LA-UR-13-26794

Exact intersection remapping of multi-material domain-decomposed polygonal meshes

M. A. Kenamond, D. E. Burton

X-Computational Physics
Los Alamos National Laboratory

Multimat 2013

International Conference on Numerical Methods for Multi-Material Fluid Flows

San Francisco, September 2-6, 2013

Acknowledgements:

U.S. DOE LANL LDRD & ASC programs

S. Doebling, T. Gianakon

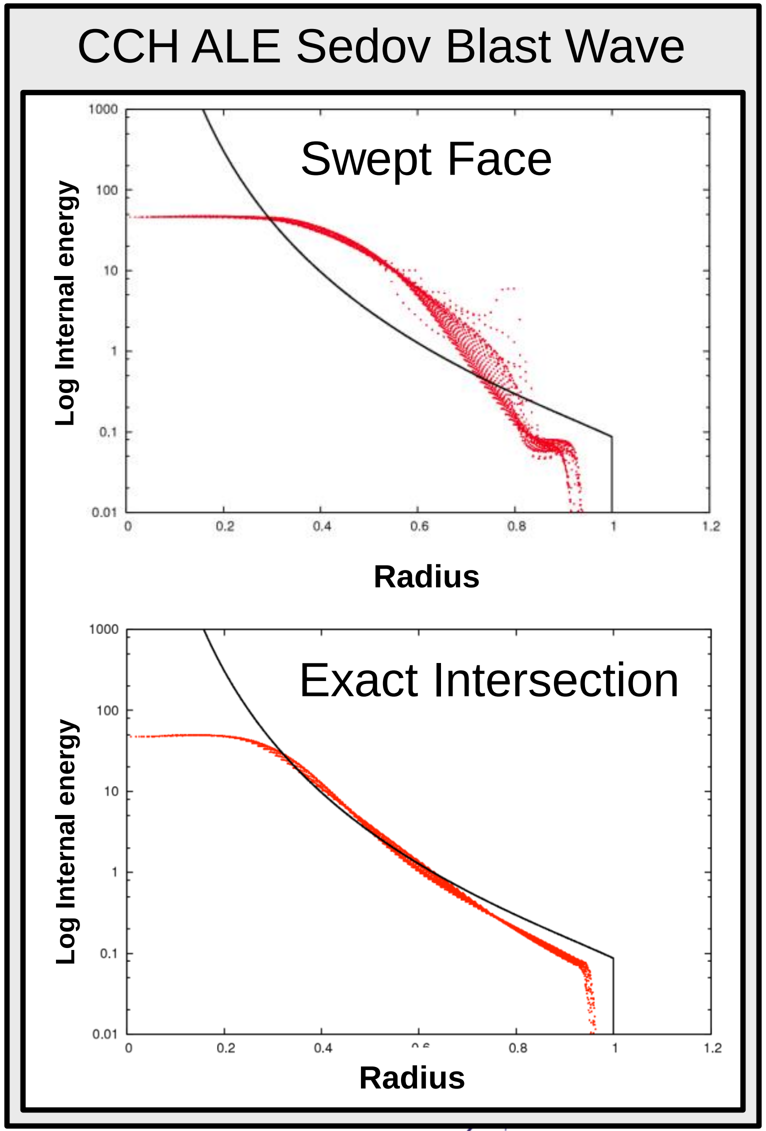
Los Alamos NATIONAL LABORATORY

Outline

- Introduction
- eXact method overview
- 2^{nd}-order remap
- Edge tracking and polygon generation
- Multi-material remap with VOF
- Results
  - Accuracy
  - Performance
  - Examples
- Summary, Conclusions and Future work

Introduction (why do you care?)

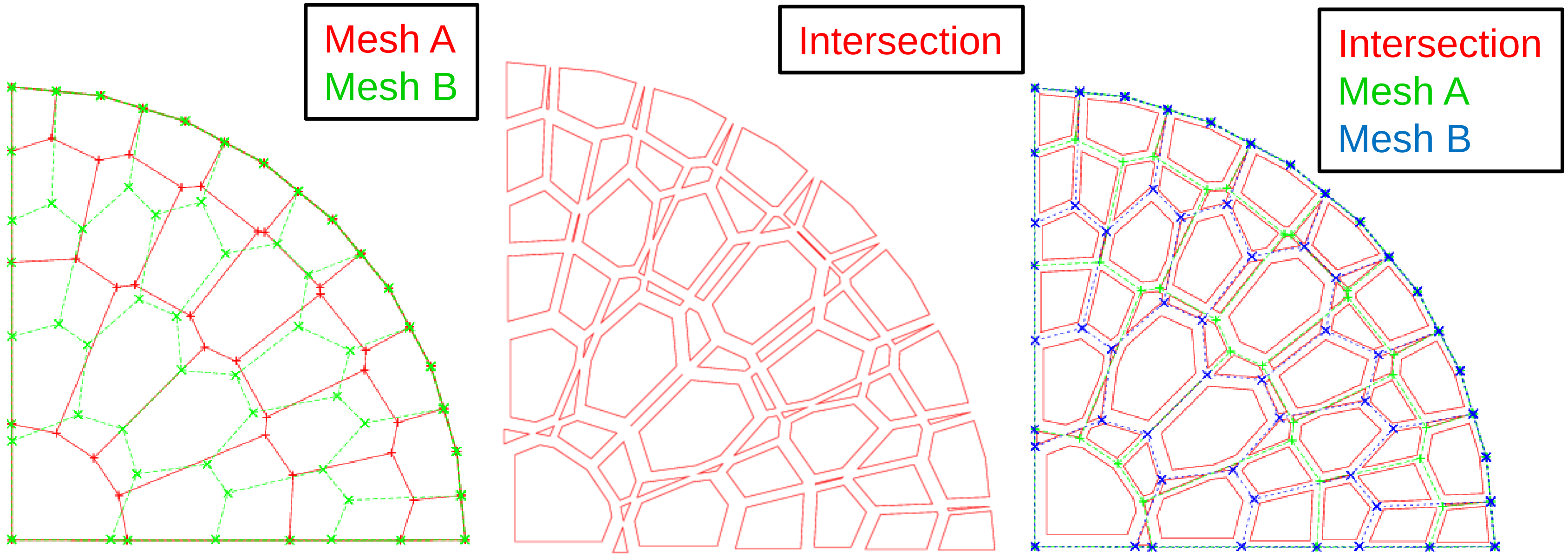
- Remapping is required for Lagrange+remap ALE hydrodynamics
  - Complete Lagrange cycle
  - Move or “relax” the mesh points, usually to improve mesh quality
  - Remap physics state to the relaxed mesh
  - Repeat
- Typical ALE 2nd-order remap is based on swept-face remap (advection)
  - Flux material across faces between donor and acceptor cells
  - Flux volumes limited to some fraction of donor cell volume
- Exact intersection remap is better
  - Fluxes across corners are included, improving accuracy for general flow
  - Not limited by swept face flux volume, decreasing cycles to solution
- Doesn’t have to be prohibitively expensive
  - Presented method is O(n) time



Los Alamos NATIONAL LABORATORY

Method overview

- Remap requires intersection of pre-relaxed mesh A with relaxed mesh B
- Overlay of both meshes generates intersection polygons
- Each polygon is the intersection of a mesh A (donor) zone with a mesh B (acceptor) zone
- Each polygon represents a flux from the donor zone to the acceptor zone
- Remap fields by subtracting fluxes from donors and adding to acceptors



Los Alamos
NATIONAL LABORATORY

2nd Order Remapping

- 2nd-order remapping of a volume-weighted intensive field f

- Requires integration over the intersection polygon of a linear reconstruction of field $f(\mathbf{x})$ based on known donor zone centered field $f(\mathbf{x}_c)$ at known donor zone centroid $\mathbf{x}_c$

$$f(\mathbf{x}) = f(\mathbf{x}_c) + \mathbf{G} \cdot (\mathbf{x} - \mathbf{x}_c)$$

- Where $\mathbf{G} = \langle \boldsymbol{G_x}, \boldsymbol{G_y} \rangle$ is the limited gradient of the field

- Choose your favorite limited gradient method

2^{nd} Order Remapping

- Extensive flux F from donor zone to acceptor zone is the integral of the linear field $f(\mathbf{x})$ over the intersection polygon volume V

Constant in V

$$F = \int_V f(\mathbf{x})dV = \int_V \big(f(\mathbf{x}_c) + \mathbf{G} \cdot (\mathbf{x} - \mathbf{x}_c)\big)dV$$

$$F = f(\mathbf{x}_c) \int_V (1)dV + \mathbf{G} \cdot \int_V (\mathbf{x})dV - \mathbf{G} \cdot \mathbf{x}_c \int_V (1)dV$$

$$F = (f(\mathbf{x}_c) - \mathbf{G} \cdot \mathbf{x}_c)J_0 + \mathbf{G} \cdot \mathbf{J}$$

$$J_0 = V = \int_V (1)dV\,, \qquad \mathbf{J} = \langle J_x, J_y \rangle, \qquad J_x = \int_V (x)dV\,, \qquad J_y = \int_V (y)dV$$

$$\boxed{F = (f(\mathbf{x}_c) - \mathbf{G} \cdot \mathbf{x}_c)J_0 + G_x J_x + G_y J_y}$$ Cartesian

$$\boxed{F = (f(\mathbf{x}_c) - \mathbf{G} \cdot \mathbf{x}_c)J_0 + G_r J_r + G_z J_z}$$ Cylindrical

- Forms of J_0, J_x, J_y depend on Cartesian vs. cylindrical geometry
- Integrals J_0, J_x, J_y can be re-used to remap all fields

Los Alamos NATIONAL LABORATORY

2^{nd} Order Remapping

- Integrals are various moments of area

- Cartesian ($dV = dxdy$):

$$J_0 = V = A = \iint (1) dxdy, \qquad J_x = \iint (x) dxdy, \qquad J_y = \iint (y) dxdy$$

- Cylindrical ($dV = r\, drdz$):

$$J_0 = V = \iint (1) r\, drdz, \qquad J_r = \iint (r) r\, drdz, \qquad J_z = \iint (z) r\, drdz$$

2^{nd} Order Remapping

- Discrete integral for polygon p with n edges is sum of discrete edge integrals

$$J_p = \sum_{e=1}^{n} J_e$$

- Discrete integrals for edge e with endpoints $\mathbf{x}_1$ and $\mathbf{x}_2$ are:
  - Cartesian:

$$J_0 = \frac{1}{2}(x_1 + x_2)(y_2 - y_1)$$

$$J_x = \frac{1}{6}(x_1^2 + x_1 x_2 + x_2^2)(y_2 - y_1)$$

$$J_y = \frac{1}{6}(y_1^2 + y_1 y_2 + y_2^2)(x_1 - x_2)$$

  - Cylindrical:

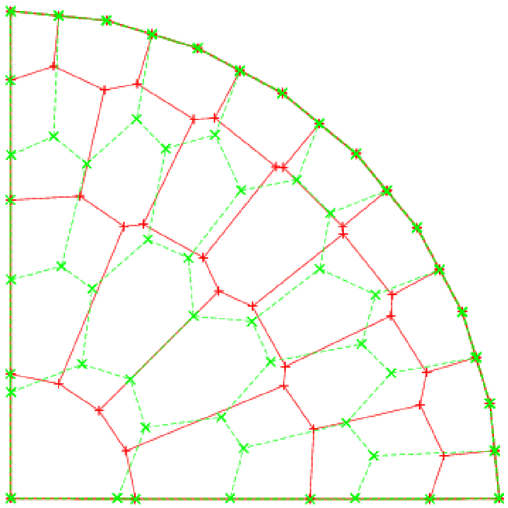
$$J_0 = \frac{1}{6}(r_1^2 + r_1 r_2 + r_2^2)(z_2 - z_1)$$

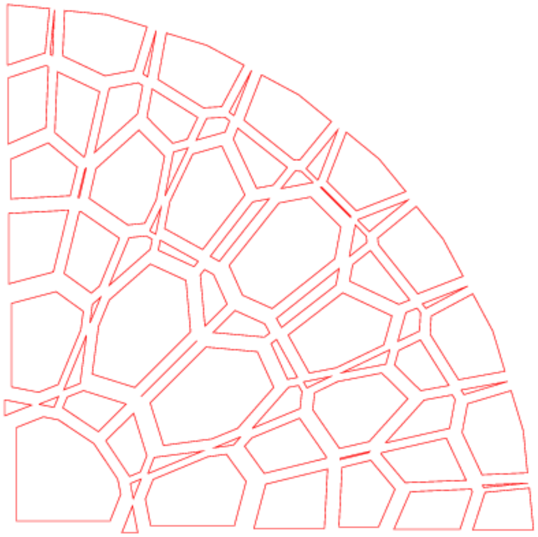
$$J_r = \frac{1}{12}(r_1 + r_2)(r_1^2 + r_2^2)(z_2 - z_1)$$

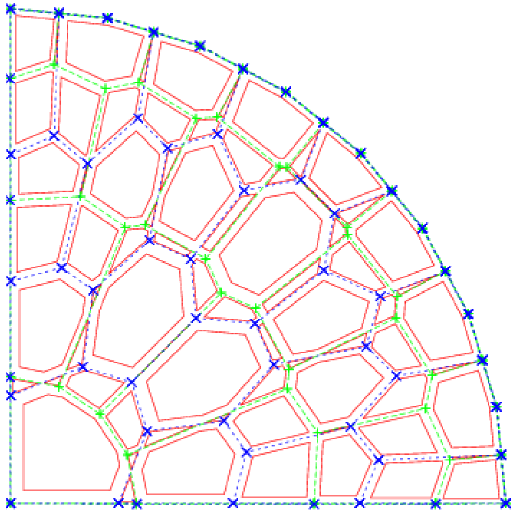
$$J_z = \frac{1}{24}(r_1^2(3z_1 + z_2) + r_2^2(3z_2 + z_1) + 2r_1 r_2(z_1 + z_2))(z_2 - z_1)$$

2nd Order Remapping

- Compute relaxed mesh zone volumes
- Compute edge integrals
- Sum to polygon to get J_0, J_x, J_y
- For each field
  - Compute limited gradient $\mathbf{G}$
  - Compute flux F for each polygon
  - Subtract flux from donor, add to acceptor
  - Convert new extensive value back to intensive form if necessary
- But we still need to determine the intersection polygons







