Parallel ALE Hydrodynamics

Presented at

Salishan Conference on High Speed Computing

April 21, 2004

Rob Neely

LLNL

rneely@llnl.gov

Lawrence Livermore National Laboratory, PO Box 808, Livermore CA 94551

This work performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

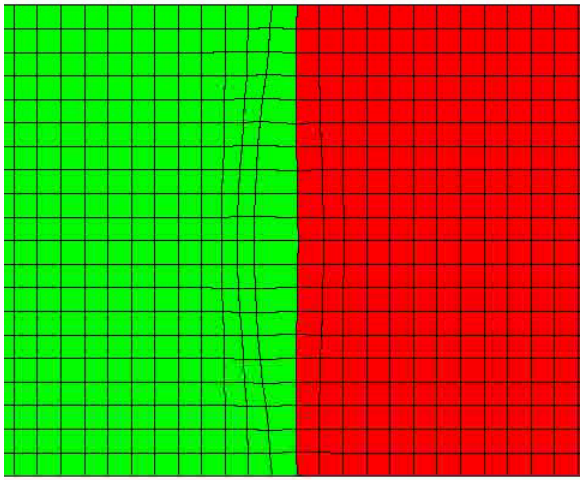
Overview

- **ALE – what is it?**
- **Overview of algorithms key to implementing ALE**
  - Mesh relaxation
  - Interface reconstruction
  - 2nd order advection
  - Momentum advection
  - Advection + slide surfaces
- **Performance implications of ALE on CPU's and parallel machines**

Topics in this talk cover some subset of ALE algorithm space, and represent the models and implementations best known by the author.

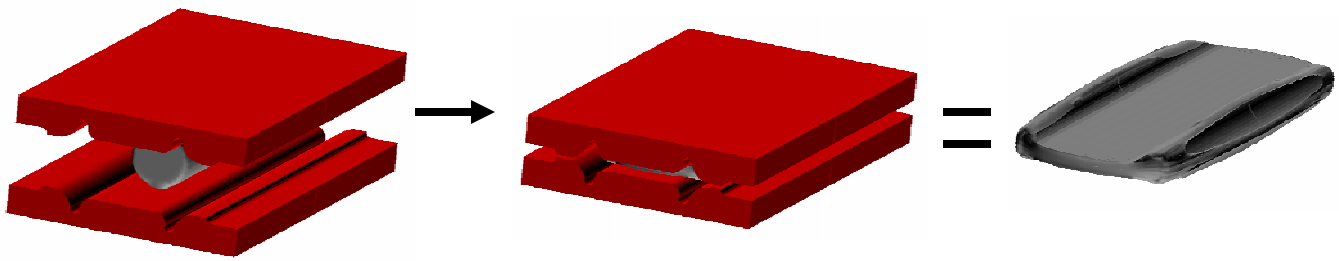
ALE Hydrodynamics Definitions

- **ALE = Arbitrary Lagrange Eulerian**
- **Hydrodynamics**
  - Modeling physical motion of material (matter) through space
- **Lagrange**
  - Mesh moves with the material
    - Pros: Accurate, need fewer elements
    - Cons: Mesh can tangle
- **Eulerian**
  - Fixed mesh, material flows through
    - Pros: Easy to set up mesh, robust (mesh can't tangle)
    - Cons: Diffusive, Requires more zones for accuracy
- **ALE**
  - Somewhere in between pure lagrange and pure eulerian
  - Offers both the best *and worst* of Lagrange and Eulerian



Example of pure lagrange simulation in an instability calculation (movie)

