FINAL EXAMINATION

Department of Civil and Environmental Engineering
University of Illinois

Adaptive numerical simulation of fracture and failure

Sofie Leon

December 10th, 2014

Committee Members:
Glaucio H. Paulino, Chair
Waldemar Celes
Ahmed E. Elbanna
James W. Foulk, III
Iwona Jasiuk

Acknowledgments

Support from

Philanthropic Education Organization Scholars Award

National Science Foundation Graduate Research Fellowship

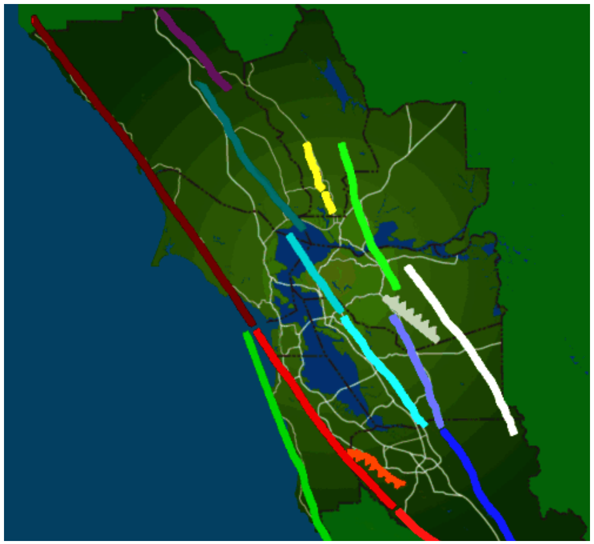
Society of Women Engineers Scholarship

Collaborations with

Computer Scientists at PUC-Rio

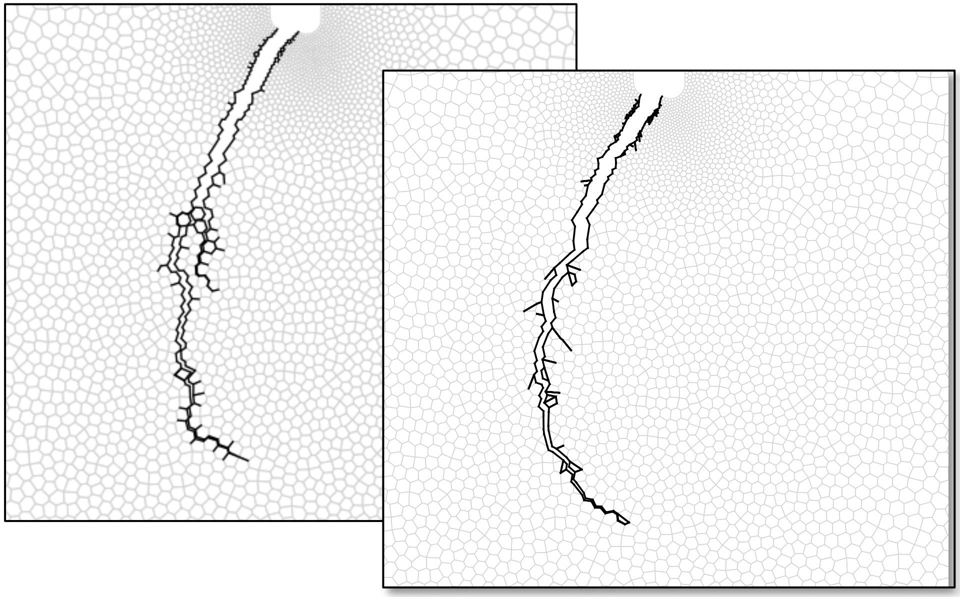
Researchers at Sandia National Laboratories

The motivation to study fracture and failure exists in many fields

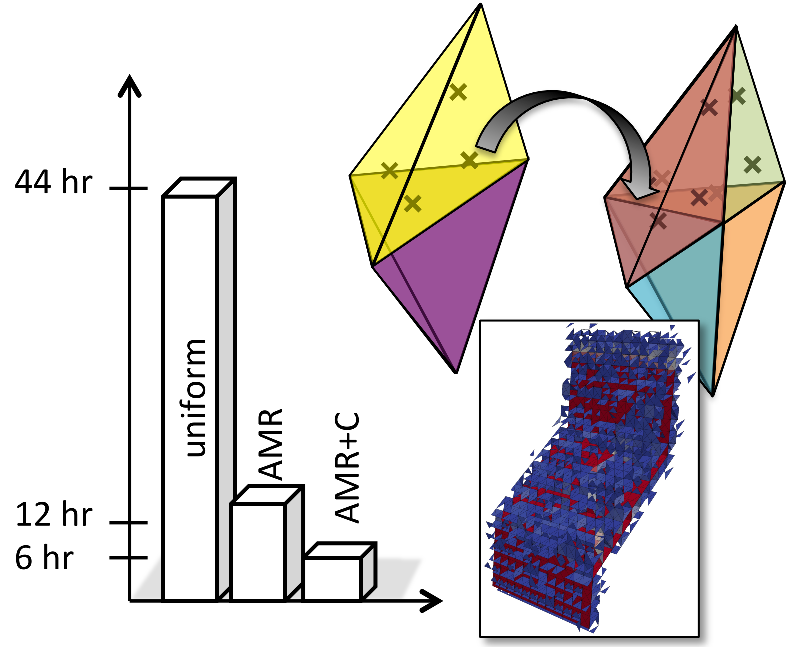


Hambli, R., and S. Allaoui. *Annals of Biomedical Engineering* 41, no. 12 (2013): 2515–2527.

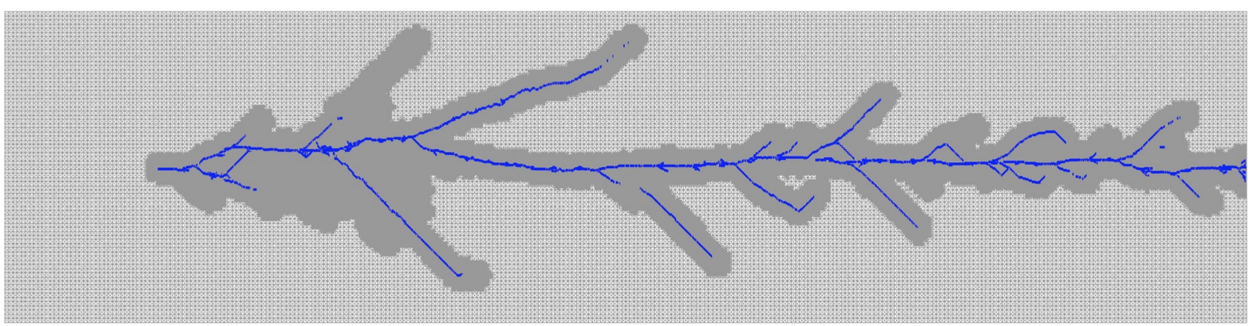
The adaptive schemes explored in this work result in:



Improved solutions over non adaptive schemes – **Adaptive polygonal splitting**

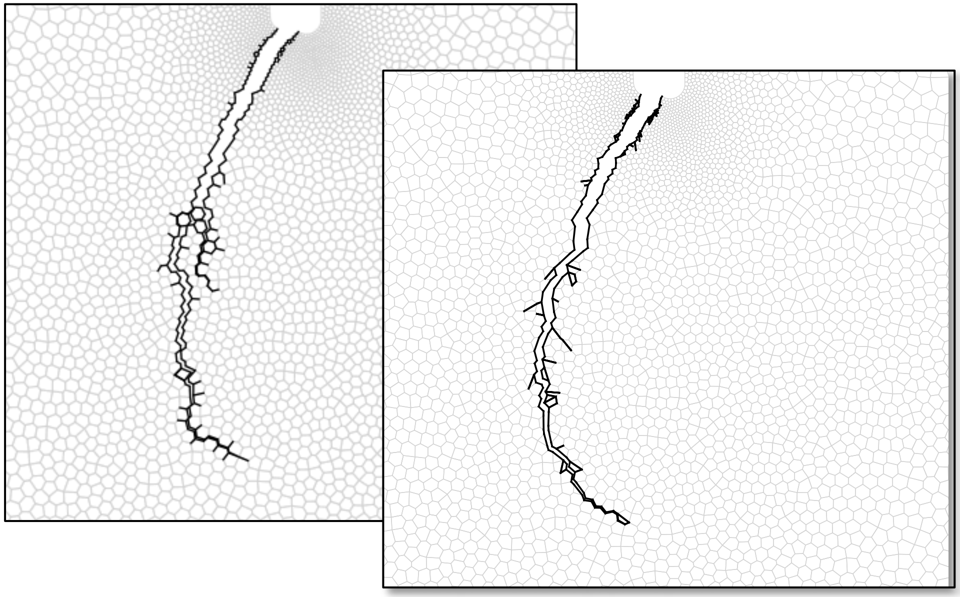


Increased computational efficiency – **3D refinement and coarsening**



Enables solutions to complicated problems – **GPU Adaptivity**

The adaptive schemes explored in this work result in:



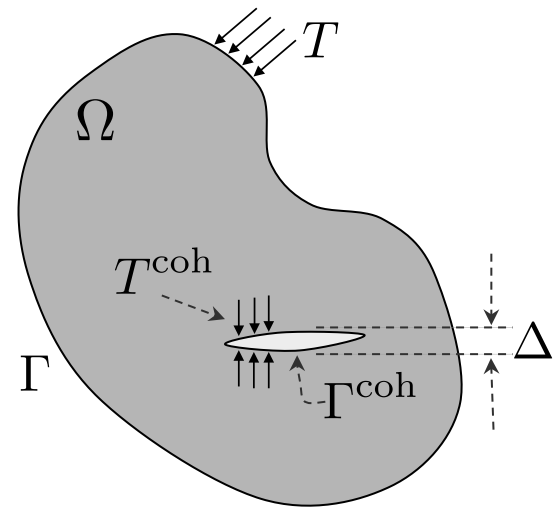
Improved solutions over non adaptive schemes – **Adaptive polygonal splitting**

Increased computational efficiency – **3D refinement and coarsening**

Enables solutions to complicated problems – **GPU Adaptivity**

Mathematical formulation for dynamic fracture

Consider the case of an arbitrary domain that is subjected to surface tractions, along the boundary and cohesive tractions along the fractured surfaces



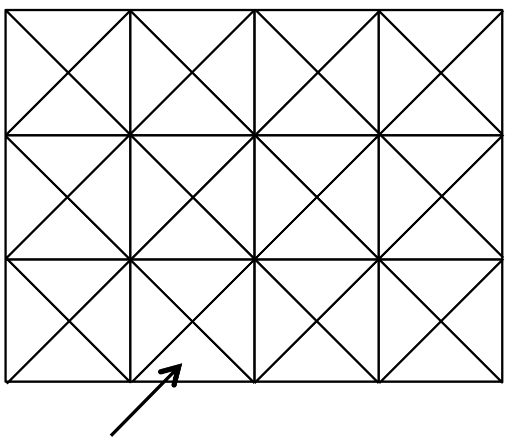
Neglecting body forces and damping, the principle of virtual work of the dynamic fracture problem states:

$$\underbrace{\int_{\Omega} \delta \mathbf{u}^T \rho \ddot{\mathbf{u}} d\Omega}_{\mathbf{M}\ddot{\mathbf{u}}} + \underbrace{\int_{\Omega} \delta \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} d\Omega}_{\mathbf{K}\mathbf{u}} = \underbrace{\int_{\Gamma} \delta \mathbf{u}^T \mathbf{T} d\Gamma}_{\mathbf{F}_{\mathrm{ext}}} + \underbrace{\int_{\Gamma^{\mathrm{coh}}} \delta \boldsymbol{\Delta} \boldsymbol{T}^{\mathrm{coh}} d\Gamma}_{\mathbf{F}_{\mathrm{coh}}}$$

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}_{\mathrm{ext}} + \mathbf{F}_{\mathrm{coh}}$$

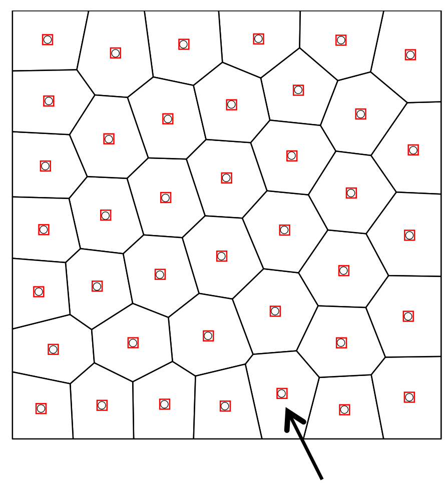
Compare three 2D mesh discretizations for dynamic fracture simulation

Structured 4K



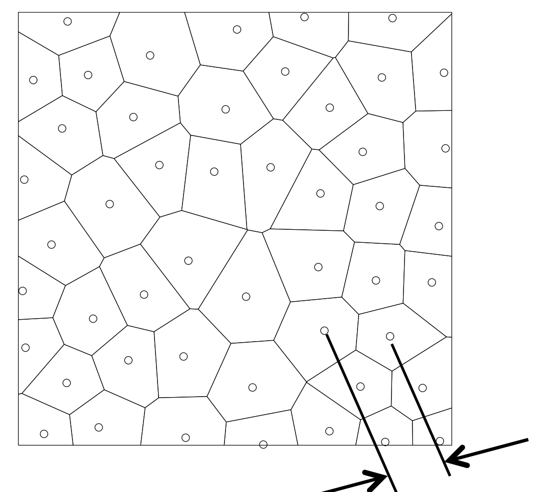
Repeated 4k patched

Unstructured CVT Polygonal



Voronoi seed = cell centroid

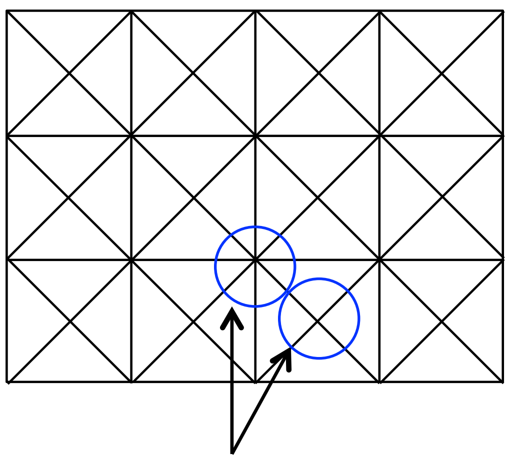
Unstructured Random Polygonal

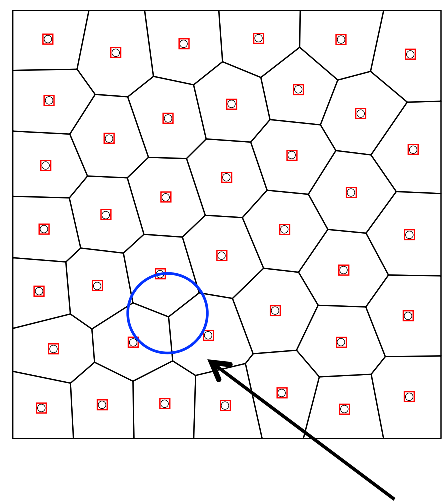


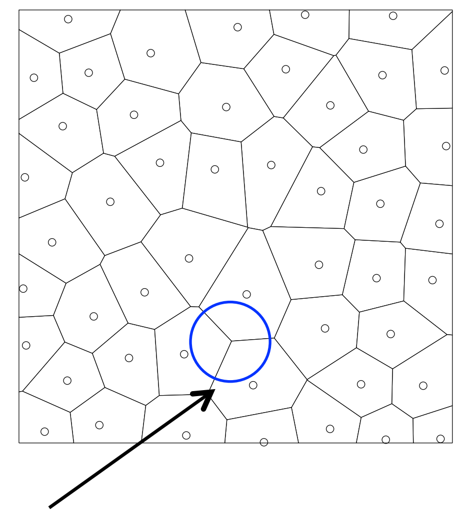
$d > r_{\min}$

Each mesh type is powerful for certain applications

	Structured 4K	Unstructured CVT Polygonal	Unstructured Random Polygonal
Pros:	Easily generated, refinement schemes are readily available	Efficiently mesh complex domains, regular elements	No mesh bias, simple mesh construction







Cons:

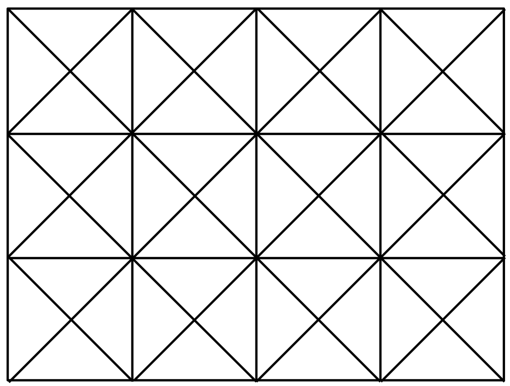
Structured 4K: Potential directions not equal at each junction

0°, 45°, 90°, etc. only

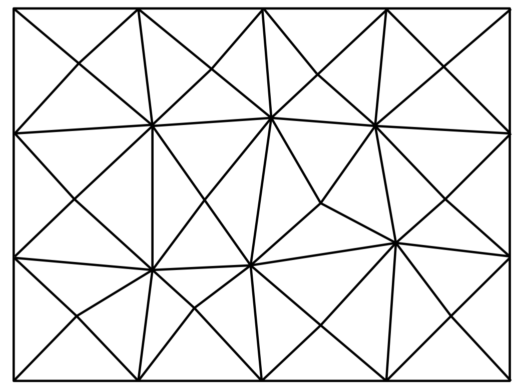
Unstructured CVT Polygonal and Unstructured Random Polygonal: Typically only 3 possible directions per junction

Adaptive mesh operators are introduced improve fracture patterns on the 4k mesh

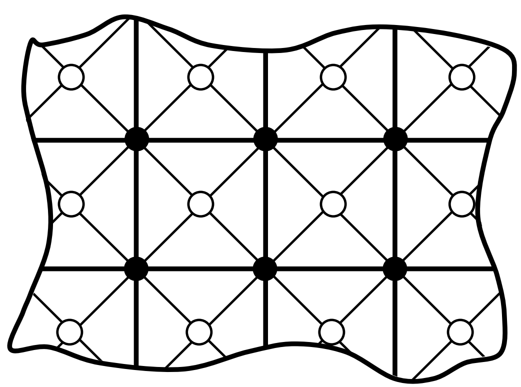
Original 4k mesh



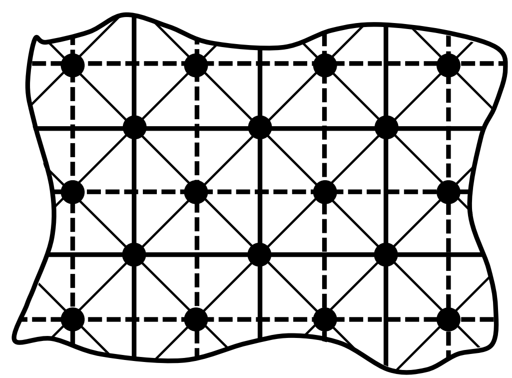
Nodal perturbation



Original 4k mesh



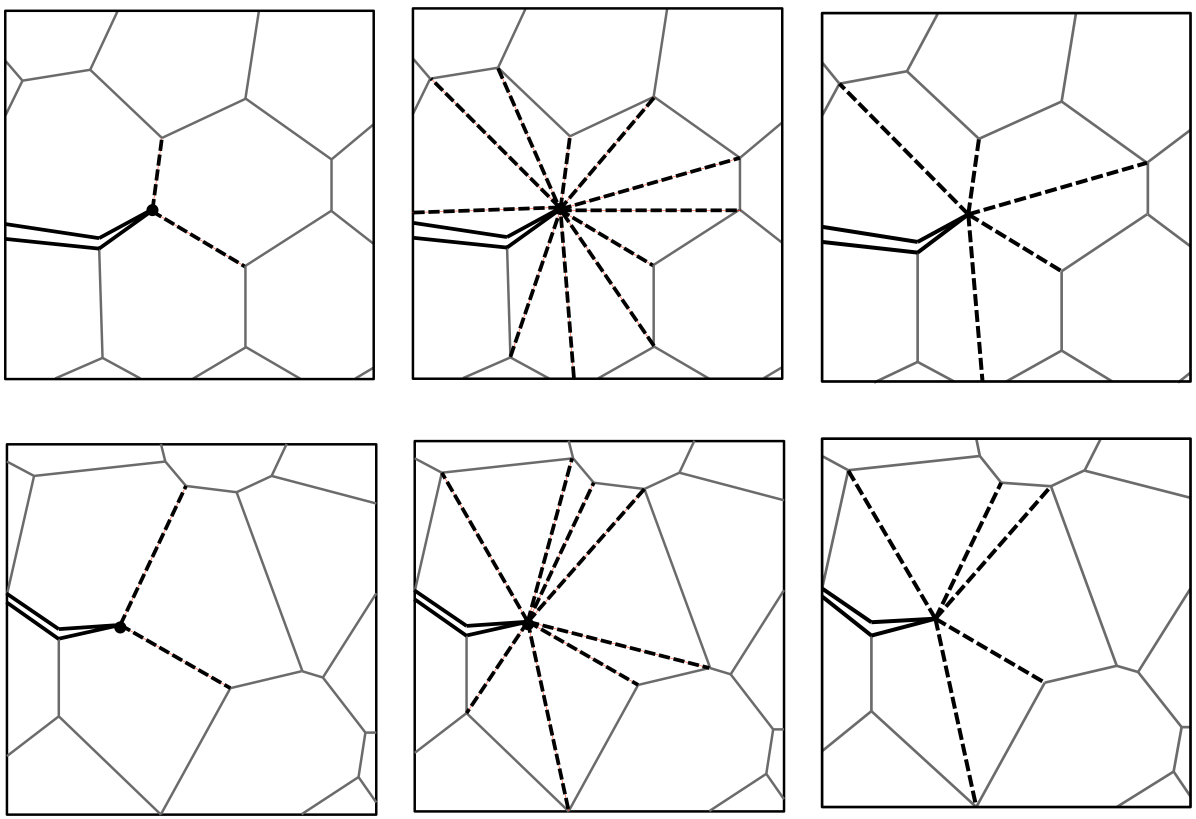
Edge swap

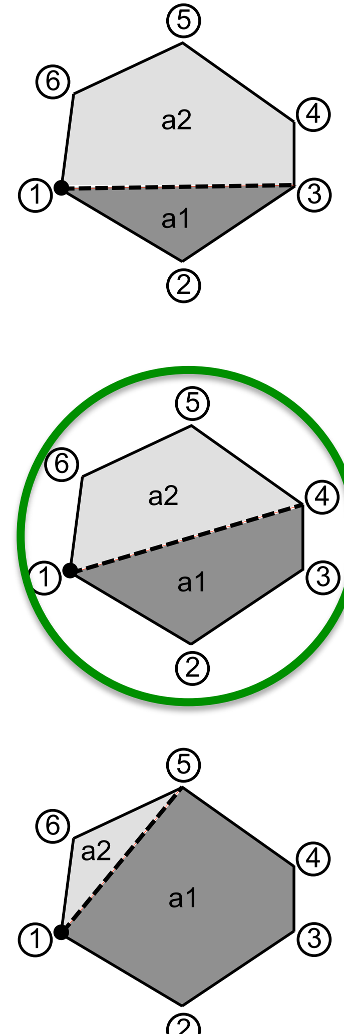


Paulino, G. H., Park, K., Celes, W., & Espinha, R. *IJNME*, 84(11), 1303–1343. 2010.

Element splitting provides more directions for the crack to propagate on polygonal meshes

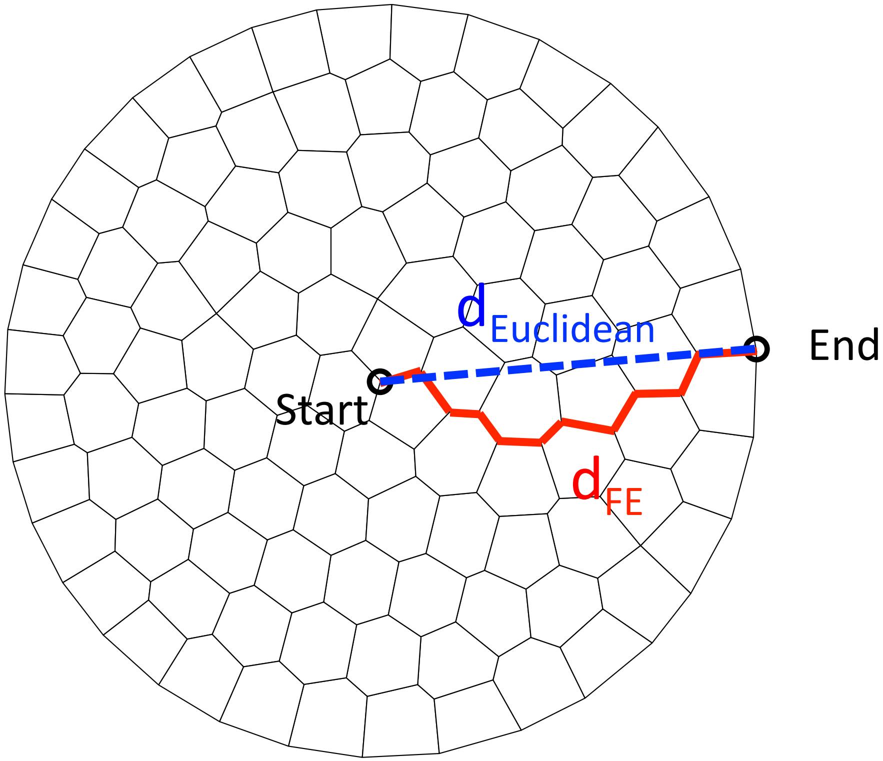
	No splitting	Unrestricted splitting	Restricted splitting
CVT polygonal mesh			
Random polygonal mesh			





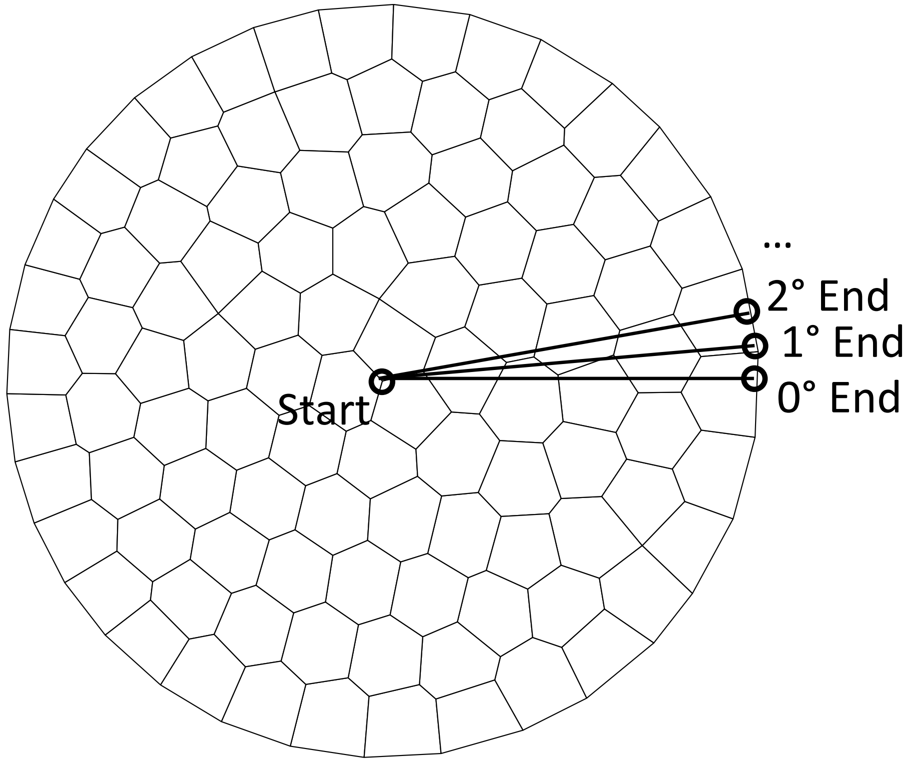
S. E. Leon*, D. W. Spring*, and G. H. Paulino. *IJNME*, 100(8): 555–76, 2014.

