q} \sim \rho\left(\Delta\mathbf{u}^2 + c_s \Delta\mathbf{u}\right)$$

If we had a perfect velocity gradient $\mathbf{G}$

$$\mathbf{G} = \begin{pmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} \end{pmatrix}$$

we could use this to get the $\Delta\mathbf{u}$ we want by

$$\Delta\mathbf{u} = \begin{pmatrix} \frac{\partial u_x}{\partial x}\Delta x & \frac{\partial u_x}{\partial y}\Delta y \\ \frac{\partial u_y}{\partial x}\Delta x & \frac{\partial u_y}{\partial y}\Delta y \end{pmatrix}$$

But what we have been doing is

$$\Delta\mathbf{u}^{\text{bad}} = \begin{pmatrix} \frac{\partial u_x}{\partial x}L & \frac{\partial u_x}{\partial y}L \\ \frac{\partial u_x}{\partial y}L & \frac{\partial u_y}{\partial y}L \end{pmatrix}$$

$\Delta\mathbf{u}$ in $\mathbf{Q}$ is wrong, because we are multiplying by a scalar L.

We are making an error for all the components *not* in the direction we picked to be the length scale direction.

The fix is to use a tensor length scale. Don't send a scalar to do a tensor's job!

Truth in advertising---this not a big deal until aspect ratios get beyond ~50. YMMV.

So we tried using **Δu = G L**, where **L** is the square root of the moment of inertia tensor for the zone.

$$\mathbf{Q} \sim \rho\left(\Delta\mathbf{u}^2 + c_s \Delta\mathbf{u}\right)$$

G is the symmetrized velocity gradient, with expanding components removed

$$\mathbf{G} = \begin{pmatrix} \dfrac{\partial u_x}{\partial x} & \dfrac{1}{2}\left(\dfrac{\partial u_x}{\partial y} + \dfrac{\partial u_y}{\partial x}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial u_x}{\partial y} + \dfrac{\partial u_y}{\partial x}\right) & \dfrac{\partial u_y}{\partial y} \end{pmatrix}$$

L is the square root of the moment of inertia tensor

$$\mathbf{L} \approx \frac{1}{N}\sqrt{\begin{pmatrix} \sum_n^z x_n^2 & \sum_n^z x_n y_n \\ \sum_n^z x_n y_n & \sum_n^z y_n^2 \end{pmatrix}}$$

where x_n is x-distance of node n from zone center

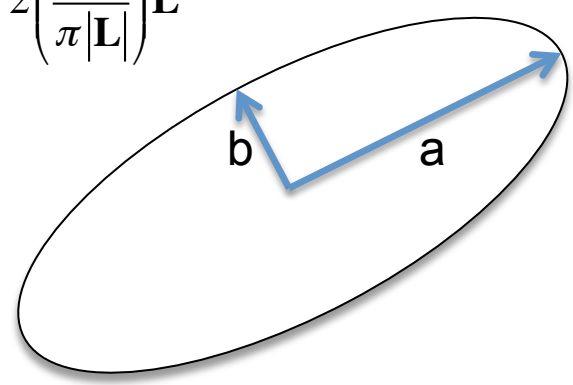
We scale the **L** tensor so that the ellipse it describes has 4x the area of the zone.

$$\mathbf{L} = \sqrt{(\mathbf{I})}$$

area of ellipse, $A_e = \pi ab = \pi|\mathbf{L}|$

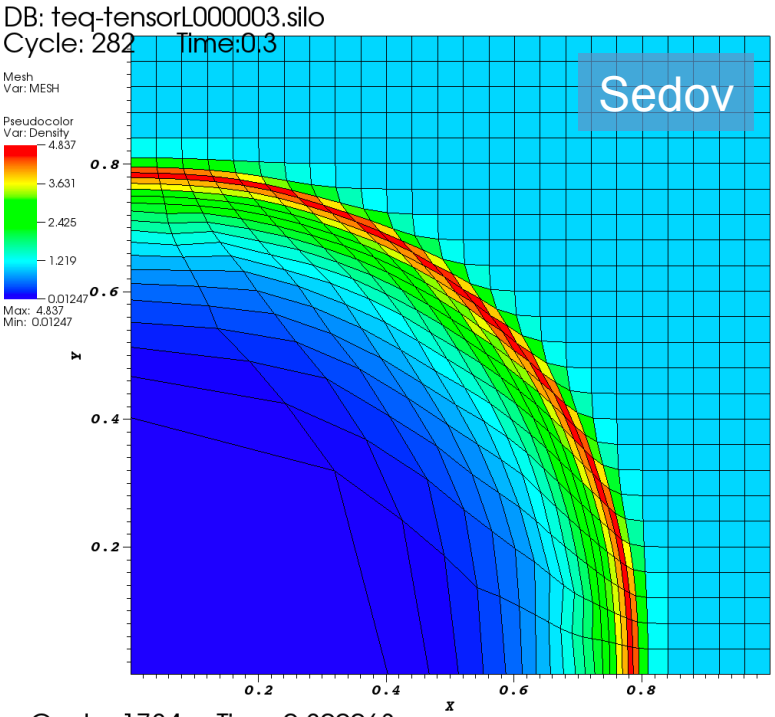
$a = \delta x / 2,\ b = \delta y / 2,$ because a, b are semi-axes

