Accelerating Implicit Contact via Matrix-Free Operators on Non-Conforming Interfaces

41st Colorado Conference on Iterative and Multigrid Methods

Zachary R. Atkins Collaborators: Jed Brown and Jeremy L. Thompson

24 June 2026

University of Colorado Boulder
Department of Computer Science

Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Outline

1. General Matrix-Free FEM

2. Numerical Contact Evaluation

   Mortar Method for Segmentation-Based Contact

   libCEED At-Points Operators

3. Numerical Results

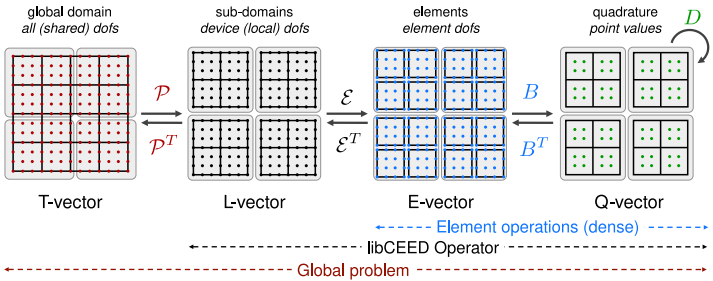
Acknowledgments

This work was supported by the Department of Energy, National Nuclear Security Administration, Predictive Science Academic Alliance Program (PSAAP III) under Award Number DE-NA0003962, and by the Department of Energy Frameworks, Algorithms, and Scalable Technologies for Mathematics (FASTMath) SciDAC Institute and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under contract DE-AC02-05CH11231 and Award Number DE-SC0016140.

General Matrix-Free FEM

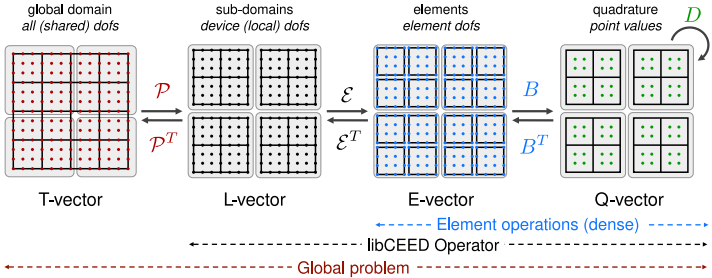
libCEED Matrix-Free Formulation

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



libCEED Matrix-Free Formulation

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$

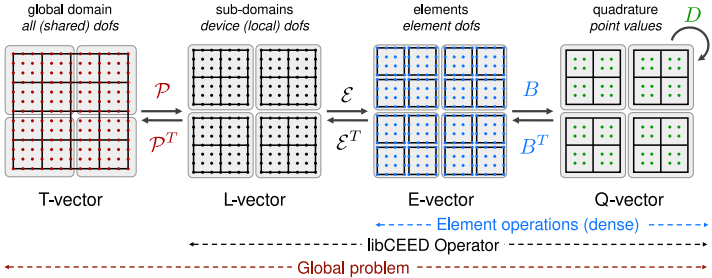


Partitioning

The operator $\mathcal{P}$ represents mapping from the parallel vector to the local vector, including inserting ghost values — this is done using PETSc's `DMPlex`

libCEED Matrix-Free Formulation

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$

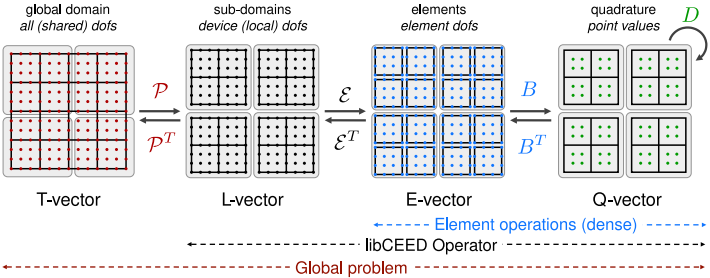


Element Restriction

The operator $\mathcal{E}$ maps local degrees-of-freedom to the finite element vector via duplicating values at nodes shared between elements