Geometric study to evaluate ability of mesh to represent a straight line using Dijkstra's algorithm



$$\text{Error} = \frac{d_{FE} - d_{Euclidean}}{d_{Euclidean}}$$

Geometric study to evaluate ability of mesh to represent a straight line using Dijkstra's algorithm

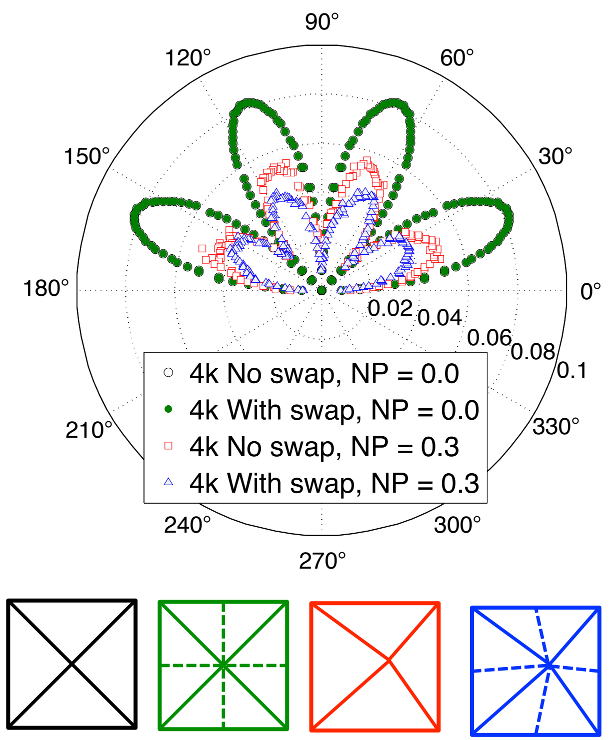


$$\text{Error} = \frac{d_{FE} - d_{Euclidean}}{d_{Euclidean}}$$

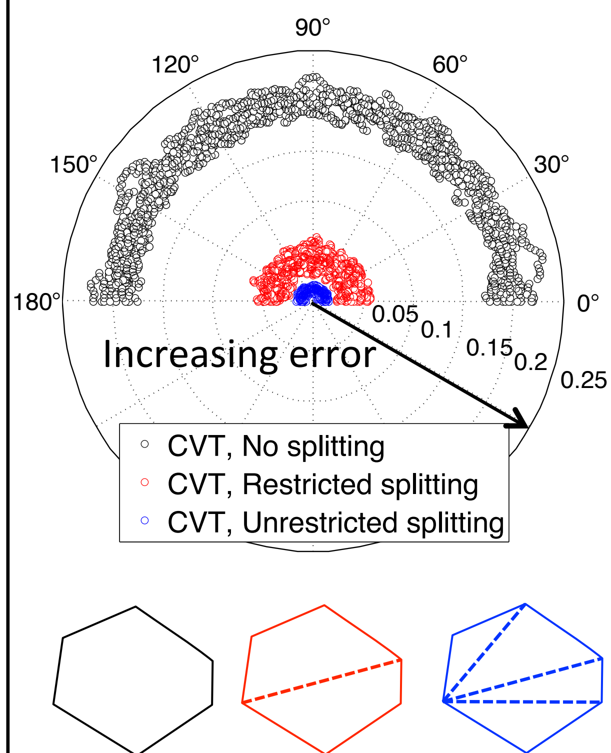
Evaluate error from 0° to 180° in 1° increments

Polygonal meshes provide an alternative to structured meshes that reduces mesh bias

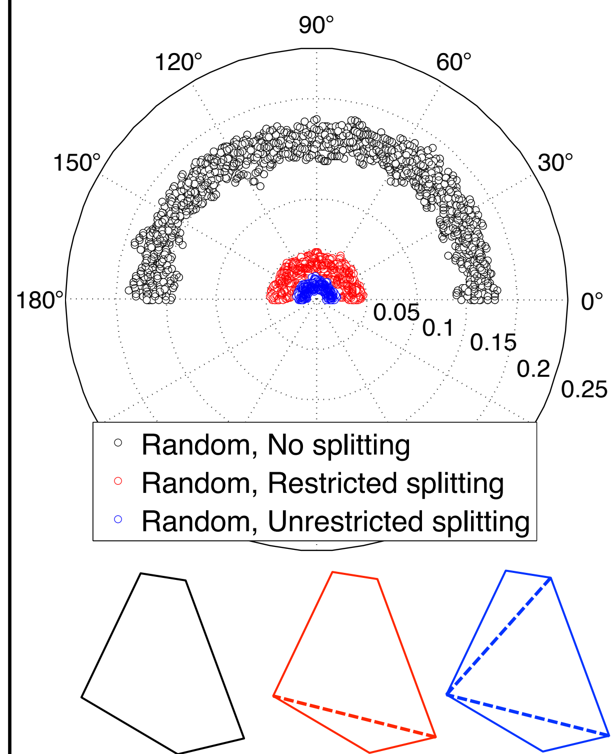
4k mesh



CVT polygonal



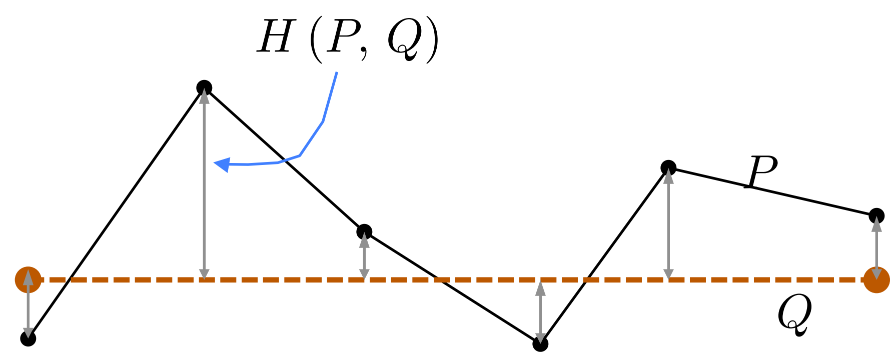
Random polygonal

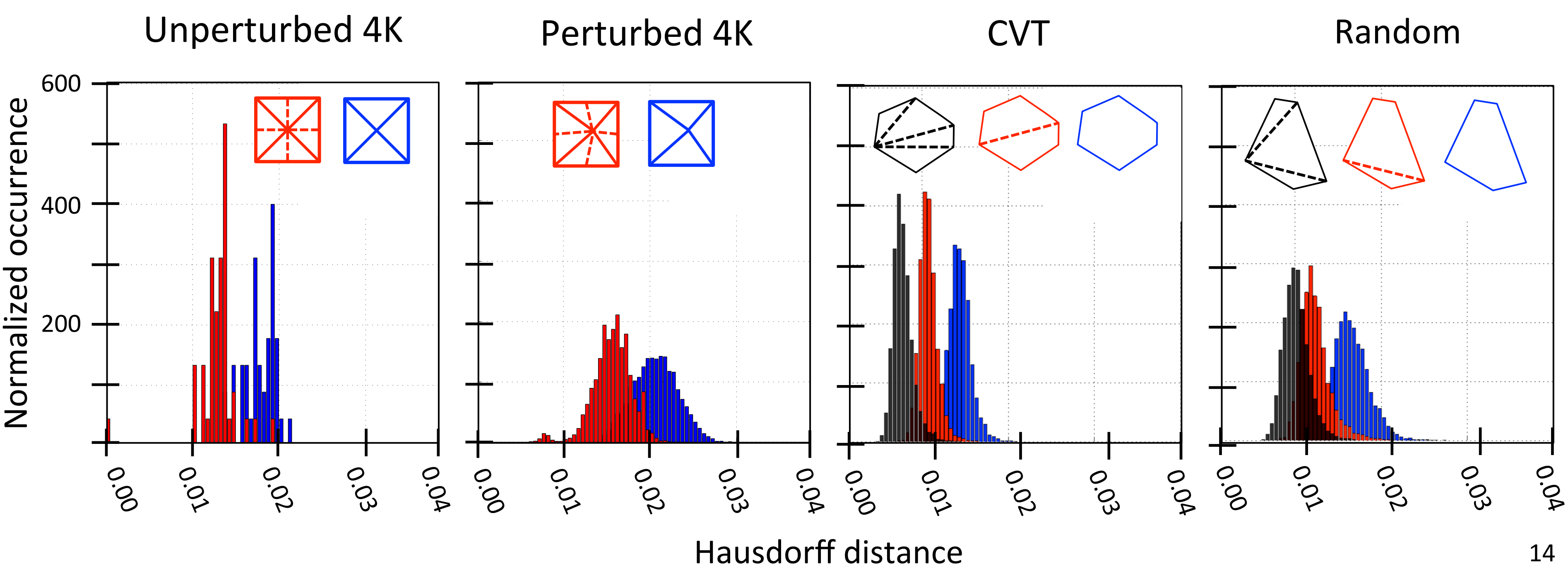


Hausdorff distances are also lower for polygonal meshes compared to 4k

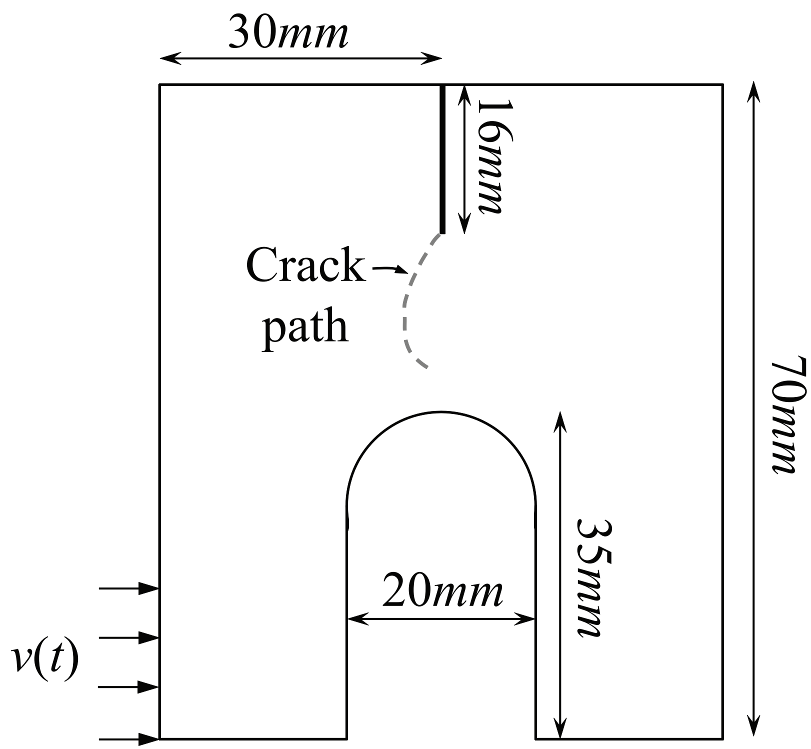
Given a discretized path, P, whose vertices are p, and a mathematical path Q, the Hausdorff distance is

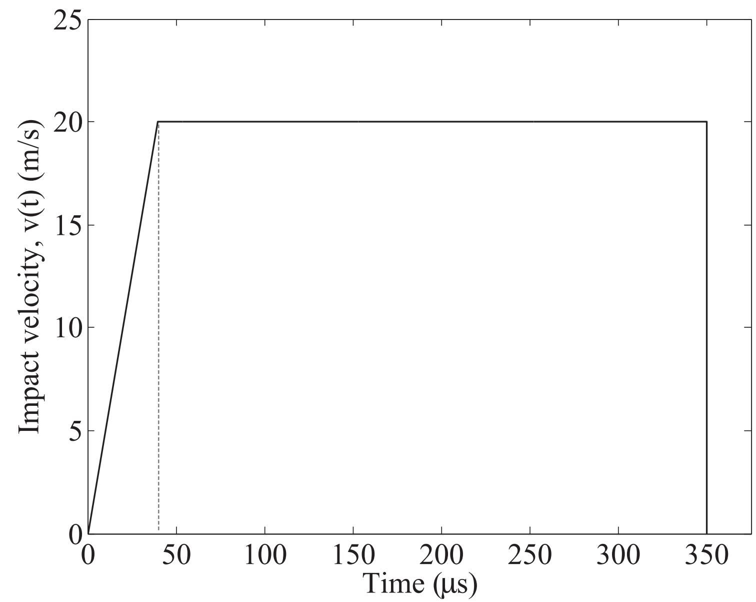
$$H\,(P,\,Q) = \max_{p\in P}\left[\min_{q\in Q}\left[\mathrm{dist}\,(p,\,q)\right]\right]$$



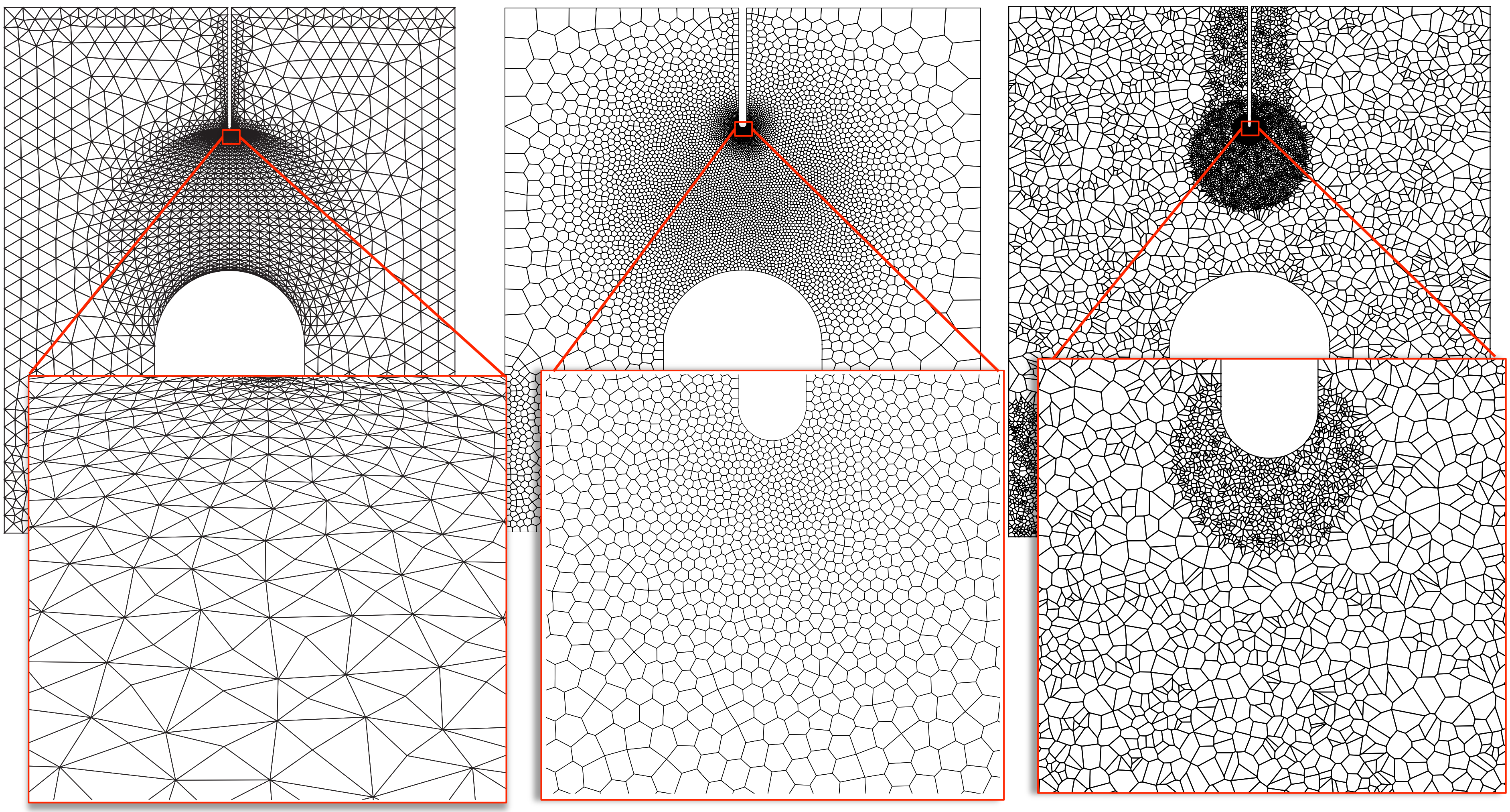


Numerical investigation of different meshes with Compact Compression Specimen

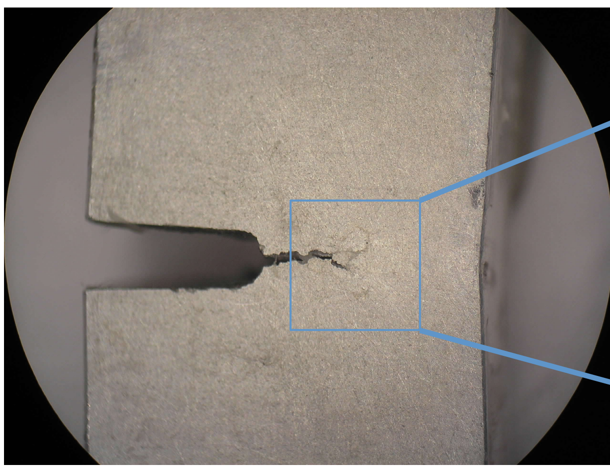


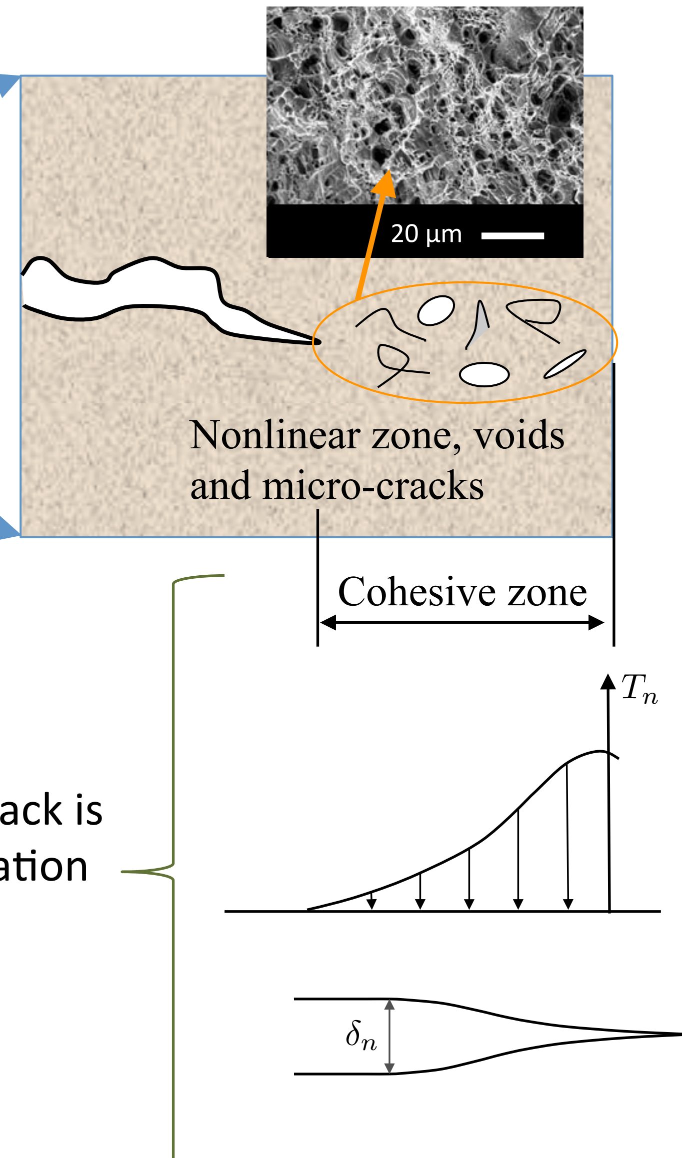


Numerical investigation of polygonal meshed with Compact Compression Specimen



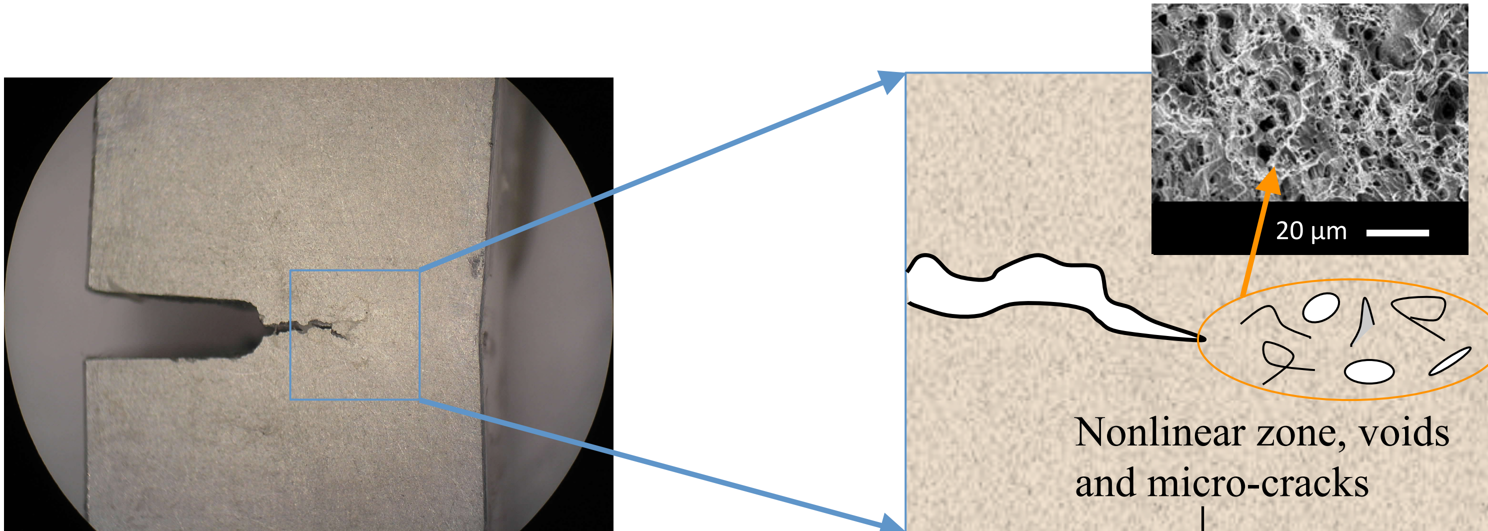
Cohesive elements aim to capture the nonlinear behavior in the zone ahead of a crack tip





Region ahead of the macro crack is idealized by a traction-separation relationship

Cohesive elements aim to capture the nonlinear behavior in the zone ahead of a crack tip

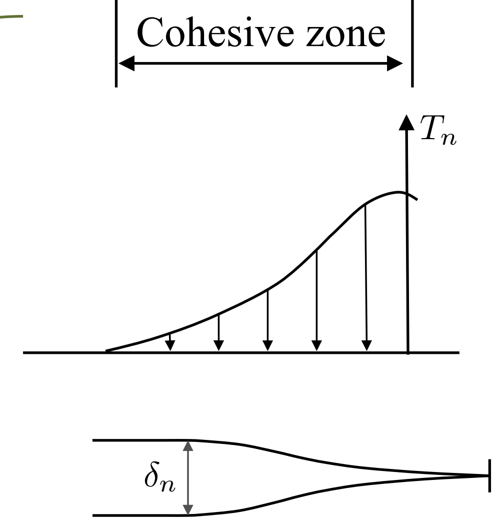


PPR Cohesive Model:

$$\Psi = \min(\phi_n, \phi_t) + \left[\Gamma_n \left(1 - \frac{\Delta_n}{\delta_n}\right)^{\alpha} + \langle \phi_n - \phi_t \rangle\right] \left[\Gamma_t \left(1 - \frac{|\Delta_t|}{\delta_t}\right)^{\beta} + \langle \phi_t - \phi_n \rangle\right]$$

$$T_n = -\alpha \frac{\Gamma_n}{\delta_n}\left(1 - \frac{\Delta_n}{\delta_n}\right)^{\alpha-1} \left[\Gamma_t \left(1 - \frac{|\Delta_t|}{\delta_t}\right)^{\beta} + \langle \phi_t - \phi_n \rangle\right]$$

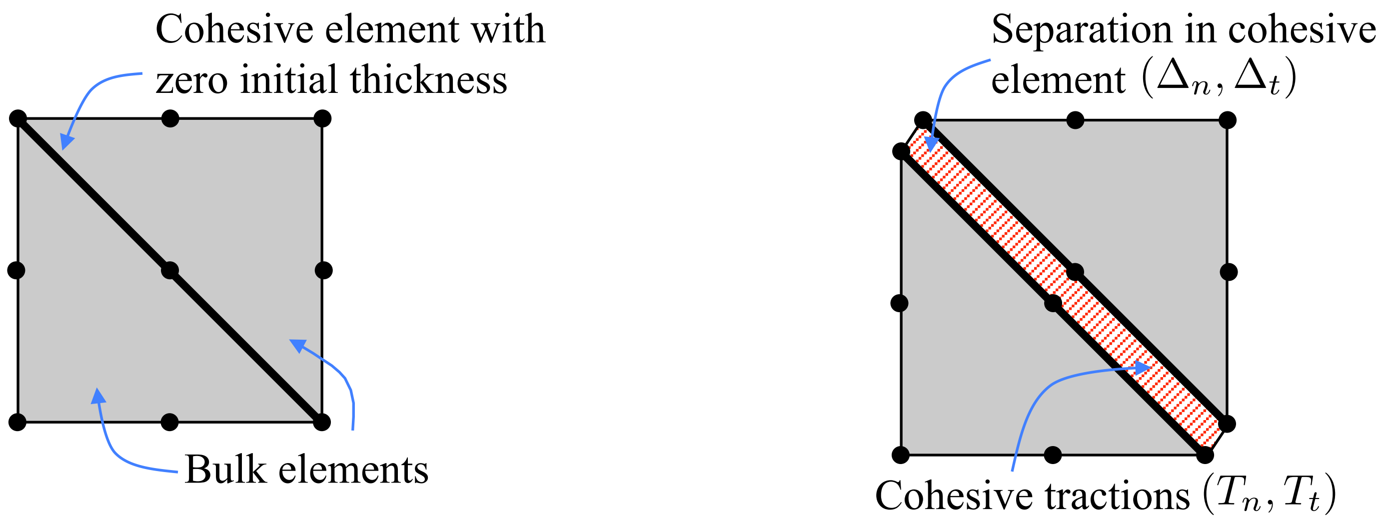
$$T_t = -\beta \frac{\Gamma_t}{\delta_t}\left(1 - \frac{|\Delta_t|}{\delta_t}\right)^{\beta-1} \left[\Gamma_n \left(1 - \frac{\Delta_n}{\delta_n}\right)^{\alpha} + \langle \phi_n - \phi_t \rangle\right] \frac{|\Delta_t|}{\delta_t}$$



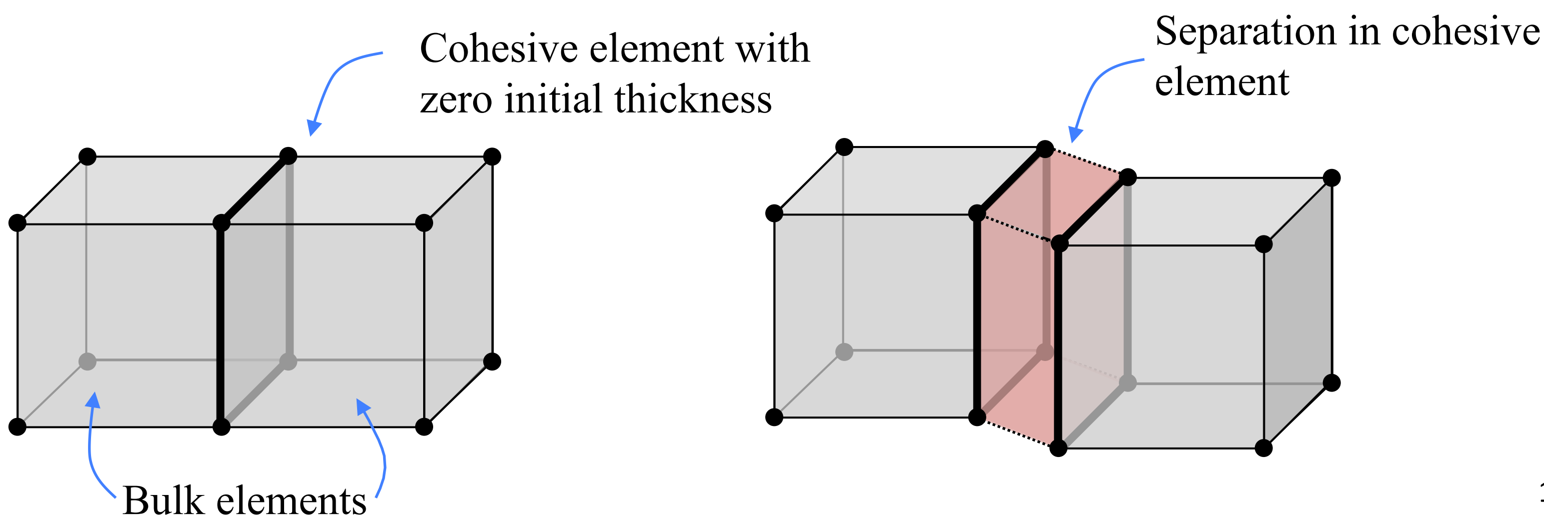
K. Park, G. H. Paulino, and J. R. Roesler. JMPS57, no. 6 (2009): 891–908. 2008.10.003.