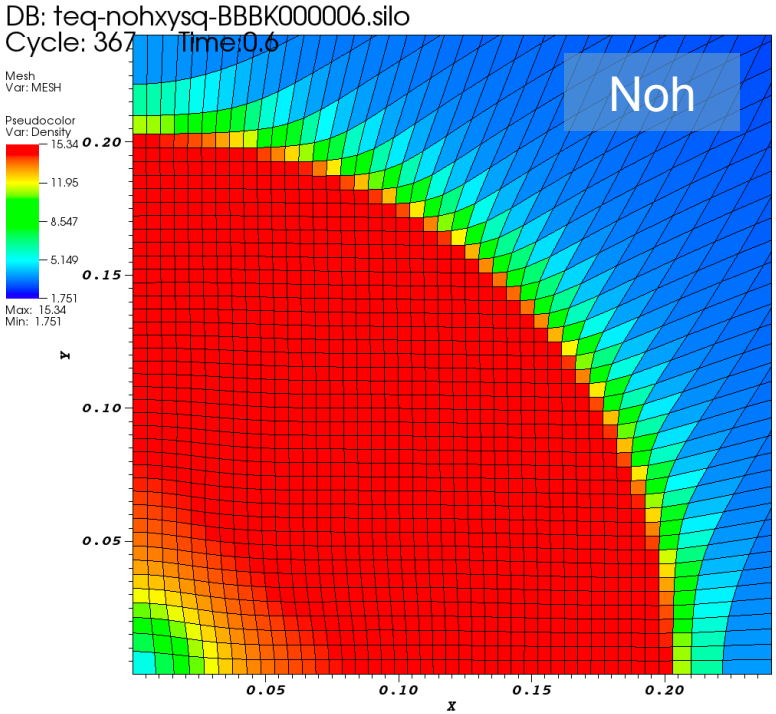
$$\mathbf{L}_z = 2\left(\frac{A_z}{\pi|\mathbf{L}|}\right)\mathbf{L}$$

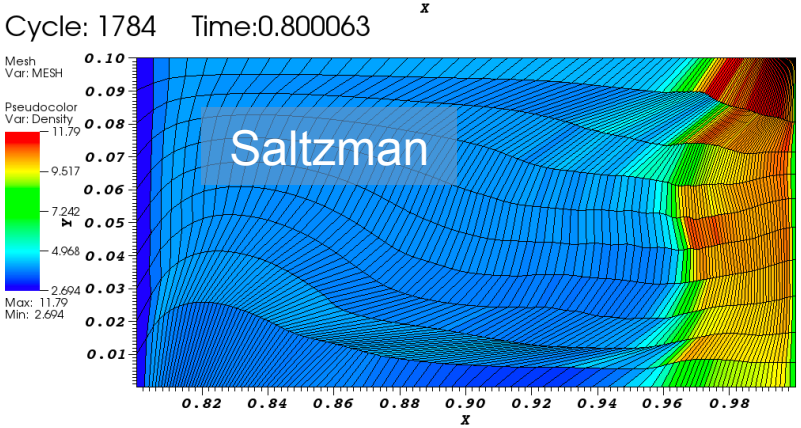


L gives us a shape (2nd moment). Scaling to the zone area captures 0th moment.

Using **L** = √**I** for length scale produced encouraging results.