libCEED Matrix-Free Formulation

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$

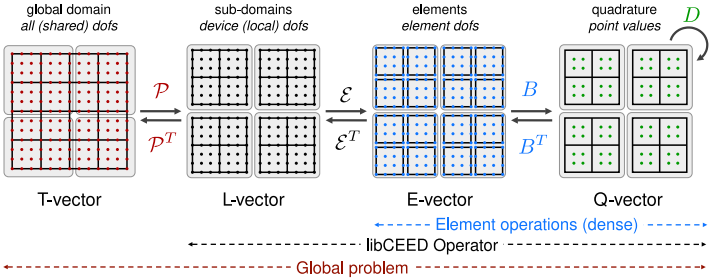


Basis

The operator B represents interpolation and/or gradient from nodes to quadrature points

libCEED Matrix-Free Formulation

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



Quadrature Function

The operator D represents the action of the PDE & its linearization at individual quadrature points — this is where all of the physics is defined

Generic Weak Form

Find $\boldsymbol{u} \in \mathcal{V} \subset H^1(\Omega)$ such that for all $\boldsymbol{v} \in \mathcal{V}$,

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle = \int_\Omega \left[\boldsymbol{v} \cdot f_0\left(\boldsymbol{u}, \nabla \boldsymbol{u}\right) + \nabla \boldsymbol{v} : f_1\left(\boldsymbol{u}, \nabla \boldsymbol{u}\right) \right] \mathrm{d}V + \int_{\partial\Omega} \boldsymbol{v} \cdot g_0\left(\boldsymbol{u}, \nabla \boldsymbol{u}\right) \mathrm{d}A = 0,$$

Generic Weak Form

Find $\boldsymbol{u} \in \mathcal{V} \subset H^1(\Omega)$ such that for all $\boldsymbol{v} \in \mathcal{V}$,

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle = \int_\Omega \left[\boldsymbol{v} \cdot f_0\left(\boldsymbol{u}, \nabla \boldsymbol{u}\right) + \nabla \boldsymbol{v} : f_1\left(\boldsymbol{u}, \nabla \boldsymbol{u}\right) \right] \mathrm{d}V + \int_{\partial\Omega} \boldsymbol{v} \cdot g_0\left(\boldsymbol{u}, \nabla \boldsymbol{u}\right) \mathrm{d}A = 0,$$

Tensor Product Assumptions

We will assume that $\mathcal{V}$ is a sum-factorizable finite element space defined over tensor product elements.

Numerical Contact Evaluation

Model Problem

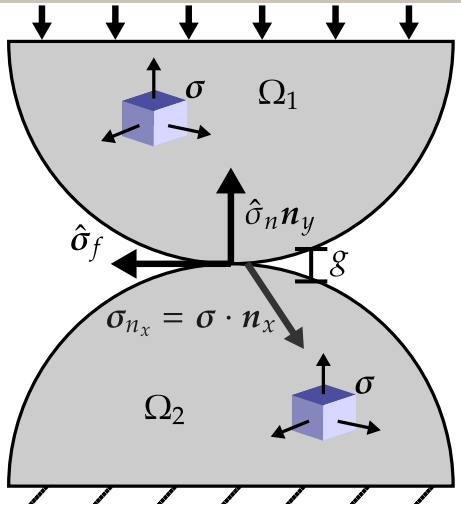
Consider two elastic bodies Ω_1, Ω_2.
In the following,

$$f_0(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{0},\ f_1(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{P}(\nabla \boldsymbol{u})$$
$$g_0(\boldsymbol{u}, \nabla \boldsymbol{u}) = \lambda_n(u) \text{ (contact forces)}$$
$$= -\left[\epsilon g(\boldsymbol{u})\right]_{\mathbb{R}^-} \boldsymbol{n}_x$$