Nodes of bulk elements are duplicated when cohesive elements are inserted

2D

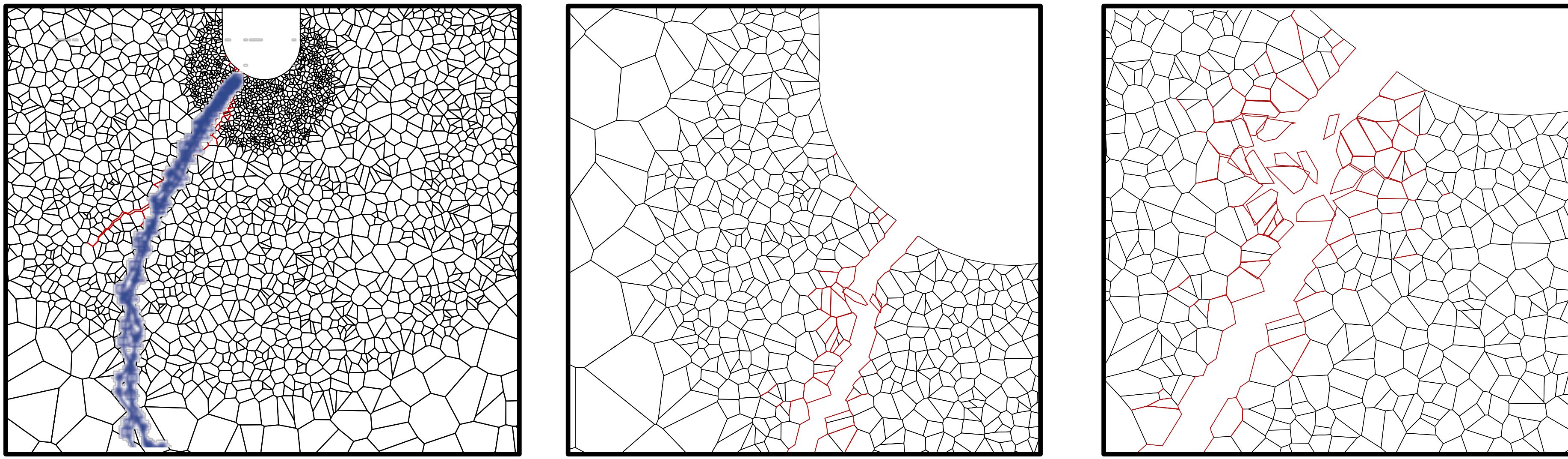


3D



Random polygonal elements are not reliable for dynamic fracture simulation

- CVT time step = 2.5e-9 to 1e-9
- Random time step without splitting = 1e-10 → requires over 1,000,000 steps!



Explicit time integration scheme update

$$\mathbf{u}_{\mathrm{n+1}} = \mathbf{u}_{\mathrm{n}} + \Delta t \dot{\mathbf{u}}_{\mathrm{n}} + \frac{\Delta t^2}{2} \ddot{\mathbf{u}}_{\mathrm{n}}$$

$$\ddot{\mathbf{u}}_{\mathrm{n+1}} = \mathbf{M}^{-1} \left[\mathbf{F}_{\mathrm{n+1}} - \mathbf{K} \mathbf{u}_{\mathrm{n+1}} \right]$$

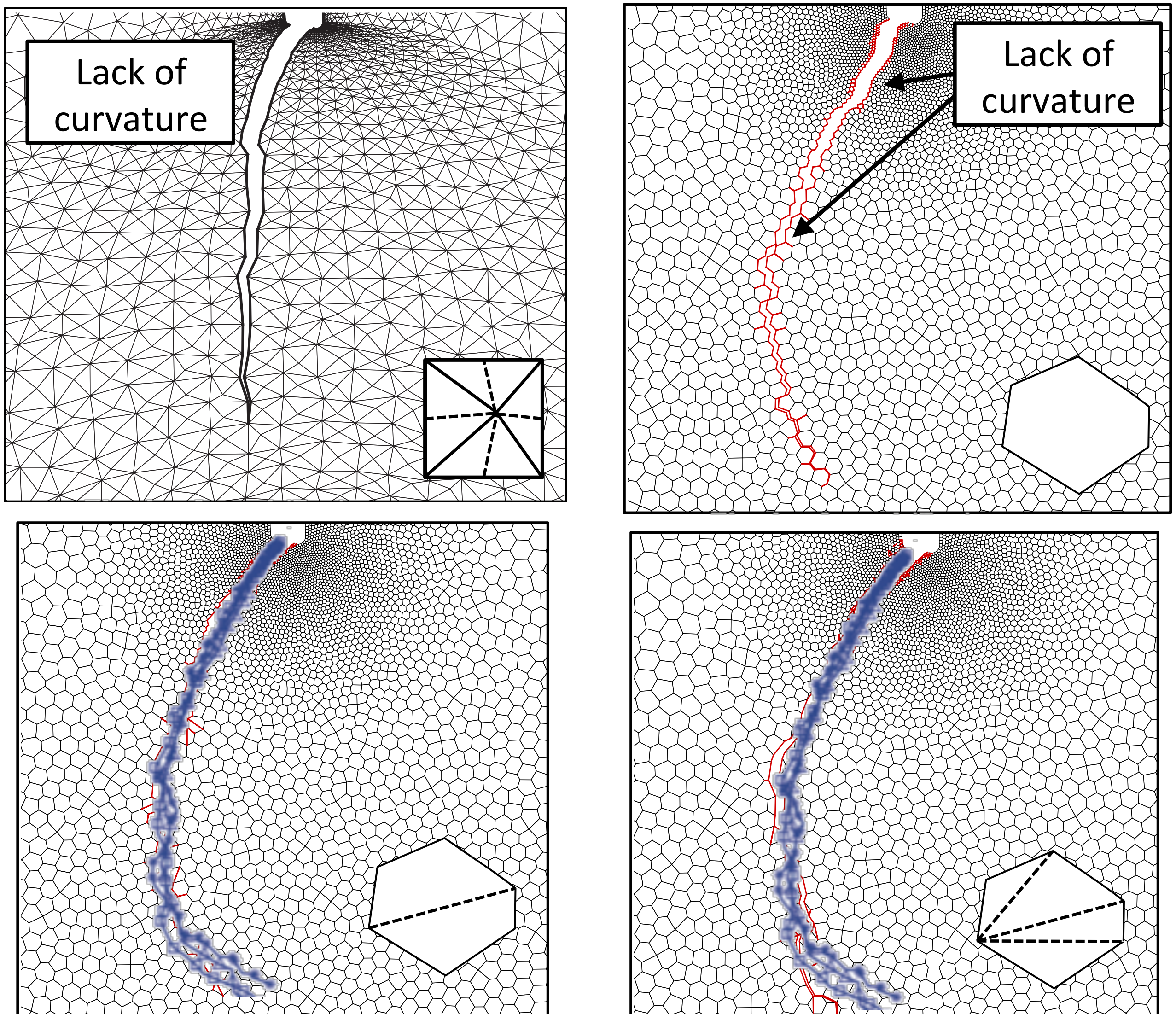
$$\dot{\mathbf{u}}_{\mathrm{n+1}} = \dot{\mathbf{u}}_{\mathrm{n}} + \frac{\Delta t}{2} \left[\ddot{\mathbf{u}}_{\mathrm{n}} + \ddot{\mathbf{u}}_{\mathrm{n+1}} \right]$$

Time step restriction

$$\Delta t \leq \frac{2}{\omega}$$

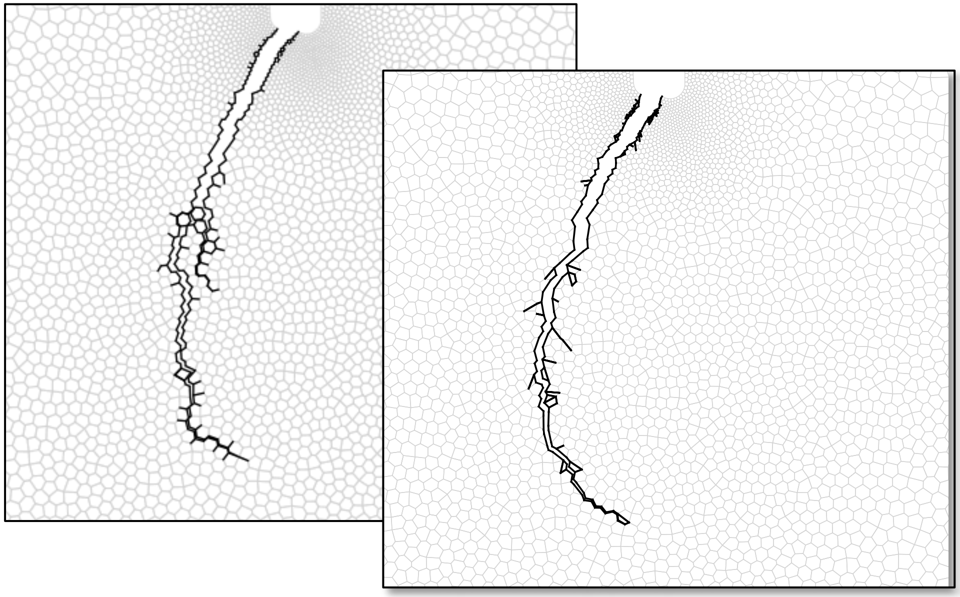
$$\Delta t \leq \frac{l_e}{C}$$

CVT polygonal elements with splitting provide excellent results for CCS test



Excellent agreement

The adaptive schemes explored in this work result in:



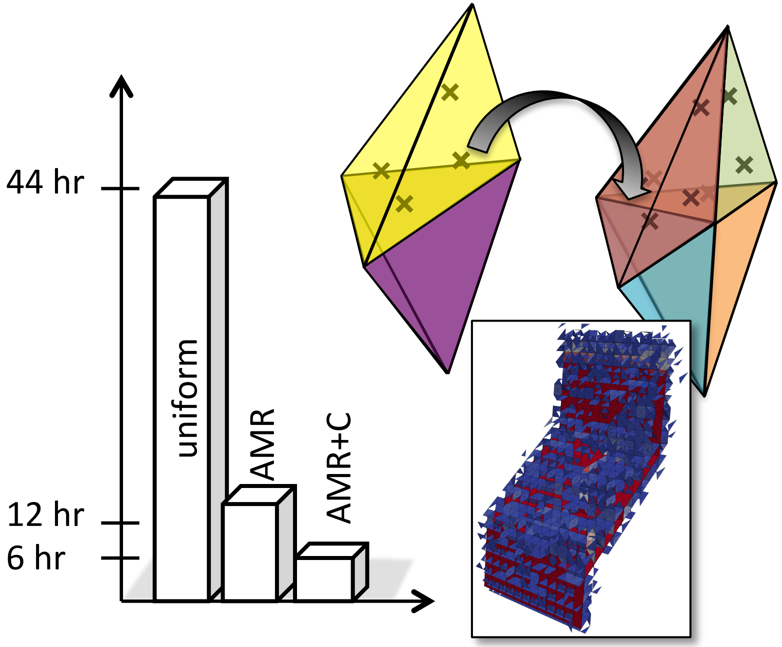
Improved solutions over non adaptive schemes – **Adaptive polygonal splitting**

Increased computational efficiency – **3D refinement and coarsening**

Enables solutions to complicated problems – **GPU Adaptivity**

The adaptive schemes explored in this work result in:

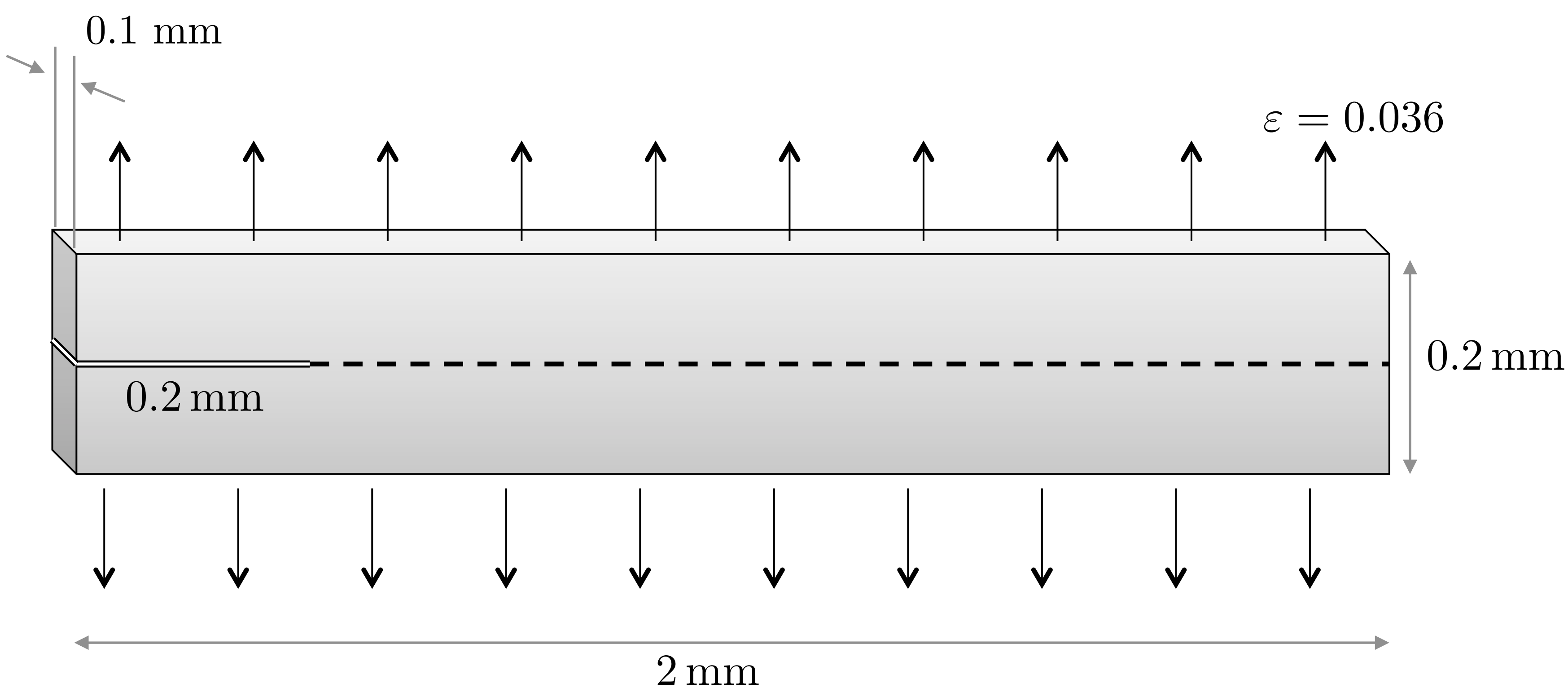
Improved solutions over non adaptive schemes – **Adaptive polygonal splitting**



Increased computational efficiency – **3D refinement and coarsening**

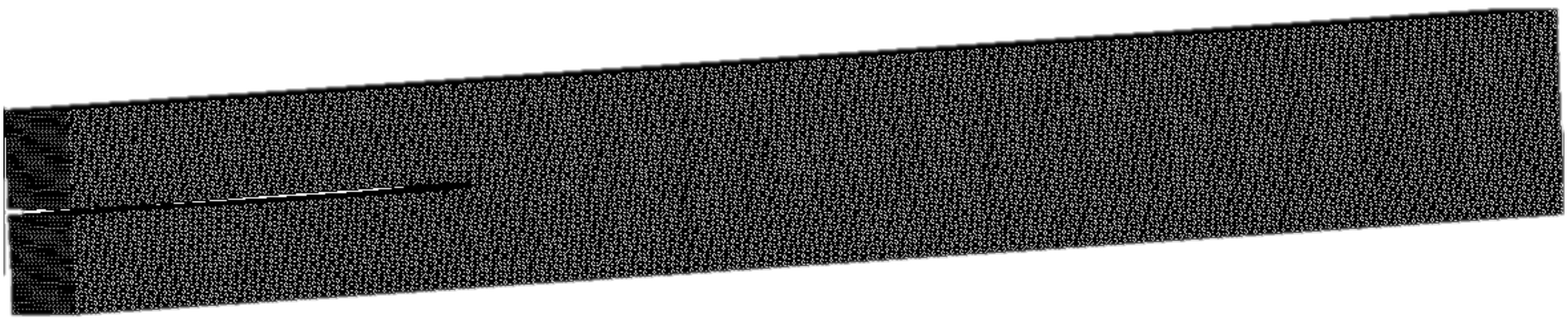
Enables solutions to complicated problems – **GPU Adaptivity**

3D simulation



- Linear tetrahedral elements with edge length of 6.25μm results in 3,932,160 elements & 852,359 nodes
- Simulate for 2 μsec with dt = 4e-10 seconds = 5000 steps

3D simulation



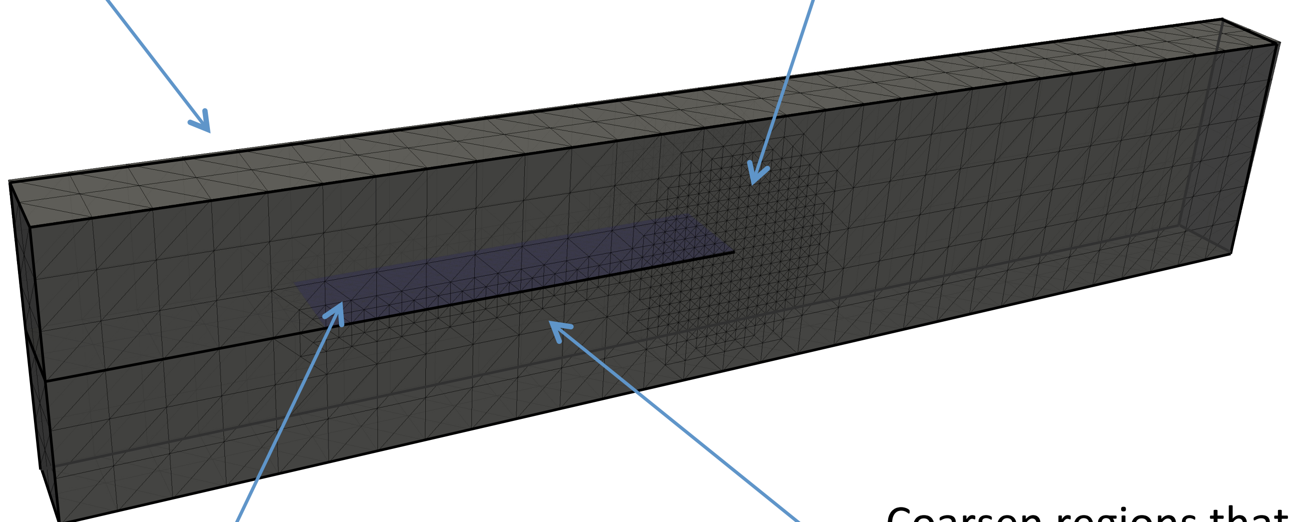
Clock time = 15.5 hours

- Linear tetrahedral elements with edge length of 6.25μm results in 3,932,160 elements & 852,359 nodes
- Simulate for 2 μsec with dt = 4e-10 seconds = 5000 steps

Adaptive mesh refinement and coarsening algorithms improve computational efficiency

Coarse mesh throughout the domain

Use fine elements only where needed – around crack tips

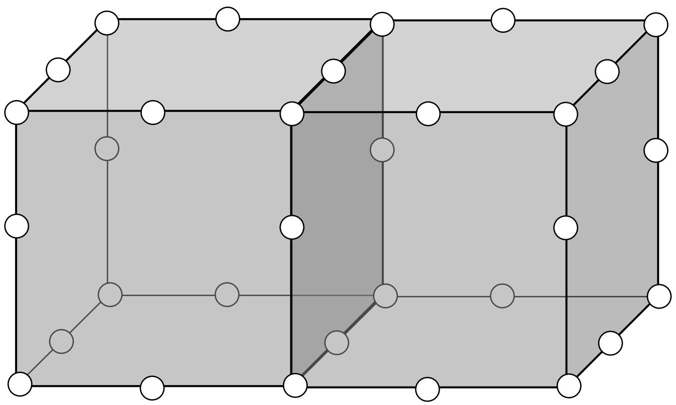


Keep neighborhood around cohesive elements refined to avoid losing detail of the crack pattern

Coarsen regions that were previously refined but now behind the crack front

The mesh is represented by a complete and compact topological data structure, TopS

- The application has access to all entities and all adjacencies
- Node and elements are stored explicitly, the rest are stored implicitly and can be retrieved on the fly
- Oriented entities used to perform local searched when obtaining adjacency information

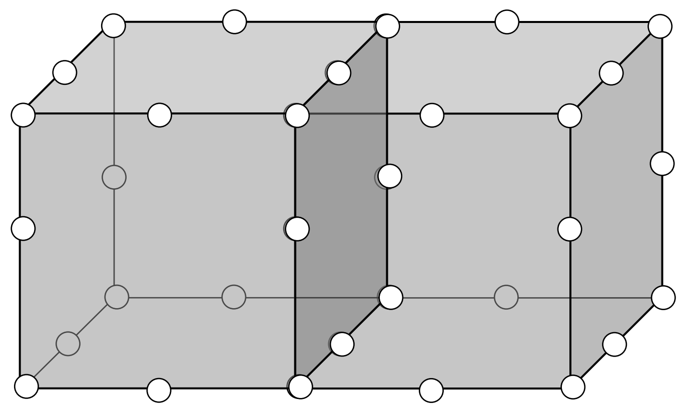


Nodes and elements

Celes, W., G. H. Paulino, and R. Espinha. *IJNME* 64, no. 11 (2005): 1529–1556.

The mesh is represented by a complete and compact topological data structure, TopS

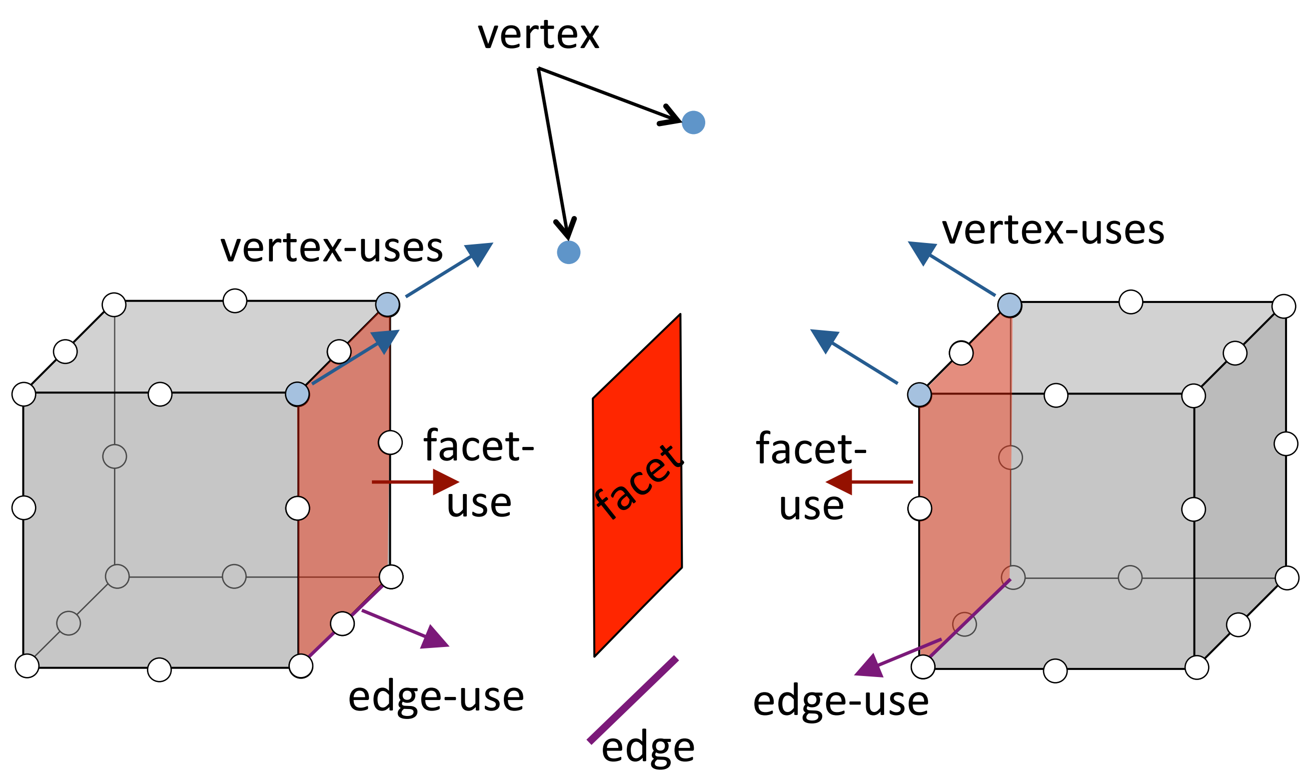
- The application has access to all entities and all adjacencies
- Node and elements are stored explicitly, the rest are stored implicitly and can be retrieved on the fly
- Oriented entities used to perform local searched when obtaining adjacency information



Celes, W., G. H. Paulino, and R. Espinha. *IJNME* 64, no. 11 (2005): 1529–1556.