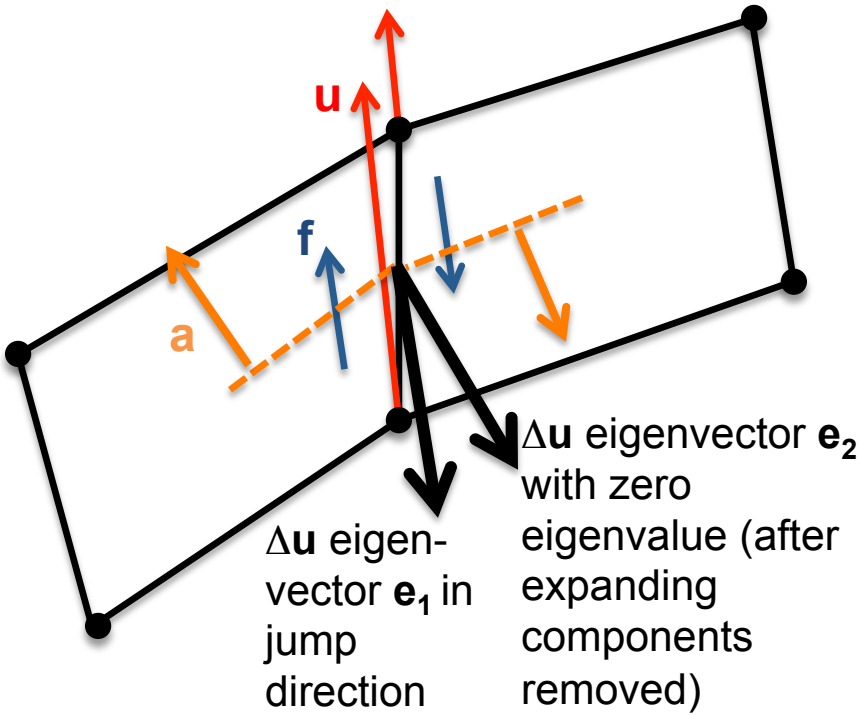




Noh and Sedov look good.
Saltzman, less so.

Once again, however, high aspect ratio zoning exposes issues.

- **G** and **L** are both symmetric, but **Δu=G•L** is not.
- Non-symmetry implies non-perpendicular eigenvectors.
- Recall, we set the eigenvalue associated with the non-compressing direction to zero, to remove expanding velocity components
- Area normal vectors pointing near the zero-eigenvalue's eigenvector direction can rapidly swing the force direction (**f=Q•a**) through 180°.



We can recover symmetry and perpendicular eigenvectors by doing something different: $\Delta\mathbf{u} = \sqrt{\mathbf{L}} \bullet \mathbf{G} \bullet \sqrt{\mathbf{L}}$

$$\sqrt{\mathbf{L}} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$\Delta\mathbf{u} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} G_{xx} & G_{xy} \\ G_{yx} & G_{yy} \end{pmatrix} \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} a^2 G_{xx} + 2abG_{xy} + b^2 G_{yy} & \boxed{abG_{xx} + (ac+bb)G_{xy} + bcG_{yy}} \\ \boxed{abG_{xx} + (ac+bb)G_{xy} + bcG_{yy}} & b^2 G_{xx} + 2bcG_{xy} + c^2 G_{yy} \end{pmatrix}$$

Quick proof the eigenvectors of $\Delta\mathbf{u}$ are orthogonal:

$$\Delta\mathbf{u} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$(a-\lambda)(c-\lambda) - b^2 = 0$$

$$\lambda = \frac{1}{2}\left((a+c) \pm \sqrt{(a-c)^2 + 4b^2}\right), \quad \text{always two real eigenvalues}$$

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ \dfrac{(c-a) + \sqrt{(a-c)^2 + 4b^2}}{2b} \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} 1 \\ \dfrac{(c-a) - \sqrt{(a-c)^2 + 4b^2}}{2b} \end{pmatrix}$$

$$\mathbf{e}_1 \bullet \mathbf{e}_2 = 1 + \frac{1}{4b^2}\left((c-a)^2 - (a-c)^2 - 4b^2\right) = 0, \quad \text{always orthogonal e-vectors}$$