Edge tracking and polygon generation

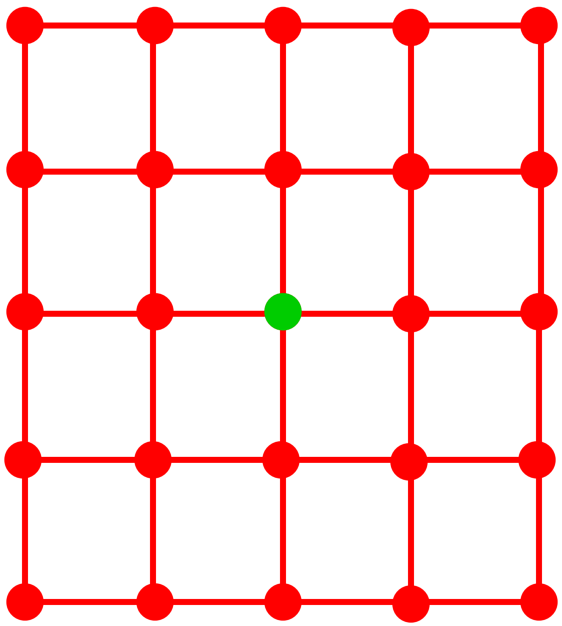
- Want to intersect every edge in donor mesh with acceptor mesh edges (and vice versa) with O(n) time complexity
  - Edges broken into segments at intersection points
  - Resulting segments bound intersection polygons
  - Segment geometry required to remap fields

- This method is an improvement to Miller and Burton method
  - They perturbed one mesh in order to avoid exact point-point, point-edge, or edge-edge coincidence
  - Works perfectly…most of the time
  - Not 100% robust

- This method:
  - Uses an advancing front algorithm (Burton et al, LA-UR 12-20613)
  - Requires no perturbation

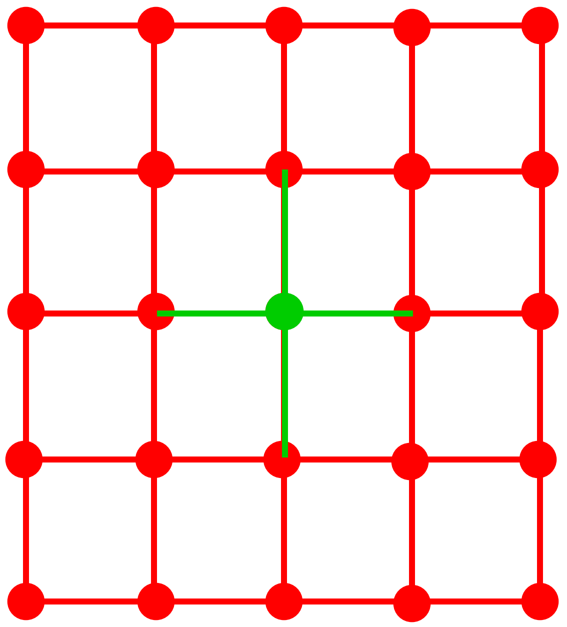
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



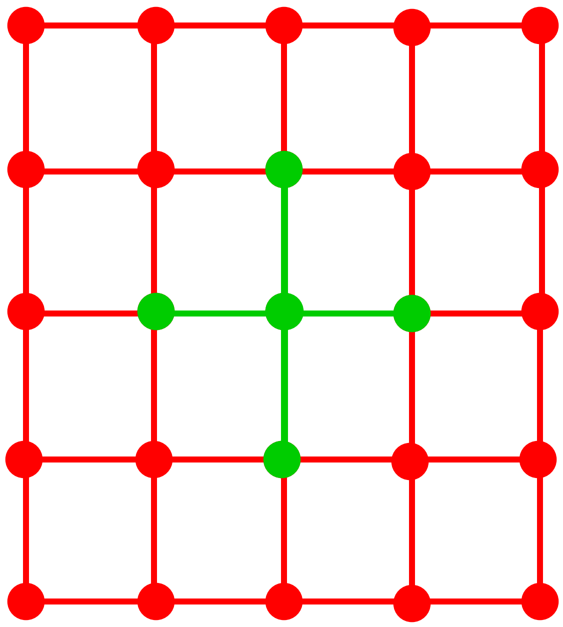
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



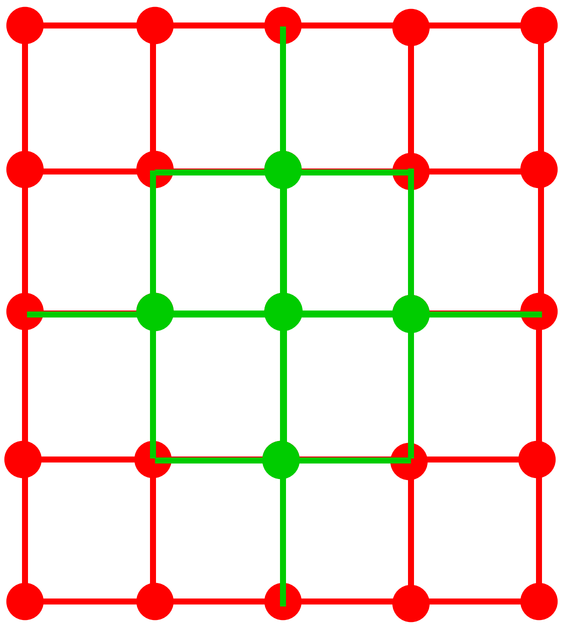
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



Edge tracking: Advancing wavefront

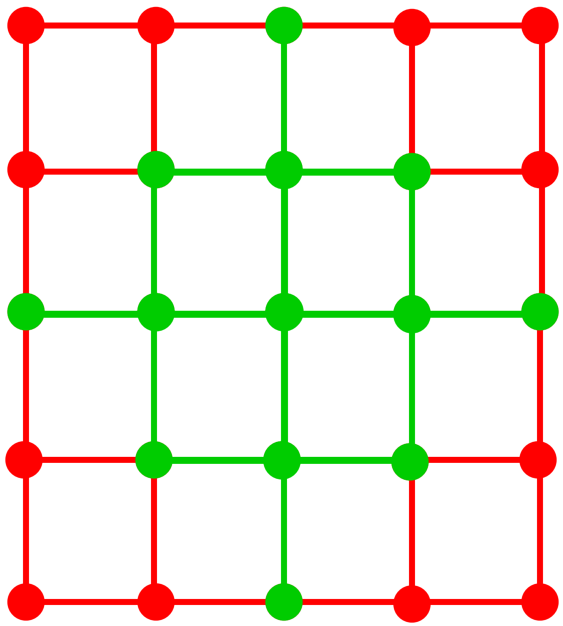
- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



Los Alamos
NATIONAL LABORATORY

Edge tracking: Advancing wavefront

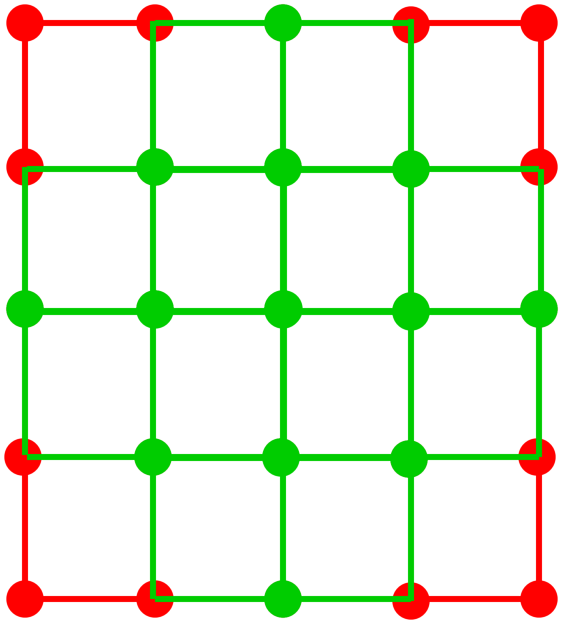
- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



Los Alamos
NATIONAL LABORATORY

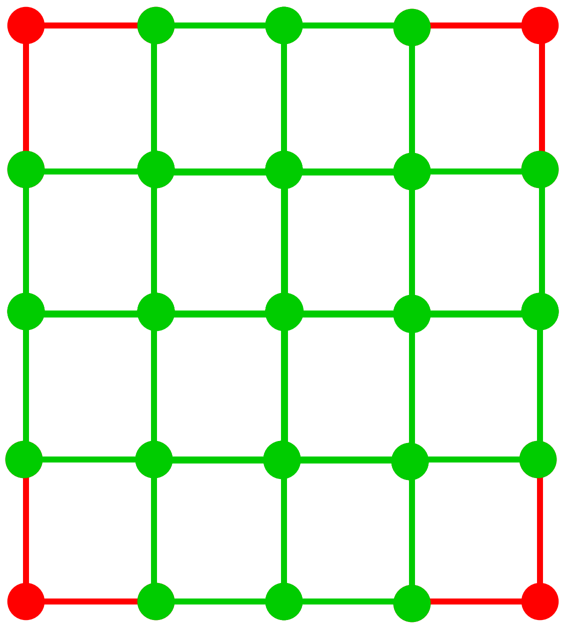
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



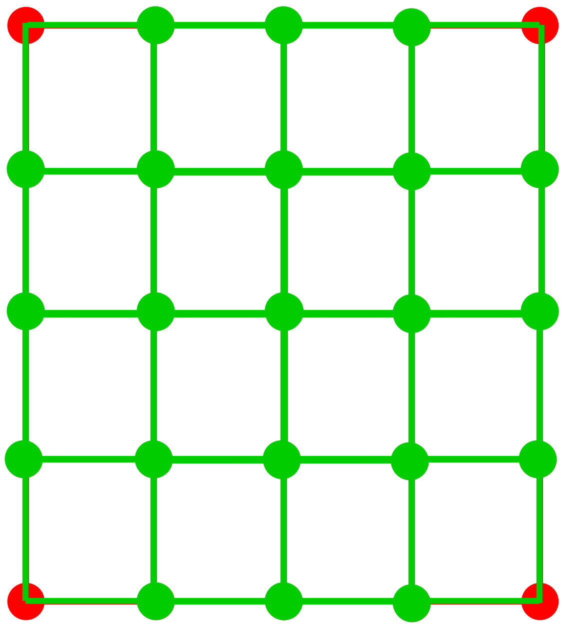
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



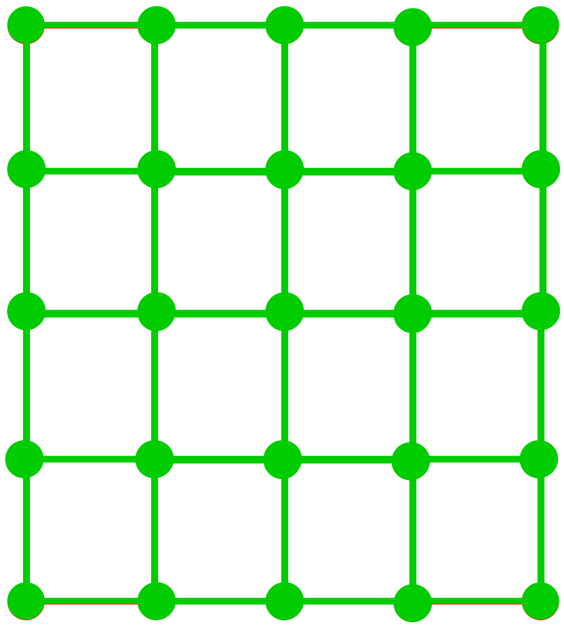
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



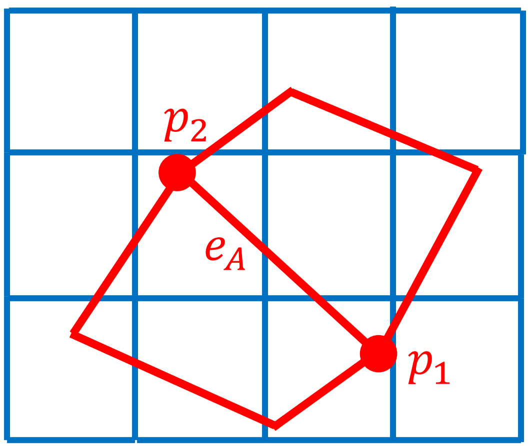
Edge tracking: Advancing wavefront

- In order to track an edge in mesh A through mesh B, we need to know where it starts in mesh B
  - Pick a point in mesh A
  - Determine where it is in mesh B
    - Single KD-tree log(n) search
  - All connected edges now know where they start
  - Track these edges through mesh B
  - All endpoints now know where they are
  - All edges connected to these endpoints now know where they start
  - Repeat until all edges have been tracked
- Repeat but track mesh B edges through mesh A



Los Alamos
NATIONAL LABORATORY

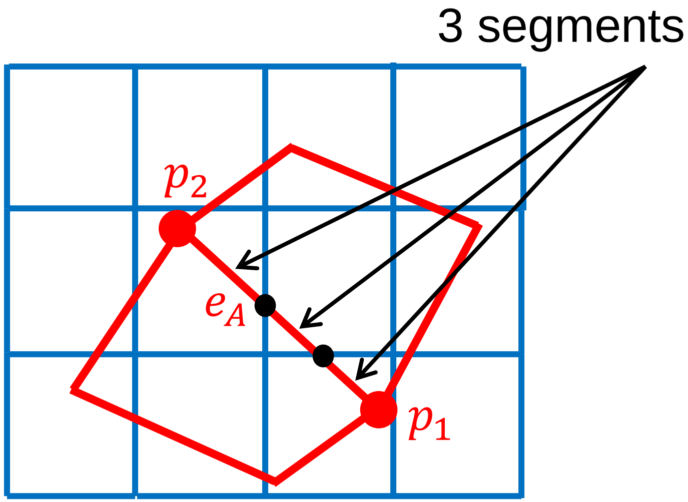
Edge tracking: Segment generation



- We must track each edge from its start to end and generate segments at each intersection
- Each edge e_A in mesh A is tracked through mesh B from start point p_1 to end point p_2
- At each intersection, a mesh A segment is generated
- For each mesh A segment, we must store:
  - Start and end coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$
  - Which zone z_B in mesh B the segment tracks through
  - The mesh A zones z_L and z_R that are on the left and right side of e_A
- Each segment bounds two intersection polygons
  - $z_B \cap z_L$
  - $z_B \cap z_R$
- Two types of segments
  - Donor mesh edge segments
  - Acceptor mesh edge segments
- Fluxes to/from z_R are negative due to reversed $\mathbf{x}_1$ and $\mathbf{x}_2$