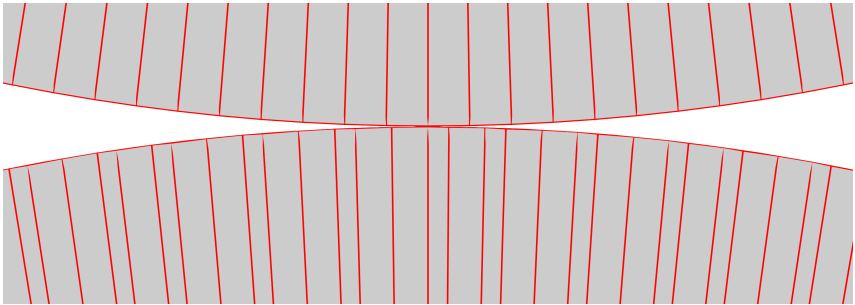
In general, KKT conditions:

$$g(u) \geq 0, \quad \lambda_n(u) \geq 0, \quad g(u) \cdot \lambda_n(u) = 0.$$

Need to integrate gap function $g(\boldsymbol{u})$ over common contact surface

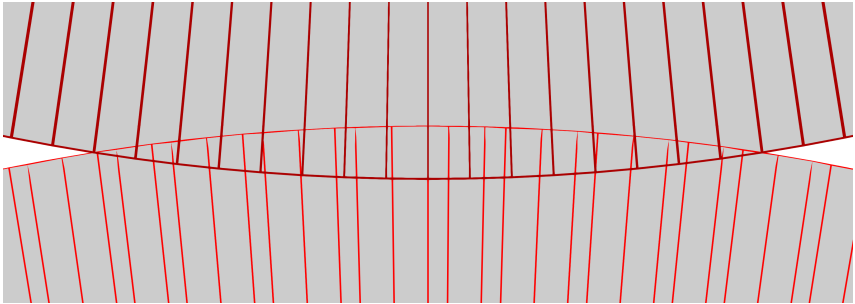


Discretized Model Problem



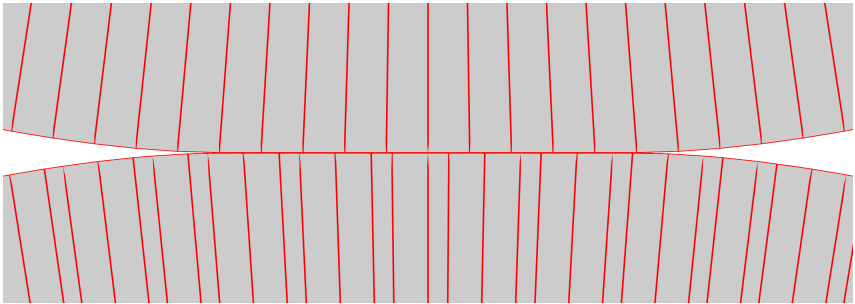
In reality, bodies are represented as discrete meshes, usually with non-conforming/non-matching contact interfaces

Discretized Model Problem



When penetration occurs, it does so with arbitrary overlap between the face elements on either side

Discretized Model Problem

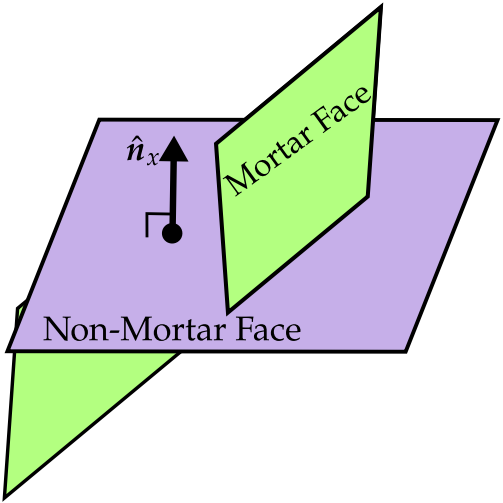


Ultimately, need to find displacements $\boldsymbol{u}^h$ such that the discrete gap $g(\boldsymbol{u}^h) \rightarrow 0$ in a weak sense along the contact interface