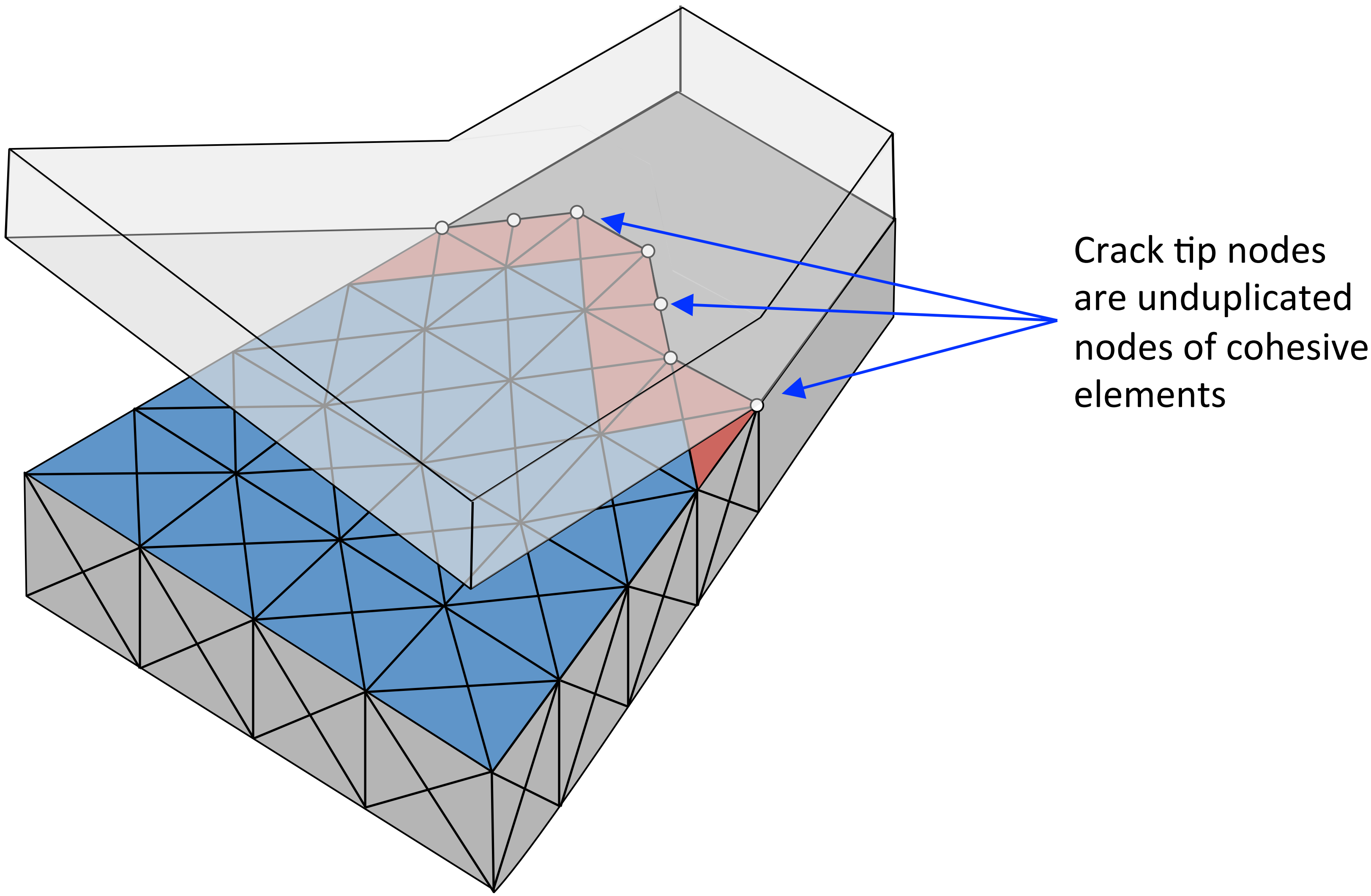
The mesh is represented by a complete and compact topological data structure, TopS

- The application has access to all entities and all adjacencies
- Node and elements are store explicitly, the rest are stored implicitly and can be retrieved on the fly
- Oriented entities used to perform local searched when obtaining adjacency information

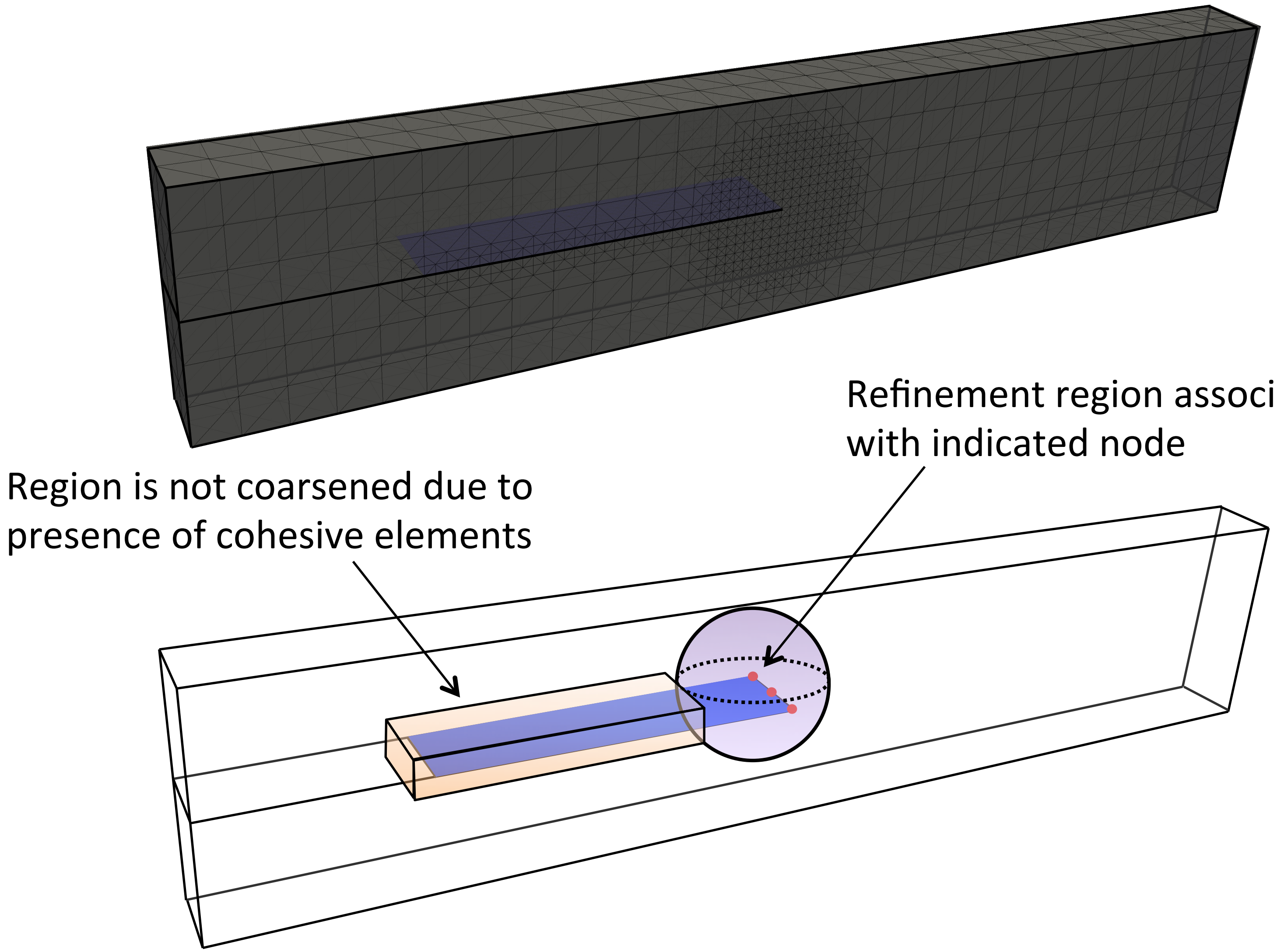


Celes, W., G. H. Paulino, and R. Espinha. *IJNME* 64, no. 11 (2005): 1529–1556.

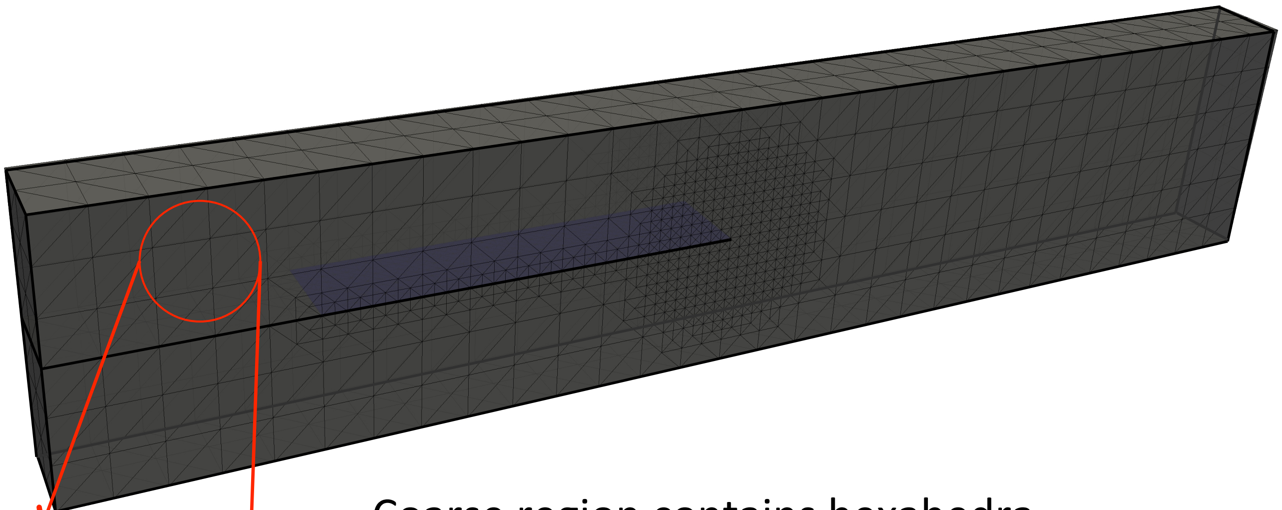
In numerical simulation automatic crack tip tracking modifies mesh discretization on the fly



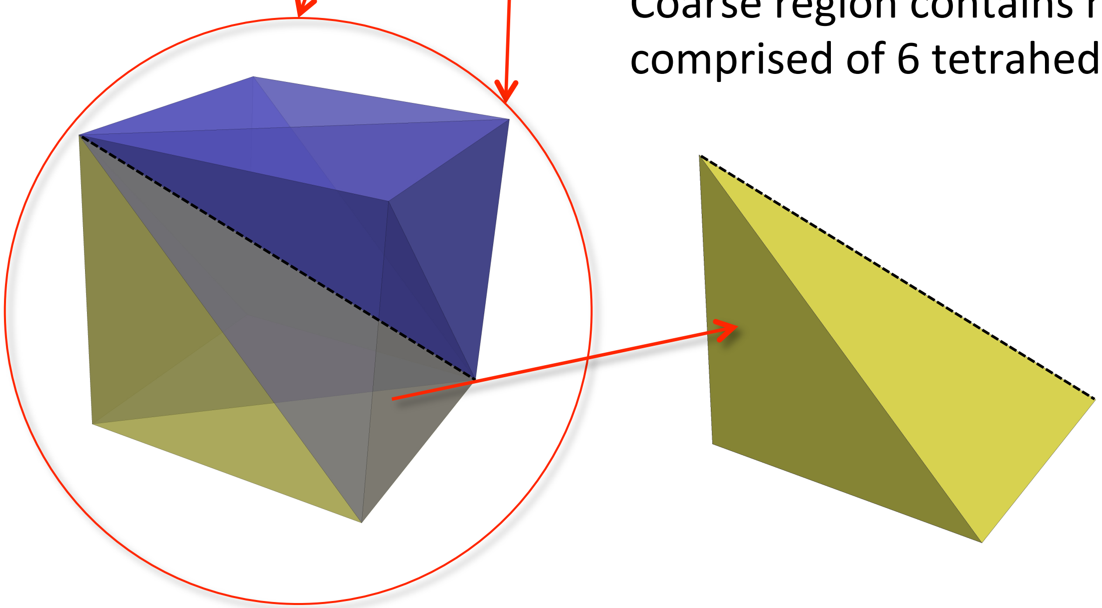
As cracks propagate regions around the crack tips are refined, and others are coarsened



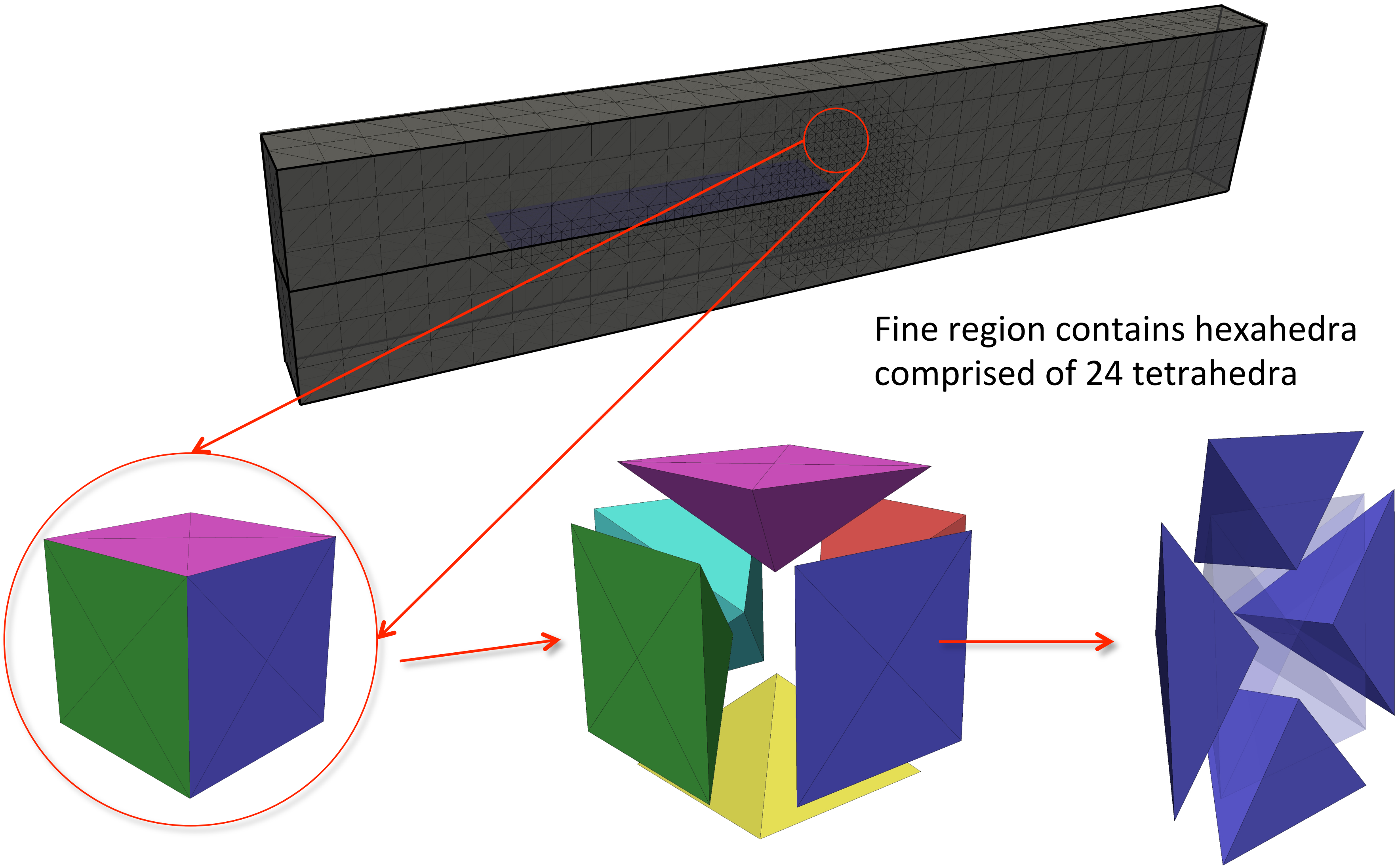
Geometric aspects of the hierarchical 3D 4k mesh



Coarse region contains hexahedra comprised of 6 tetrahedra

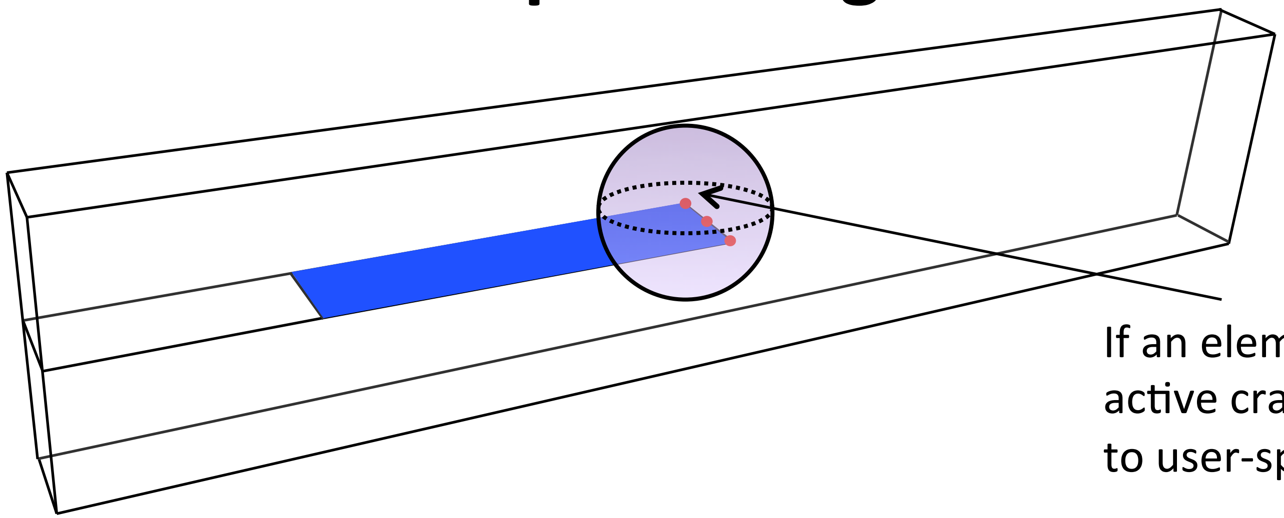


Geometric aspects of the hierarchical 3D 4k mesh



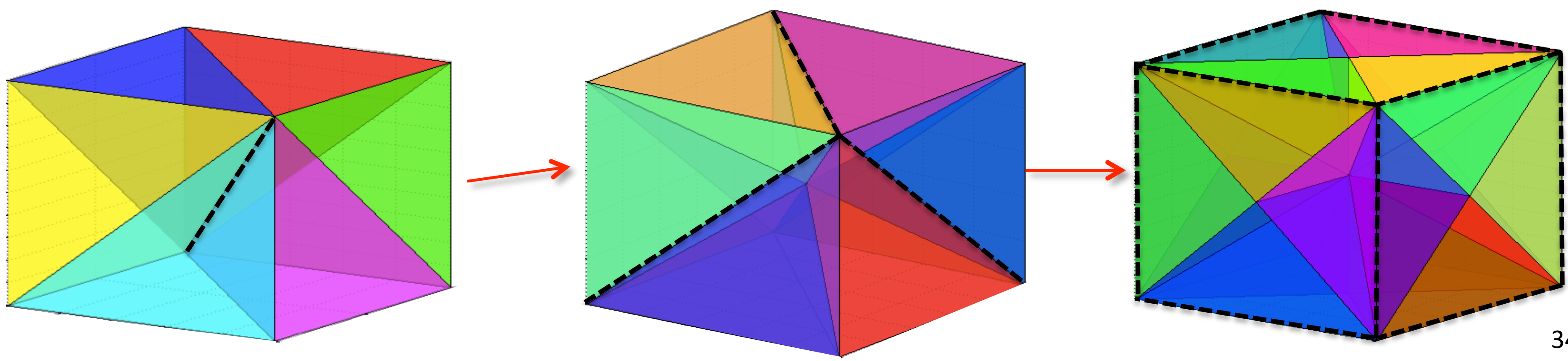
Fine region contains hexahedra comprised of 24 tetrahedra

Mesh refinement is executed by means of the automatic crack tip cracking

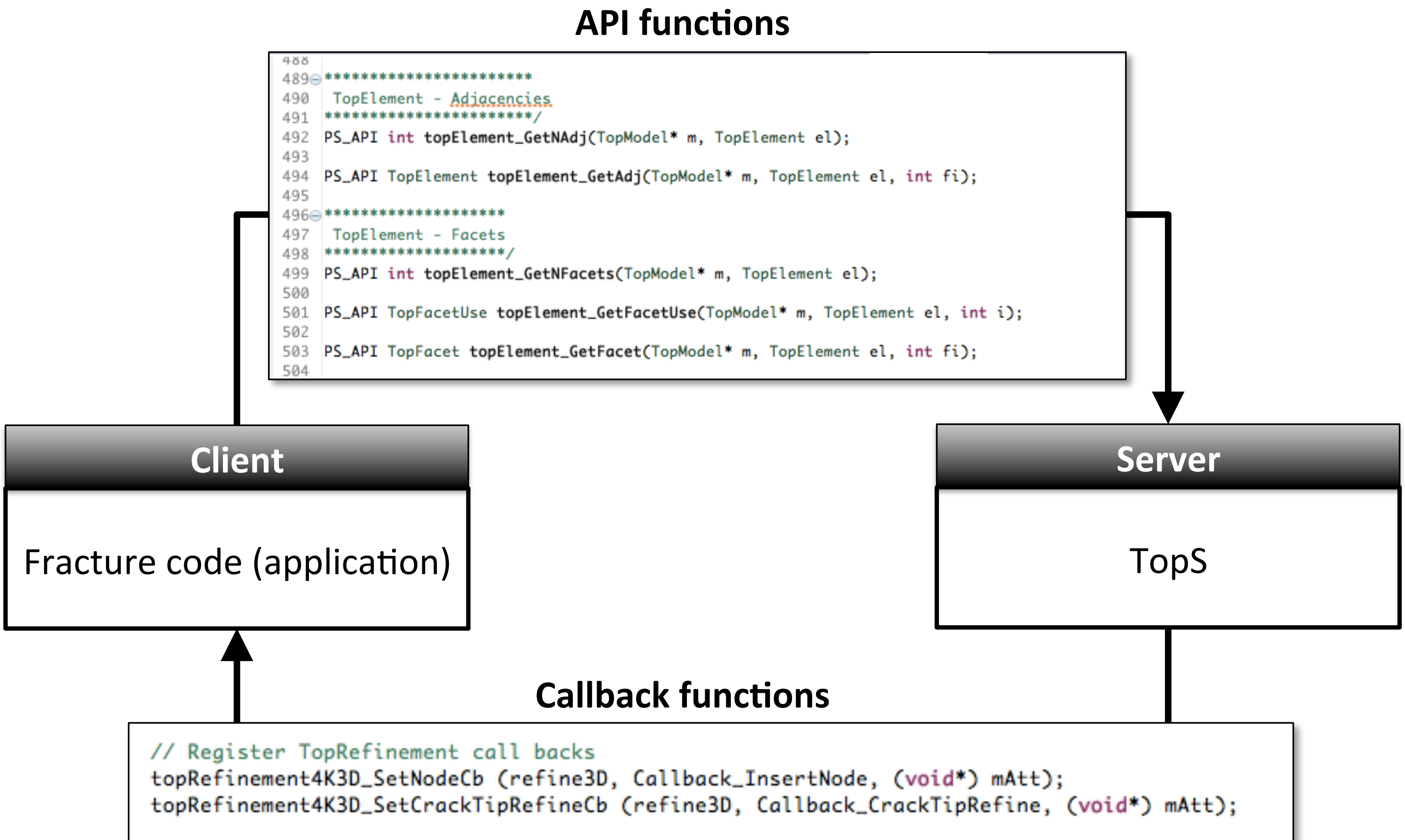


If an element falls inside an active crack tip region, refine to user-specified depth

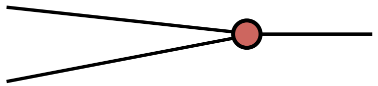
Refine elements by splitting along longest edge: Insert a node at the midpoint of the longest edge and updating connectivity accordingly



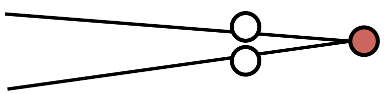
As TopS updates the mesh, the application updates the physical model via callback functions



Communication between TopS and application during mesh refinement

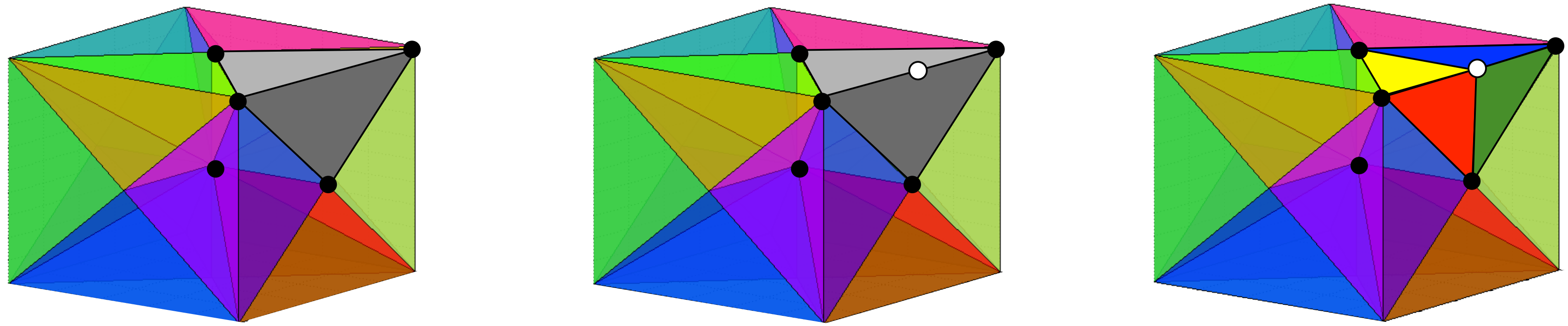


After the cracks have propagated, the application calls the UpdateMeshRefinement Function



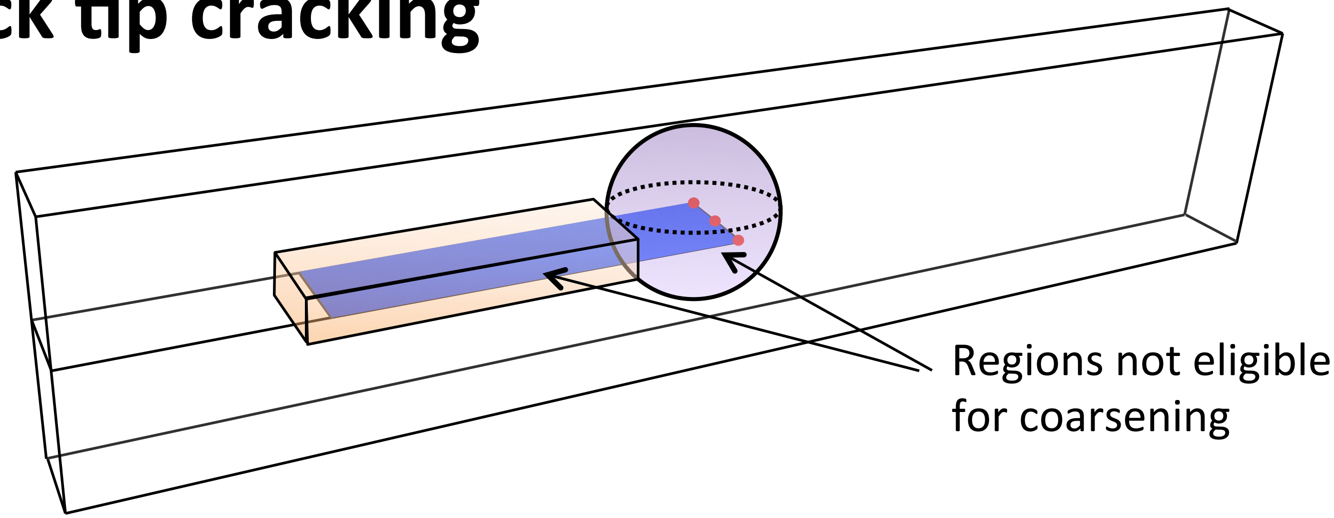
TopS identifies new crack tips and begins refining elements that fall within the refinement regions

TopS notifies the client when a new node is inserted. The client initializes the new node and interpolates its displacement field from neighboring nodes using standard finite element shape functions of parent (grey) elements.

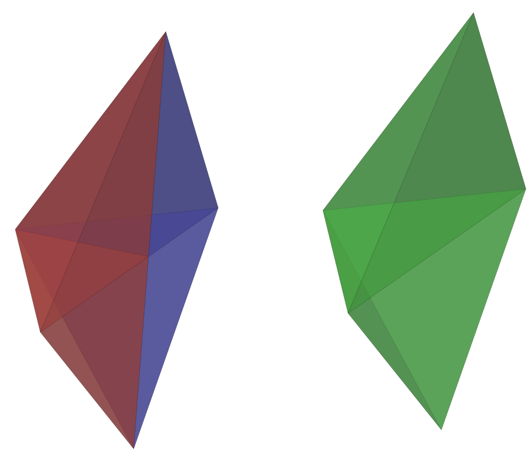


Mesh coarsening is executed by means of the automatic crack tip cracking

Nodes outside current refinement regions are eligible to be removed.