Edge tracking: Segment generation

- We must track each edge from its start to end and generate segments at each intersection
- Each edge e_A in mesh A is tracked through mesh B from start point p_1 to end point p_2
- At each intersection, a mesh A segment is generated
- For each mesh A segment, we must store:
  - Start and end coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$
  - Which zone z_B in mesh B the segment tracks through
  - The mesh A zones z_L and z_R that are on the left and right side of e_A
- Each segment bounds two intersection polygons
  - $z_B \cap z_L$
  - $z_B \cap z_R$
- Two types of segments
  - Donor mesh edge segments
  - Acceptor mesh edge segments
- Fluxes to/from z_R are negative due to reversed $\mathbf{x}_1$ and $\mathbf{x}_2$



Los Alamos
NATIONAL LABORATORY

Edge tracking: Segment generation

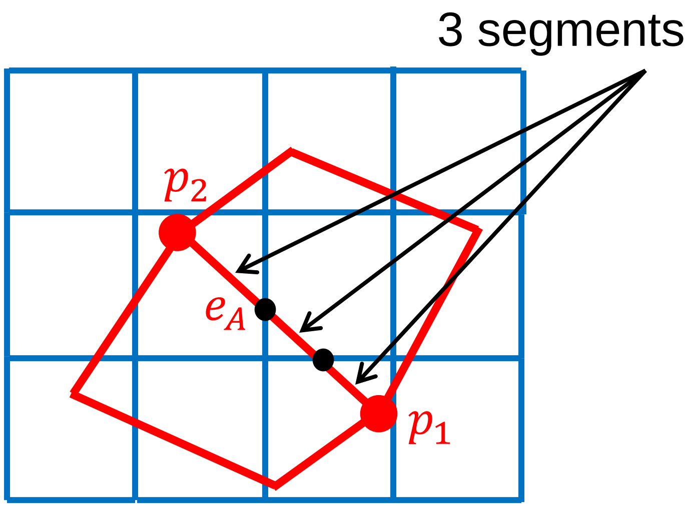
- We must track each edge from its start to end and generate segments at each intersection

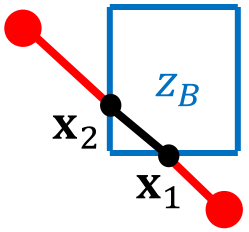
- Each edge e_A in mesh A is tracked through mesh B from start point p_1 to end point p_2

- At each intersection, a mesh A segment is generated

- For each mesh A segment, we must store:
  - Start and end coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$
  - Which zone z_B in mesh B the segment tracks through
  - The mesh A zones z_L and z_R that are on the left and right side of e_A
- Each segment bounds two intersection polygons
  - $z_B \cap z_L$
  - $z_B \cap z_R$
- Two types of segments
  - Donor mesh edge segments
  - Acceptor mesh edge segments

- Fluxes to/from z_R are negative due to reversed $\mathbf{x}_1$ and $\mathbf{x}_2$





Edge tracking: Segment generation

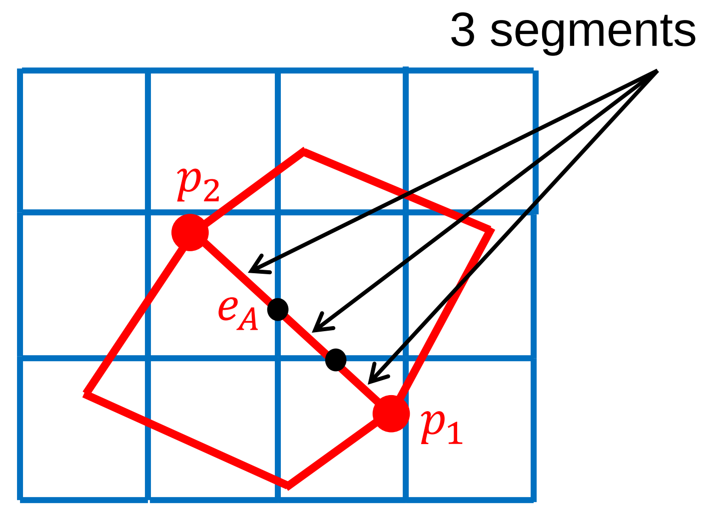
- We must track each edge from its start to end and generate segments at each intersection

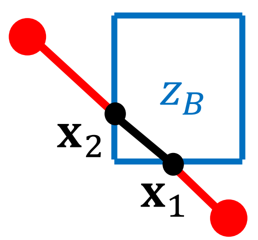
- Each edge e_A in mesh A is tracked through mesh B from start point p_1 to end point p_2

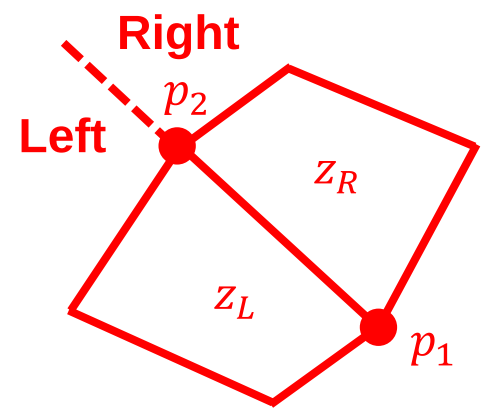
- At each intersection, a mesh A segment is generated

- For each mesh A segment, we must store:
  - Start and end coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$
  - Which zone z_B in mesh B the segment tracks through
  - The mesh A zones z_L and z_R that are on the left and right side of e_A
- Each segment bounds two intersection polygons
  - $z_B \cap z_L$
  - $z_B \cap z_R$
- Two types of segments
  - Donor mesh edge segments
  - Acceptor mesh edge segments

- Fluxes to/from z_R are negative due to reversed $\mathbf{x}_1$ and $\mathbf{x}_2$

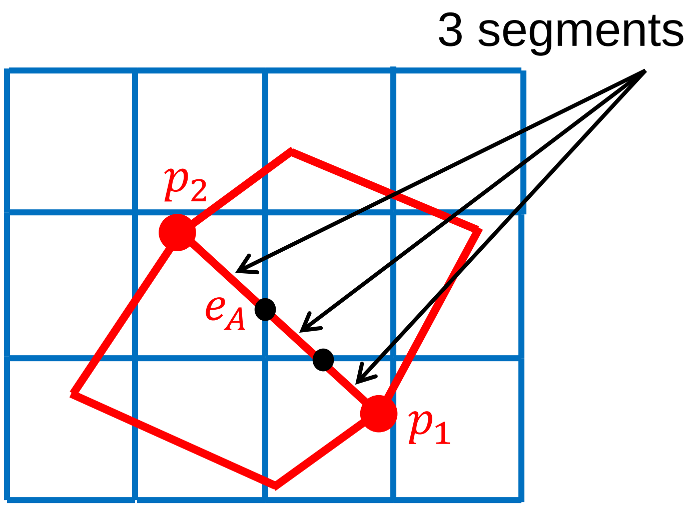


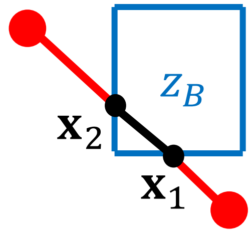


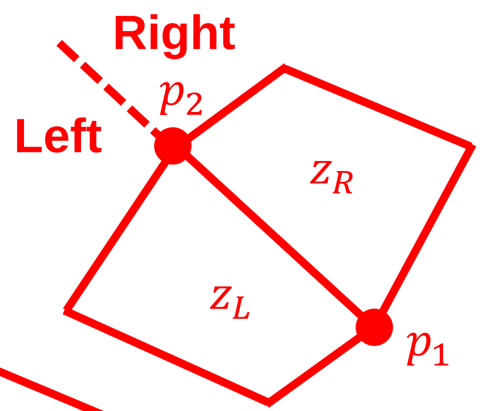


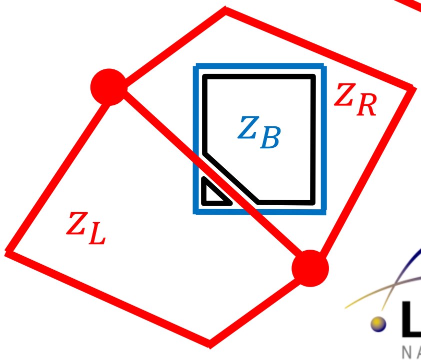
Edge tracking: Segment generation

- We must track each edge from its start to end and generate segments at each intersection
- Each edge e_A in mesh A is tracked through mesh B from start point p_1 to end point p_2
- At each intersection, a mesh A segment is generated
- For each mesh A segment, we must store:
  - Start and end coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$
  - Which zone z_B in mesh B the segment tracks through
  - The mesh A zones z_L and z_R that are on the left and right side of e_A
- Each segment bounds two intersection polygons
  - $z_B \cap z_L$
  - $z_B \cap z_R$
- Two types of segments
  - Donor mesh edge segments
  - Acceptor mesh edge segments
- Fluxes to/from z_R are negative due to reversed $\mathbf{x}_1$ and $\mathbf{x}_2$









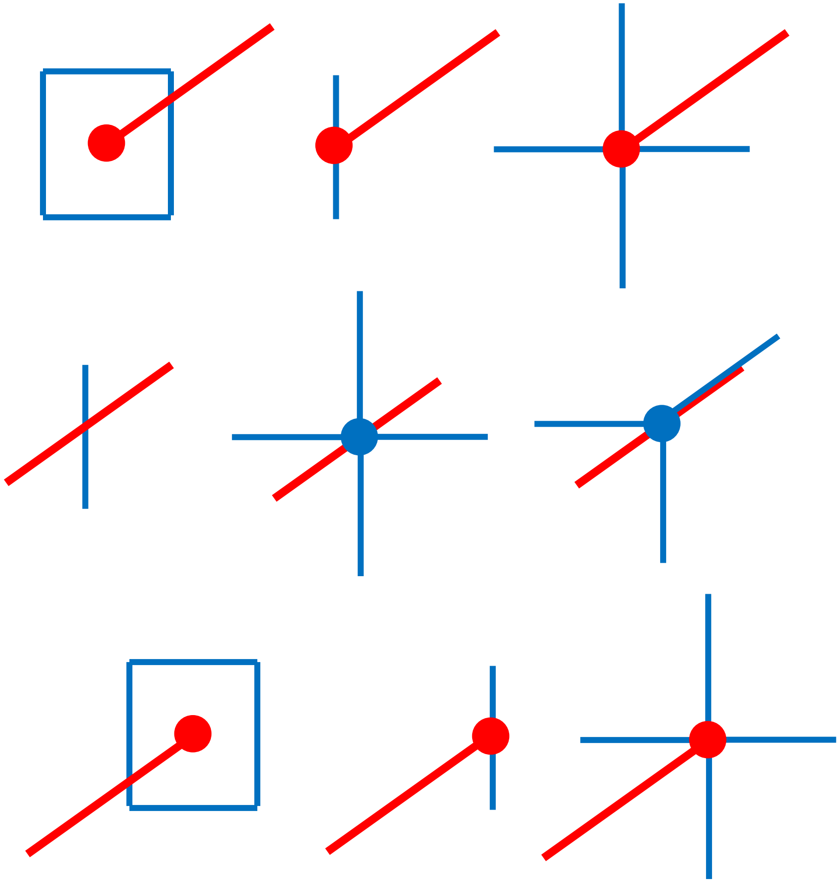
Edge tracking

- The start, intersection and end conditions while edge tracking are the key to the method

- Three conditions are possible for the start point of a segment:
  - Starts within a zone
  - Starts exactly on an edge (between its endpoints)
  - Starts exactly on a point

- Three types of intersection are possible
  - Edge-edge intersection
  - Exact edge-point intersection
  - Exactly collinear edges (special but common case)

- Three conditions are possible for the end point of a segment:
  - Ends within a zone
  - Ends exactly on an edge (between its endpoints)
  - Ends exactly on a point



- The “trick” is to identify the exact cases
  - Finite precision computers aren’t exact
  - This introduces possible inconsistencies or even geometrically impossible situations
  - Zero tolerances: “The road to hell is paved with tolerances.”