Example of ALE in Metal Forming Analysis

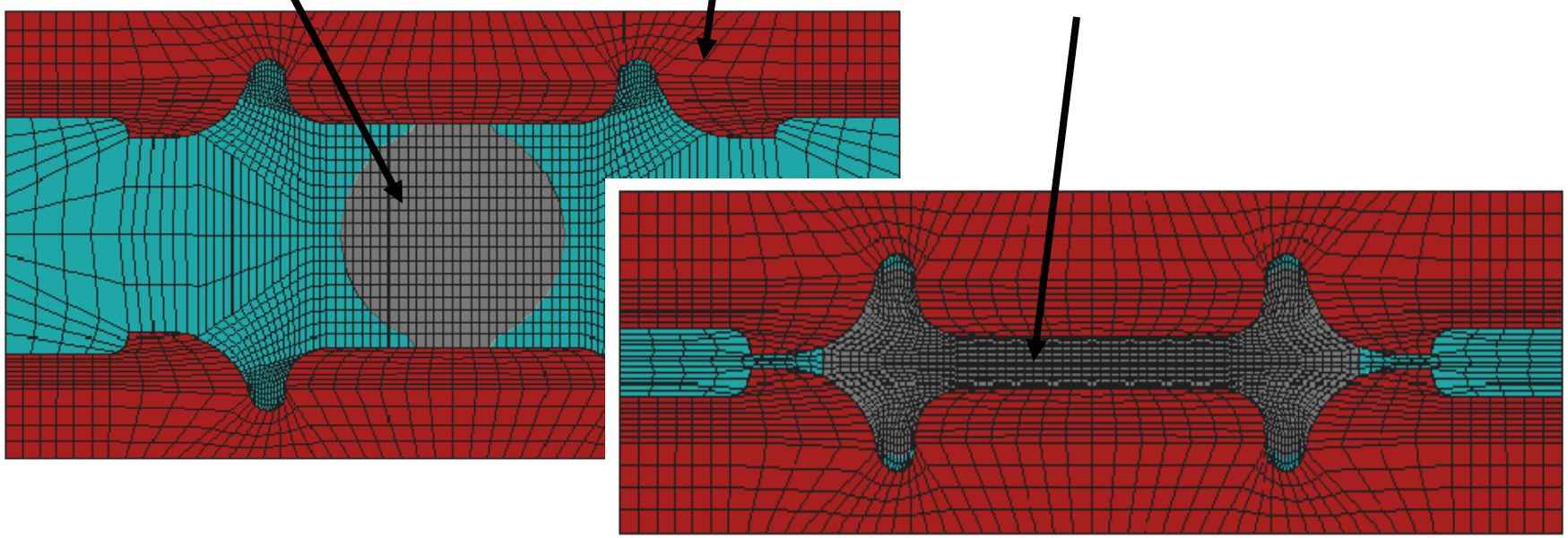


Soft Aluminum ingot compressed into desired shape

Aluminum piece is painted in over smooth mesh

Die cast mesh is kept lagrangian. Boundaries are maintained by slide surfaces

Softer material flows through mesh, which relaxes to relieve tangling

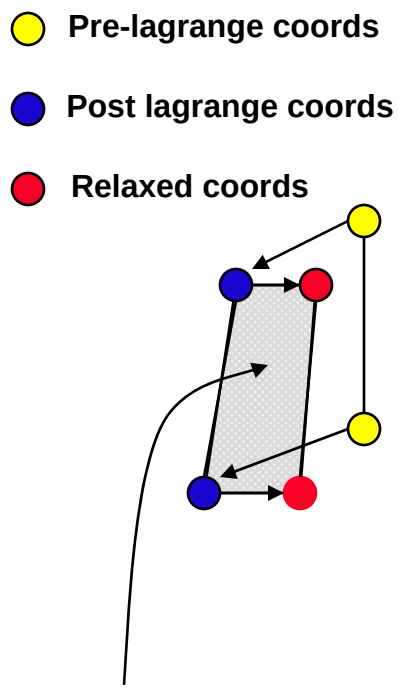


ALE = Advection

- **As discussed here, ALE is operator split (*Lagrange* + *Remap)***
  - Material is accelerated using whatever hydro algorithm
    - Explicit, implicit, time/space staggered, predictor-corrector, etc..
  - Material is remapped from the mesh at the end of the lagrange step to a new idealized mesh
- ***Advection* is the term we use for calculating this remap**
  - Fluxes are due to the mesh moving through the material
  - ie – the mesh is “pulled back” through the material
- **Advection can be done in sweeps (given a structured mesh), or simultaneously in all dimensions**
- **Advection is operator split (independent) from the lagrange algorithm**
- **Can be on meshes of any dimensionality (1D-3D)**
- **Works with structured or unstructured meshes**

Overview of Advection Step

- **Relax the lagrangian mesh**
  - Calculates new mesh coordinates based on post-lagrange positions + smoothing algorithms
- **Calculate volume fluxes at each *face* due to relaxation**
- **Approximate mixed element interfaces through reconstruction**
- **Apportion volume fluxes to material components**
- **Do 2nd order, monotonic advection for PURE elements**
- **Do 1st order upwind advection for MIXED elements**
- **Do 2nd order monotonic advection for momentum (node centered)**
- **Recompute pressure, rescale stresses, etc…**



Volume flux created by relaxation

Relaxation Algorithms

- **Elliptic partial differential equations (most widely and successfully used)**
  - Equipotential smoothing
  - Provide nice properties that guarantee no overlap of grid lines
    1. Solutions are smooth provided their coefficients are also smooth - i.e. continuous with continuous derivatives.
    2. Solutions are determined by their values on the boundary.
    3. Solutions obey an extremum principle, in that the values of all points inside and on the boundary are bounded by the values on this boundary.
- **Algebraic or interpolation methods**
  - involves the use of an interpolating function which is constrained to match specified boundary values.
- **Geometric**
  - Attempts to maintain the original grid spacing

Equipotential Relaxation

- **Also known as Winslow-Crowley**
- **Later modifications by Tipton are commonly employed**
  - Reformulated with variational integral and solved using finite element techniques
  - Added factors that tend toward equal volume elements
- **Map physical space to logical rectangular coordinates**
- **Solve inverse Laplace equation**

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \lambda x_{\eta\eta} = 0$$

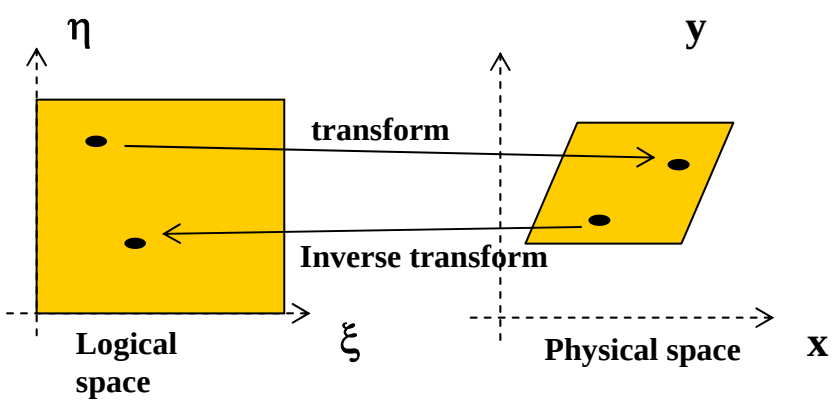
$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \lambda y_{\eta\eta} = 0$$

where

$$\alpha = x_\xi^2 + y_\xi^2$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta$$

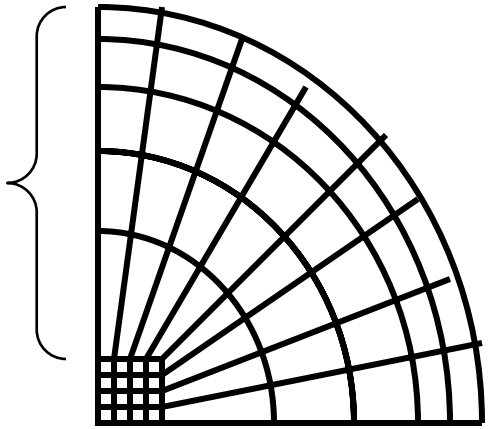
$$\lambda = x_\eta^2 + y_\eta^2$$



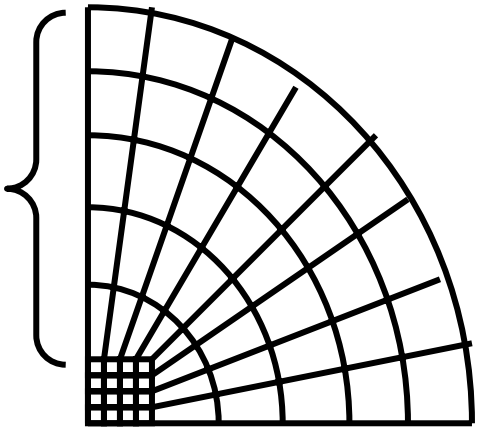
Recent Equipotential Relaxation Modifications