Numerical Contact Evaluation

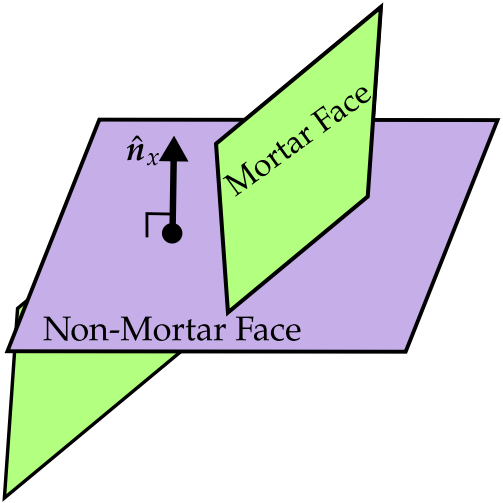
Mortar Method for Segmentation-Based Contact

Simple Case: Two Linear Quadrilateral Facets



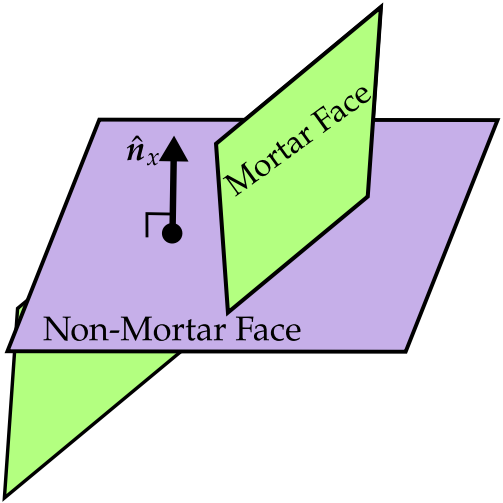
- Clearly, these facets overlap, but they are not in the same plane

Simple Case: Two Linear Quadrilateral Facets



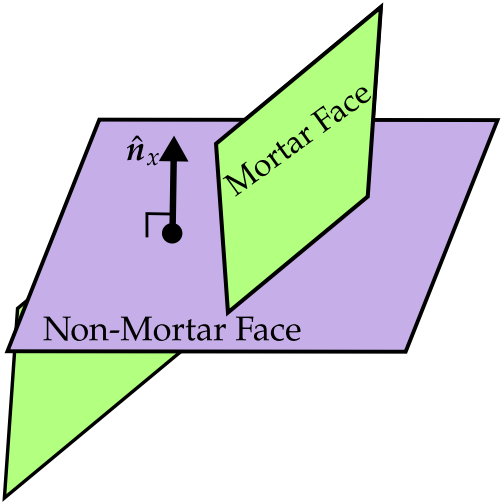
- Clearly, these facets overlap, but they are not in the same plane
  - Formally, their intersection is a segment

Simple Case: Two Linear Quadrilateral Facets



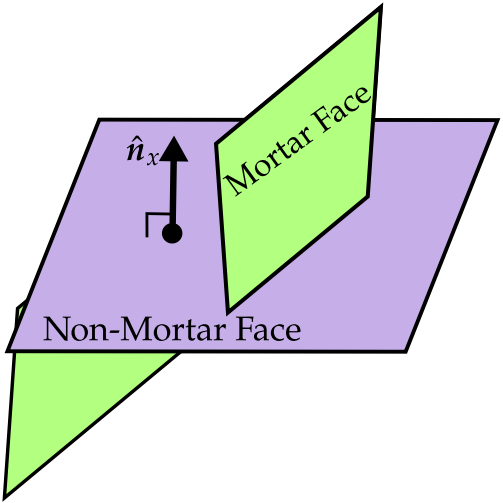
- Clearly, these facets overlap, but they are not in the same plane
  - Formally, their intersection is a segment
- Goal: define some non-zero area where the facets are in contact