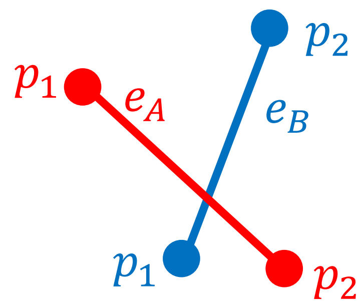
Edge tracking: Edge intersection

- Intersecting a pair of edges, e_A from mesh A and e_B from mesh B
- Identify which side s each endpoint of one edge is relative to the other edge: left, right, or exactly on
- Use cross products

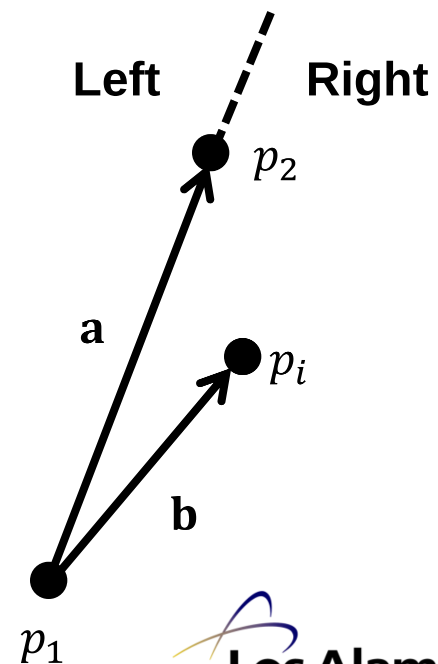
$$\mathbf{a} = \mathbf{x}_{p_2} - \mathbf{x}_{p_1}$$

$$\mathbf{b}_i = \mathbf{x}_{p_i} - \mathbf{x}_{p_1}$$

$$s = \begin{cases} -1 : (\mathbf{b} \times \mathbf{a}) < 0 \\ 0 : (\mathbf{b} \times \mathbf{a}) = 0 \\ +1 : (\mathbf{b} \times \mathbf{a}) > 0 \end{cases}$$



- $s = 0$ means $(\mathbf{b} \times \mathbf{a})$ is exactly zero
- Four values:
  - s_1 : which side of e_B point p_1 is on
  - s_2 : which side of e_B point p_2 is on
  - s_1 : which side of e_A point p_1 is on
  - s_2 : which side of e_A point p_2 is on



Edge tracking: Edge-edge intersection

- Endpoints of both edges are on opposite sides of the other edge

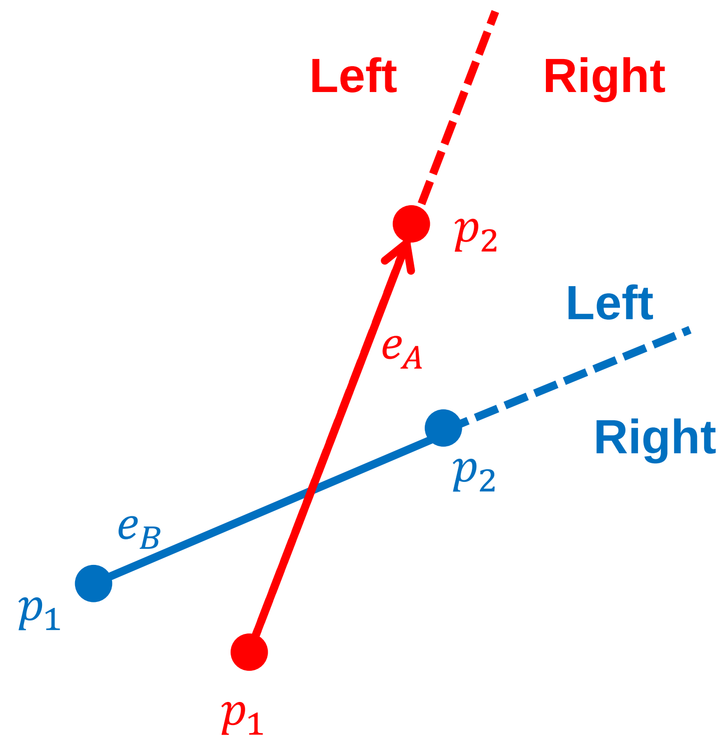
  $s_1 = -s_2, \; s_1 = -s_2$

- Compute intersection location $\mathbf{x}_{int}$
  - Must use the same exact math regardless of tracking mesh A through mesh B or mesh B through mesh A
  - Otherwise finite precision will bite you

$$\mathbf{x}_{int} = \mathbf{x}_1 + u\mathbf{a}$$

$$u = \min(1, \frac{\|(\mathbf{b}_1 \times \mathbf{a})\|}{\|(\mathbf{b}_1 \times \mathbf{a})\| + \|(\mathbf{b}_2 \times \mathbf{a})\|})$$

- Remainder of e_A tracks through zone on s_2 side of e_B



Edge tracking: Edge-edge intersection

- Endpoints of both edges are on opposite sides of the other edge

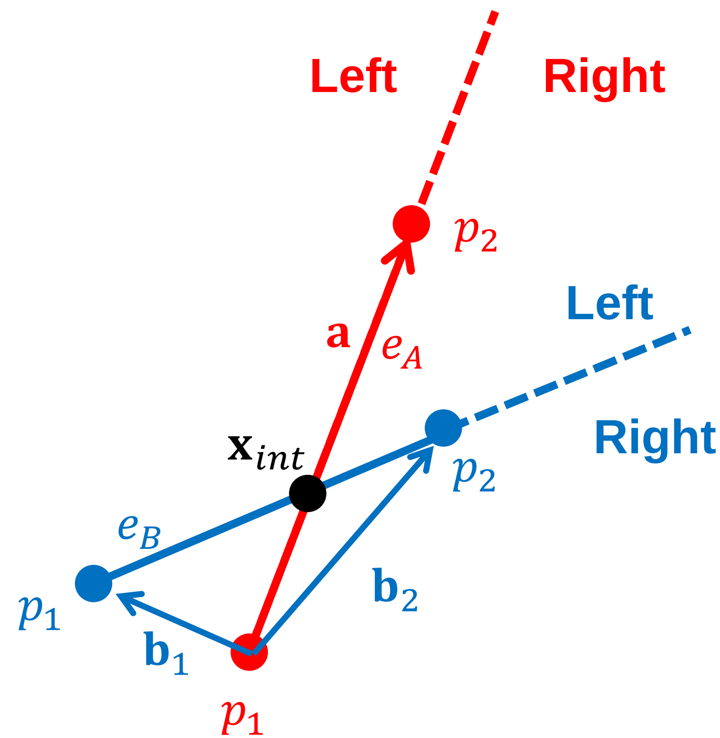
$$s_1 = -s_2, \quad s_1 = -s_2$$

- Compute intersection location $\mathbf{x}_{int}$
  - Must use the same exact math regardless of tracking mesh A through mesh B or mesh B through mesh A
  - Otherwise finite precision will bite you

$$\mathbf{x}_{int} = \mathbf{x}_1 + u\mathbf{a}$$

$$u = \min\left(1, \frac{\|(\mathbf{b}_1 \times \mathbf{a})\|}{\|(\mathbf{b}_1 \times \mathbf{a})\| + \|(\mathbf{b}_2 \times \mathbf{a})\|}\right)$$

- Remainder of e_A tracks through zone on s_2 side of e_B



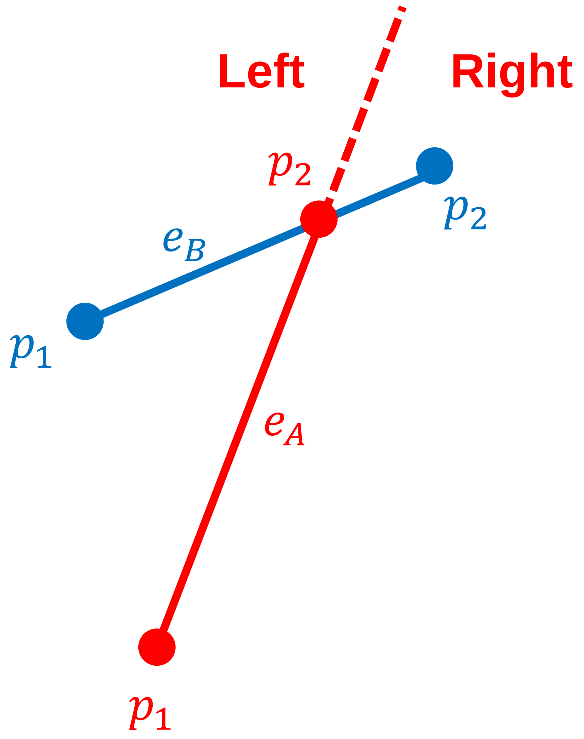
Edge tracking: Edge ends on edge

- Endpoint of e_A is exactly on e_B, e_B endpoints p_1 and p_2 on opposite sides of e_A
  - $s_1 \neq 0, s_2 = 0,\ s_1 = -s_2$

- Intersection point is at p_2

$$\mathbf{x}_{int} = \mathbf{x}_2$$

- Other mesh A edges starting at p_2 begin exactly on e_B



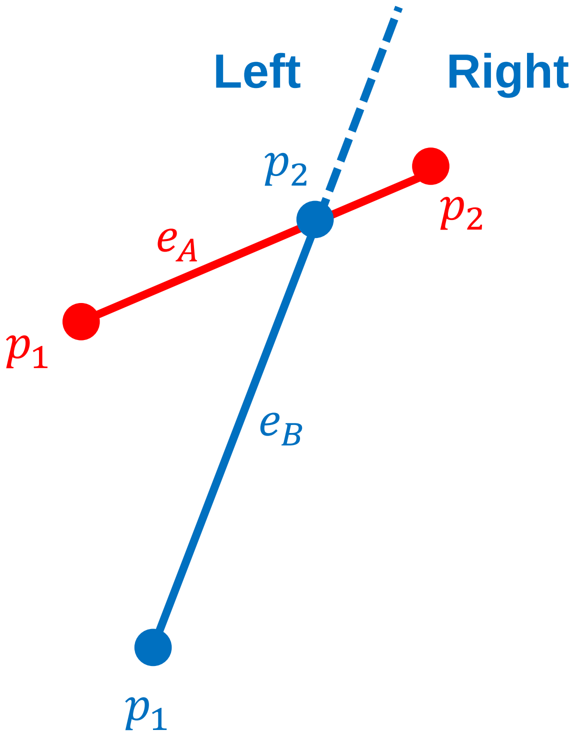
Edge tracking: Edge-point intersection

- Edge e_A exactly intersects endpoint of e_B
  - $s_1 = -s_2$
  - and $s_1 \neq 0$, $s_2 = 0$
  - or $s_2 \neq 0$, $s_1 = 0$

- Intersection point is at p_2

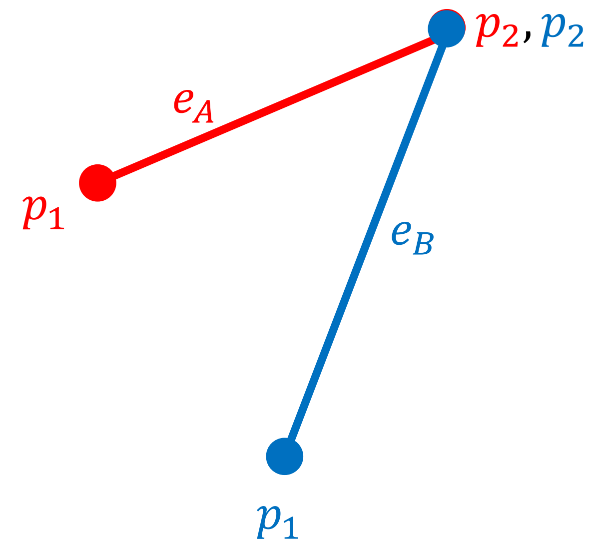
$$\mathbf{x}_{int} = \mathbf{x}_2$$

- Remainder of e_A tracks from point p_2



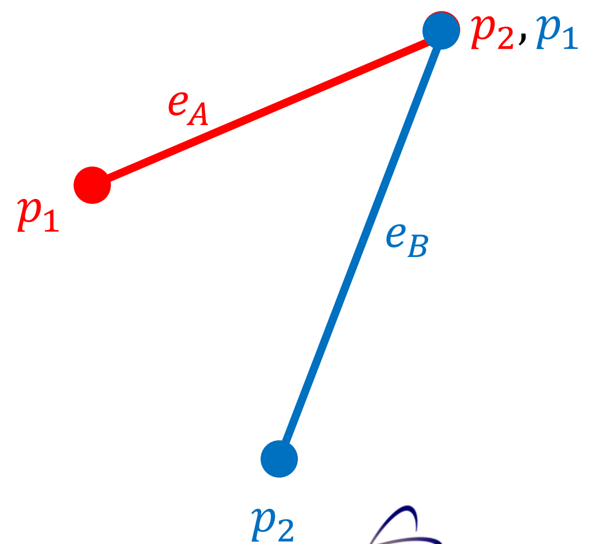
Edge tracking: Coincident points

- Edge e_A endpoint p_2 exactly on one of the endpoints of e_B
  - $s_2 = 0,$
  - and $s_2 \neq 0, s_1 = 0, \mathbf{x}_1 = \mathbf{x}_2$
  - or $s_1 \neq 0, s_2 = 0, \mathbf{x}_2 = \mathbf{x}_2$



- Intersection point is at p_2

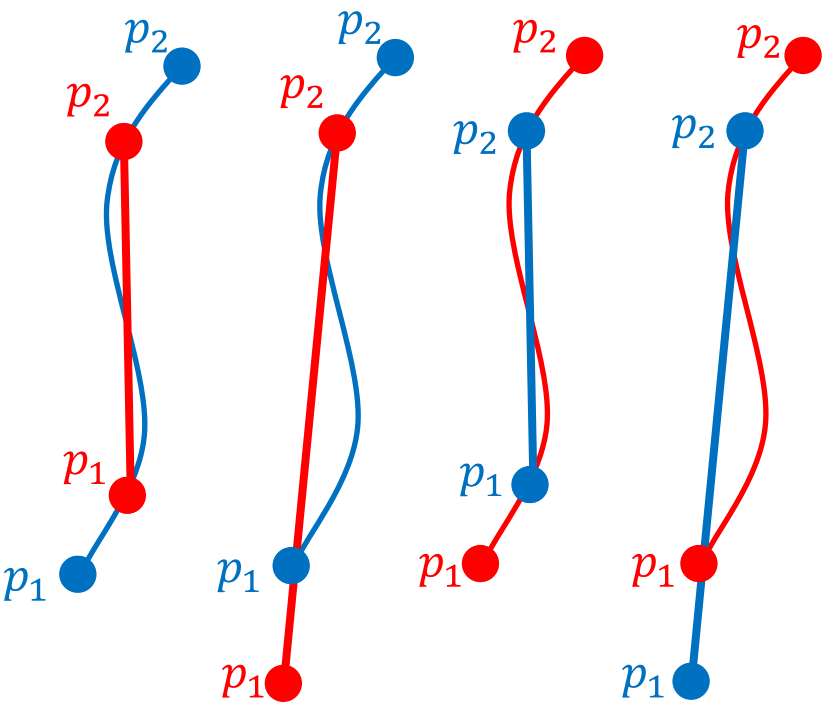
$$\mathbf{x}_{int} = \mathbf{x}_2$$



- Other mesh A edges starting at p_2 begin exactly on p_1 or p_2

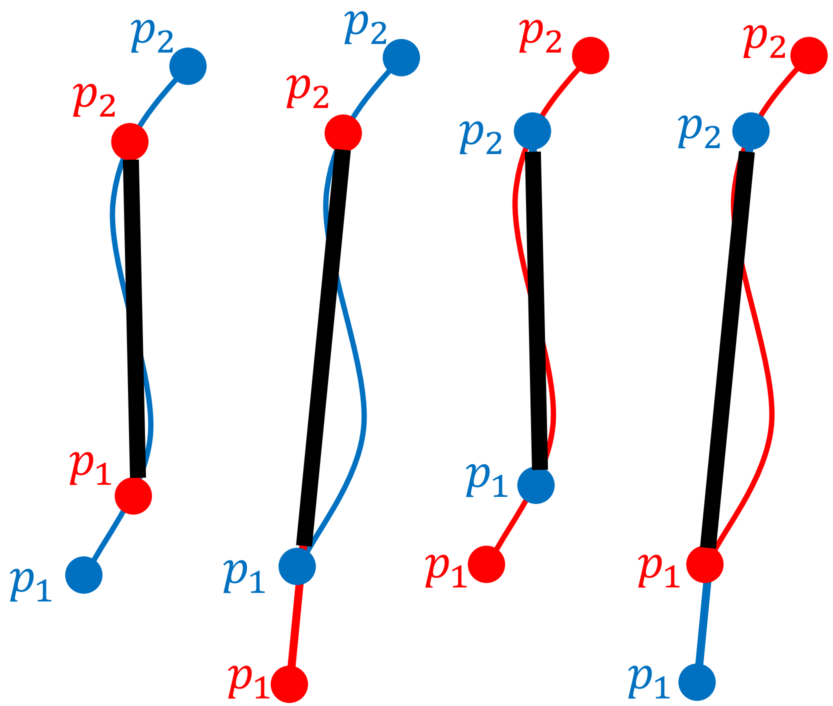
Edge tracking: Collinear edge intersection