- **Tipton equipotential algorithms shoot for equal volume per element**
  - This is not always what we want
  - In spherical geometries, radial lines tend to push away from the center

As elements get wider away from the center, they flatten out – pulling the radial mesh lines outward to maintain equal volume



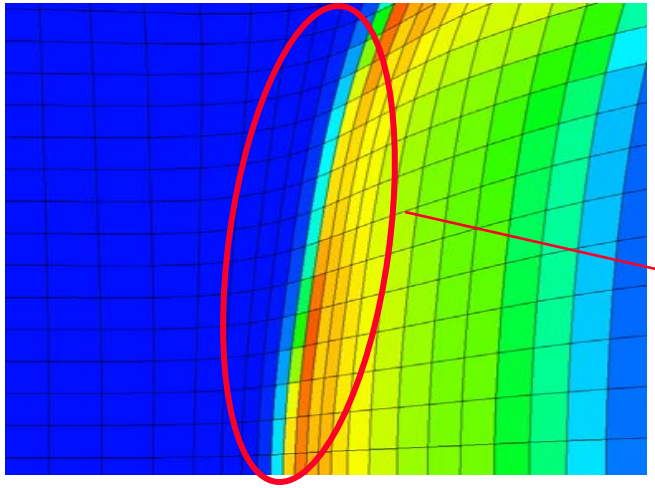
Ideally, we'd like to maintain equidistant radial mesh lines



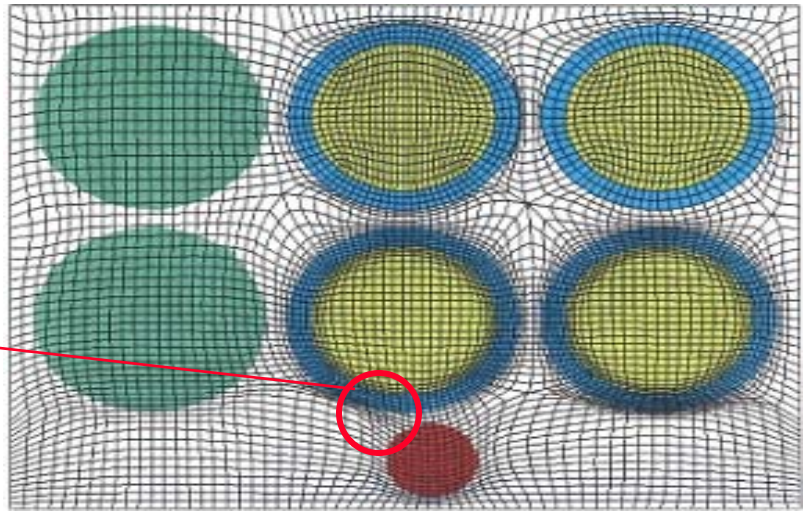
- **Additional terms have been added to the equipotential algorithms to better handle known deficiencies (B.I. Jun)**

Weighting Terms in the Relaxation

- **Weighting terms can be used on a per element basis to increase resolution in areas of interest**
  - Dynamically concentrate zones near shock fronts
  - Keep zones in regions or materials of interest
  - Based on material/region (e.g. high density materials)
  - Based on physical quantities (e.g. pressure, artificial viscosity)
- **Weights are typically smoothed and propagated to prevent sharp differences at material boundaries**

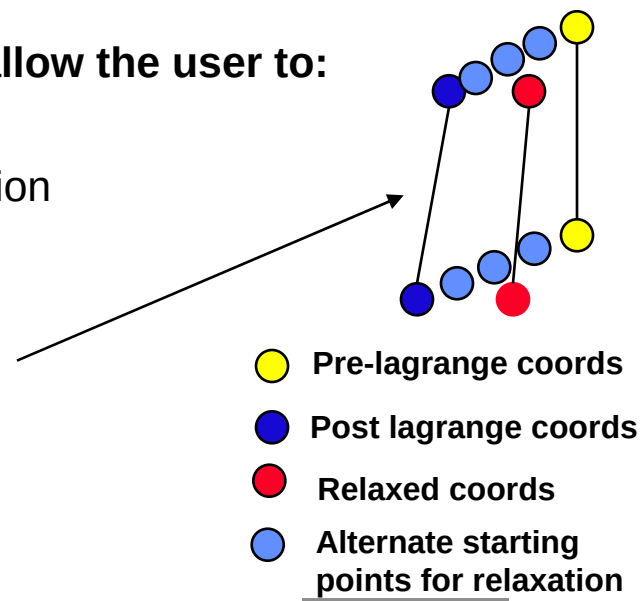


Pulling zones into shock front (left) and impact cylinder (right)



Mesh Relaxation as an Art

- **Philosophy should be to inhibit relaxation for as long as "good" mesh can be maintained**
  - Balance between accuracy of conforming material boundaries in lagrange, and better truncation error of smooth mesh in eulerian
- **Finding this balance (when to allow relaxation) can be a bit of an art**
- **User defined parameters helping control relaxation can sometimes be as important to the solution as the initial mesh**
- **ALE codes need to define relaxation parameters to allow the user to:**
  - Inhibit relaxation in subsets of the mesh
  - Define number of iterations of the equipotential solution
  - Limit relaxation on a node-by-node basis
  - Define the starting mesh used in the relaxation
    - pre-lagrange, post-lagrange, scaled in between
  - etc…