In principal axes frame, we can see

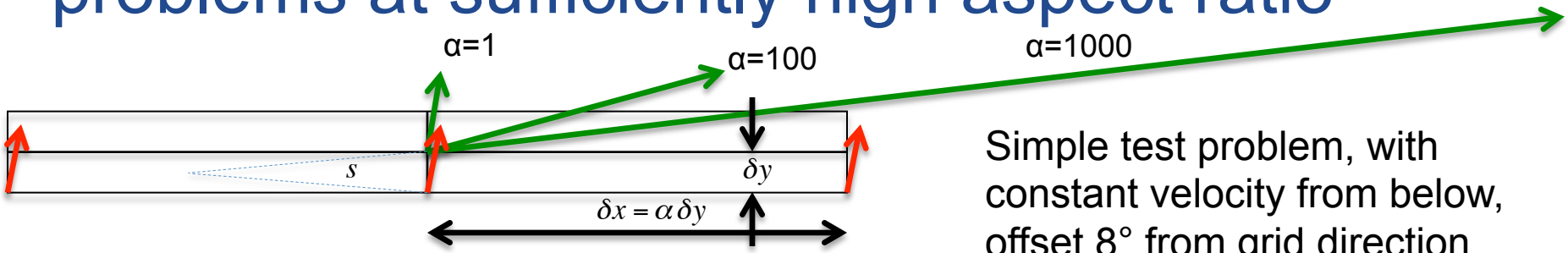
$$\sqrt{\mathbf{L}} = \begin{pmatrix} \sqrt{\delta x} & 0 \\ 0 & \sqrt{\delta y} \end{pmatrix}$$

$$\Delta\mathbf{u} = \begin{pmatrix} \sqrt{\delta x} & 0 \\ 0 & \sqrt{\delta y} \end{pmatrix} \begin{pmatrix} G_{xx} & G_{xy} \\ G_{yx} & G_{yy} \end{pmatrix} \begin{pmatrix} \sqrt{\delta x} & 0 \\ 0 & \sqrt{\delta y} \end{pmatrix}$$

$$\Delta\mathbf{u} = \begin{pmatrix} \delta x\, G_{xx} & \sqrt{\delta x \delta y}\, G_{xy} \\ \sqrt{\delta x \delta y}\, G_{xy} & \delta y\, G_{yy} \end{pmatrix}$$

This form averages length scale multiplying the cross derivative terms, providing some stability at high aspect ratio.

But, even with **Δu = √L•G•√L** we can find problems at sufficiently high aspect ratio



Simple test problem, with constant velocity from below, offset 8° from grid direction (all ∂/∂x are zero).

$$\mathbf{G} = \sqrt{\mathbf{L}}\frac{1}{2}\left(\nabla\mathbf{u}^T + \nabla\mathbf{u}\right)\sqrt{\mathbf{L}} = \begin{pmatrix} \sqrt{\delta x} & 0 \\ 0 & \sqrt{\delta y} \end{pmatrix}\begin{pmatrix} 0 & \frac{1}{2}u_{x,y} \\ \frac{1}{2}u_{x,y} & u_{y,y} \end{pmatrix}\begin{pmatrix} \sqrt{\delta x} & 0 \\ 0 & \sqrt{\delta y} \end{pmatrix}$$

$$\mathbf{Q}_s = \rho\left(C_{quad}\mathbf{G}^2 - C_{lin}C_s\mathbf{G}\right) \qquad \mathbf{f}_n^s = \frac{1}{2}\mathbf{Q}_s \bullet \mathbf{A}_s \qquad \mathbf{a}_n^s = \mathbf{f}_n^s / m_n \qquad m_n = \alpha\,\delta y^2\rho$$

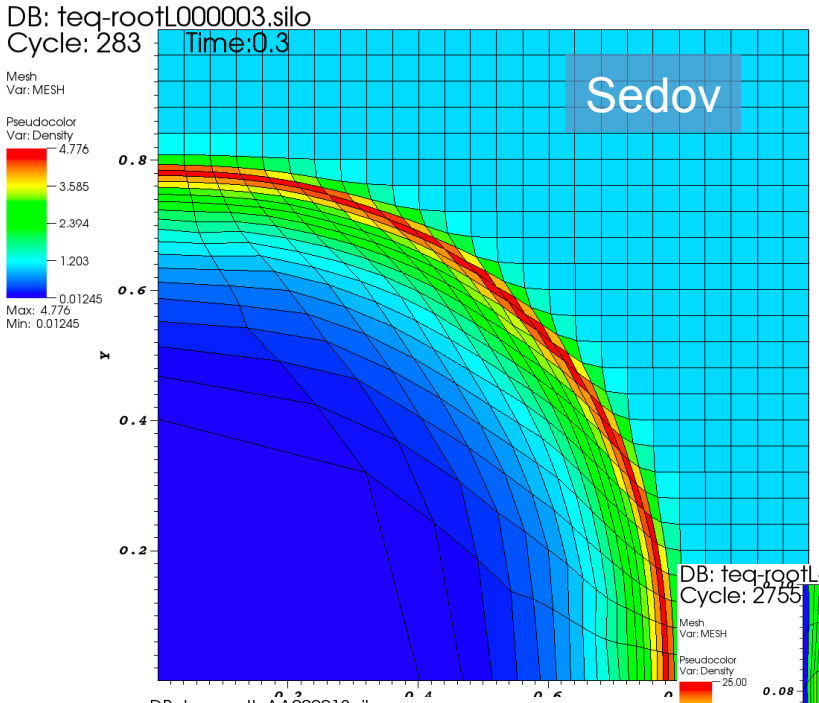
$$\mathbf{A}_s = \begin{pmatrix} 0 \\ \frac{1}{2}\alpha\,\delta y \end{pmatrix}$$