- Many possible combinations
- If $s = 0$ for start and end of segment, edges are exactly collinear
- The zone through which the segment tracks is ambiguous
- Force consistent selection of zone
  - If tracking mesh A through mesh B, segment tracks through e_B right zone
  - If tracking mesh B through mesh A, segment tracks through mesh A zone that is on left side of e_B
- Generates a degenerate polygon that can be culled later



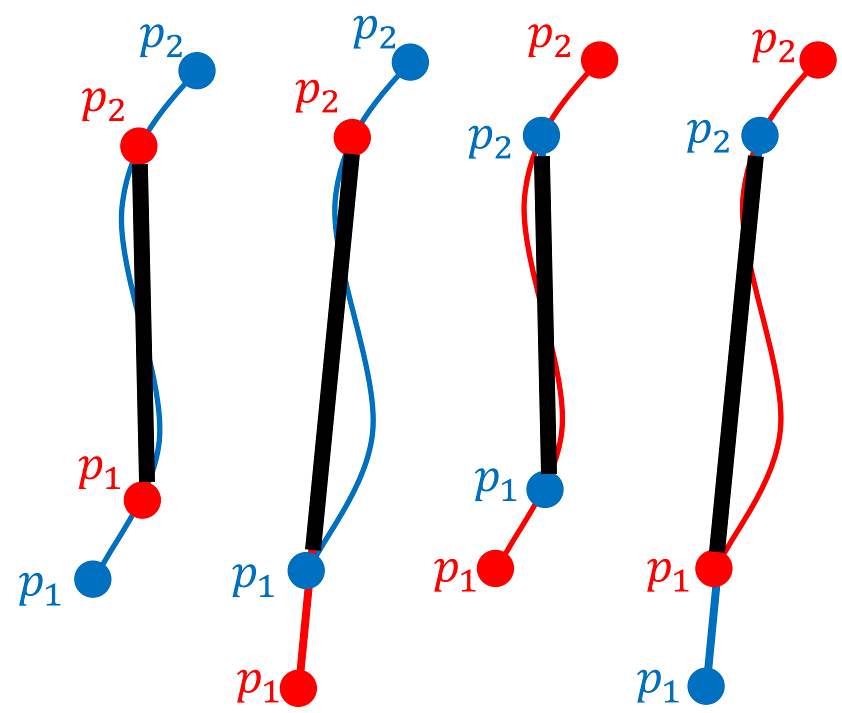
Edge tracking: Collinear edge intersection

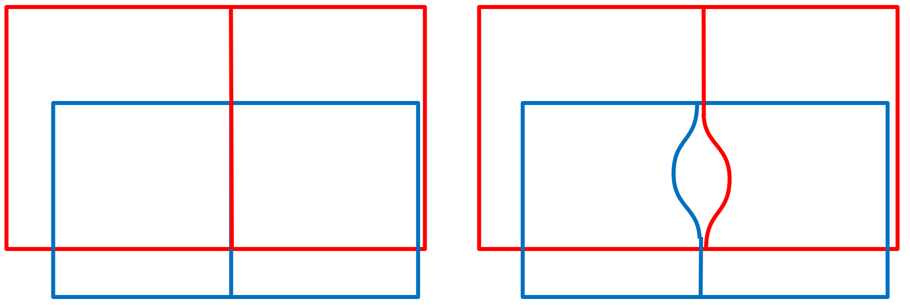
- Many possible combinations
- If $s = 0$ for start and end of segment, edges are exactly collinear
- The zone through which the segment tracks is ambiguous
- Force consistent selection of zone
  - If tracking mesh A through mesh B, segment tracks through e_B right zone
  - If tracking mesh B through mesh A, segment tracks through mesh A zone that is on left side of e_B
- Generates a degenerate polygon that can be culled later



Edge tracking: Collinear edge intersection

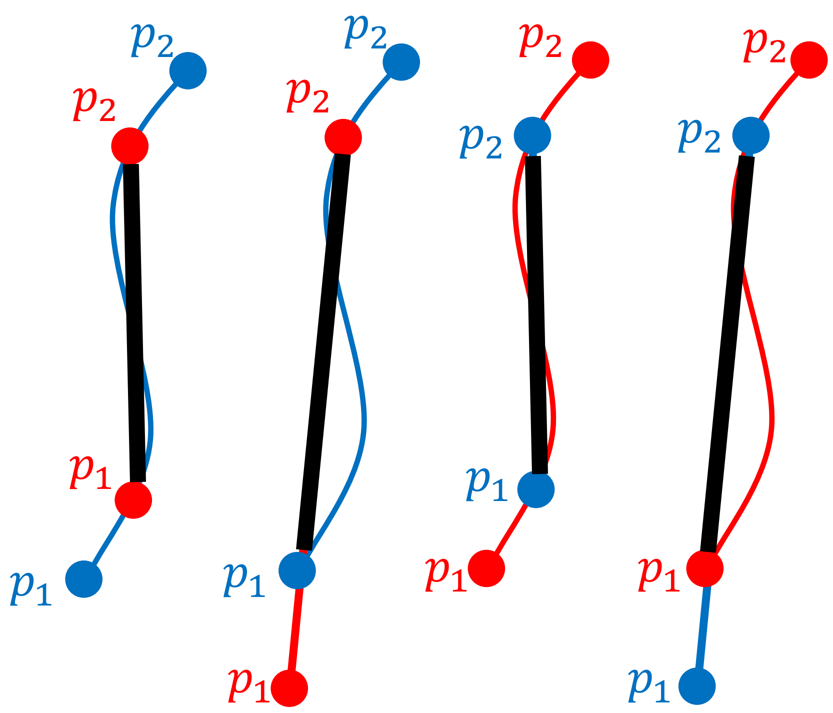
- Many possible combinations
- If $s = 0$ for start and end of segment, edges are exactly collinear
- The zone through which the segment tracks is ambiguous
- Force consistent selection of zone
  - If tracking mesh A through mesh B, segment tracks through e_B right zone
  - If tracking mesh B through mesh A, segment tracks through mesh A zone that is on left side of e_B
- Generates a degenerate polygon that can be culled later

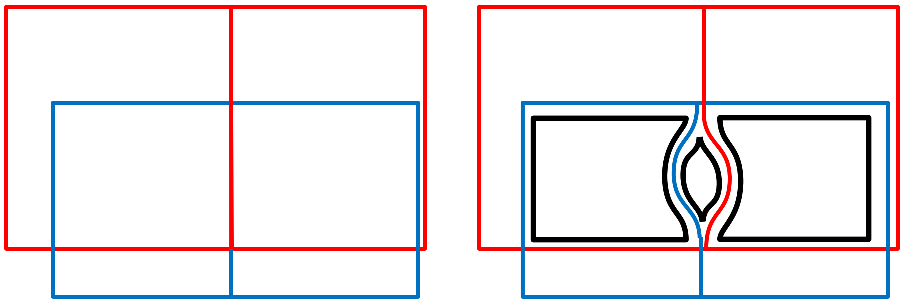




Edge tracking: Collinear edge intersection

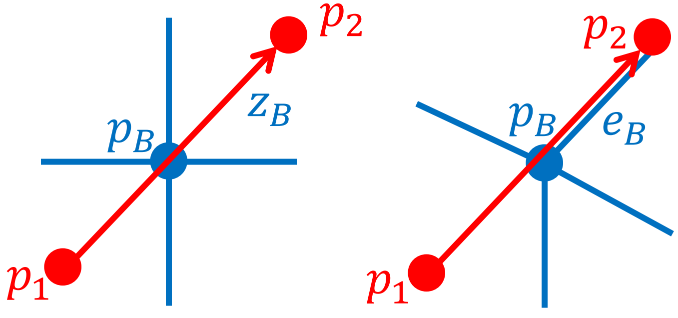
- Many possible combinations
- If $s = 0$ for start and end of segment, edges are exactly collinear
- The zone through which the segment tracks is ambiguous
- Force consistent selection of zone
  - If tracking mesh A through mesh B, segment tracks through e_B right zone
  - If tracking mesh B through mesh A, segment tracks through mesh A zone that is on left side of e_B
- Generates a degenerate polygon that can be culled later

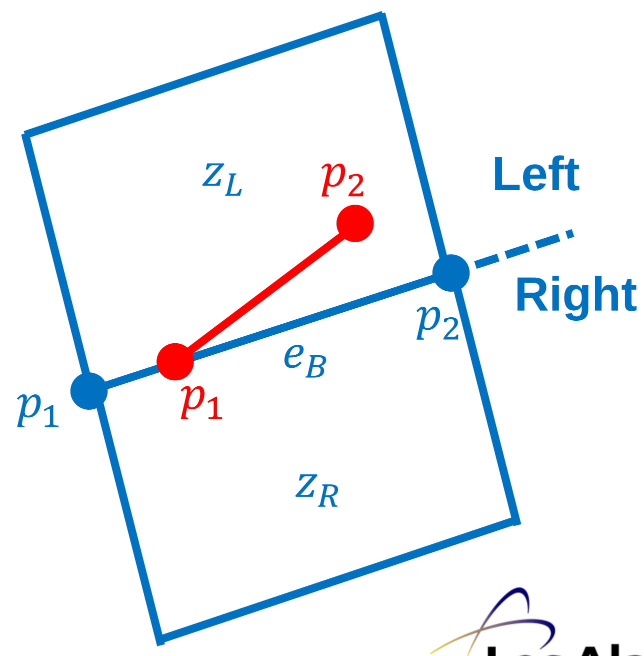




Edge tracking: Start conditions

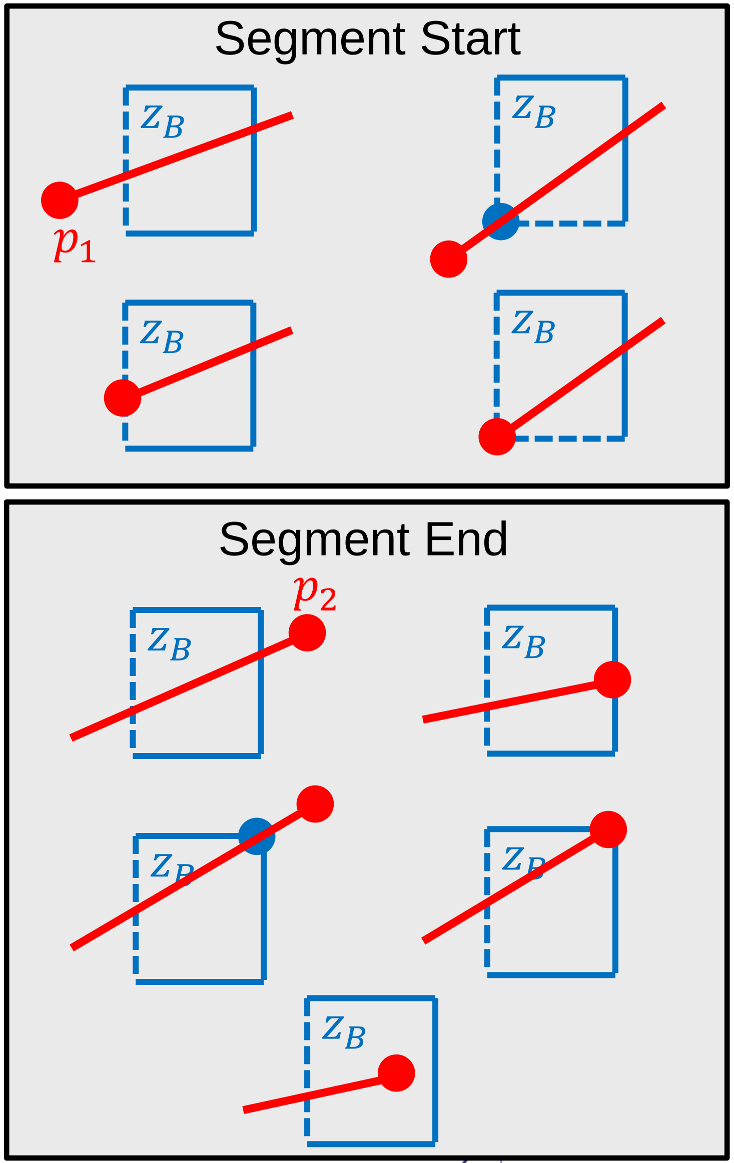
- While tracking edge e_A, the start condition for each new segment is known: in zone, on edge, on point
- On point p_B:
  - Determine whether e_A tracks through adjacent zone z_B or edge e_B
  - Re-evaluate e_A with new start condition
- On edge e_B:
  - Determine whether e_A is collinear with e_A or tracks through one of its adjacent zones z_L or z_R
  - If not collinear, re-evaluate e_A with new start condition