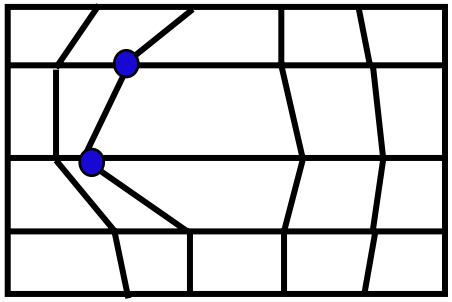


Constraints on Relaxation

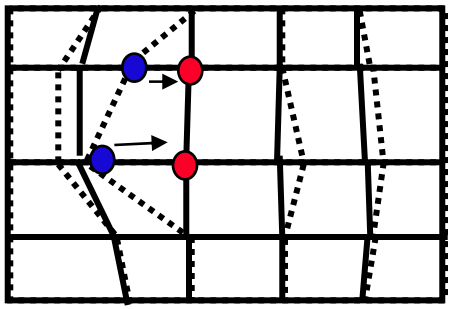
- **Nodes can be entirely *inhibited* from relaxing.**
  - Not reached a specified velocity yet
    - This prevents the mesh from moving until activity is present
  - User-specified start time has not been reached
  - Material based properties (has not detonated, reacted, etc…)
  - Nodes on mesh boundaries (free surface, inflow/outflow, etc…)
  - Manually specified by the user
- **Relaxation can be *limited* via displacement constraints**
  - A fraction of the shortest leg around a node
  - A multiple of the lagrangian distance moved
- **Likewise, nodes can be *forced* to relax (to prevent inevitable tangling due to distortion)**
  - Acute angles at nodal vertices
  - Large difference in element volumes surrounding nodes

Preventing too much advection

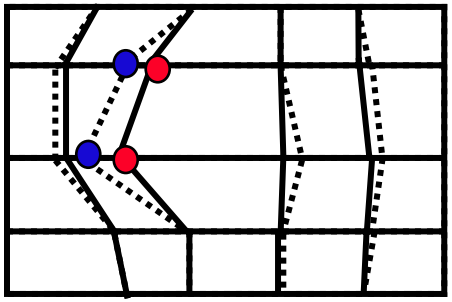
- **Accuracy will suffer if the face-centered flux is greater than some ratio of the volume in the neighboring element(s)**
  - e.g. If flux requires more than 50% of an element's volume to be advected in one cycle
- **Timestep controls can slow down lagrange motion and allow mesh to "catch up" over several timesteps**
- **Also, for a node that is far from equilibrium, the default equipotential solution can cause it to "jump off" too far when it first is allowed to relax**



Pre-relaxed mesh, far from equilibrium. Suppose blue nodes have been constrained up to this point



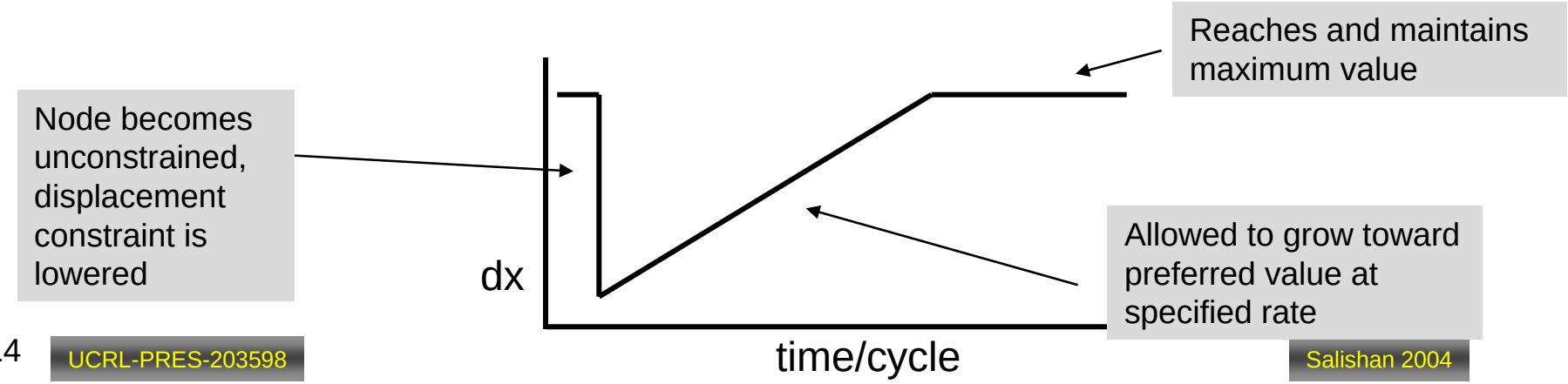
Blue nodes jump to their equipotential solution, generating too large a flux



Displacement constraint needs to control distance node is allowed to relax in one cycle

Dynamically Adjusting Displacement Constraints

- **The desired displacement constraint may need to be dynamically adjusted**
  - Initially it should be small, to prevent phenomena outlined in previous slide
  - Eventually, should be larger, to allow equipotential solution maximum effectiveness
- **We allow for the displacement constraint to be dynamically determined by the code**
  - Initially small when node first begins to relax
  - Grows toward “preferred” value at specified rate
  - Done on a *node-by-node* basis



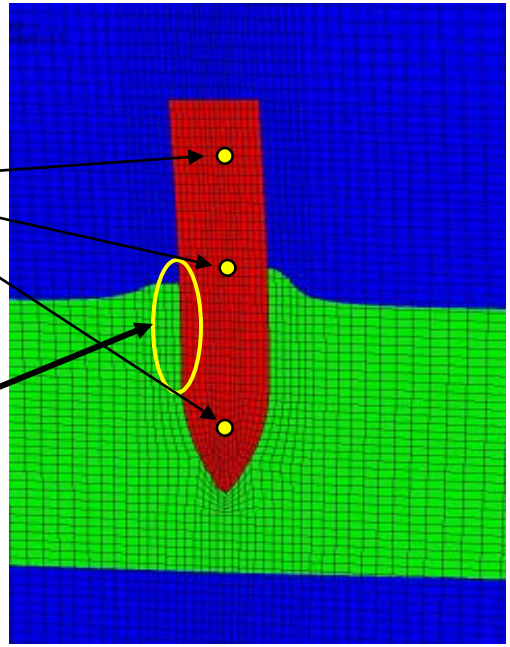
“Programmed” Relax

- **Sometimes a more global solution is required. e.g.**
  - Disallow relaxation in given dimension
  - Have mesh follow a rotating rigid body
  - Uniform mesh compression (maintain ratio zoning)
  - Following mesh deformation