Mortar Method for Segmentation-Based Contact



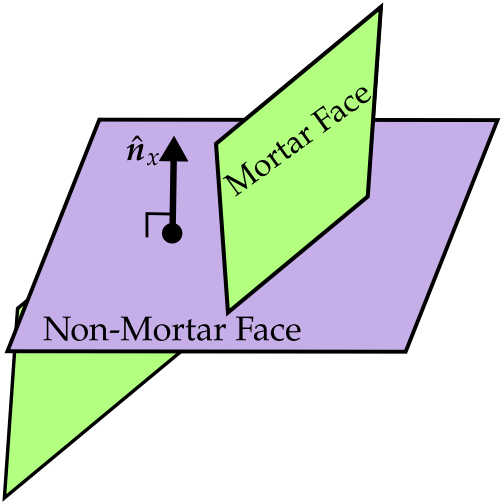
- Consider a well established approach from Puso and Laursen 2004

Mortar Method for Segmentation-Based Contact



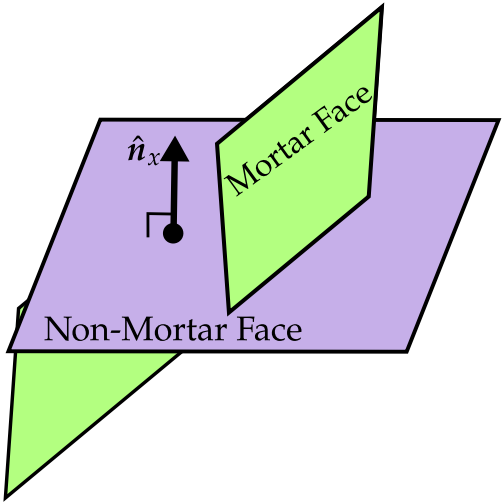
- Consider a well established approach from Puso and Laursen 2004
- Label one face as the ”non-mortar” face, where integration will be well defined

Mortar Method for Segmentation-Based Contact



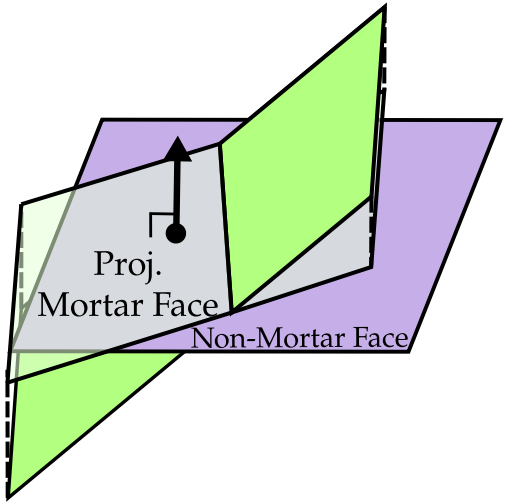
- Consider a well established approach from Puso and Laursen 2004
- Label one face as the ”non-mortar” face, where integration will be well defined
- Label the other face as the ”mortar” face, which will experience some aliasing during integration

Mortar Method for Segmentation-Based Contact



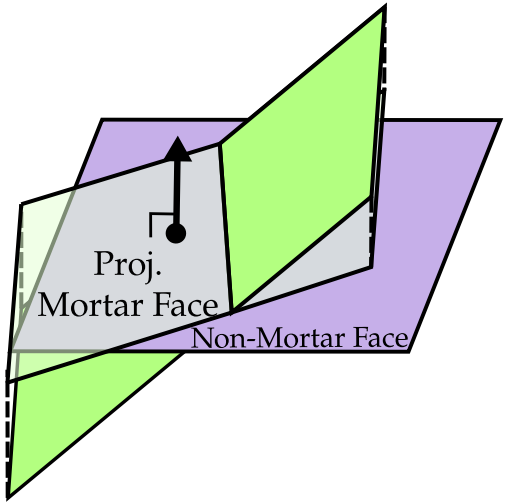
- Consider a well established approach from Puso and Laursen 2004
- Label one face as the ”non-mortar” face, where integration will be well defined
- Label the other face as the ”mortar” face, which will experience some aliasing during integration
- We briefly outline the method (performed once per face pair per residual evaluation)