Edge tracking: Through zones

- Segments are only generated for collinear edges or when tracking through a zone z_B
- Intersect e_A with all edges bounding z_B except entry edges
- Temporarily store every intersection
  - Type of intersection (edge or point)
  - Intersected index e_B or p_B
  - Intersection coordinates $\mathbf{x}_{int}$
  - Parametric u value
  - Whether e_A terminates or not
- If intersections found, choose intersection with minimum u value
  - Generate segment for e_A tracking through z_B
  - Use stored intersection information to continue tracking
- If no intersection found, edge terminates within z_B, done tracking e_A
  - Other mesh A edges starting at p_2 will begin tracking within zone z_B

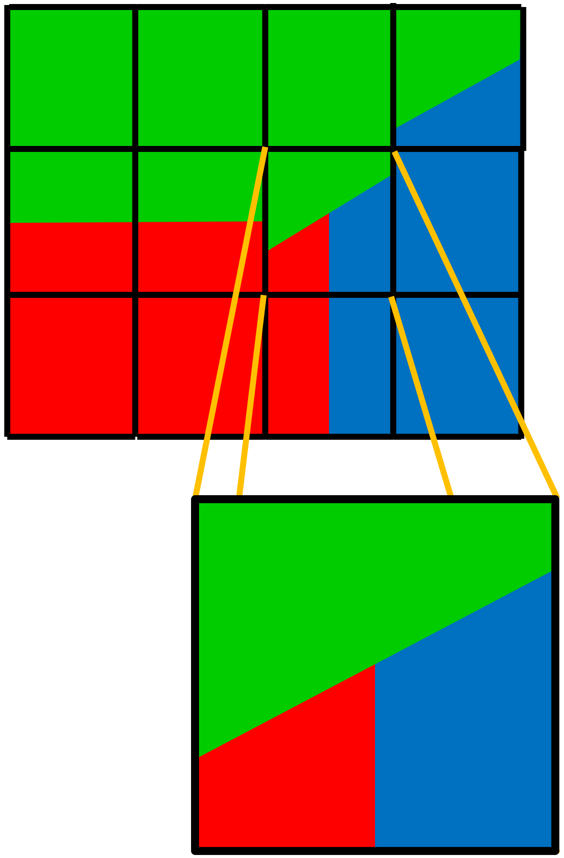


Remapping using polygons or segments

- Polygon fluxes are required for remapping multi-material donor zones
  - VOF interface reconstructions must be considered
  - Covered in a few slides
- Segment fluxes can be used for remapping single-material donor zones
- Flux equation valid for segments or polygons
  - $F = (f(\mathbf{x}_c) - \mathbf{G} \cdot \mathbf{x}_c) J_0 + \mathbf{G} \cdot \mathbf{J}$
  - $f(\mathbf{x}_c)$, $\mathbf{x}_c$, and $\mathbf{G}$ are donor zone values
  - F is a flux added to the acceptor zone
  - J_0 and $\mathbf{J}$ can be segment or polygon values

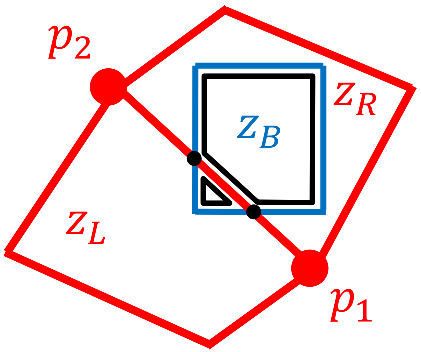
$$J_p = \sum_{e=1}^{n} J_e$$

- Must construct polygons from segments

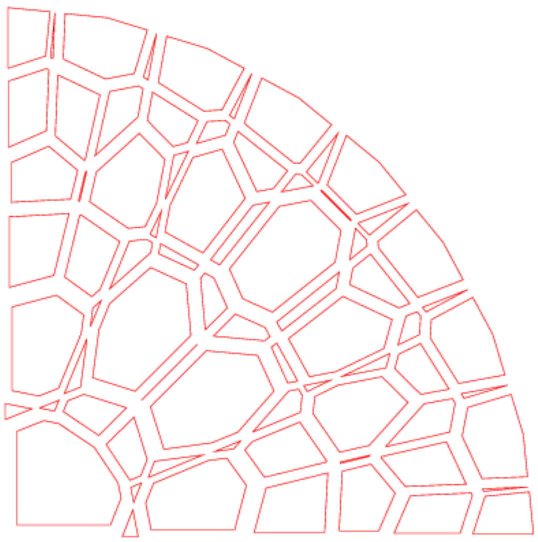


Polygon generation

- Two types of segments
  - Donor mesh edge segments
    - z_B from acceptor mesh
    - z_L and z_R from donor mesh
    - Contribute to $z_L \cap z_B$ and $z_R \cap z_B$
  - Acceptor mesh edge segments
    - z_B from donor mesh
    - z_L and z_R from acceptor mesh
    - Contribute to $z_B \cap z_L$ and $z_B \cap z_R$



- Identify all segments intersecting the same donor and acceptor zone pair
  - These define the boundary of the same intersection polygon



- Construct polygon from segments
  - Order vertices counter-clockwise

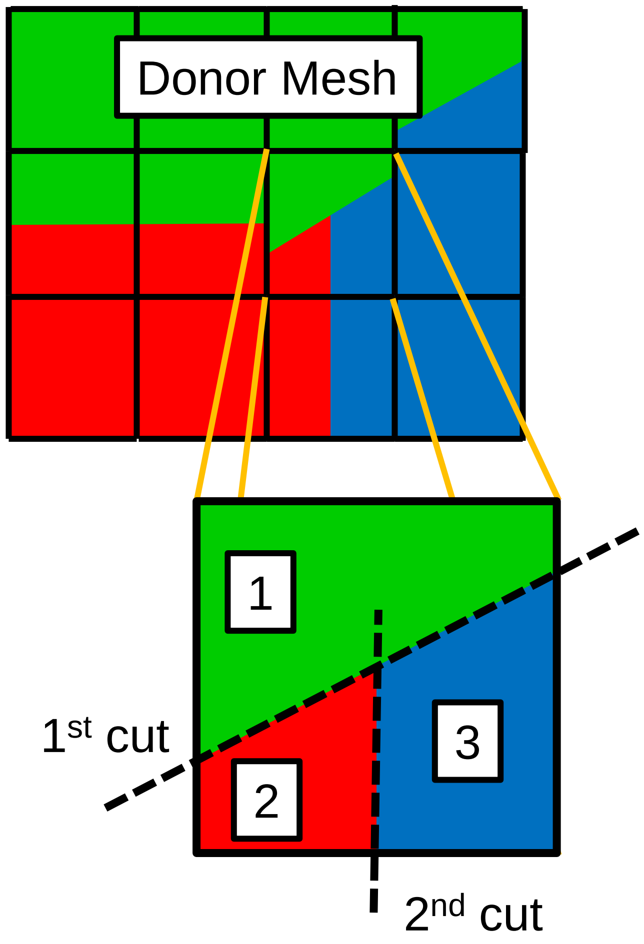
Pure material sub-polygons

- Interfaces are reconstructed in multi-material zones (using J. Mosso code)
- Interface reconstruction (IR) module computes and stores interfaces (line and outward normal)
- Pure material sub-polygons obtained by cutting intersection polygon with interfaces
  - IR module returns remainder polygon after each cut
  - Compute $J_{remainder}$ for remainder polygon
  - J values for cut-off polygon recovered

$$J_{whole} = J_{cutoff} + J_{remainder}$$

$$J_{cutoff} = J_{whole} - J_{remainder}$$

  - N-1 cuts for N-material zone
  - Repeat for each cut
  - N^{th} material sub-polygon is remainder of cut N-1
- Only the sub-polygon J values are needed
- Remap is still 2^{nd}-order



Los Alamos NATIONAL LABORATORY

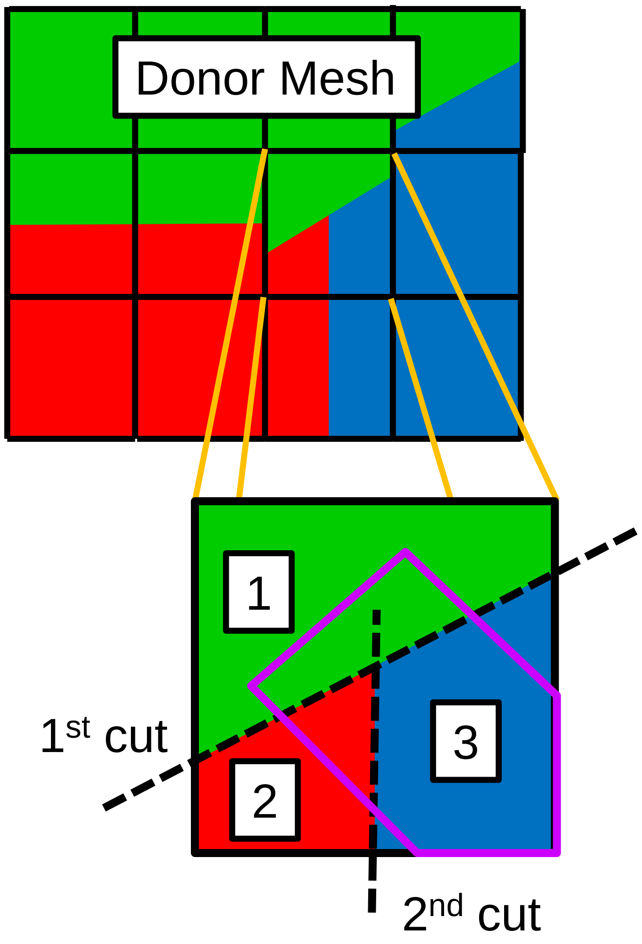
Pure material sub-polygons

- Interfaces are reconstructed in multi-material zones (using J. Mosso code)
- Interface reconstruction (IR) module computes and stores interfaces (line and outward normal)
- Pure material sub-polygons obtained by cutting intersection polygon with interfaces
  - IR module returns remainder polygon after each cut
  - Compute $J_{remainder}$ for remainder polygon
  - J values for cut-off polygon recovered

$$J_{whole} = J_{cutoff} + J_{remainder}$$

$$J_{cutoff} = J_{whole} - J_{remainder}$$

  - N-1 cuts for N-material zone
  - Repeat for each cut
  - N[th] material sub-polygon is remainder of cut N-1
- Only the sub-polygon J values are needed
- Remap is still 2[nd]-order



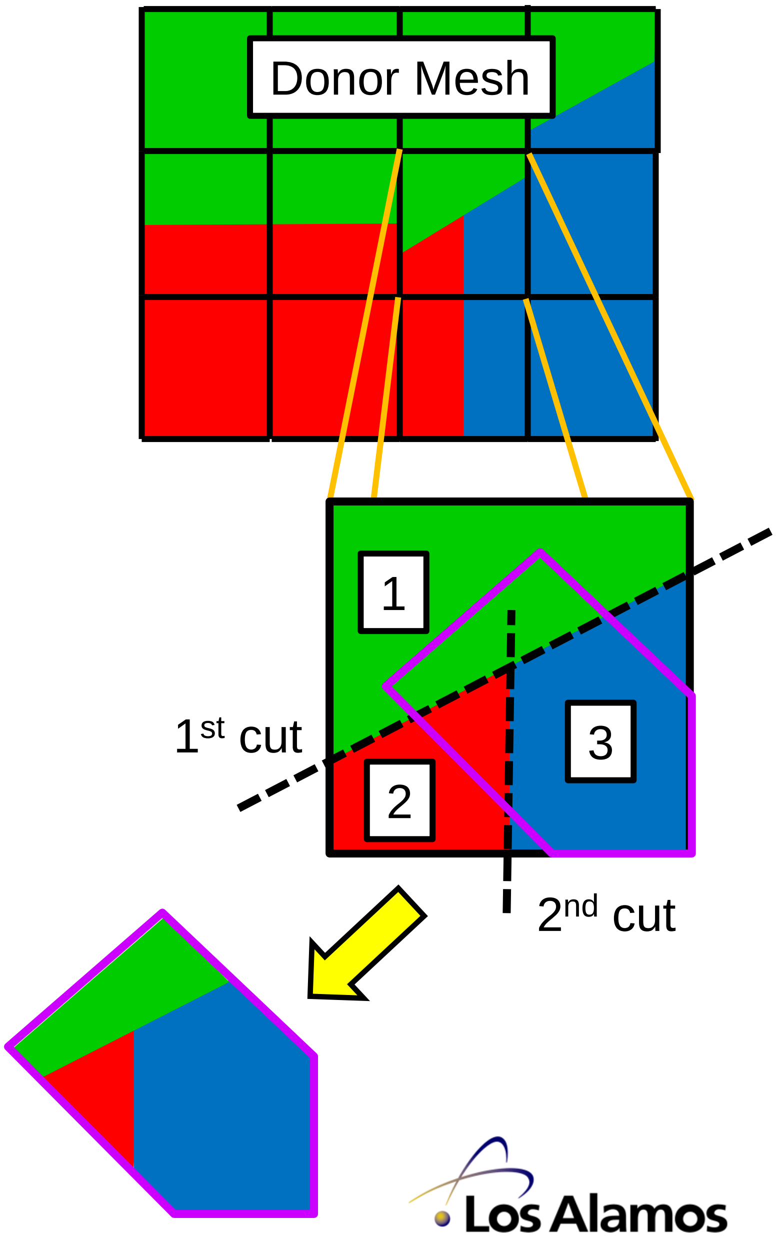
Pure material sub-polygons

- Interfaces are reconstructed in multi-material zones (using J. Mosso code)
- Interface reconstruction (IR) module computes and stores interfaces (line and outward normal)
- Pure material sub-polygons obtained by cutting intersection polygon with interfaces
  - IR module returns remainder polygon after each cut
  - Compute $J_{remainder}$ for remainder polygon
  - J values for cut-off polygon recovered

$$J_{whole} = J_{cutoff} + J_{remainder}$$

$$J_{cutoff} = J_{whole} - J_{remainder}$$

  - N-1 cuts for N-material zone
  - Repeat for each cut
  - N^{th} material sub-polygon is remainder of cut N-1
- Only the sub-polygon J values are needed
- Remap is still 2^{nd}-order