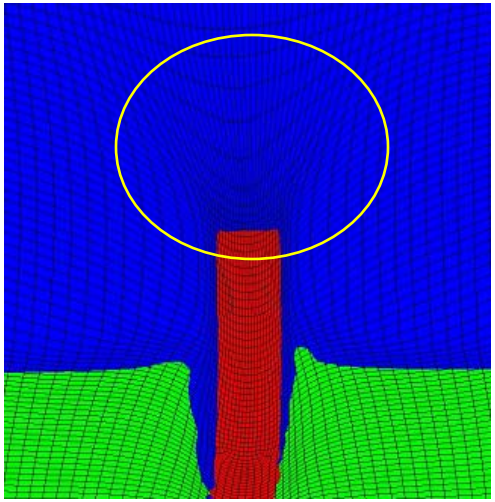
Default equipotential solution drags mesh behind the high velocity projectile

“Peg points” in the mesh direct relaxation in the rest of the mesh

Slide surface between projectile and target keeps non-eroding penetrator lagrangian



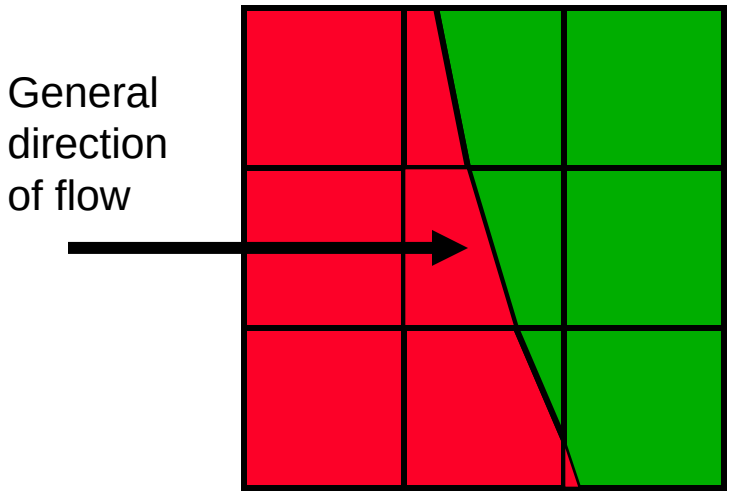
Programmed Relax



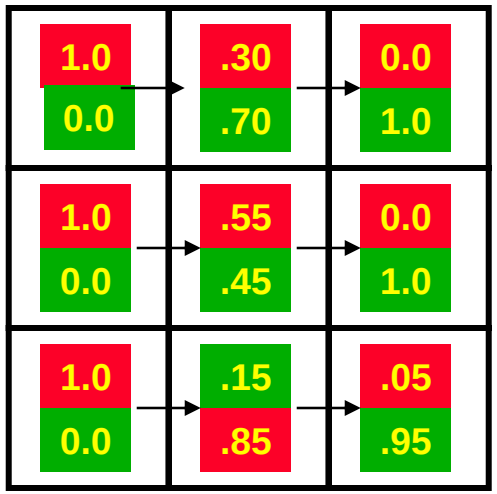
Equipotential Relax

Interface Reconstruction

- **In mixed material elements, interfaces between material components usually are not explicitly tracked**
  - Only volume fractions are known
- **How to determine material to transfer through volume fluxes?**
- **Estimate layout of materials in a zone by looking at volume fractions of materials in surrounding zones and direction of flux at each face**



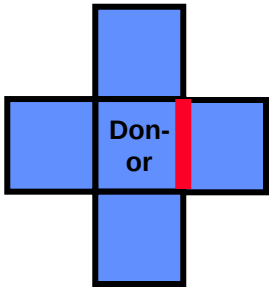
What's really happening



What the code sees

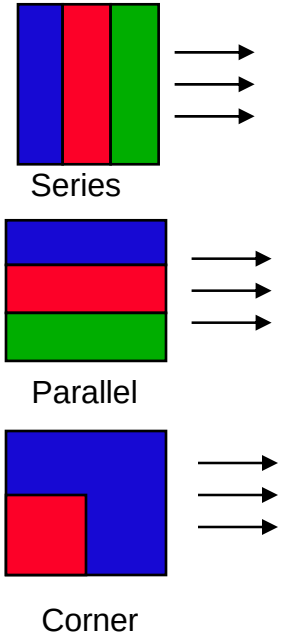
Interface Reconstruction Algorithm

- **CALE93 (Tipton) algorithm is one way to determine which materials will advect through a given volume flux**
- **A normalized slope is calculated at each face for each material in the upwind (donor) element.**



Slope at faces calculated from volume fractions in donor and its neighbors

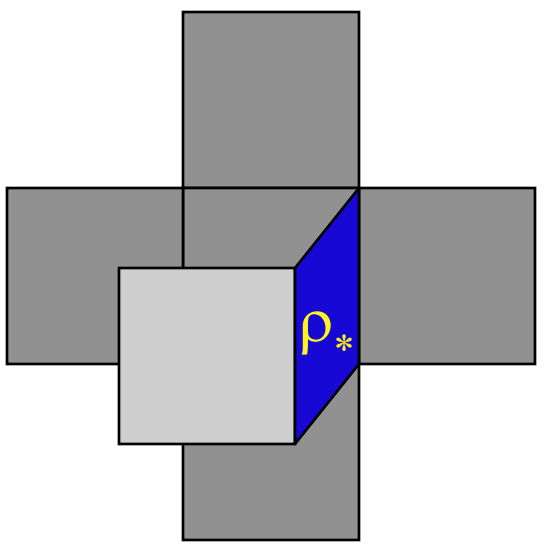
- ***Series* flow: Materials are moved in order in which they are stacked up relative to the face**
- ***Parallel* Flow: The components are moved simultaneously.**
- ***Corner* Flow: Treated as series flow until a critical volume fraction is achieved, at which point it becomes parallel flow.**
- **In the case where materials are advected simultaneously in all dimensions, care must be taken not to overdeplete the material in a zone**
  - Three passes (leading/parallel ; middle ; trailing)
  - After each pass, material flux volumes must be rescaled