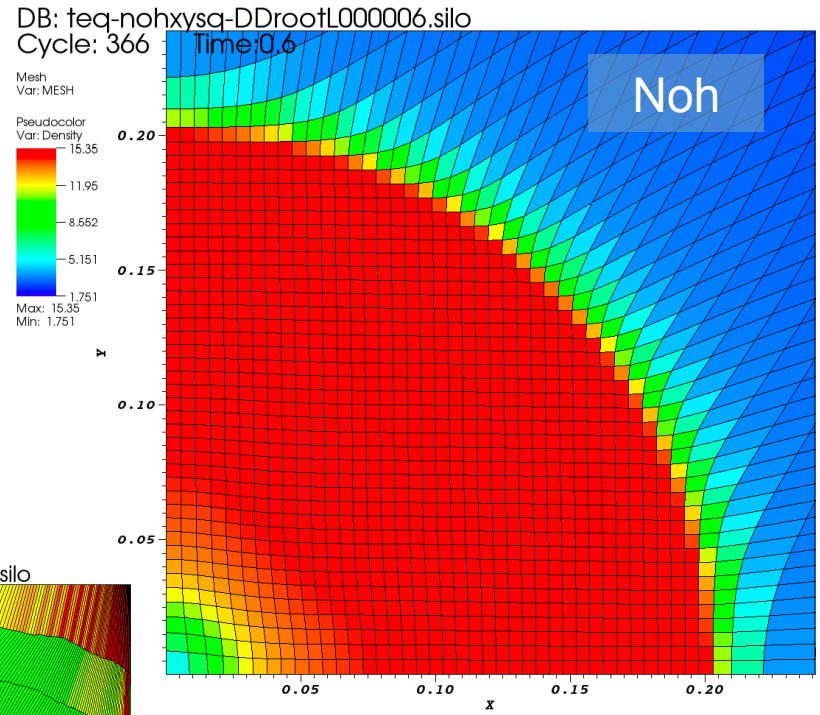
$$\mathbf{a}_n^s = C_{quad}\begin{pmatrix} \frac{1}{4}\sqrt{\alpha}\,\delta y\,u_{x,y}\,u_{y,y} \\ \frac{1}{8}\alpha\delta y\,u_{x,y}^2 + \frac{1}{2}\delta y\,u_{y,y}^2 \end{pmatrix} - C_{lin}C_s\begin{pmatrix} \frac{1}{4}\sqrt{\alpha}\,u_{x,y} \\ \frac{1}{2}u_{y,y} \end{pmatrix}$$

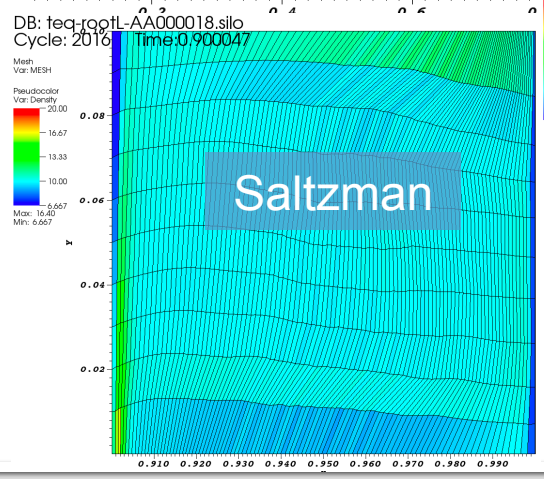
Acceleration of the central node should be in same direction as imposed velocity.