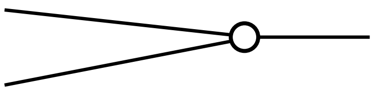


The node will be removed if the norm of the difference in the strain between the original refined patch and the potential coarse patch is less than a certain threshold.

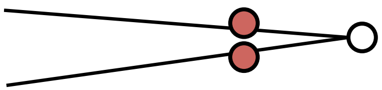


$$e_{\text{patch}} = \left\| \varepsilon_{\text{refined}} - \varepsilon_{\text{coarsened}} \right\|$$

Communication between TopS and application during mesh coarsening

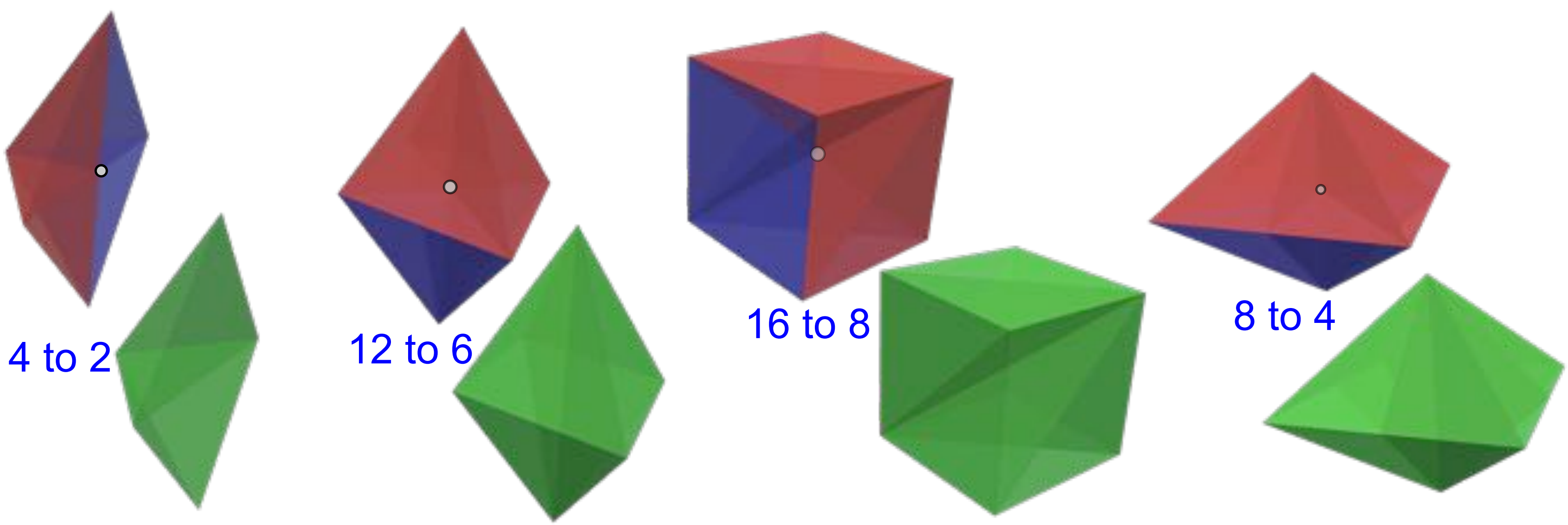


After the cracks have propagated, the application calls the UpdateMeshCoarsening Function



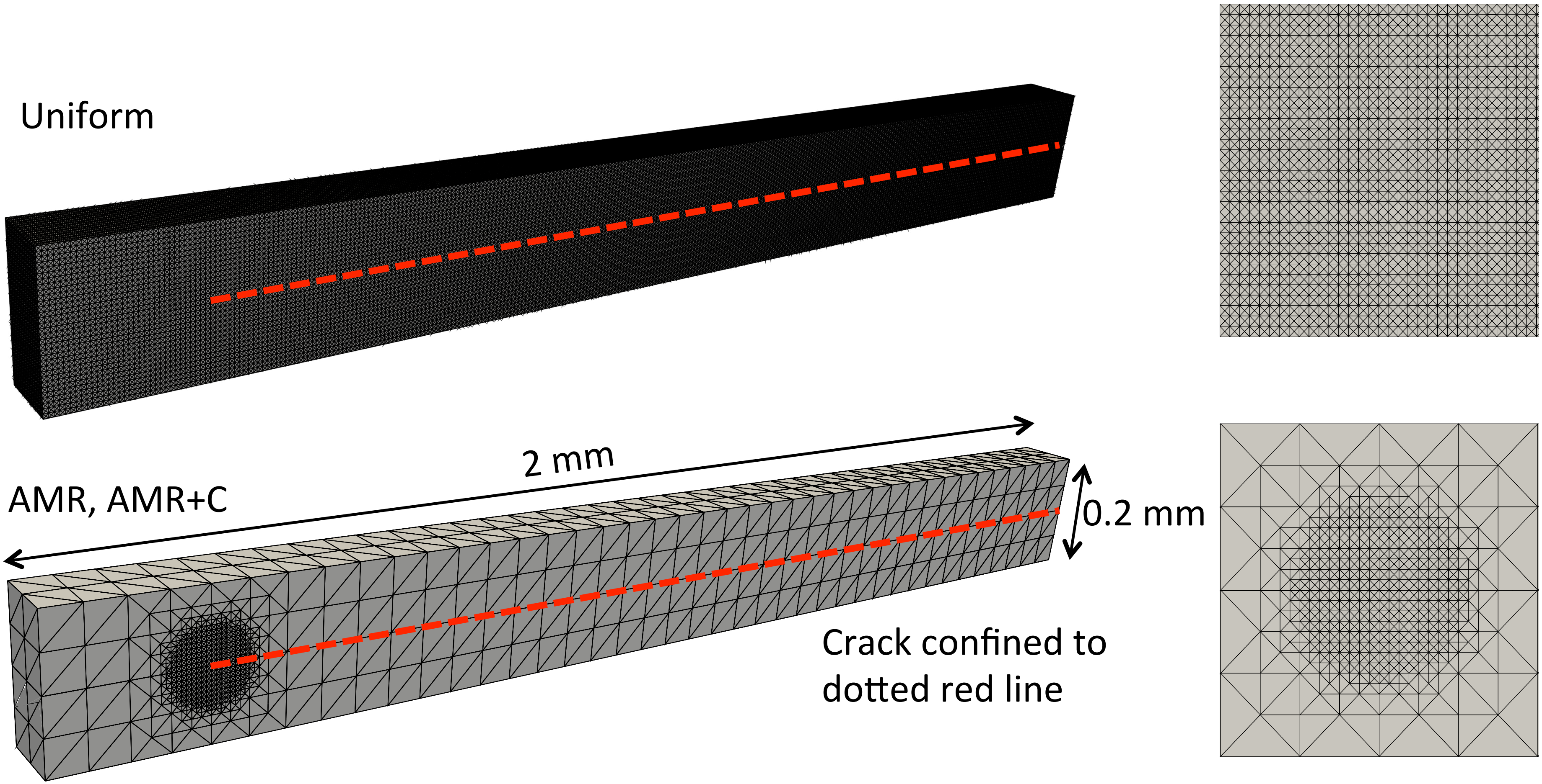
TopS identifies old refinement regions and asks the application if the patches associated these nodes can be coarsened

The application computes the strain on the refined and coarsened patch to determine if the error threshold is met.



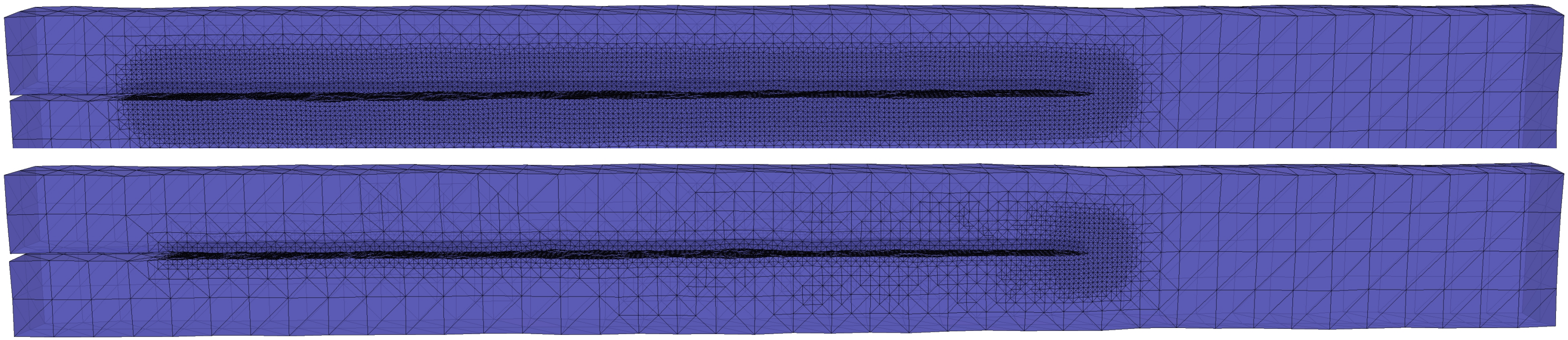
To maintain conformity of the mesh, all elements adjacent to the node to be removed are considered in the patch.

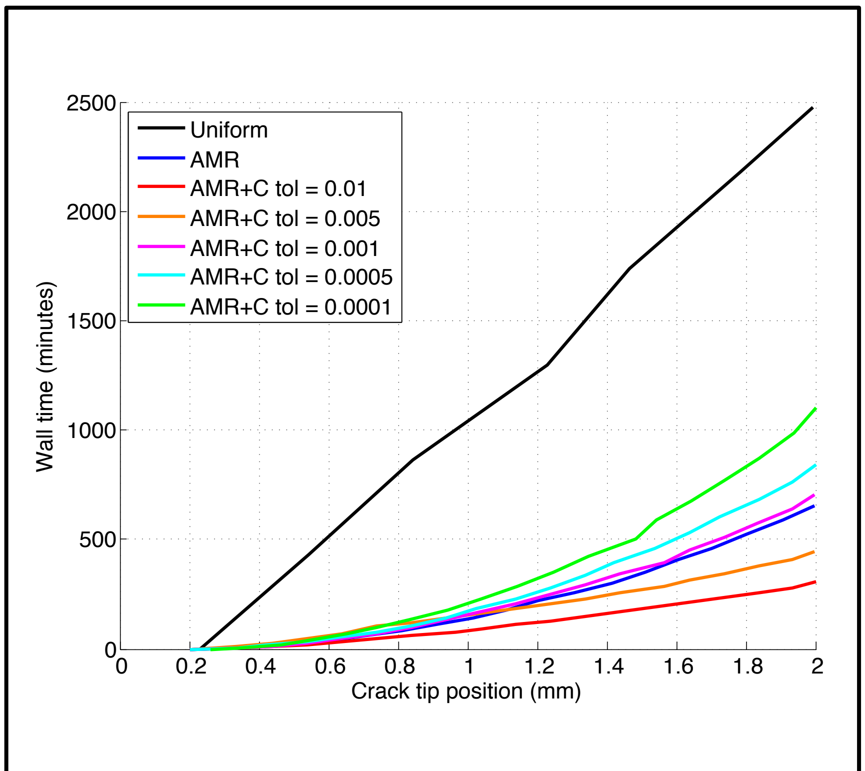
Benchmark problem to investigate AMR+C schemes

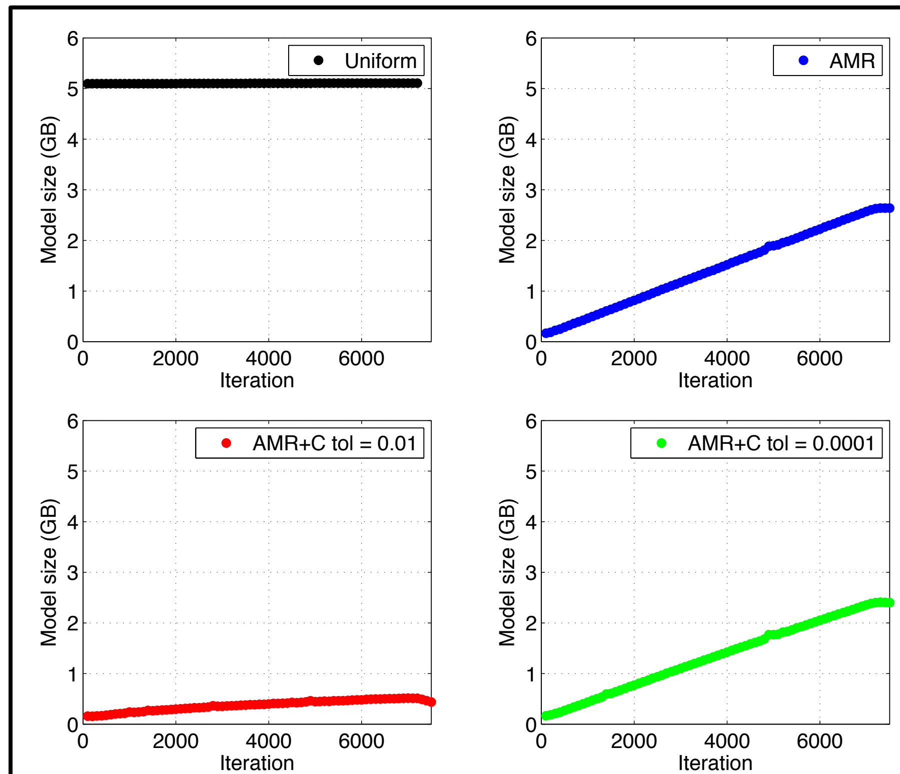


Specimen thickness: 0.05-0.01 mm
Coarse mesh resolution: 12.5-50 μm
Coarsening strain error tolerance: 0.01-0.0001

Refinement saves computational time and space, coarsening is dependent on criteria

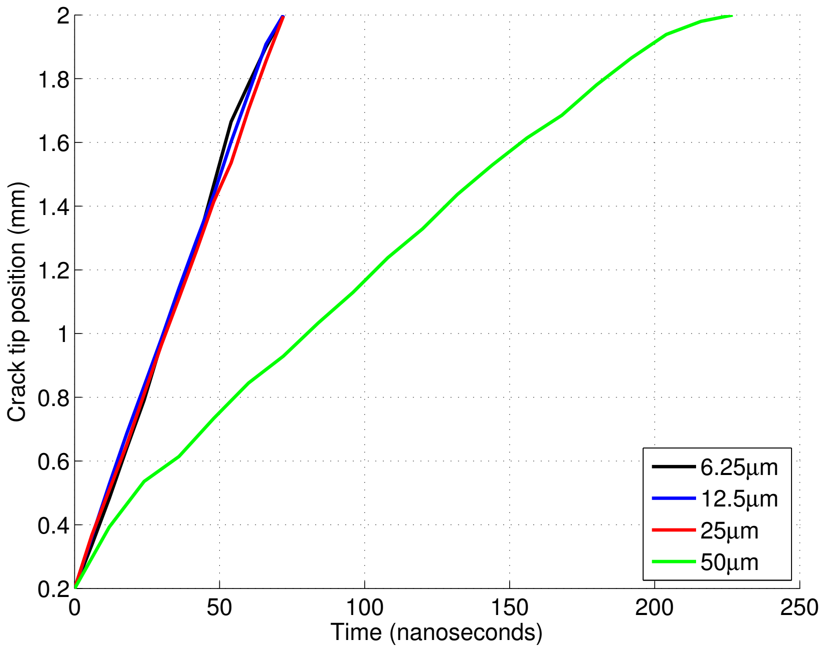






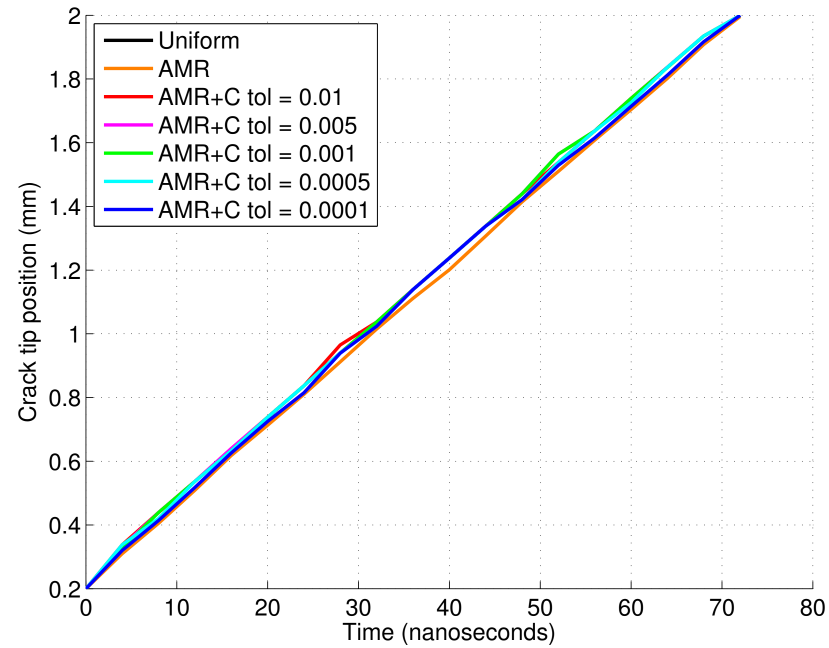
Velocity changes with level of refinement, but is consistent for different coarsening criteria

Variation in size of coarse elements



Propagation is faster in finer far-field mesh

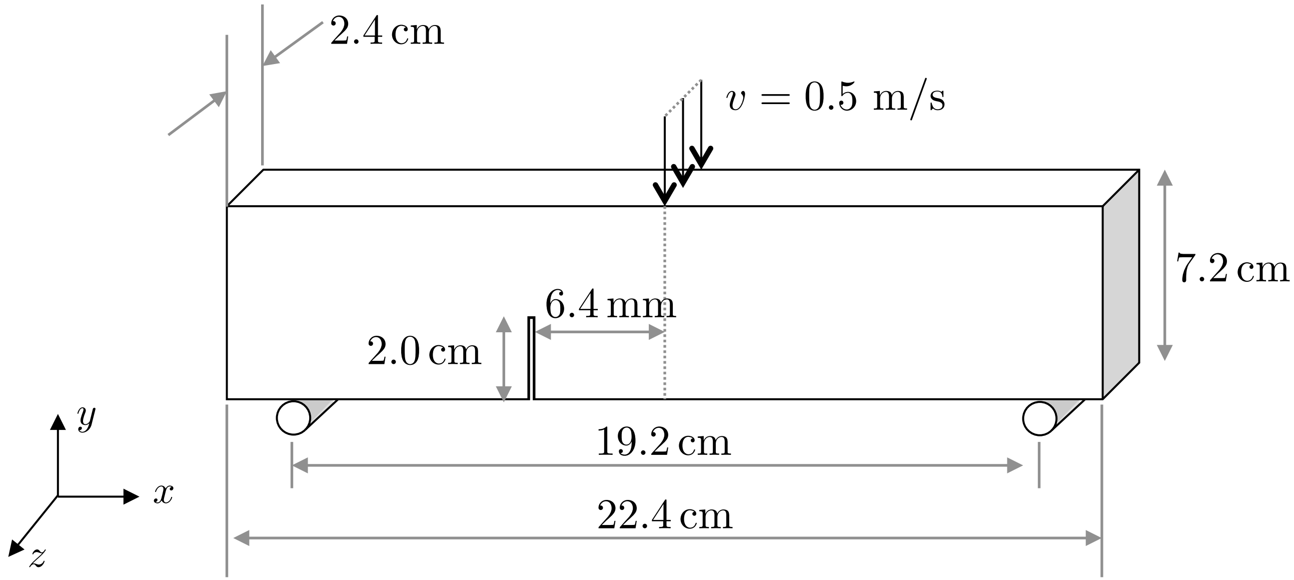
Variation in coarsening tolerance



Refinement and coarsening do not have an impact on the crack front velocity

Leon, S. E., Espinha, R., Celes, W. and Paulino, G. H. "Adaptive refinement and coarsening on structured 3D meshes" In preparation.

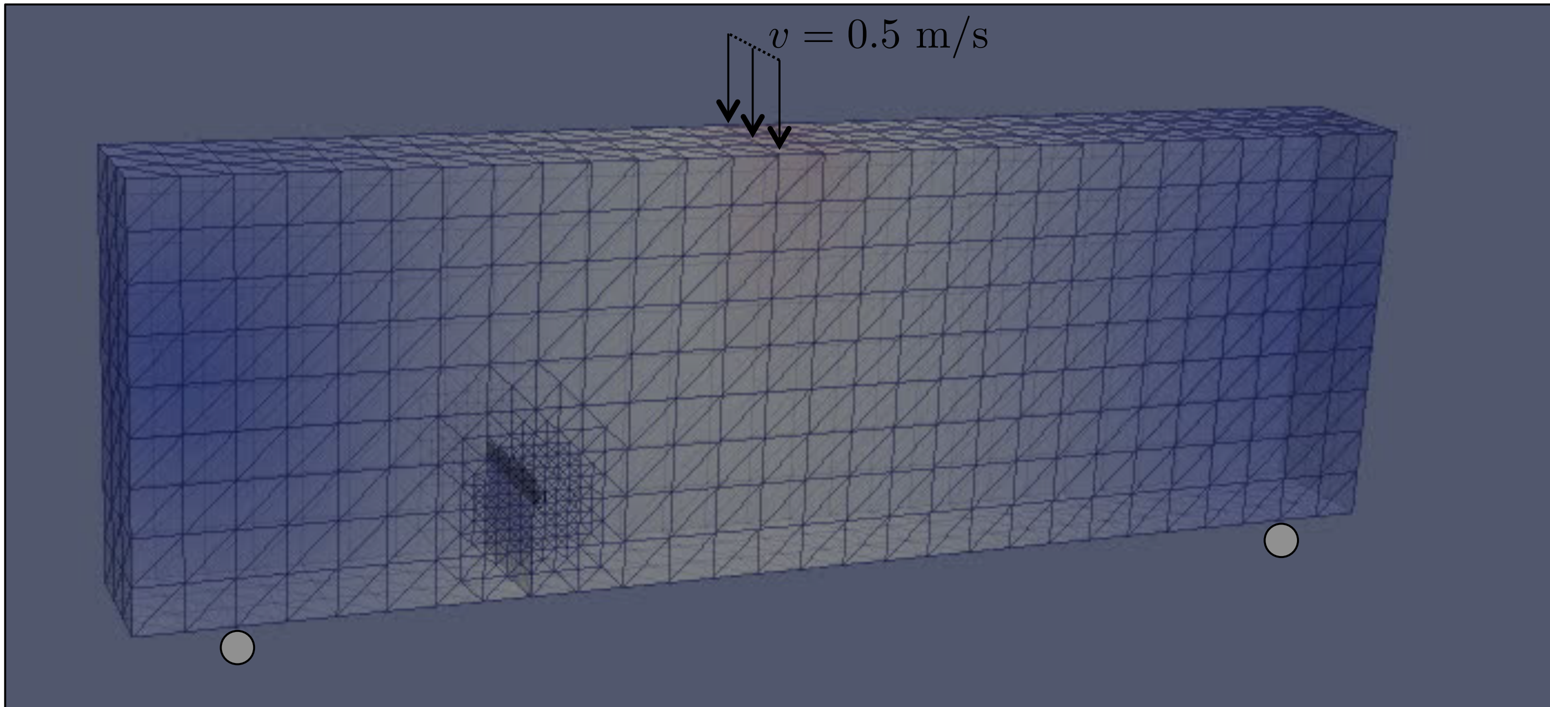
A second benchmark problem demonstrates the use of the method for mixed mode problems



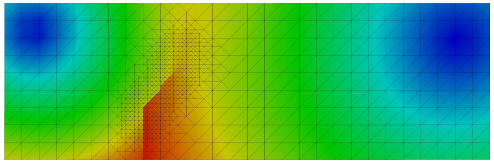
Model size and run time restrictions make this problem computationally challenging:

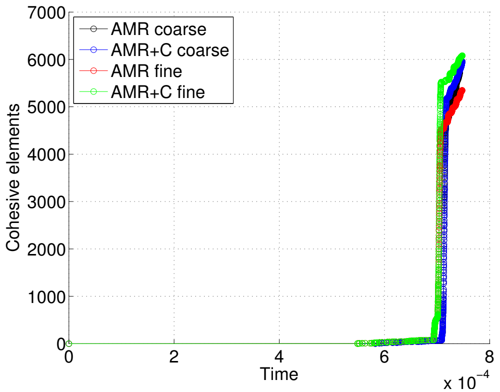
- Uniform refinement of 224x24x72 4k patches ≈ 10 million elements ≈ **47 GB of RAM**
- AMR+C initial refinement ≈ **610 MB of RAM** initially, but takes over **48 hours**
- Choose a coarser level of refinement to make this possible

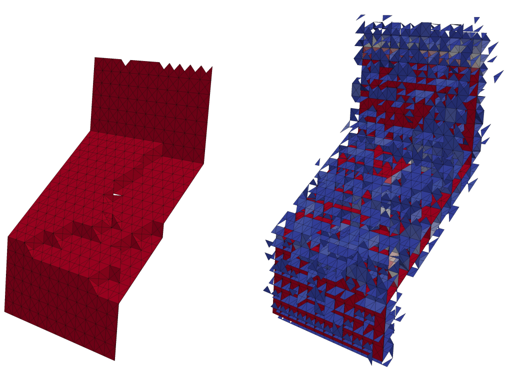
Three point bend simulation



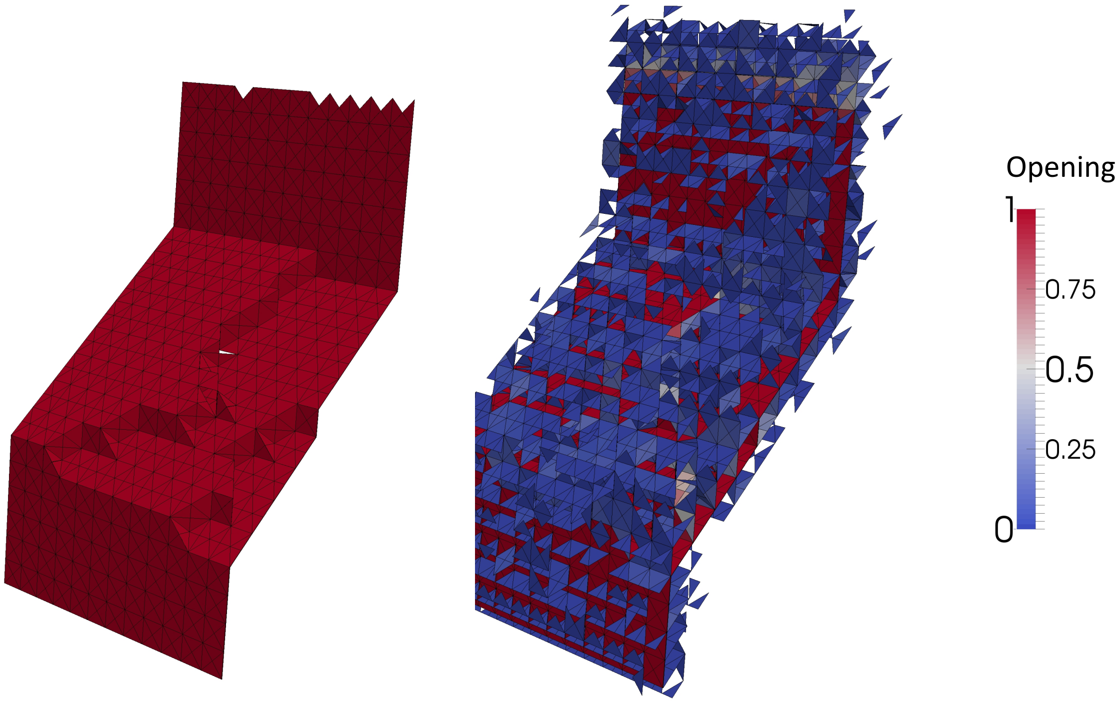
TPB Results



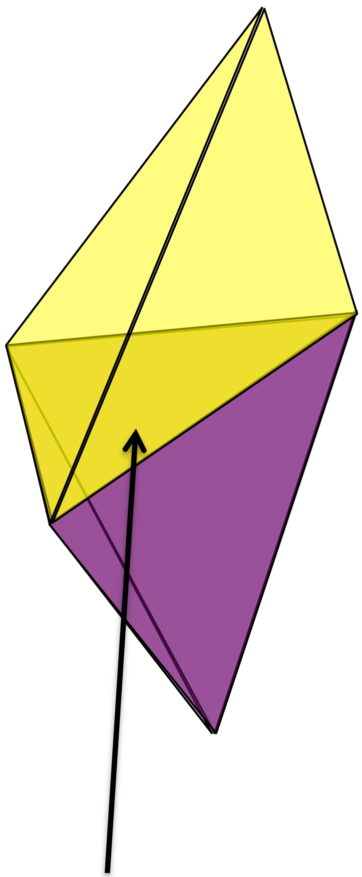




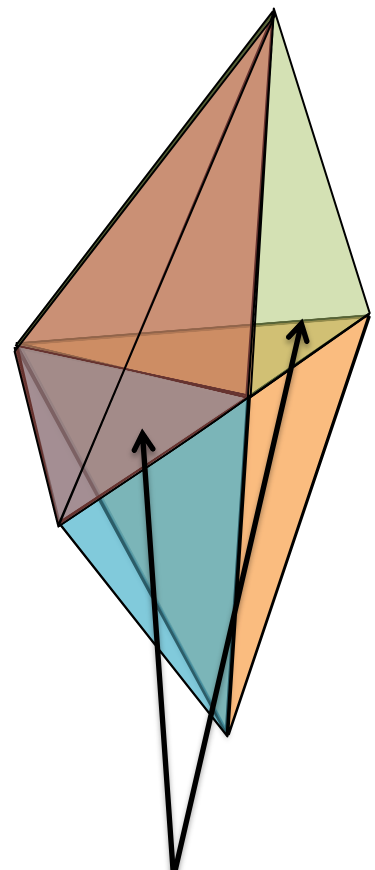
Many cohesive elements are inserted but not all are completely separated