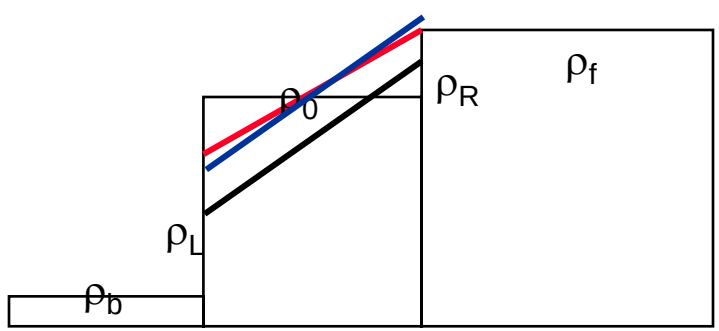
2nd Order Advection

- **Advection done simultaneously in all directions**
  - Need to find density at midpoint of advected volume
  - Function of density at upwind element and its neighbors
  - Can be shown to be 2nd order accurate

S enforces monotonicity
ρ is constant in an element



$$\rho_* = S\frac{\Delta\rho}{\Delta_\xi}\xi + S\frac{\Delta\rho}{\Delta_\eta}\eta + S\frac{\Delta\rho}{\Delta_\zeta}\zeta + \rho_0$$

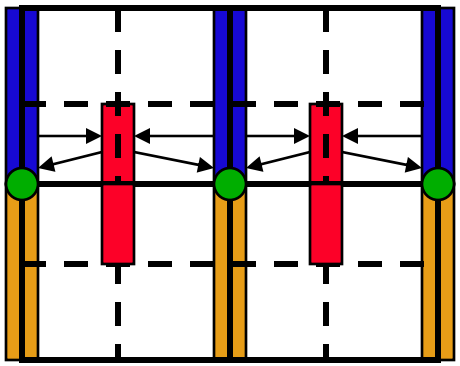


VanLeer limiter

Unlimited slope
Shift the line up so it goes through ρ_0 (mass conserved)
Limit slope to avoid overshoot

Nodal Momentum Advection

- **Need to recompute nodal velocities due to momentum transfer on relaxed mesh**
- **Material independent**
- **Compute momentum fluxes on dual mesh**



•Recenter mass fluxes (solid blue) on dual mesh (dotted red)

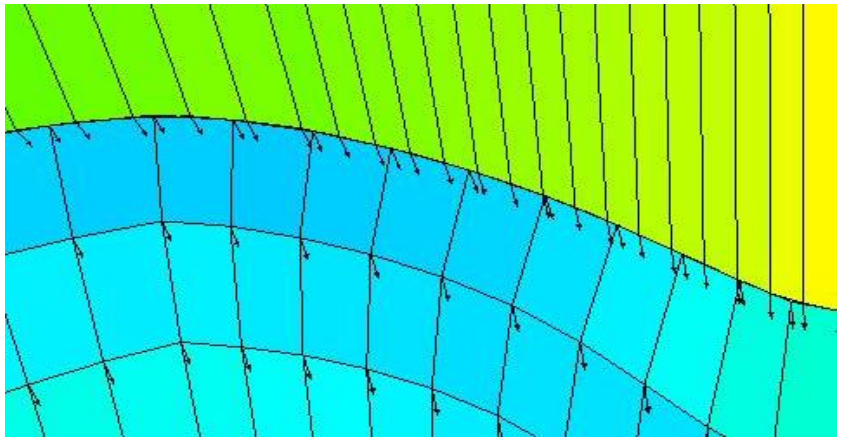
•Interpolate velocities on dual to get momentum fluxes (limit on $\nabla\vec{u}$ is monotonicity constraint)

•Reapportion momentum fluxes on dual to nodes

•Evaluate new nodal velocity

Advection and Slide Surfaces

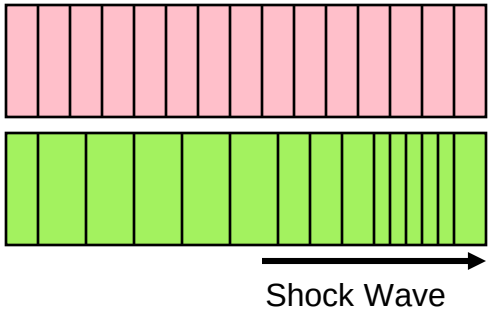
- **Slide Surfaces are discontinuities in the mesh**
- **Material is allowed to flow tangential to the surface, but not across**
  - Momentum normal to the slide surface is conserved



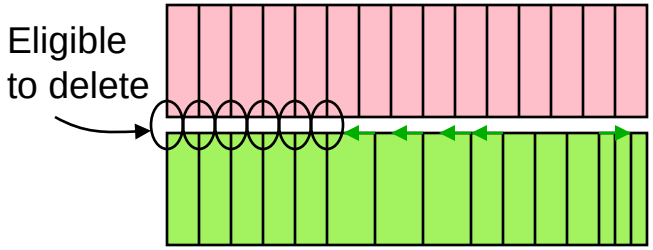
- **Slides required early in simulation can be deleted late in time to allow advection normal to surface**
  - Only if nodes match up 1-1 across surface
- **Deletion of a slide can be done all at once, patches at a time (based on user selection), or partially based on heuristics (e.g. activity + grace time)**

Aggressive Relaxation

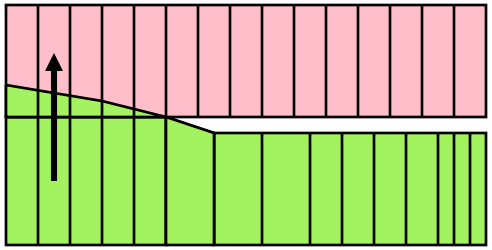
- **Allow mesh to follow a shock, using relax weights to pull mesh in**
- **As shock passes, tell node to "migrate" back to its order node**
  - Migration rate is automatically limited by displacement constraints
- **Once it returns, mark it eligible for deletion**
- **Good for problems where direction of flow changes from tangential to normal at later time**