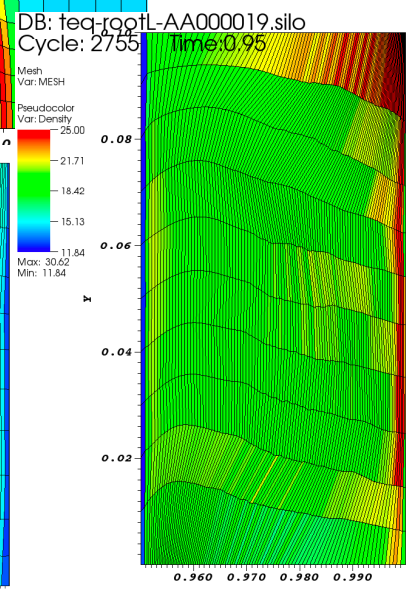
Acceleration direction explicitly changes with aspect ratio, definitely not good.

However, $\Delta u = \sqrt{L} \bullet G \bullet \sqrt{L}$ compares well with previous results on standard tests





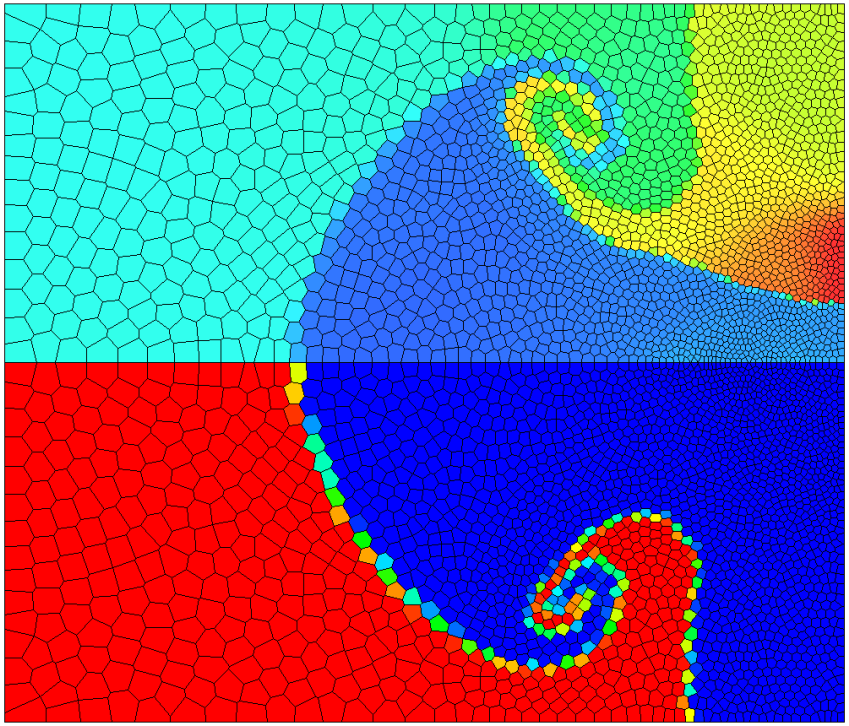




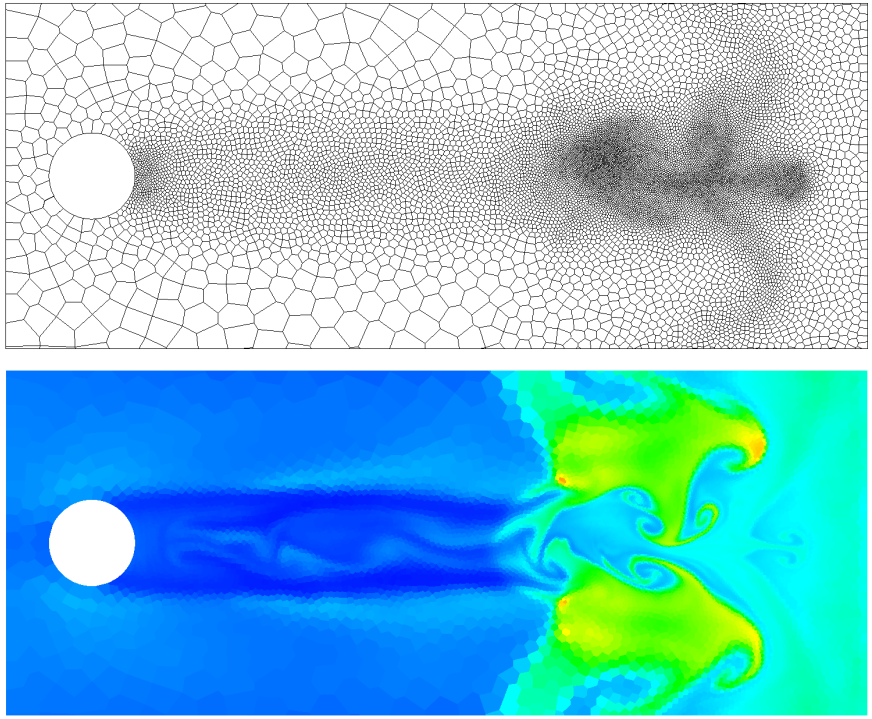
Saltzman piston runs through 3rd bounce at t=0.95.