If the material model contains Internal State Variables, we must map them to the new mesh



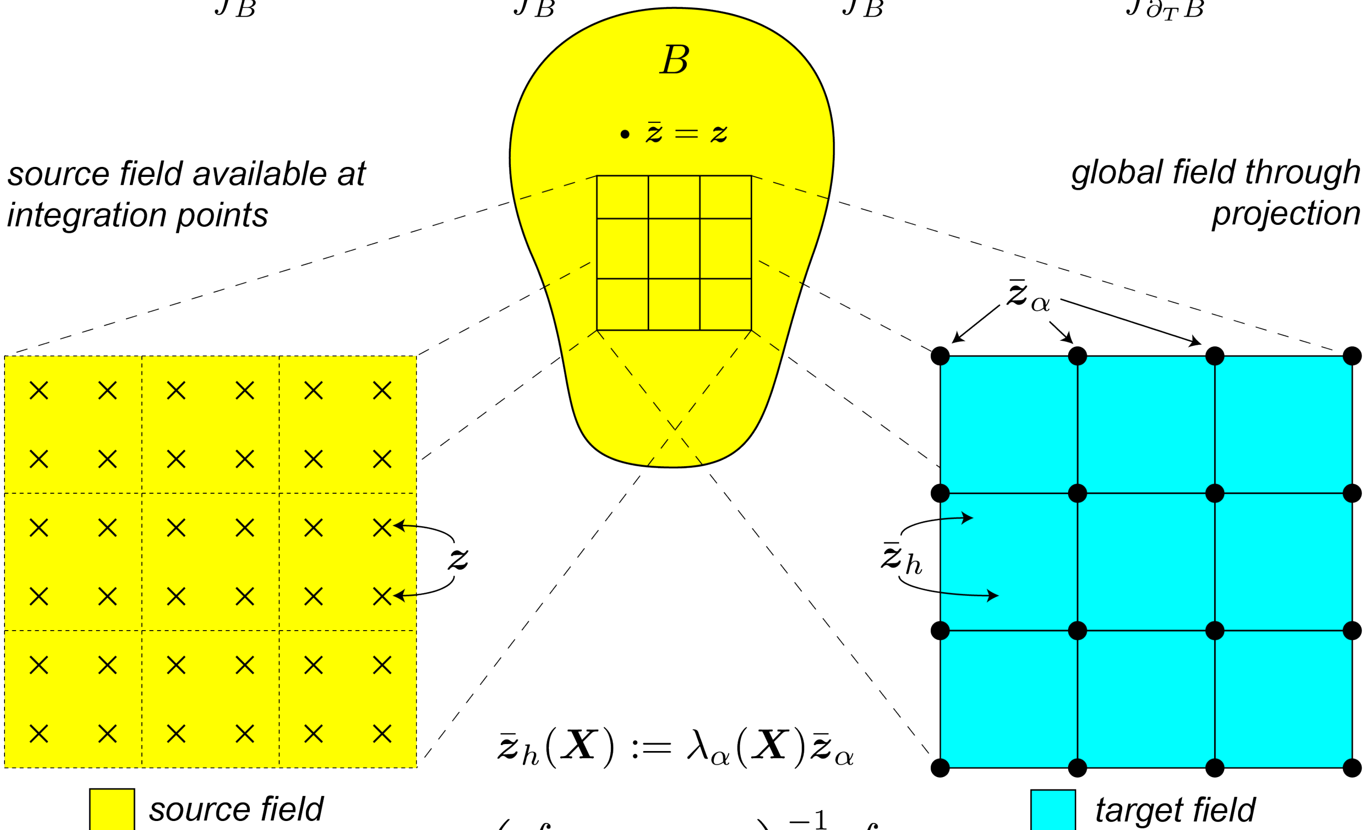
Element variables on original elements



Variables need to be transferred to new elements

Project element variables from one mesh to another by minimizing the error between them

$$\Phi[\boldsymbol{\varphi}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{y}}] := \int_B W(\boldsymbol{F}, \bar{\boldsymbol{z}})\, dV + \int_B \bar{\boldsymbol{y}} \cdot (\bar{\boldsymbol{z}} - \boldsymbol{z})\, dV - \int_B \rho_0 \boldsymbol{B} \cdot \boldsymbol{\varphi}\, dV - \int_{\partial_T B} \boldsymbol{T} \cdot \boldsymbol{\varphi}\, dS$$



$$\bar{\boldsymbol{z}}_h(\boldsymbol{X}) := \lambda_\alpha(\boldsymbol{X}) \bar{\boldsymbol{z}}_\alpha$$

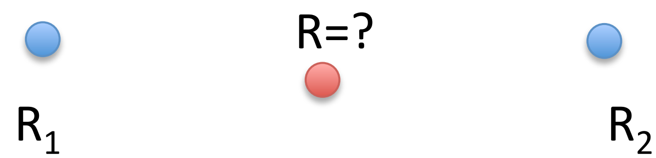
$$\bar{\boldsymbol{z}}_h = \lambda_\alpha \left(\int_B \lambda_\alpha \lambda_\beta \boldsymbol{I}\, dV \right)^{-1} \int_B \lambda_\beta \boldsymbol{z}\, dV$$

Sandia National Laboratories

Mota, A., Sun, W., Ostien, J. T., Foulk, J. W., & Long, K. N. (2013). Computational Mechanics.

Certain element variables can not be projected directly – example rotation matrices

Given rotations at two points, find the rotation at some other point:



Polynomial interpolation of rotations does not make sense because rotations belong to a multiplicative group, specifically the Special Orthogonal, *SO(3)*, Lie Group

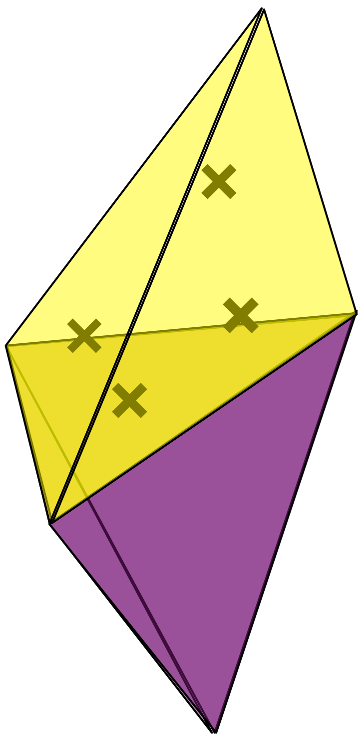
$$\mathbf{R} \neq \frac{\mathbf{R_1} + \mathbf{R_2}}{2} \qquad \mathbf{R} \in SO\left(3\right) = \left\{ \mathbf{A} \in M\left(n\right) \middle| \mathbf{A}\mathbf{A}^T = \mathbf{I},\, \det \mathbf{A} = 1 \right\}$$

In order to produce a variable that belongs to a Lie group we can map it to its Lie Algebra where addition is admitted. The Lie Algebra of SO are skew-symmetric matrices, *so(3)*

$$\log \mathbf{R} = \mathbf{r} \in so\left(3\right) = \left\{ \mathbf{B} = M\left(n\right) \middle| \mathbf{B} = -\mathbf{B}^T \right\}$$

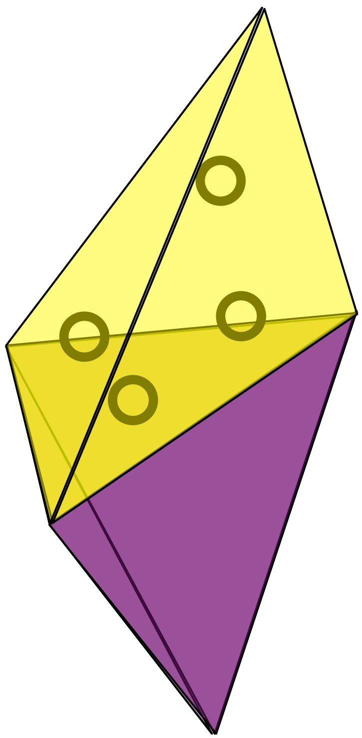
Sandia National Laboratories

Lie group interpolation and L2 error minimizing projection



Lie Group ×

Lie group interpolation and L2 error minimizing projection

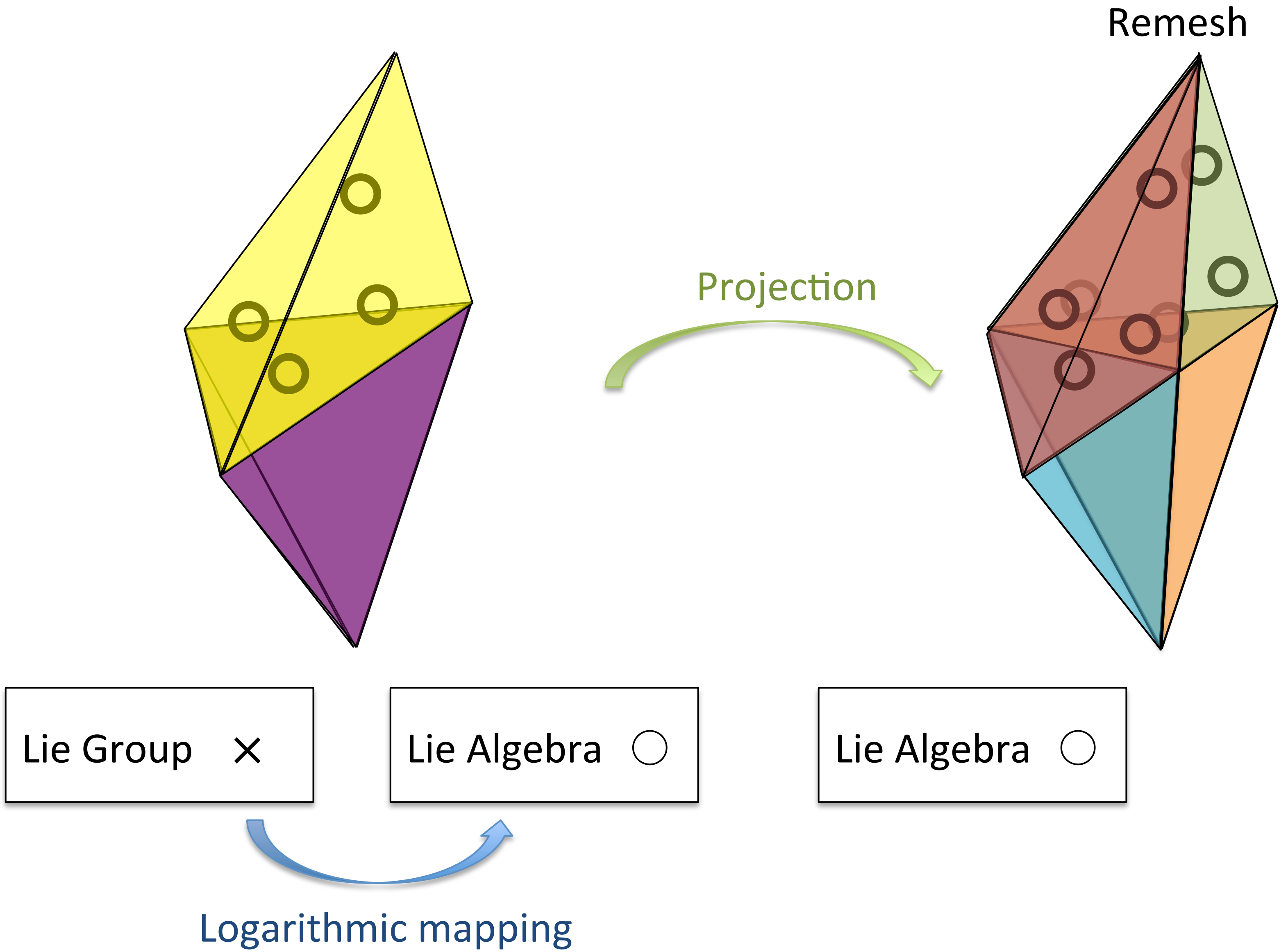


Lie Group ×

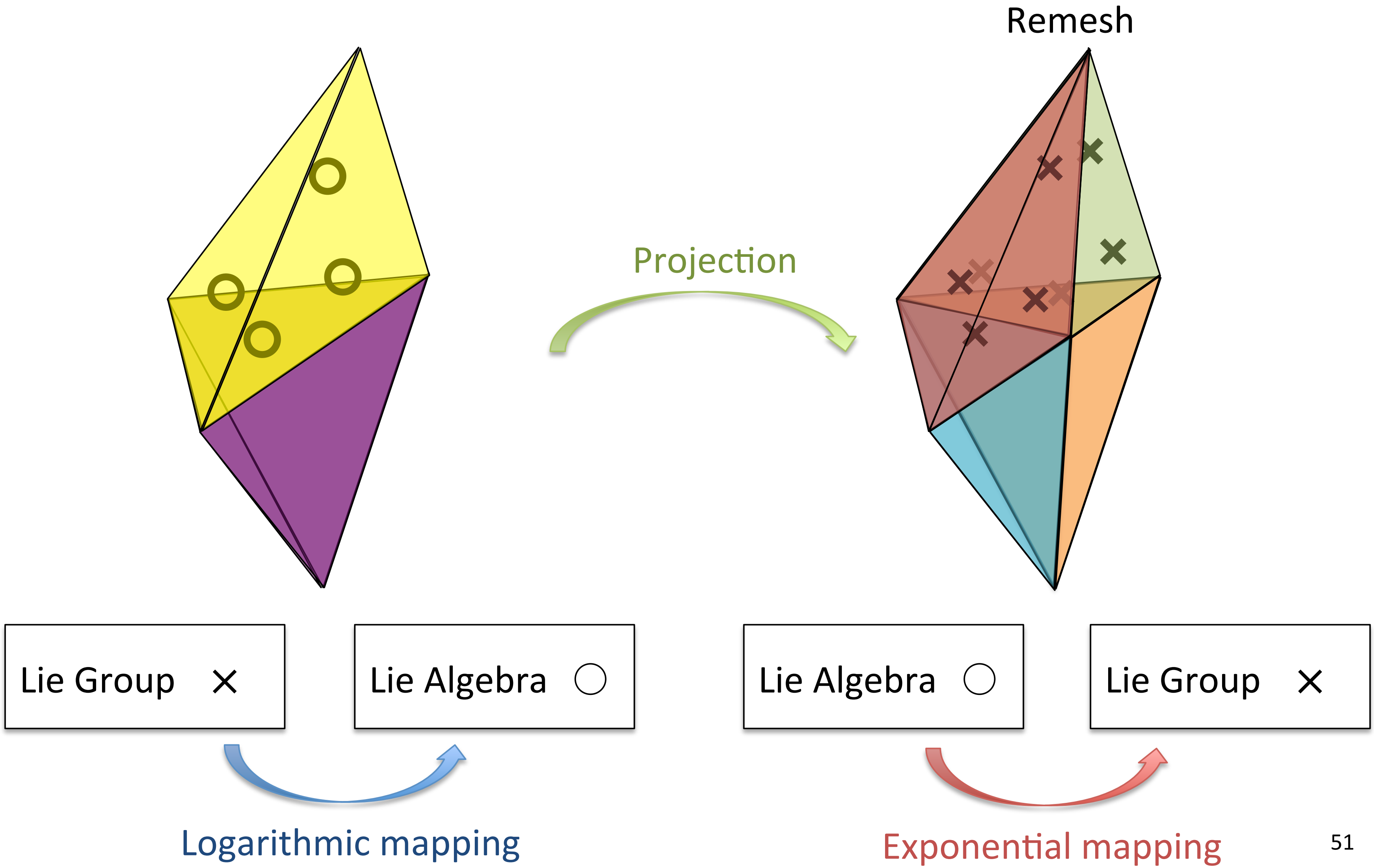
Lie Algebra ○

Logarithmic mapping

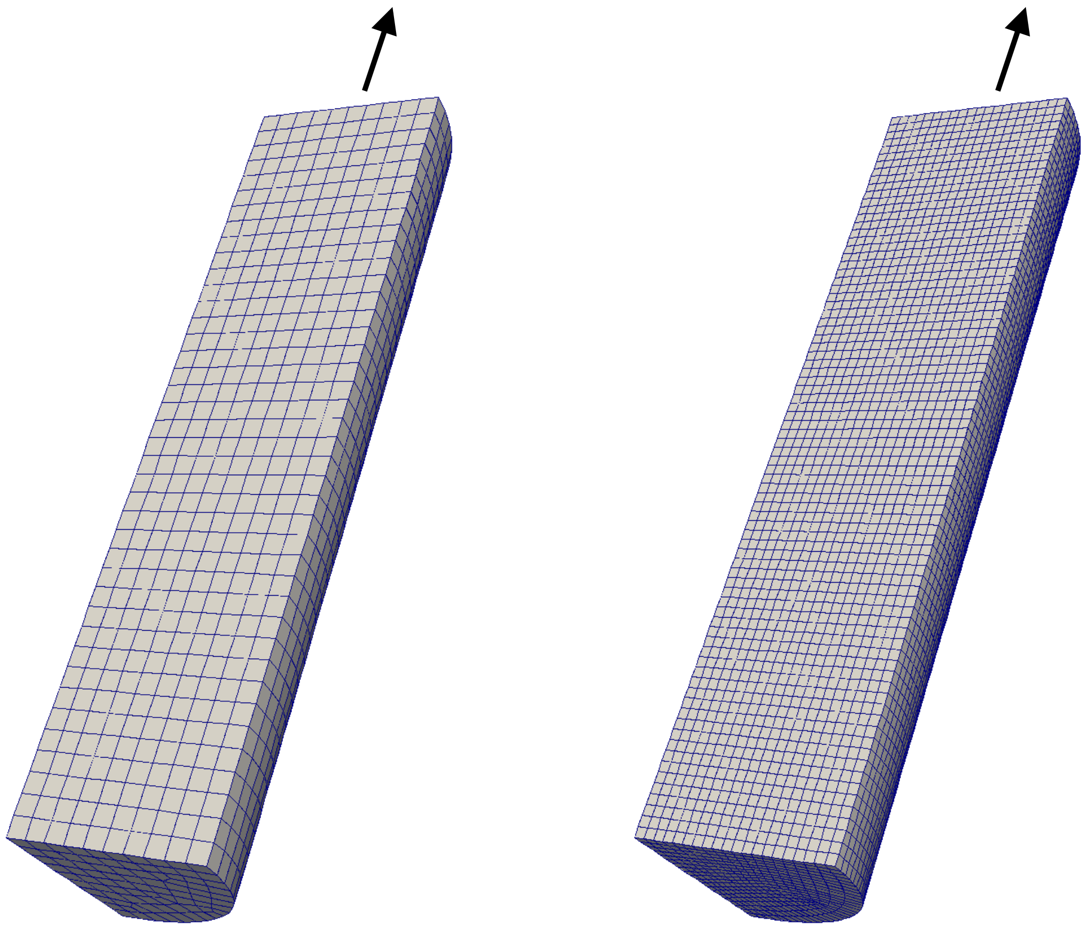
Lie group interpolation and L2 error minimizing projection



Lie group interpolation and L2 error minimizing projection



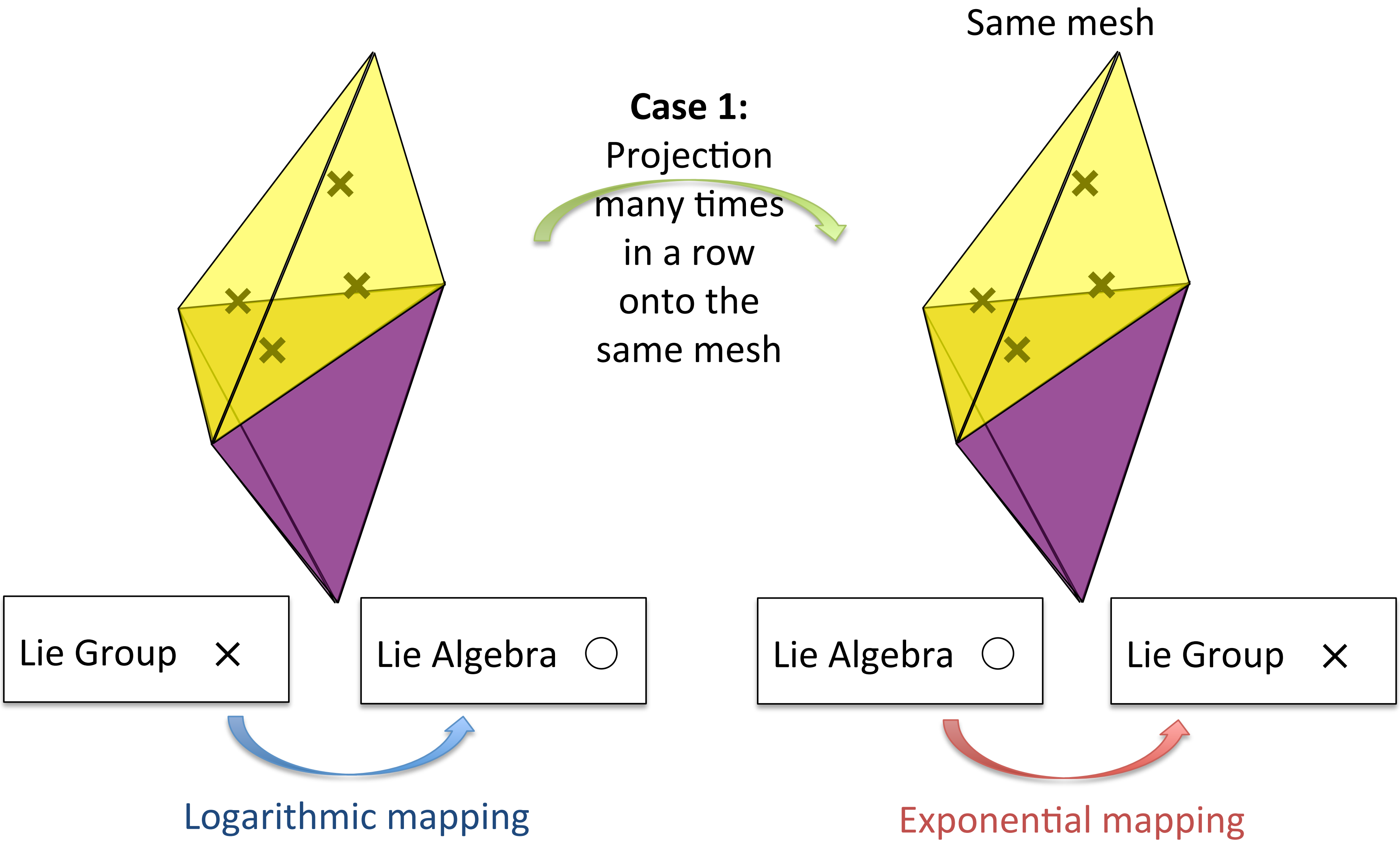
Uniaxial tension of a smooth bar is used to investigate the remeshing and mapping procedure



coarse mesh, 10 elements across thickness

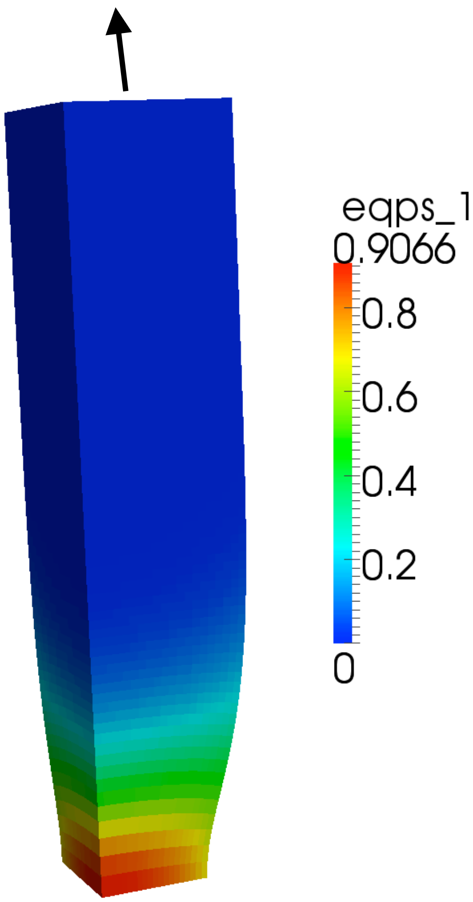
fine mesh, 20 elements across thickness

Numerical study 1: Mapping without remeshing

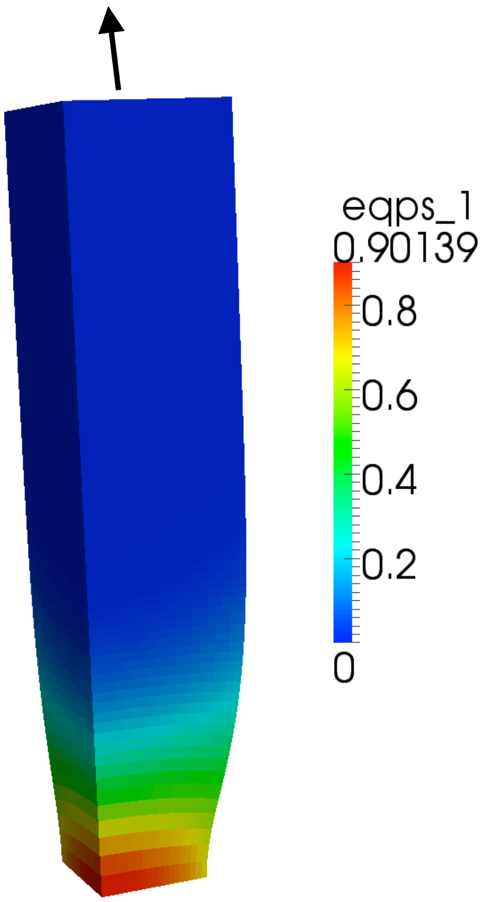


Some diffusion in the internal state variables is present when many remaps are performed

Equivalent plastic strain in fine mesh at one integration point per element at end of analysis