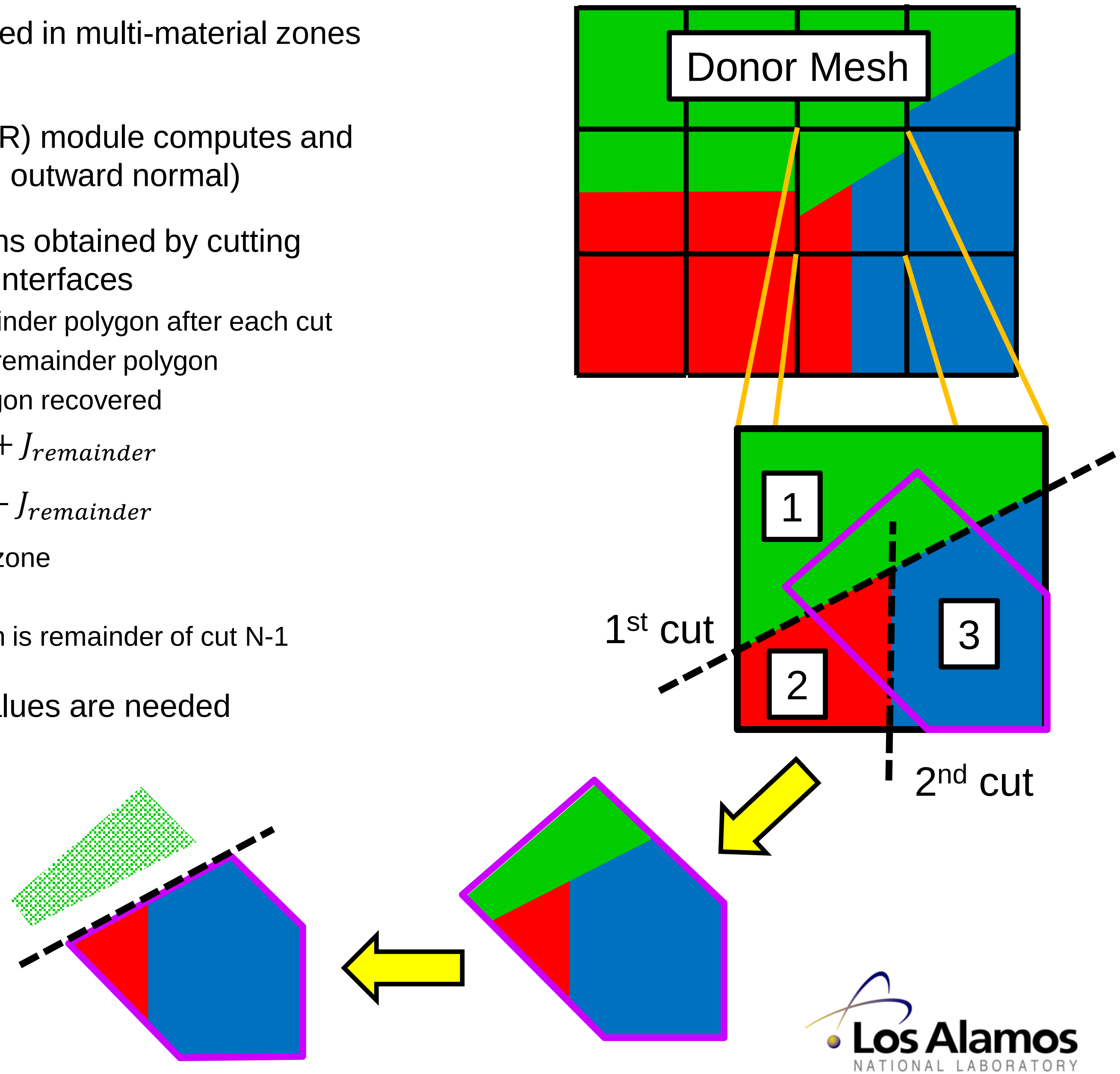
Pure material sub-polygons

- Interfaces are reconstructed in multi-material zones (using J. Mosso code)
- Interface reconstruction (IR) module computes and stores interfaces (line and outward normal)
- Pure material sub-polygons obtained by cutting intersection polygon with interfaces
  - IR module returns remainder polygon after each cut
  - Compute $J_{remainder}$ for remainder polygon
  - J values for cut-off polygon recovered

$$J_{whole} = J_{cutoff} + J_{remainder}$$

$$J_{cutoff} = J_{whole} - J_{remainder}$$

  - N-1 cuts for N-material zone
  - Repeat for each cut
  - N^{th} material sub-polygon is remainder of cut N-1
- Only the sub-polygon J values are needed
- Remap is still 2^{nd}-order



Los Alamos NATIONAL LABORATORY

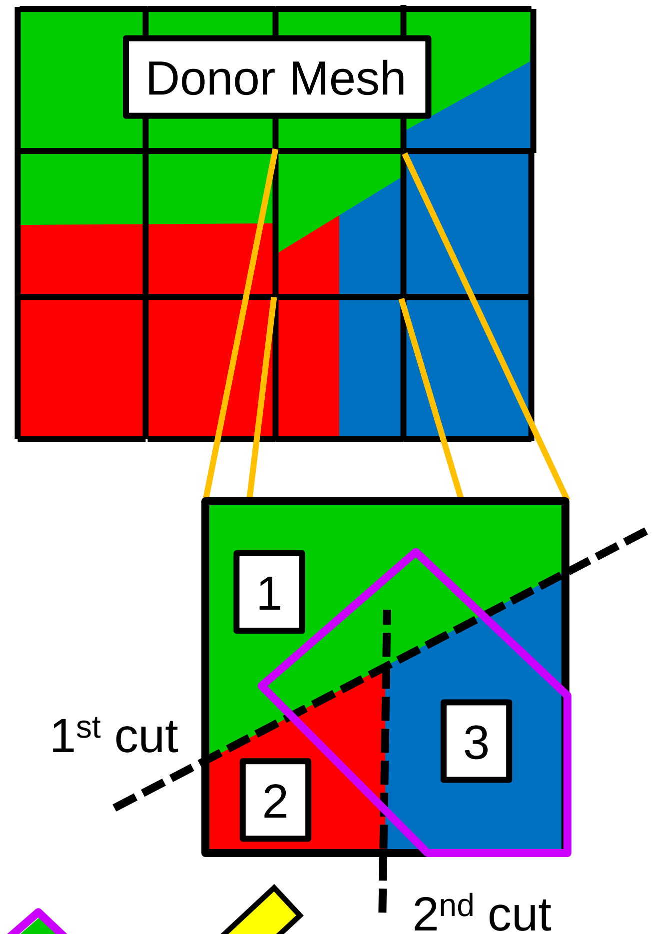
Pure material sub-polygons

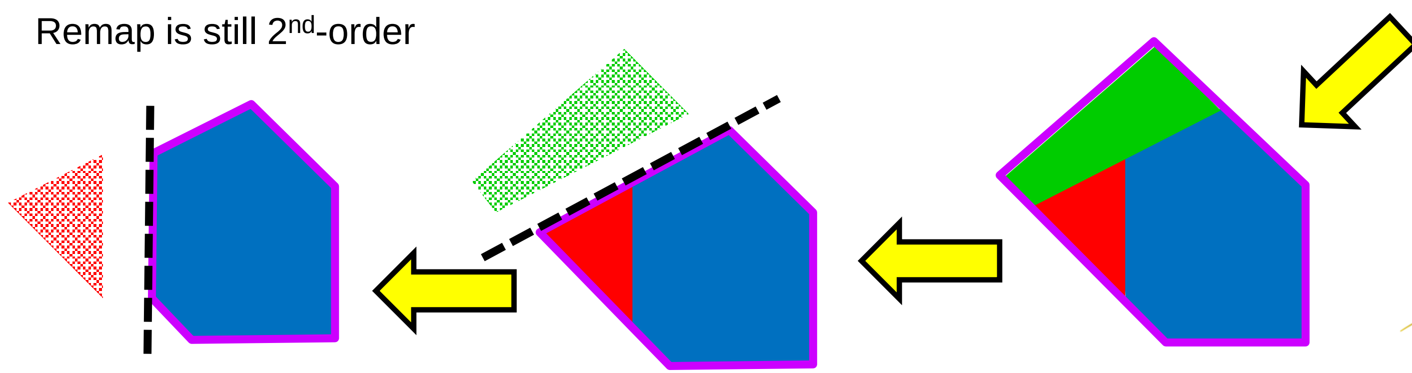
- Interfaces are reconstructed in multi-material zones (using J. Mosso code)
- Interface reconstruction (IR) module computes and stores interfaces (line and outward normal)
- Pure material sub-polygons obtained by cutting intersection polygon with interfaces
  - IR module returns remainder polygon after each cut
  - Compute $J_{remainder}$ for remainder polygon
  - J values for cut-off polygon recovered

$$J_{whole} = J_{cutoff} + J_{remainder}$$

$$J_{cutoff} = J_{whole} - J_{remainder}$$

  - N-1 cuts for N-material zone
  - Repeat for each cut
  - N^{th} material sub-polygon is remainder of cut N-1
- Only the sub-polygon J values are needed
- Remap is still 2^{nd}-order





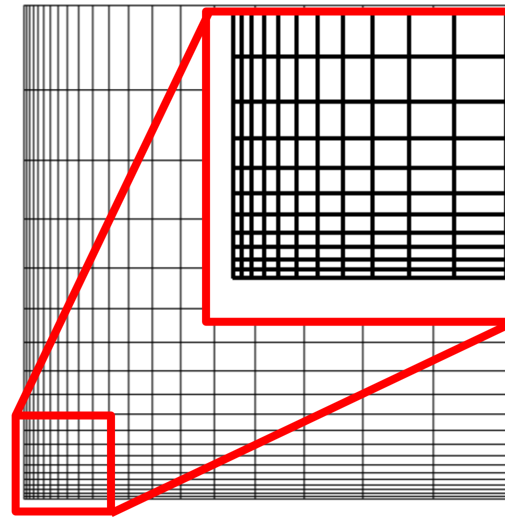
Parallelization

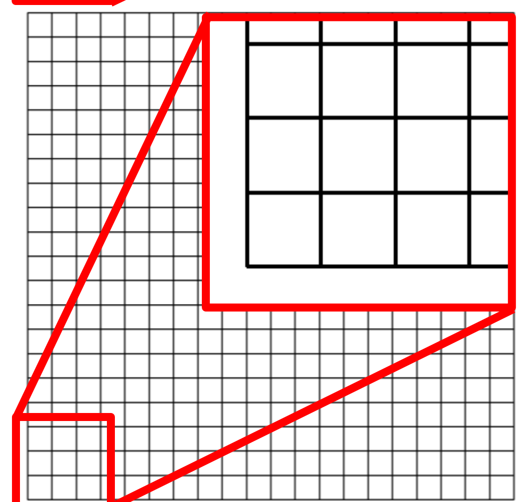
Results: Accuracy

- Remap of linear field
  - Should recover the linear field
- Remap of non-linear field
  - Should converge at 2^{nd} order with increasing resolution
- Cartesian and polygonal meshes
  - Demonstrate generality

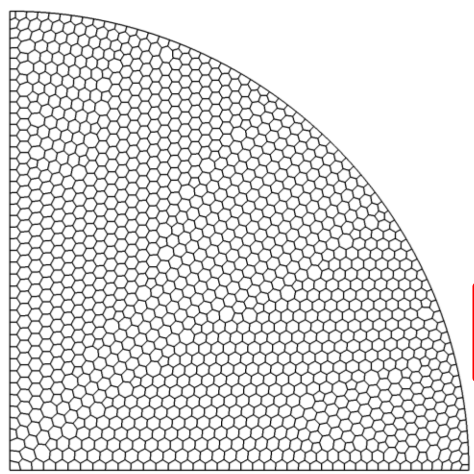
Results: Accuracy

- Cartesian and polygonal meshes
- Linear field: $f(x,y) = 2x + 3y + 4$
- Laplacian relaxer
- Force correct $\mathbf{G} = \langle 2,3 \rangle$ in boundary zones
- Remap errors are O(E-14)
- Also demonstrates that mesh can be relaxed more than a zone size
  - Limitation for swept-face advection





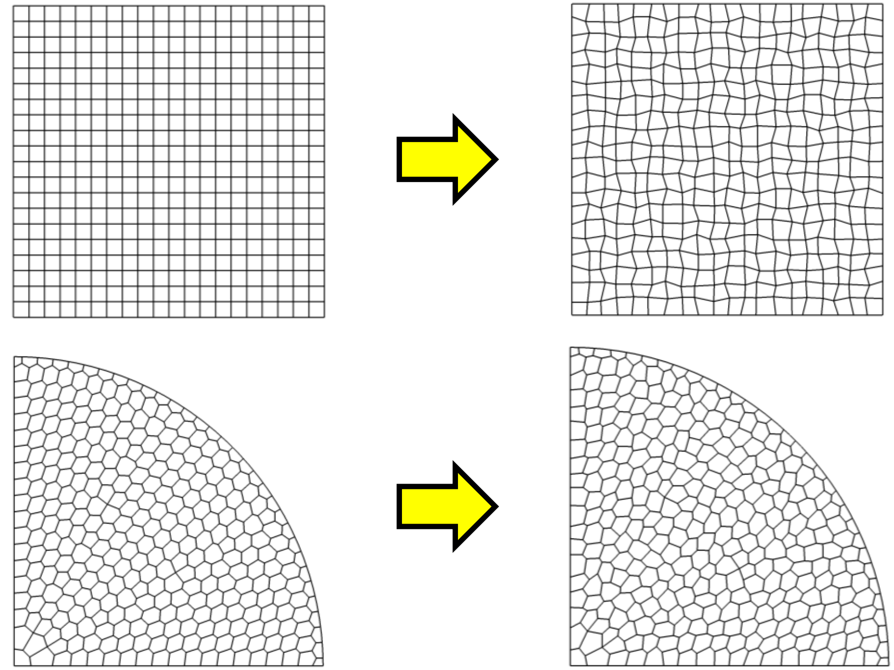
Mesh type	L1 Relative Error	L2 Relative Error
Cartesian	5.959E-15	1.198E-14
Polygonal	3.006E-14	5.602E-14