LLNL's ReALE uses the $\Delta u = \sqrt{L} \bullet G \bullet \sqrt{L}$ version of the tensor edge **Q**

Previously KULL's hydro crashed the ReALE algorithm with mesh tangling. The tensor edge **Q** stabilized it, and is now our default scheme for ReALE.



Triple point solution with ReALE



ReALE solution for shock over a disk

CONCLUSION

- The **Q** based on $\mathbf{\Delta u} = \sqrt{\mathbf{L}}\bullet\mathbf{G}\bullet\sqrt{\mathbf{L}}$ achieves most of what we were looking for:
  - Galilean invariant
  - much less sensitive to mesh topology
  - With limiter, ignores shear, rotation, uniform compression
  - Dissipative
  - Only pushes in compression direction
- Still see incorrect force directions at very high aspect ratio
- Future possibilities
  - Replace $\mathbf{Q}=\rho(C_q\,\Delta\mathbf{u}\bullet\Delta\mathbf{u} - C_s\,C_l\,\Delta\mathbf{u})$ with $\mathbf{Q}=\rho(C_q\,g(\mathbf{u}) - C_s\,C_l)\,\Delta\mathbf{u}$ (*scalar*)
  - $\mathbf{\Delta u} = \nabla\mathbf{u}\bullet\mathbf{L}$, and filter when eigenvalues go complex
  - Last ditch: transition to CSW-style edge $\mathbf{\Delta u}$ as aspect ratio increases. There are known problems with this idea.

References

① E.J. Caramana, M.J. Shashkov, P.P. Whalen, *Formulations of artificial viscosity for multi-dimensional shock wave computations*, JCP v144, p70-97 (1998)

② J.M. Owen, *A tensor artificial viscosity for SPH*, JCP v201, p601-629 (2004)

③ D.S. Miller, J.M. Owen, M. Ulitsky, A. Cook, *Examining some new artificial viscosities,* presentation at the 5th MultiMat, Arcachon, FR, (2011)

④ M.L. Wilkins, *Use of Artificial Viscosity in Multidimensional Fluid Dynamic Calculations,* JCP v36, p281-303 (1980)

⑤ R.B. Demuth, L.G. Margolin, B.D. Nichols, T.F. Adams, B.W. Smith, *SHALE: A Computer Program for Solid Dynamics*, Los Alamos National Laboratory Report LA-10236, May 1985

⑥ J.C. Campbell, M.J. Shashkov, *A Tensor Artificial Viscosity Using a Mimetic Finite Difference Algorithm,* JCP v172, 739-765 (2001)

⑦ L. G. Margolin, *A Centered Artificial Viscosity for Cells with Large Aspect Ratios*, Lawrence Livermore National Laboratory Technical Report UCRL-53882 (1988).

Lawrence Livermore
National Laboratory

Backup slides start here

Lawrence Livermore National Laboratory

Rotating to the principal axes frame of the zone makes analysis simpler

$$\nabla \mathbf{u} = \begin{pmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} \end{pmatrix}$$

$\mathbf{L}$ is a tensor describing the shape of the zone.

$\nabla \mathbf{u}' = \mathbf{R}^T \nabla \mathbf{u} \mathbf{R}$, where $\mathbf{R}$ is the rotation matrix made from the eigenvectors of $\mathbf{L}$.