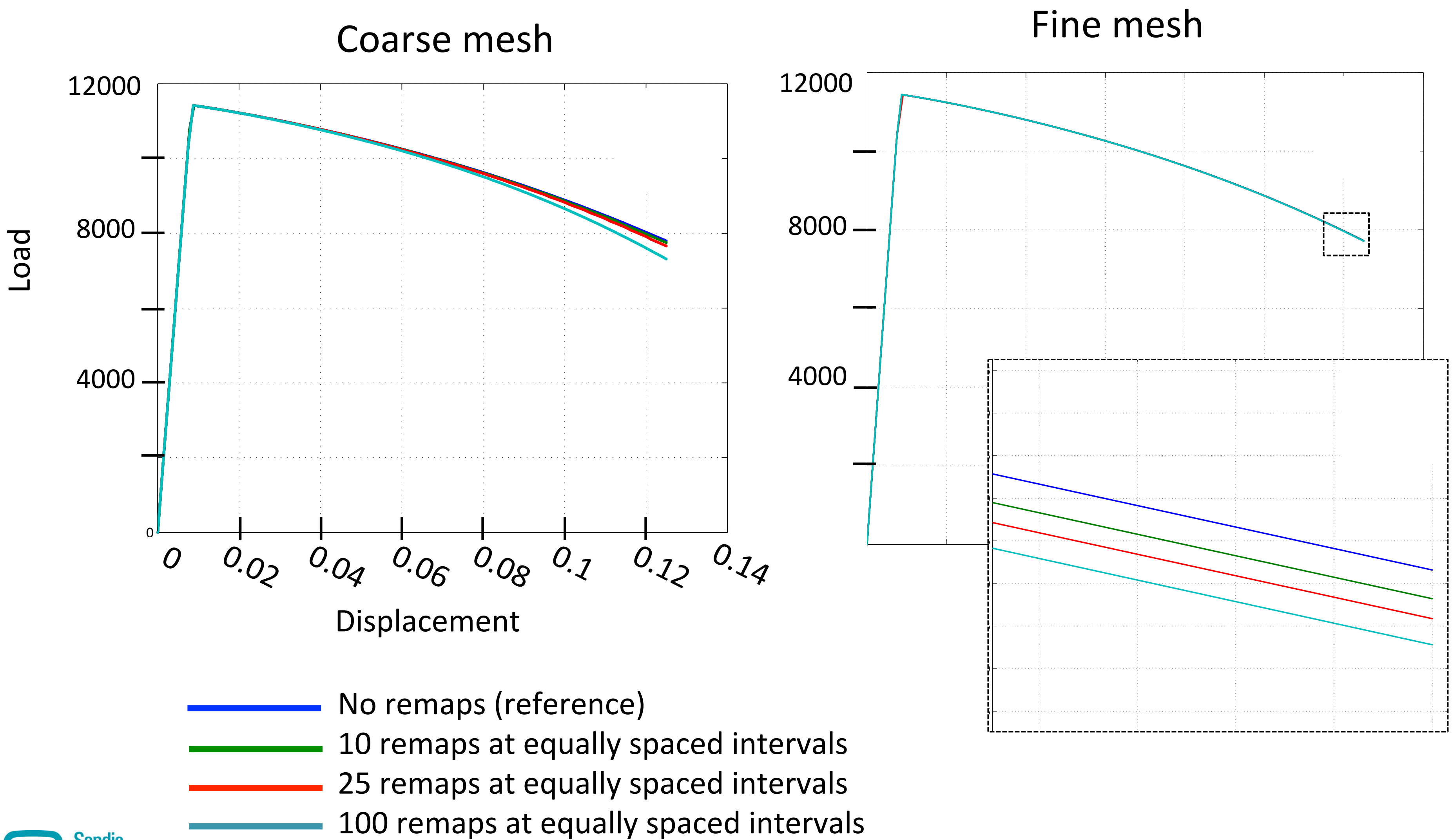


Reference -
No remapping

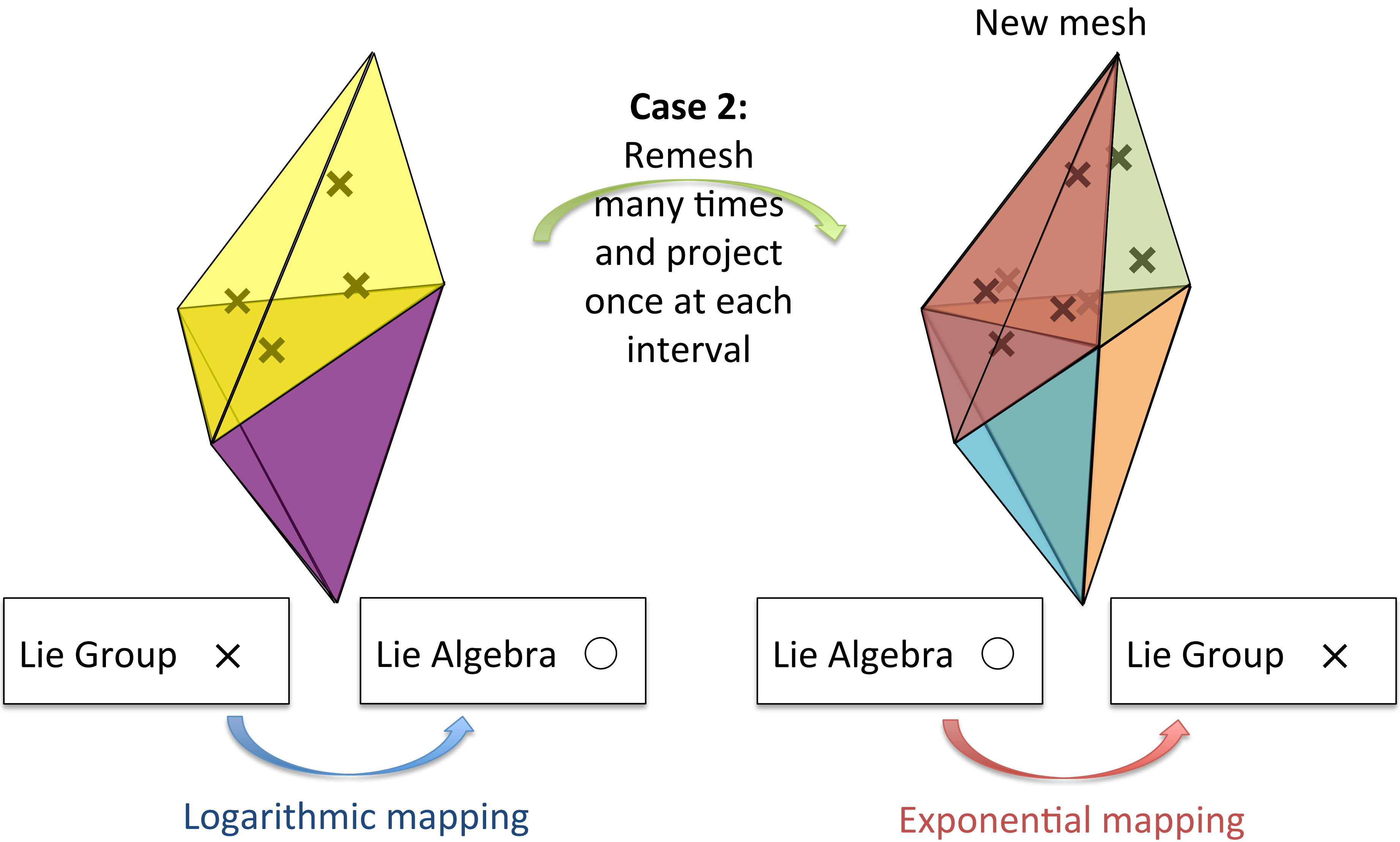


Remap 100 times between
t = 0 and t=0.25

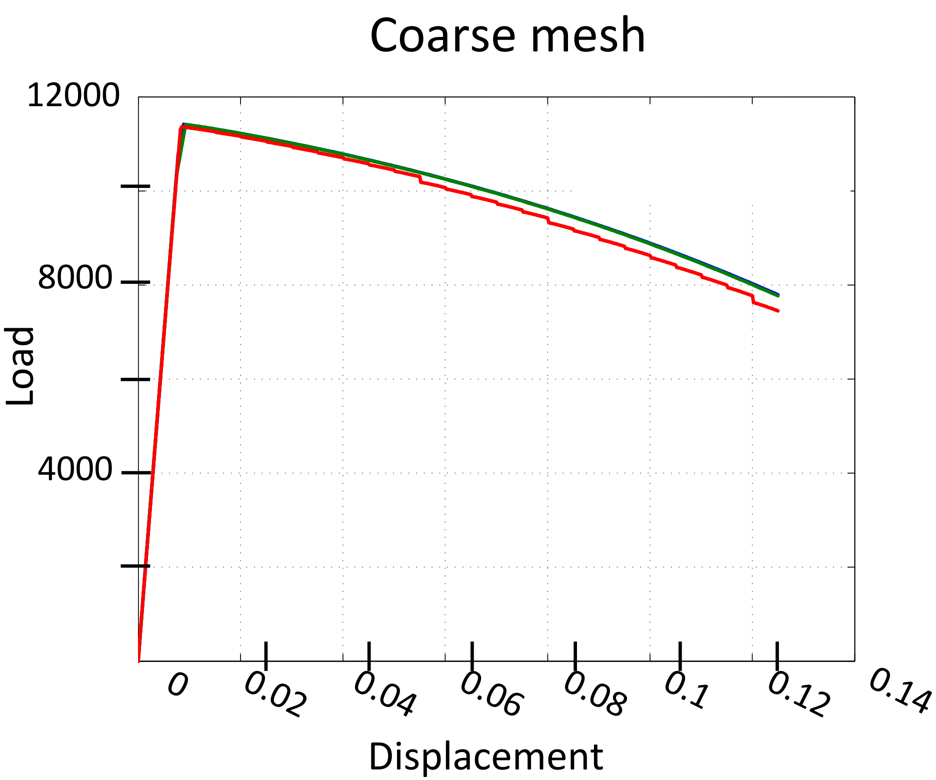
The loss is less prevalent in a fine mesh than in a coarser mesh

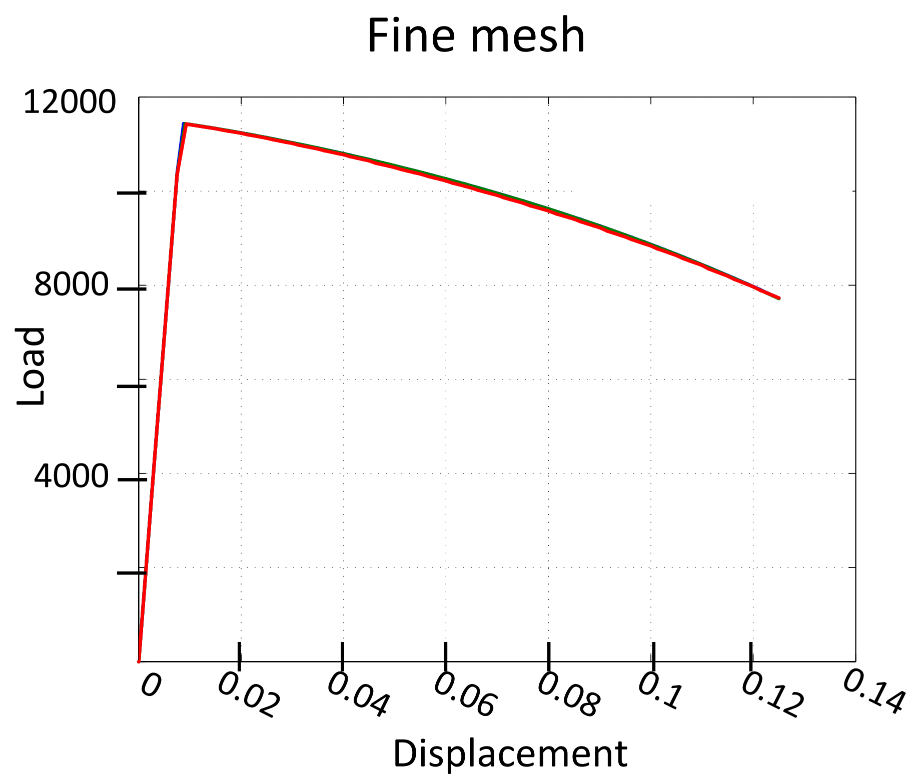


Numerical study 2: Mapping and remeshing



Some loss is present in the load-displacement curve, but it reduces with mesh refinement



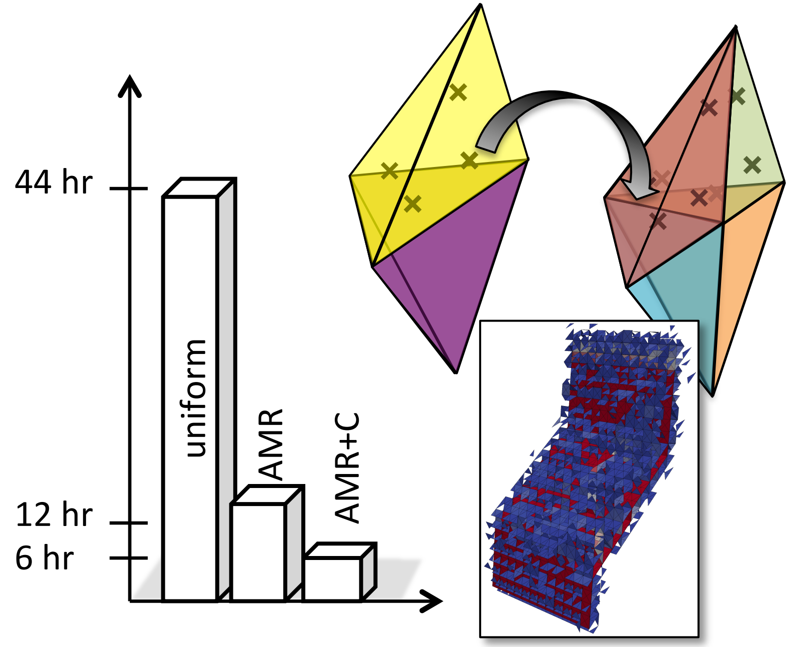


No remeshing or mapping (reference)
25 remaps at equally spaced intervals **without** remeshing
25 remaps at equally spaced intervals **with** remeshing

Sandia National Laboratories

The adaptive schemes explored in this work result in:

Improved solutions over non adaptive schemes – **Adaptive polygonal splitting**



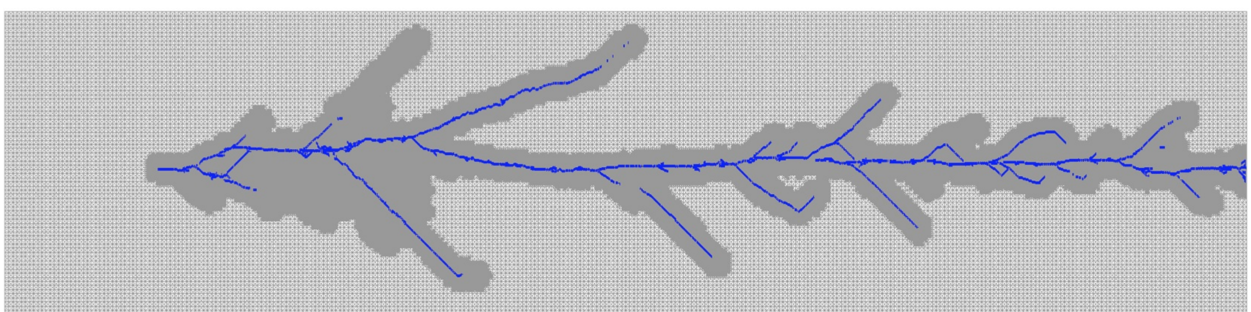
Increased computational efficiency – **3D refinement and coarsening**

Enables solutions to complicated problems – **GPU Adaptivity**

The adaptive schemes explored in this work result in:

Improved solutions over non adaptive schemes – **Adaptive polygonal splitting**

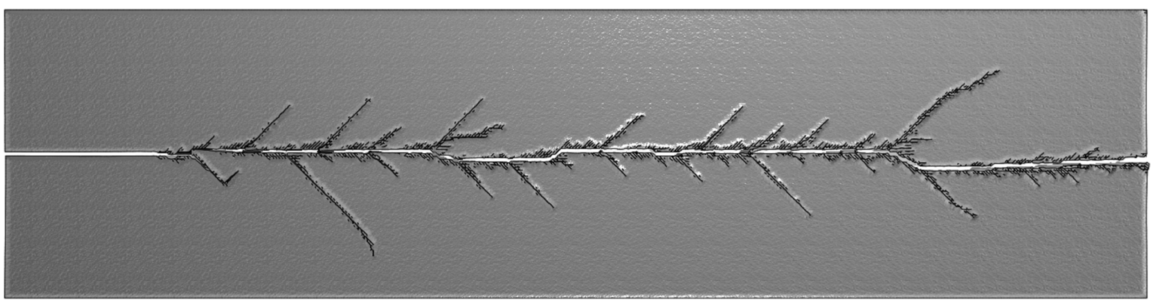
Increased computational efficiency – **3D refinement and coarsening**



Enables solutions to complicated problems – **GPU Adaptivity**

Adaptive fracture simulation on a GPU

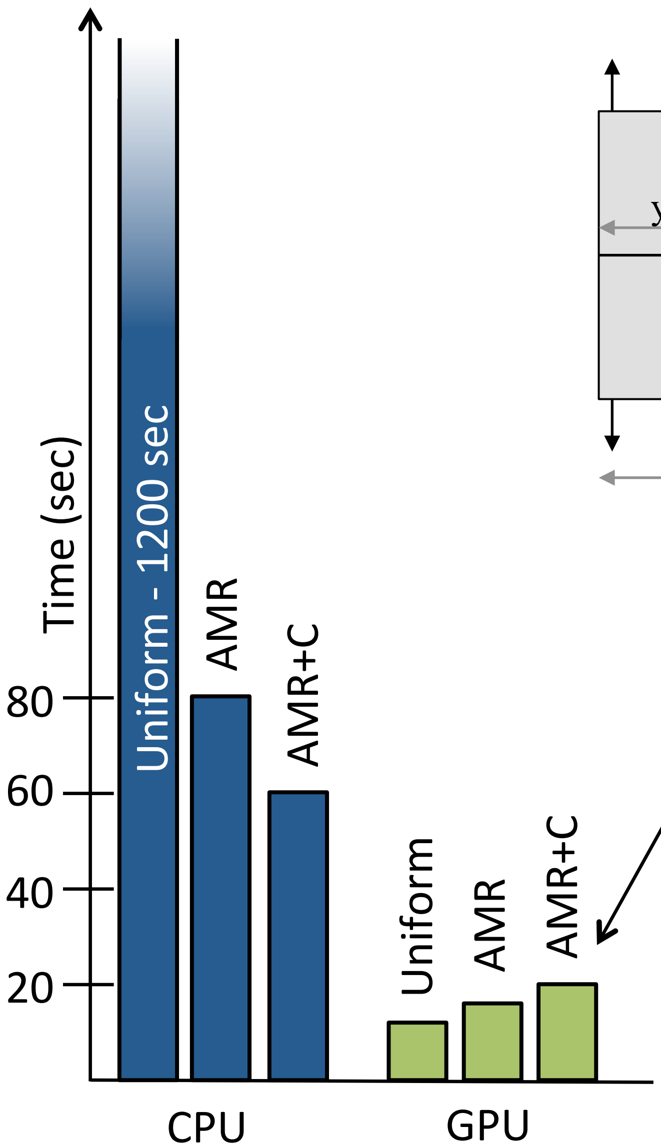
- A GPU is a massively parallel system, could run thousands of threads at once
- GPU fracture achieved speed up over the CPU implementation

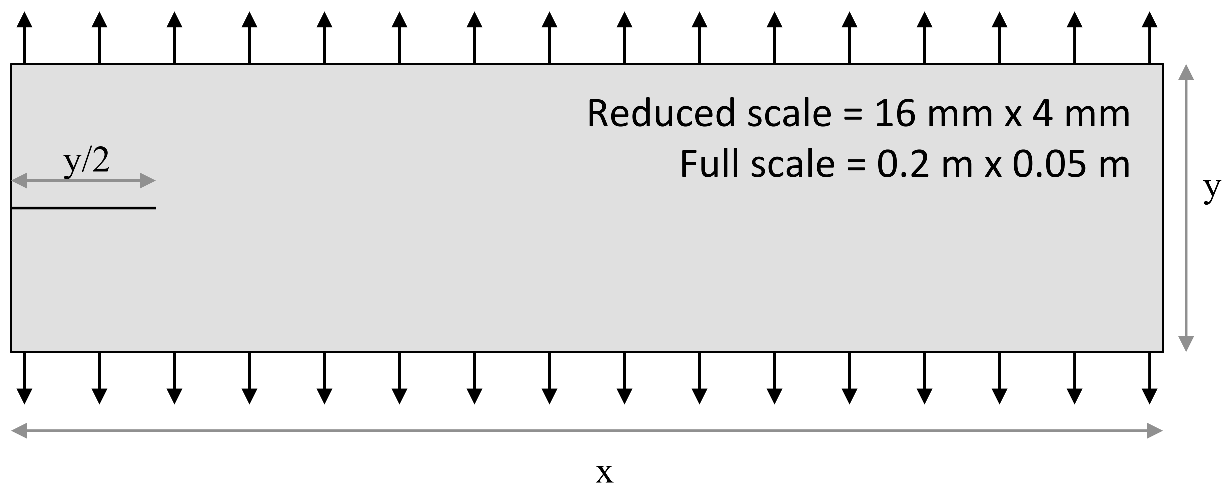


No. of bulk elements	Timestep	CPU time	GPU time	Speedup
36,864	2.0e-9	410.181 s	11.788 s	34.8
147,456	0.5e-9	6,537.839 s	153.809 s	42.5

Alhadeff, A., W. Celes, and G. H. Paulino. “Mapping Cohesive Fracture and Fragmentation Simulations to GPUs.” *IJNME.* Accepted.

GPU is significantly faster than CPU implementations and AMR+C make larger problems feasible



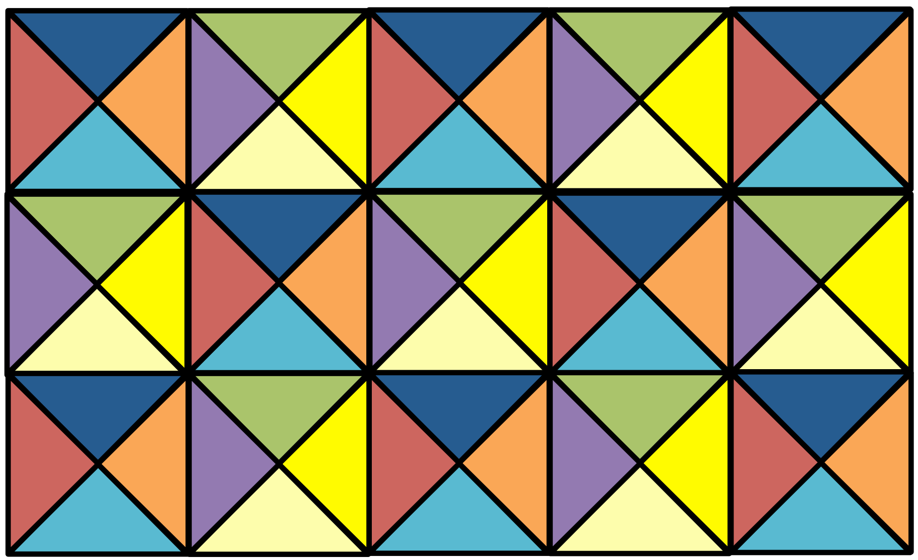


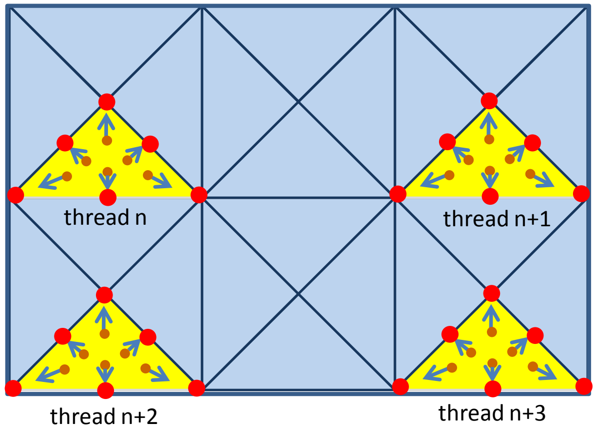
AMR and AMR+C are more expensive on the GPU than the no adaptivity case for the reduced scale model

Full scale problem is not possible with uniform refinement on the GPU due to memory restrictions

Adaptive mesh refinement and coarsening on a GPU

- In order to avoid the race condition, previous works have used a graph coloring scheme

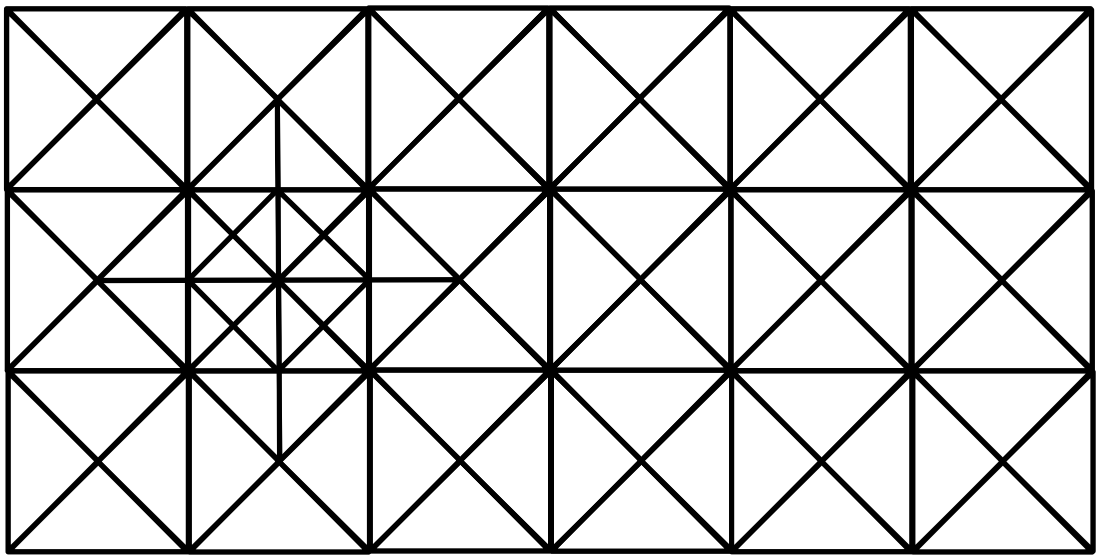


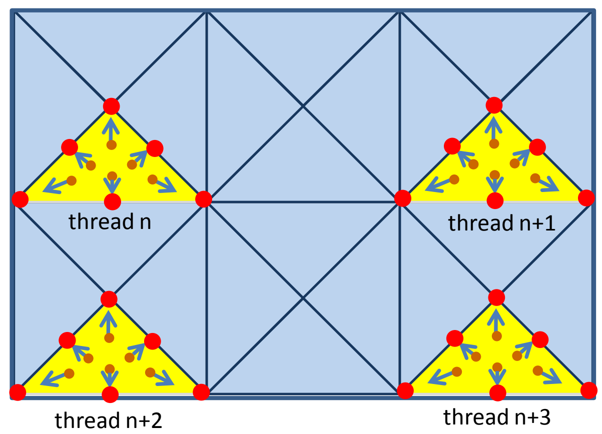


Alhadeff, A., W. Celes, and G. H. Paulino. "Mapping Cohesive Fracture and Fragmentation Simulations to GPUs." *IJNME.* Accepted.