Early time: mesh follows shock



Shock passes, nodes migrate back to be eligible for deletion. (Must line up with corresponding node on other side of slide surface)



At specified intervals, eligible nodes are merged, advection is allowed across surface

High Level Algorithm and Comm Requirements

Lagrange Step

Ghost Elem Exchange (Pure elem, mix elem, history, nodal pos/vel) (B, C x 3)

Relax Mesh

Propagate Weights (E x 3)

First order rlx pos (B)

Calc volume fluxes

Potential mixed zones (D)

Interface Reconstruction

Transfer slope (F)

Check overdepletion (D x 3)

Zonal Advection

Momentum Advection

Sum mass fluxes (A)

Sum momentum fluxes (A)

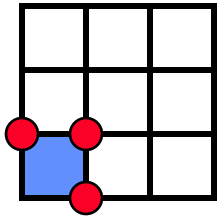
Sum partial nodal masses (A)

Update

Ghost elem exchange of new material (pure + mix) (C x 2)

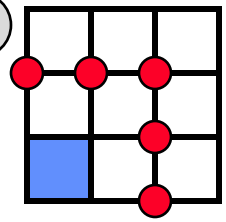
Computation
Communication

A



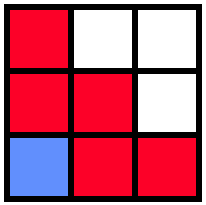
Sum partial values at boundary nodes

B



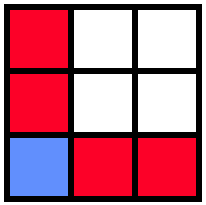
First order ghost node exchange

C



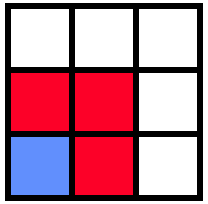
Full zonal ghost exchange (1st, 2nd, corner)

D



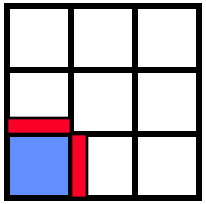
Upwind (+1) elem exchange

E



First order elem exchange

F



Exchange face calculations

Typical communication patterns required for advection and their stencils

Parallel Performance

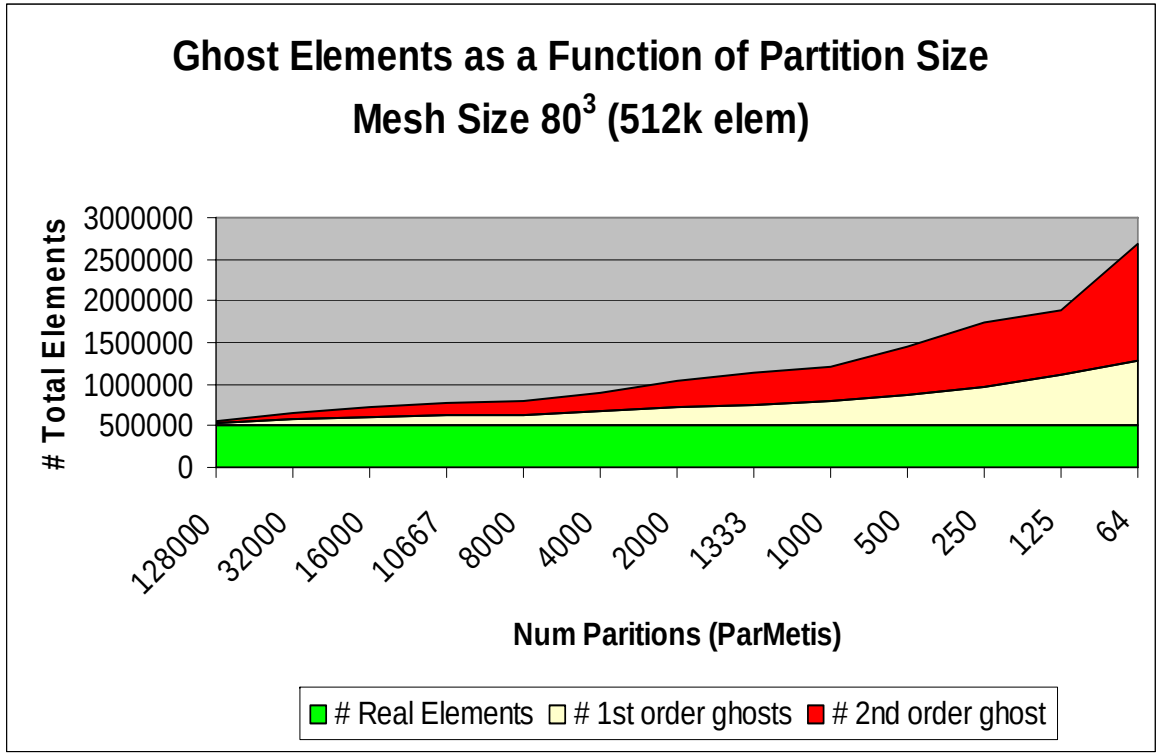
- **Nothing about ALE (advection) is particularly non-scalable**
  - Communication is mostly local, point-to-point, ghost element boundary exchanges
- **Load Balance can be a factor**
  - Areas with mixed elements require more work
    - These boundaries move between processors
  - Different material models can have drastically different compute times
    - Initial static load balance will break down at later time
- **Collective operations are mostly limited to the lagrange calculations**
  - Sparse solves (implicit hydro), minimum timestep, etc…

Parallel Issues

- **Second order advection requires two layers of ghost elements**
- **Surface area/volume ratio effectively doubles**
- **Small partitions suffer from excessive ghost elements**

Percentage of ghost elements when domain size is:

- 128000 elem/dom = 10%
- 16000 elem/dom = 41%
- 2000 elem/dom = 102%
- 250 elem/dom = 238%



CPU Performance Implications

- **Advection is characterized by not having any single kernel of the algorithm which dominates the timestep**
- **For explicit lagrange, advection can increase the cost of a timestep by 2-5x**
- **Lots of indirection between element, node, and face centered data**
- **Typically memory bandwidth limited**
  - Face centered cache issues
  - Loop over faces, gather data from upwind (+1) and downwind elements
  - Can cause very irregular memory access patterns to element data
- **Good relaxation requires lots of "bookkeeping"**

Performance of Production ALE Code on IBM Power3 CPU

- Recent performance studies done at LLNL on several different codes show advection running at between 6% and 9% of peak on the Power3 CPU

- Structured mesh codes did slightly better than unstructured

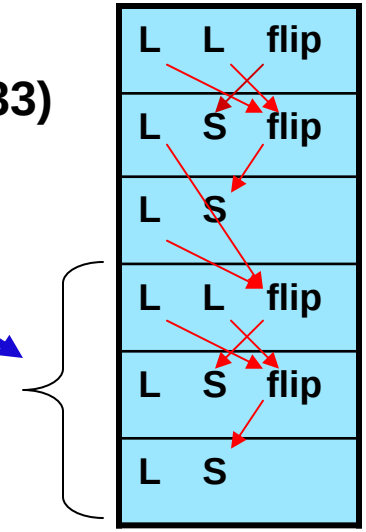
PACKAGE	Mflops	% peak Mflops	% of run
problem	101.44	6.8%	100.0 %
initialization	5.93	0.4%	2.2%
lagrange	119.29	8.0%	%
lagrange nodal	109.94	7.3%	8.7%
lagrange zonal	137.11	9.1%	%
mat (eos + str)	133.98	8.9%	6.4%
advection	98.97	6.6%	%
relaxation	106.85	7.1%	%
equip solve	197.59	13.2%	3.2%
calc vol flux	184.12	12.3%	5.9%
interface recon	3.76	0.3%	0.7%
zonal advection	68.44	4.6%	%
momentum adv	109.75	7.3%	%
mat (eos only)	134.76	9.0%	1.6%
slide surfaces	102.66	6.8%	%
restart/plot (silo)	0.28	0.0%	1.4%

Other Code Characteristics

PACKAGE	Mflops	% peak Mflops	% of run	%fp instr	%fma/fp instr	instr/ cycle	L1 hit rate	ld/st ratio	comp intens
problem	101.44	6.8%	100.0%	21.5%	25.3%	0.96	97.9%	2.63	0.58
initialization	5.93	0.4%	2.2%	1.3%	19.4%	1.08	99.4%	2.14	0.03
lagrange	119.29	8.0%	20.3%	24.4%	30.3%	0.97	98.7%	2.64	0.64
lagrange nodal	109.94	7.3%	8.7%	23.2%	26.2%	1.04	98.8%	2.73	0.54
lagrange zonal	137.11	9.1%	11.4%	26.8%	36.2%	0.86	98.2%	2.47	0.89
mat (eos + str)	133.98	8.9%	6.4%	26.0%	42.1%	0.75	97.8%	2.58	0.98
advection	98.97	6.6%	64.0%	21.5%	22.6%	0.96	97.5%	2.73	0.59
relaxation	106.85	7.1%	14.8%	21.3%	33.6%	0.97	97.4%	3.62	0.56
equip solve	197.59	13.2%	3.2%	35.4%	48.7%	1.01	97.0%	2.55	1.19
calc vol flux	184.12	12.3%	5.9%	39.9%	23.0%	1.09	97.7%	2.98	1.69
interface recon	3.76	0.3%	0.7%	1.0%	0.0%	1.24	97.5%	2.27	0.02
zonal advection	68.44	4.6%	20.1%	16.2%	12.3%	0.93	97.5%	3.4	0.49
momentum adv	109.75	7.3%	11.7%	24.8%	17.8%	1.03	97.1%	1.48	0.48
mat (eos only)	134.76	9.0%	1.6%	26.7%	34.2%	0.79	98.1%	2.04	1.07
slide surfaces	102.66	6.8%	10.5%	19.7%	38.7%	1	98.9%	3.91	0.52
restart/plot (silo)	0.28	0.0%	1.4%	0.0%	0.0%	0.84	99.7%	1.4	0

Simplified CPU models help us target performance benefit efforts

- Measured performance data gives us *algorithmic characteristics* that help us define an approx upper bound on **Power3 CPU's**
  - ***Measured*: .25 FMA/FP (thus, 1 flip – 1.25 flops)**
  - ***Measured*: ~ 2 loads + 1 store per flip (comp intens = .33)**
  - ***Limitation*: 2 loads or 1 load and 1 store per cycle**
  - ***Possible*: 2 flips per 3 cycles (2.5 flops per 3 cycles)**
  - ***Peak*: 4 flops/cycle (12 flops per 3 cycles)**
  - **2.5/12 = 20.8% upper bound**



- This percentage is further eroded by
  - **Cache misses, branch mispredictions, dependencies, integer operations, communication, etc…**
- This same idea can be applied to different instruction mixes:
  - **Comp intens = .5, and 50% FMA (1.5 flops / cyc) = 37.5%**
  - **Comp intens = .25 and 20% FMA (2.5 flops / 4 cyc) = 15.6%**

The future of ALE

- **What sorts of things are people working on related to ALE?**
  - ALE + AMR
  - More accurate mixed zone interface reconstruction/tracking
  - Better heuristics for automating mesh control

Conclusions

- **ALE attempts to capture the best of lagrange and eulerian methods**
- **Algorithms discussed here are well understood and commonly used**
- **Advection is moderately expensive**
  - Memory intensive
  - Scalable
- **Aside:**
  - MPI and stable compilers (read: portability) have done more for productivity than virtually any other advances in CS in the past decade
  - We spend O(n^2) more time porting 3rd party libraries than we do worrying about parallelism
- **Acknowledgements**
  - Richard Sharp, Brad Wallin, Evi Dube