In the rotated frame, the eigenvalues of $\mathbf{L}$ have a simple physical interpretation.

$$\mathbf{L}' = \mathbf{R}^T \mathbf{L} \mathbf{R} = \begin{pmatrix} \Delta x' & 0 \\ 0 & \Delta y' \end{pmatrix}$$

The principal axes frame for **L** is especially convenient to work in

Now a very nice obvious thing happens when we multiply;

$$\Delta \mathbf{u}' = \nabla \mathbf{u}' \mathbf{L}' = \begin{pmatrix} \frac{\partial u'_{x'}}{\partial x'} \Delta x' & \frac{\partial u'_{x'}}{\partial y'} \Delta y' \\ \frac{\partial u'_{y'}}{\partial x'} \Delta x' & \frac{\partial u'_{y'}}{\partial y'} \Delta y' \end{pmatrix}$$

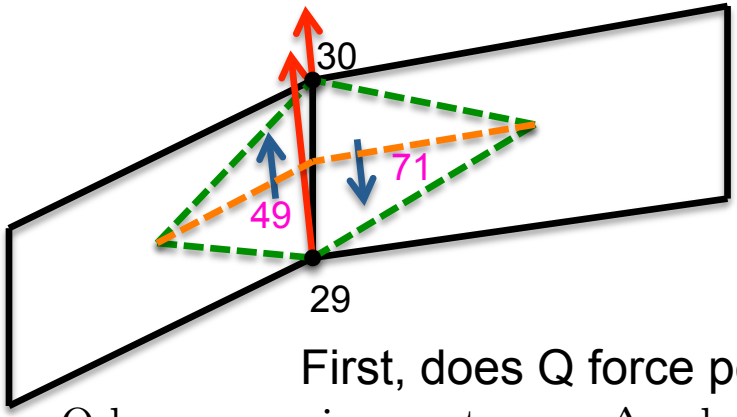
And this is exactly the expression we want for $\Delta \mathbf{u}$. So we rotate it back, via

$\Delta \mathbf{u} = \mathbf{R} \Delta \mathbf{u}' \mathbf{R}^T$, which simplifies trivially:

$$\Delta \mathbf{u} = \mathbf{R} \Delta \mathbf{u}' \mathbf{R}^T = \mathbf{R} \nabla \mathbf{u}' \mathbf{L}' \mathbf{R}^T = \mathbf{R} \mathbf{R}^T \nabla \mathbf{u} \mathbf{R} \, \mathbf{R}^T \mathbf{L} \mathbf{R} \, \mathbf{R}^T = \nabla \mathbf{u} \mathbf{L}$$

$$\Delta \mathbf{u} = \nabla \mathbf{u} \mathbf{L}$$

Small problem leads us to eigenvector issue with area normals used in computing **Q** forces



Why does side 49 produce a force opposite side 71?

First, does Q force point in delta U direction? Yes.

Q has same eigenvectors as $\Delta\mathbf{u}$, because squaring a matrix does not change eigenvectors. Does $\Delta\mathbf{u} = \nabla\mathbf{u}\cdot\mathbf{L}$ have same eigenvectors at $\nabla\mathbf{u}$? Obviously, in general, "no", but in our case $\nabla\mathbf{u}$ has only one non-zero eigenvalue, after expanding components are removed (it remains symmetric). In that case,
$\Delta\mathbf{u} = \nabla\mathbf{u}\cdot\mathbf{L} = \nabla\mathbf{u}\cdot(a\hat{\mathbf{e}}_1 + b\hat{\mathbf{e}}_2, c\hat{\mathbf{e}}_1 + d\hat{\mathbf{e}}_2) = (\lambda_1 a\hat{\mathbf{e}}_1, \lambda_1 c\hat{\mathbf{e}}_1)$
This $\Delta\mathbf{u}$ is singular (det=0), non-symmetric, and has eigenvector $\hat{\mathbf{e}}_1$.

$$\mathbf{F} = \mathbb{Q}\cdot\mathbf{A}$$
$$\mathbb{Q}\cdot\mathbf{A} = \mathbb{Q}\cdot(a\hat{\mathbf{e}}_1 + b\hat{\mathbf{e}}_2)$$
$$\mathbb{Q}\cdot\mathbf{A} = a\lambda_1\hat{\mathbf{e}}_1 + b\lambda_1\hat{\mathbf{e}}_2)$$
$$\mathbf{F} = a\lambda_1\hat{\mathbf{e}}_1$$

F points in correct direction for any area **A**

If **A** points near $\mathbf{e_2}$, the "a" coefficient gets very small. If **A** crosses $\mathbf{e_2}$, "a" changes sign. Similar areas can give different force directions!