Adaptive mesh refinement and coarsening on a GPU

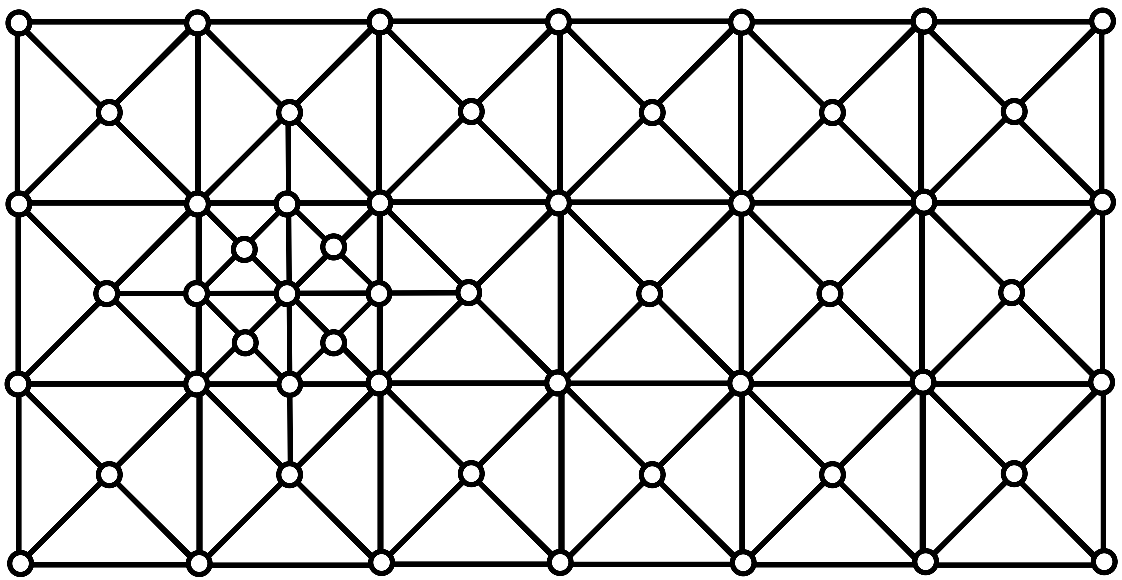
- In order to avoid the race condition, previous works have used a graph coloring scheme
- However, it is too expensive to color the mesh every time the number of elements change

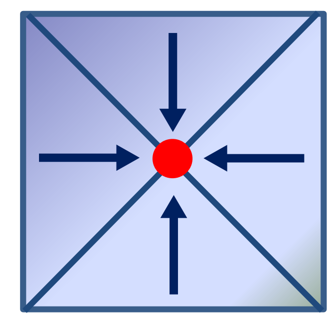




Adaptive mesh refinement and coarsening on a GPU

- We will employ a node-by-node implementation, rather than an element by element approach, so no coloring is necessary

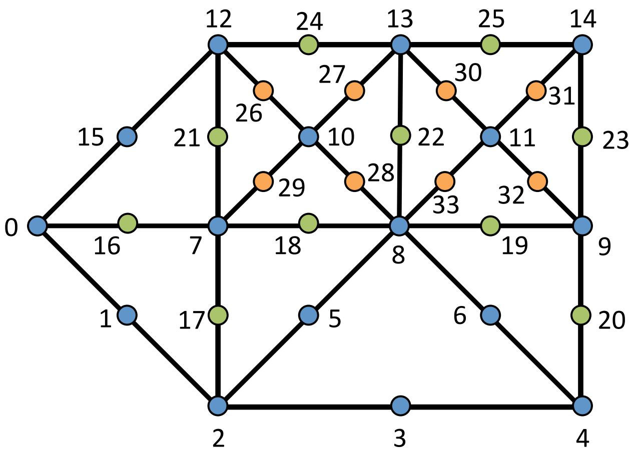


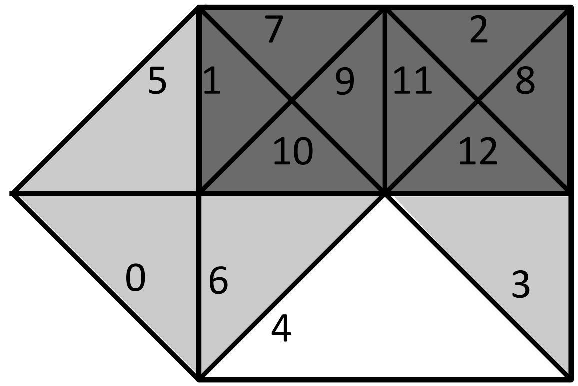


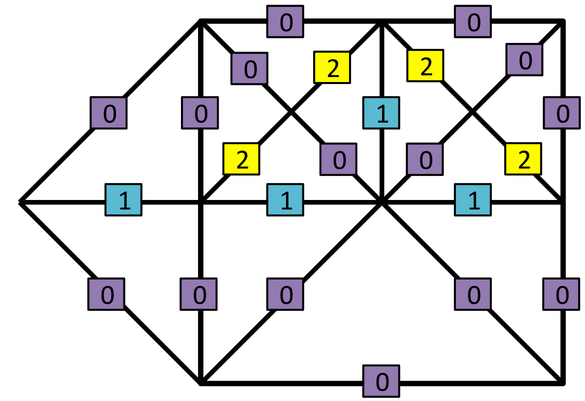
Launch one thread per node and gather contributions from each of its adjacent elements

- Also requires changes to the data structure in order to account for the changing number of bulk elements

A node and element table contains the necessary information for AMR+C on the structured mesh







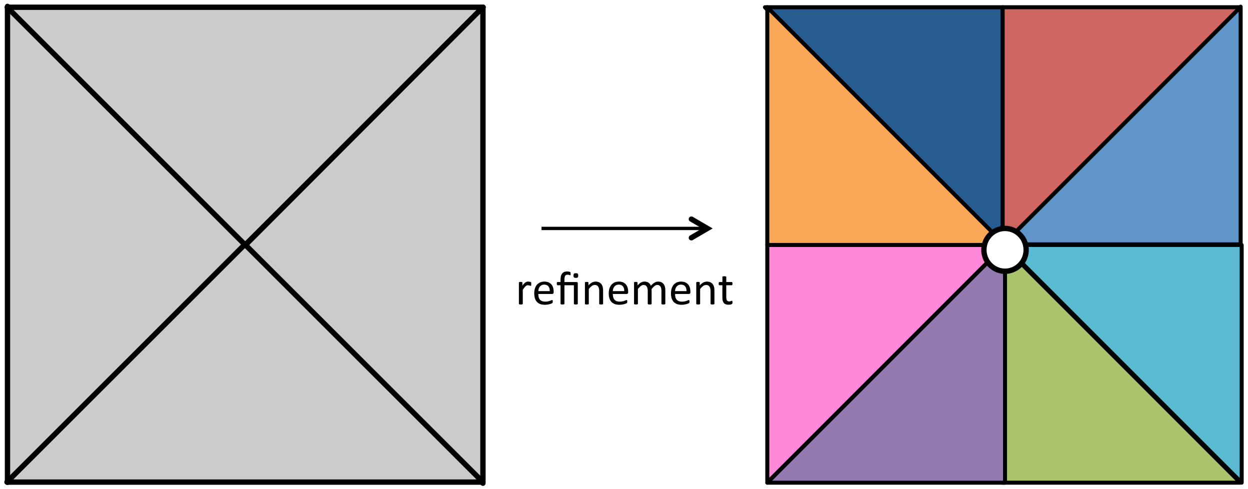
Node table

Id	x	y	Adj Elem
0	x	y	0
1	x	y	0
2	x	y	4
…			
16	x	y	0
17	x	y	1
…			
32	x	y	8
33	x	y	12

Element table

Id	v0	v1	v2	v3	v4	v5	O_0	O_1	O_2	Level	Ref	Labels
0	0	2	7	1	17	16	6	5	-1	1	0	0-0-1
1	12	7	10	21	29	26	10	7	5	2	1	0-2-0
..												
3	8	4	9	6	20	19	-1	12	4	1	3	0-0-1
4	2	4	8	3	6	5	3	6	-1	0	4	0-0-0
..												
11	13	8	11	22	33	30	12	2	9	2	2	1-0-2
12	8	9	11	19	32	33	8	11	2	2	2	1-2-0

The AMR+C GPU implementation makes the variation of the numerical solution evident



The order in which the new elements are inserted are random. So, the way their contributions are added to the white node will be different from one simulation to the next

$$u_{node} = u_{e1}+u_{e2}+u_{e3}+u_{e4}+u_{e5}+u_{e6}+u_{e7}+u_{e8}$$

or

$$u_{node} = u_{e1}+u_{e2}+u_{e3}+u_{e4}+u_{e5}+u_{e6}+u_{e7}+u_{e8}$$

or

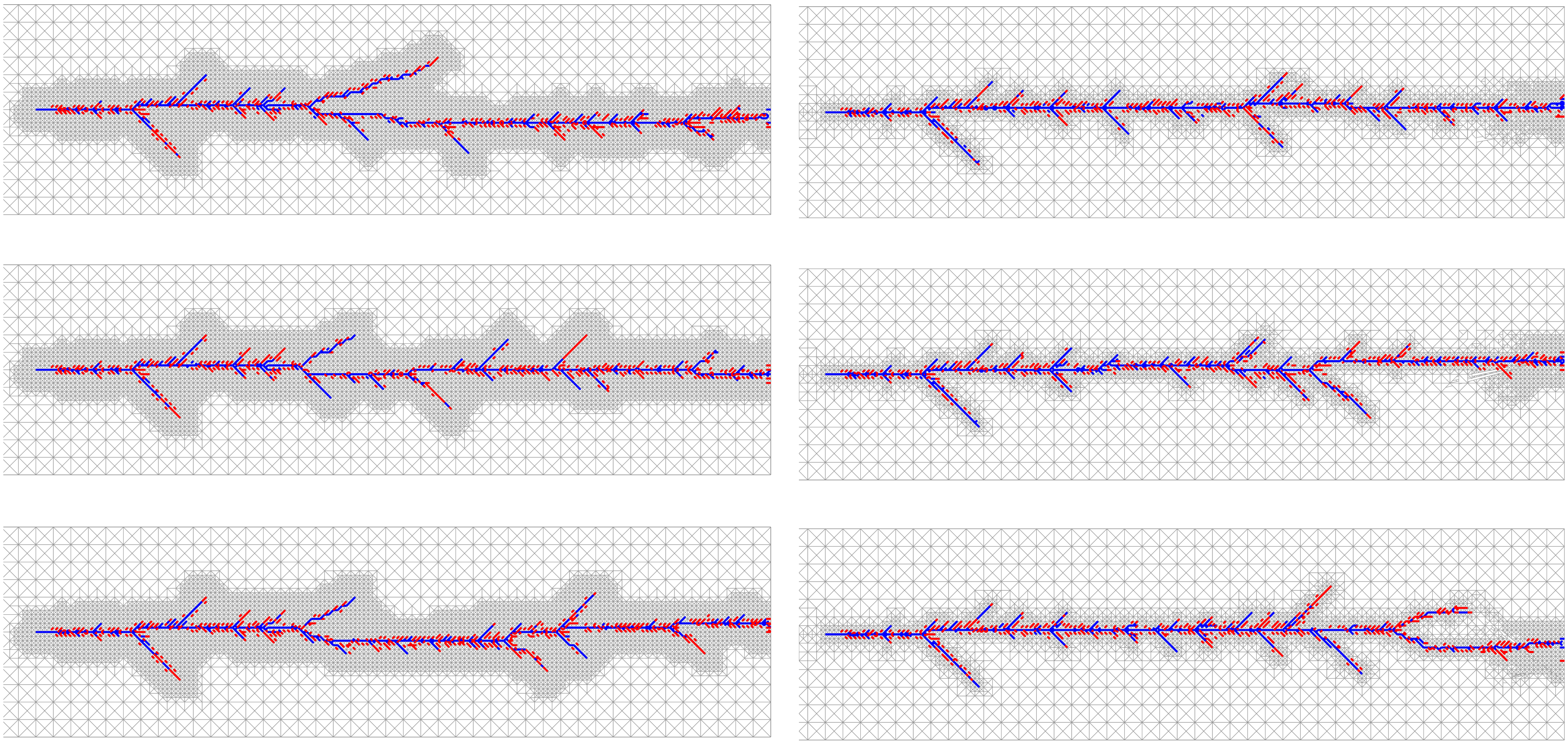
$$u_{node} = u_{e1}+u_{e2}+u_{e3}+u_{e4}+u_{e5}+u_{e6}+u_{e7}+u_{e8}$$

or …

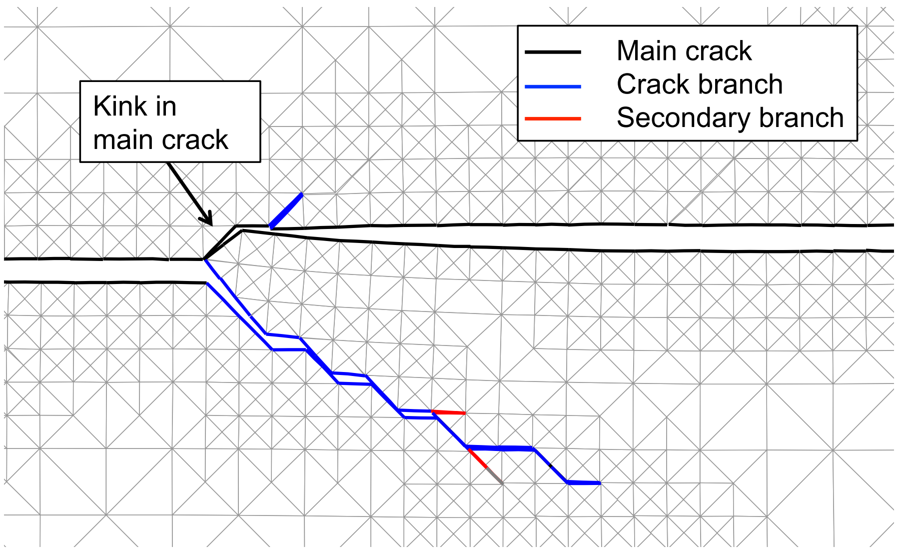
A+B≠B+A

Adaptive results on GPU appear quite different from one simulation to another

AMR | AMR+C



Study of physically relevant properties reveals minimal influence from randomness



Post processing algorithm traces the final crack path of open elements and identifies branches

A cohesive element is considered open if the separation between nodes is greater than a user-defined threshold

Variation for AMR and AMR+C with both thresholds

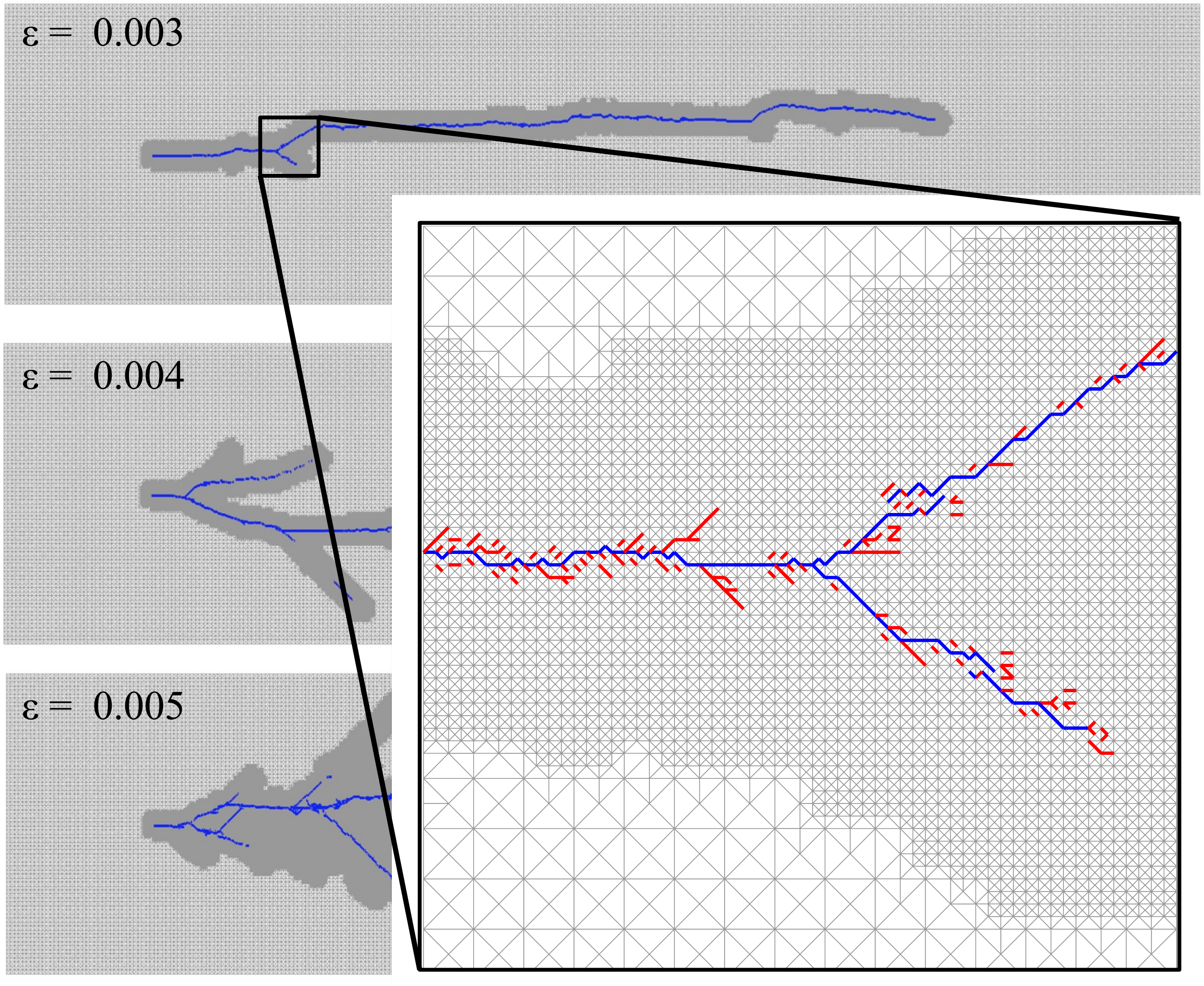
Total crack length (energy released): 1.5% - 4.5%

Number of branches: 20% - 100%

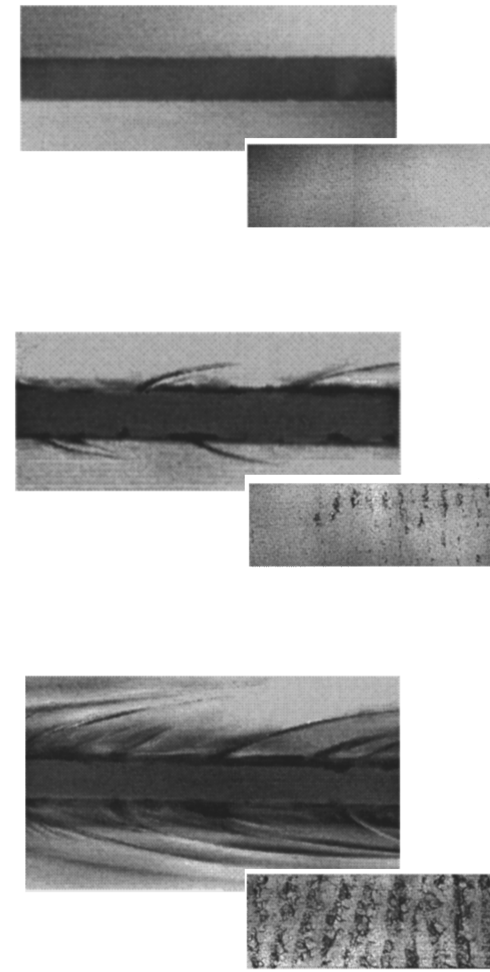
Length of branches: 80% - 130%

Leon, S. E*., Alhadeff, A.*, W. Celes, and G. H. Paulino. "Massively parallel adaptive mesh refinement and coarsening for dynamic fracture simulations ." In preparation.

Adaptivity on GPU makes the larger scale version of the problem computationally tenable



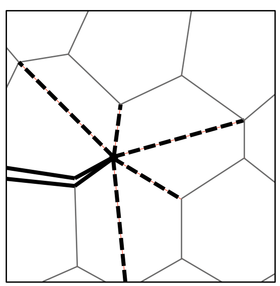
412,565 Bulk Elements
17,565 Cohesive Elements
856,473 Nodes

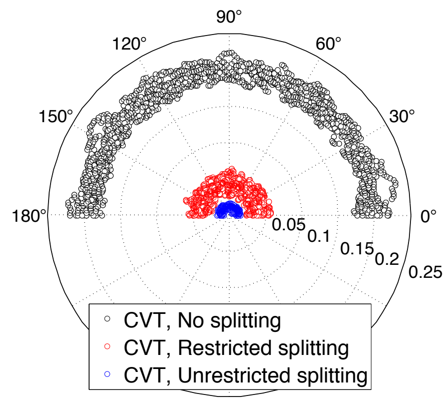


Sharon, E., and J. Fineberg. *Physical Review B* 54, no. 10 (1996): 7128–39.

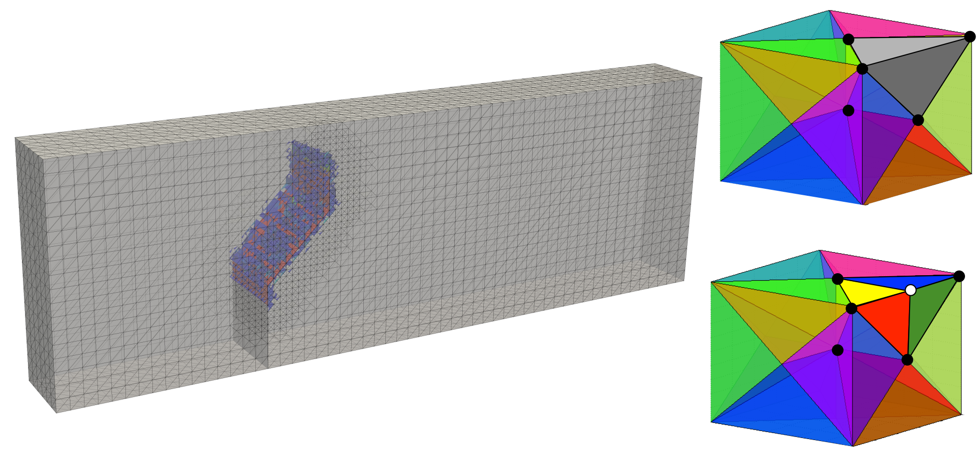
Summary of PhD research

Dynamic fracture simulation using polygonal elements with adaptive element splitting

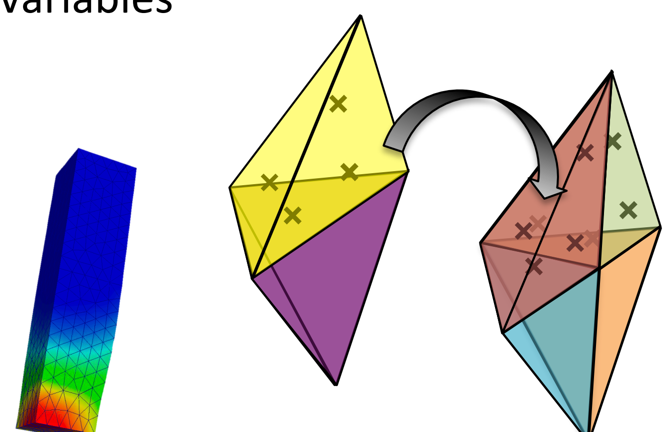




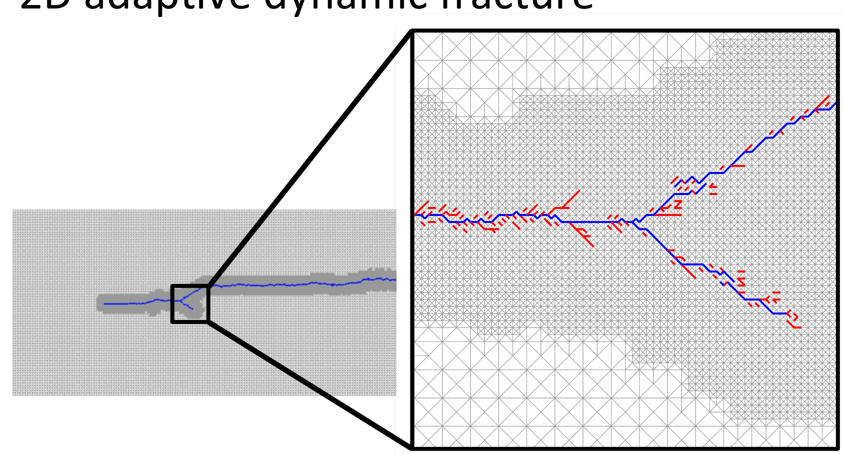
Adaptive mesh refinement and coarsening on 3D 4K meshes



Investigation of mapping internal state variables



Study of the effect of GPU implementation on 2D adaptive dynamic fracture



Contributions

Addressed in this presentation:

1. **S. E. Leon***, D. W. Spring*, and G. H. Paulino. “Reduction in mesh bias for dynamic fracture using adaptive splitting of polygonal finite elements.” *International Journal for Numerical Methods in Engineering*, 100(8): 555–76, 2014.

2. **S. E. Leon**, R. Espinha, W. Celes, W. and G. H. Paulino. “Adaptive refinement and coarsening on structured 3D meshes” In preparation.

3. **S. E. Leon***, A. Alhadeff*, W. Celes, and G. H. Paulino. “Massively parallel adaptive mesh refinement and coarsening for dynamic fracture simulations .” In preparation.

4. J. M. Emery, J. Foulk, **S. E. Leon**, A. Mota, J. Ostein, W. C. Sun, M. Veilleux. “Mapping internal state variables for large deformation simulation in preparation.” In preparation.

Not covered in this presentation:

5. D. W. Spring, **S. E. Leon**, and G. H. Paulino. “Unstructured Polygonal Meshes with Adaptive Refinement for the Numerical Simulation of Dynamic Cohesive Fracture.” *International Journal of Fracture*, 2014.

6. **S. E. Leon**, E. N. Lages, C. N. Araújo, and G. H. Paulino. “On the effect of constraint parameters on the generalized displacement control method.” *Mechanics Research Communications* 56 (March 1, 2014): 123–29.

7. **S. E. Leon**, G. H. Paulino, A. Pereira, I. F. M. Menezes, and E. N. Lages. “A Unified Library of Nonlinear Solution Schemes.” *Applied Mechanics Reviews* 64, no. 4 (2011): 040803.

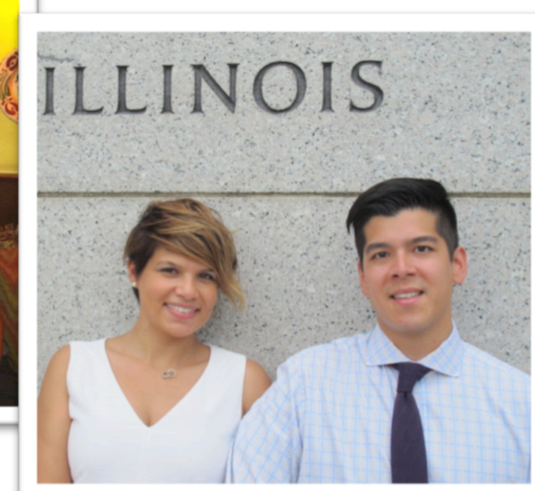
8. E. V. Dave, **S. E. Leon**, and K. Park. “Thermal Cracking Prediction Model and Software for Asphalt Pavements.” *T&DI Congress 2011 Integrated Transportation and Development for a Better Tomorrow*, 2011, 667–76.

9. E. V. Dave, W. G. Buttlar, **S. E. Leon**, B. Behnia, and G. H. Paulino. “IlliTC – Low-Temperature Cracking Model for Asphalt Pavements.” *Road Materials and Pavement Design* 14, no. 2 (2013): 57–78.

Mentors, colleagues, and friends: Thank you for your support!

Glaucio H. Paulino, Waldemar Celes, Ahmed Elbanna, Jay Foulk, Iwona Jasiuk, Petros Sofronis

Rodrigo, Andrei, Anderson, Nobre, Ivan, Adeildo, Viviane
Cam, Kyoungsoo, Eshan, Daniel, Marco, Luis, Lauren, Daiane, Heng, Junho, Will, Emily,
Maryam, Evgueni, Arun, Ludimar, Chris, Tam, Ying, Tomas, Shelly, Tuo
Evan, Ashley, Danny, Zach, Kelley, Gabbie
Sam, Liz, Danielle, Ritu, Jin, Angeli, Neera, Ashley, Marybeth, Meredith, Jackie, Jamie
Wendy, Fritz, Jake, Carly, and Wylie



Thank you for your attention

Questions?