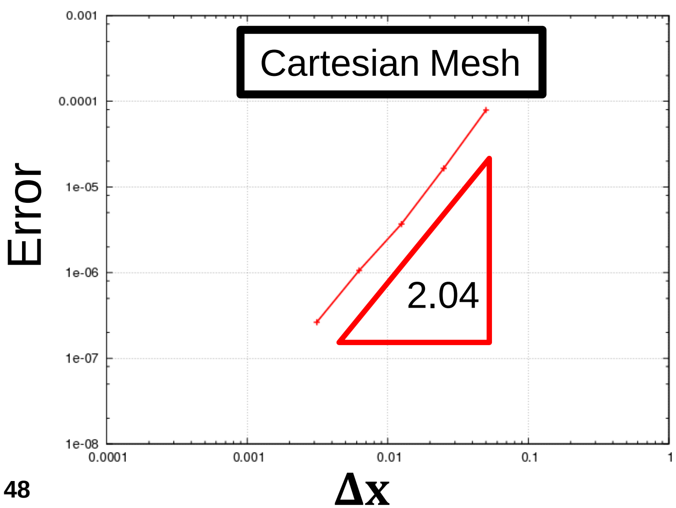


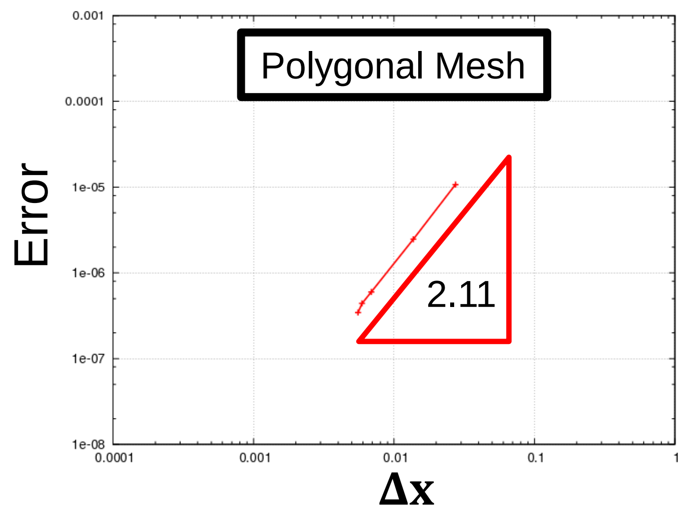
Los Alamos NATIONAL LABORATORY

Results: Accuracy

- Non-linear field
  - $f(x,y) = x^2 + y^2 + 1$
- Random perturbation relaxer
- Convergence rates:
  - Cartesian: 2.04
  - Polygonal: 2.11



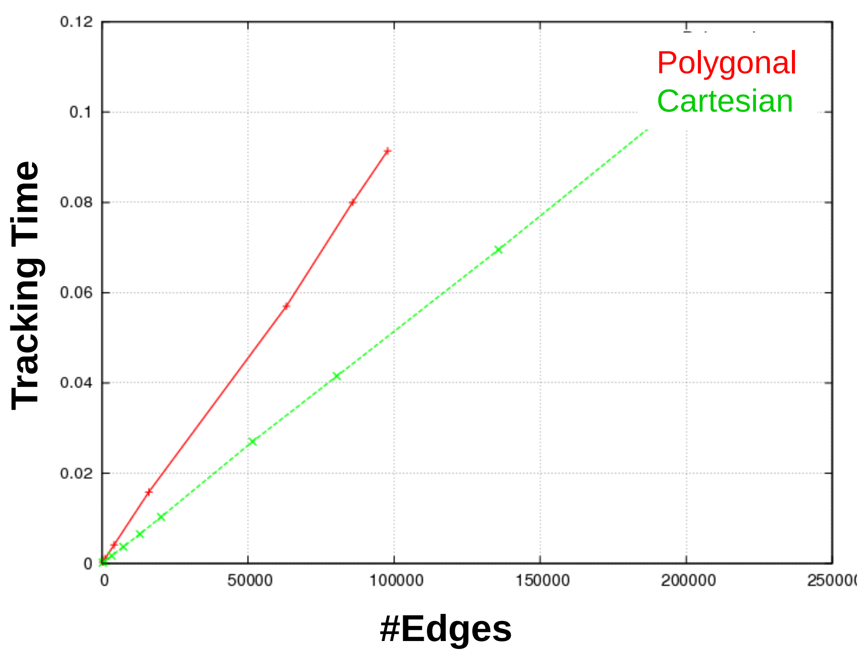


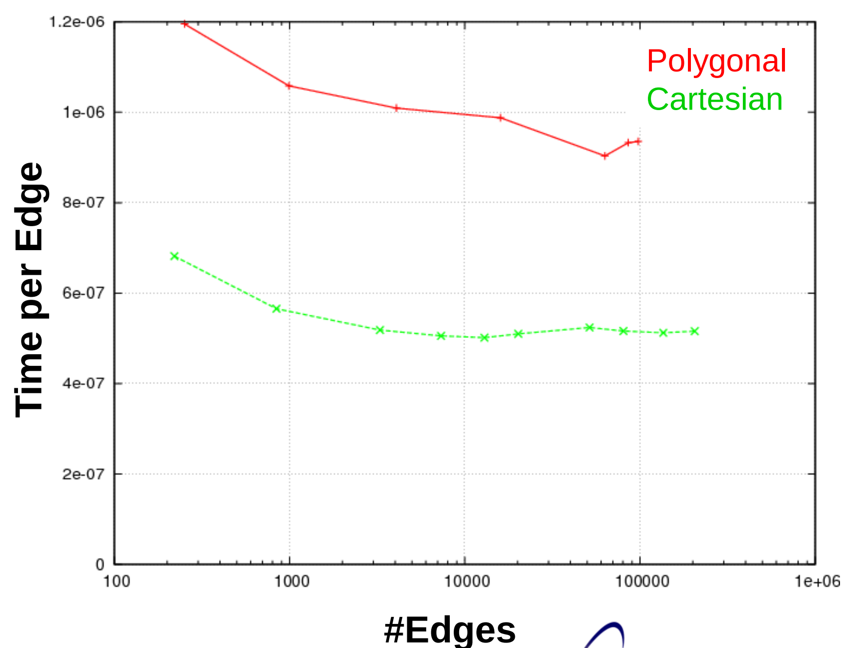


Los Alamos
NATIONAL LABORATORY

Results: Performance

- Should observe O(n) time complexity
- Increasing mesh resolution
  - Cartesian meshes
  - Polygonal meshes

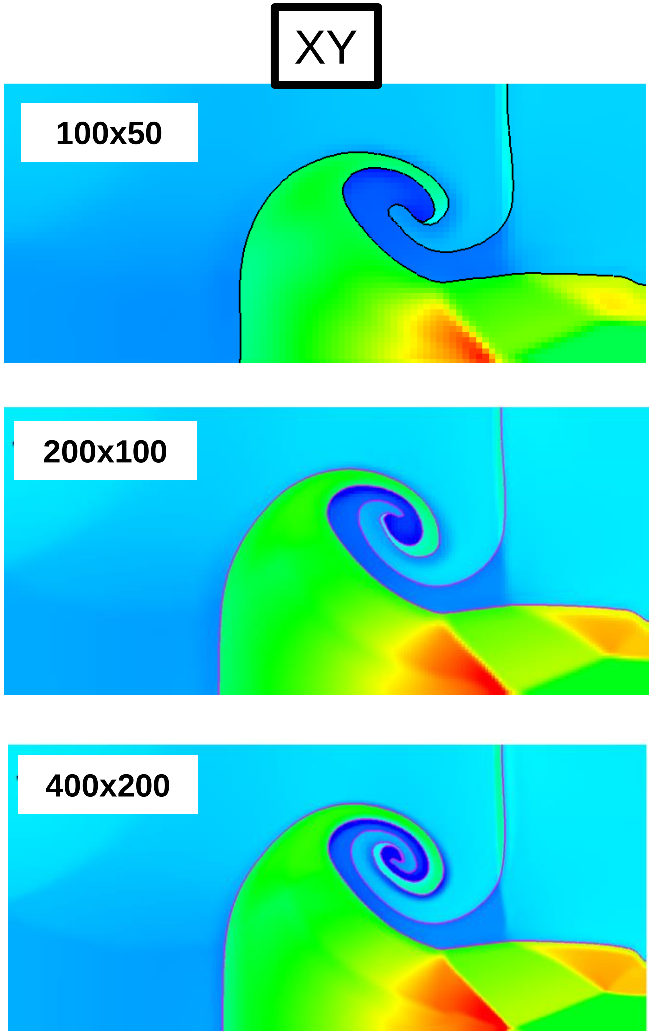


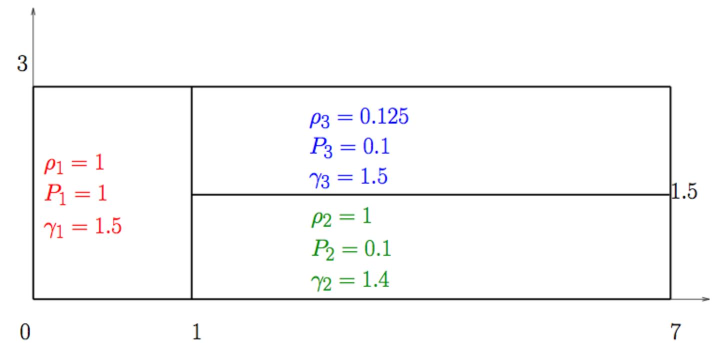


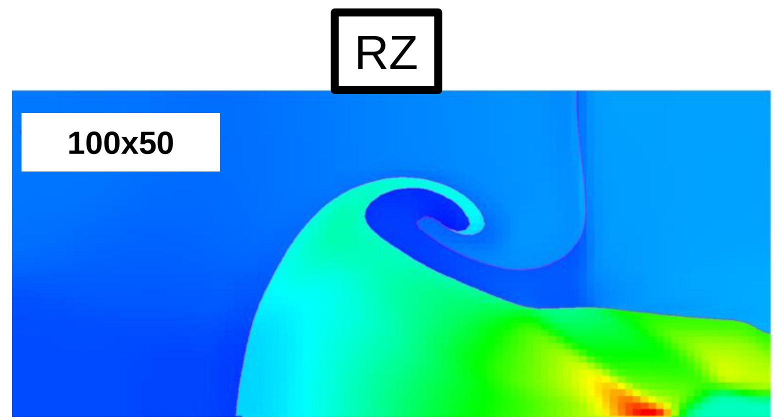
Los Alamos
NATIONAL LABORATORY

Results: Vortex problem

- Exact remapper integrated with CCH (xALE)
  - 2D Cartesian (XY) and Cylindrical (RZ)

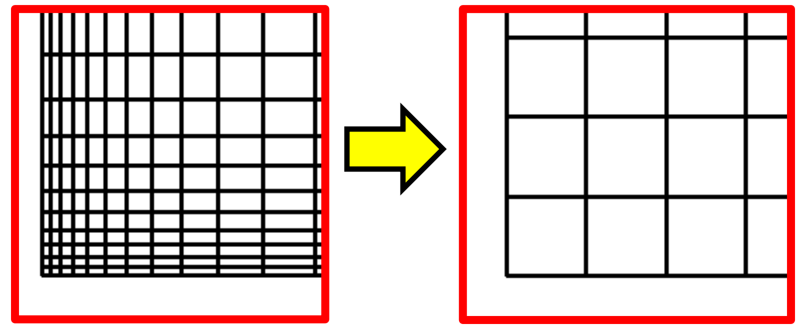




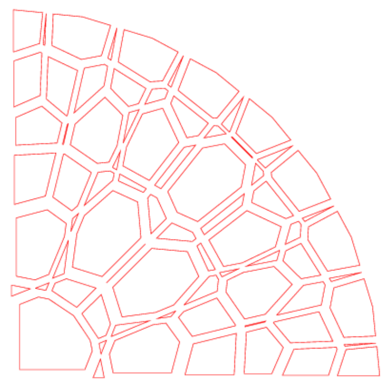


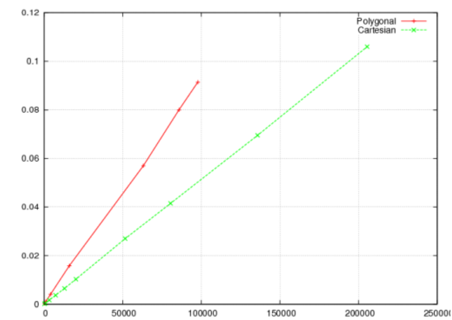
Los Alamos NATIONAL LABORATORY

Summary and Conclusions

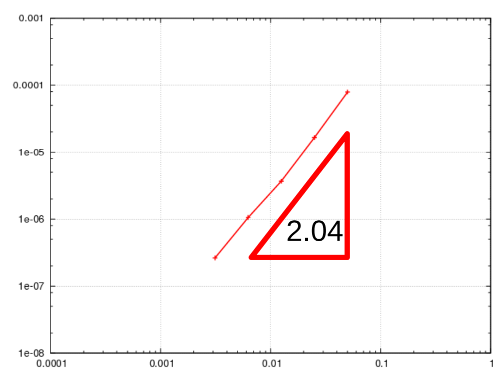


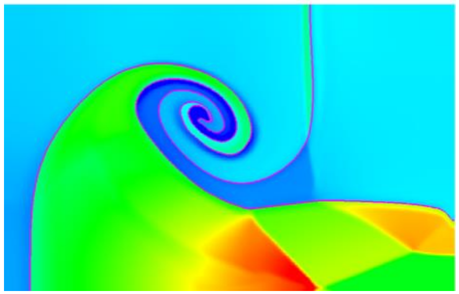
- 2D exact intersection remap
  - Alternative to swept face advection or directional splitting
  - No limit on relaxer displacements
  - Polygonal meshes





- No perturbations or tolerances required
  - Must handle special start, end, and intersection cases
    - On point, on edge, collinear edges
  - Robust
    - So far, so good
- O(n) time complexity
  - Advancing wavefront guarantees that edge start conditions are known
  - Only one log(n) search required
- 2nd-order spatial accuracy
- Multi-material remapping with VOF





Los Alamos
NATIONAL LABORATORY

Future Work

- Parallelism
- Remap point- and/or corner-centered fields (SGH)
- Interface reconstruction work
  - Moment of Fluids (MOF)
  - Automatic material ordering
- ReALE
- 3D
- Investigate performance improvements

Backup

This page intentionally left blank.

Backup

This page intentionally left blank.

Backup

This page intentionally left blank.