Project Mortar to Non-Mortar



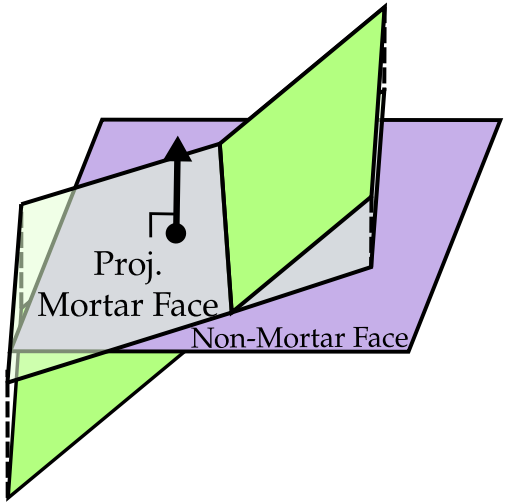
- Project the vertices of the other face onto the common plane of the non-mortar face

Project Mortar to Non-Mortar



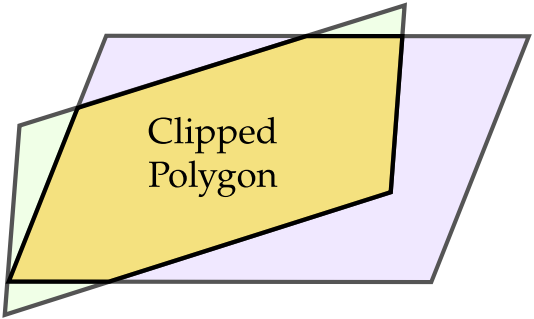
- Project the vertices of the other face onto the common plane of the non-mortar face
  - If the non-mortar face vertices are not coplanar, use the midpoint and average face normal

Project Mortar to Non-Mortar



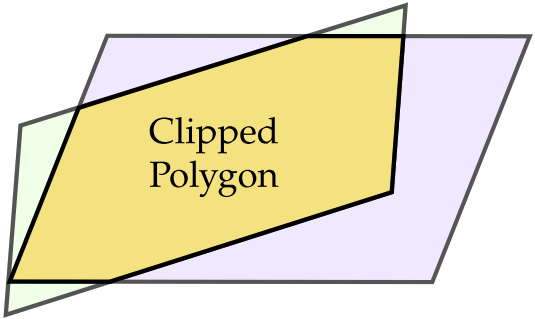
- Project the vertices of the other face onto the common plane of the non-mortar face
  - If the non-mortar face vertices are not coplanar, use the midpoint and average face normal
- The mortar face will experience some aliasing during integration due to the projection

Clip the 2D Polygons



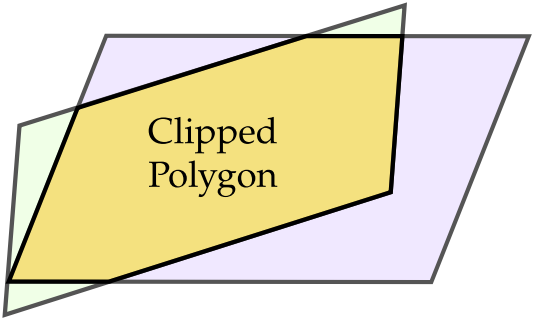
- Coplanar facets $\implies$ can directly find their intersection

Clip the 2D Polygons



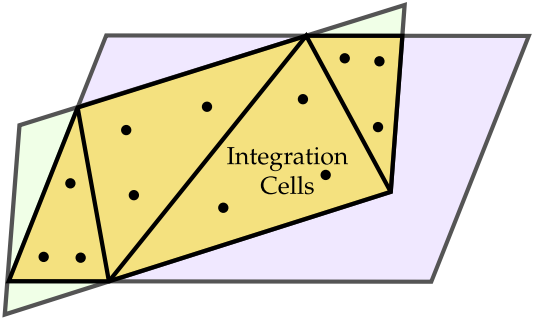
- Coplanar facets $\implies$ can directly find their intersection
- Very difficult to do robustly, especially for non-convex facets

Clip the 2D Polygons



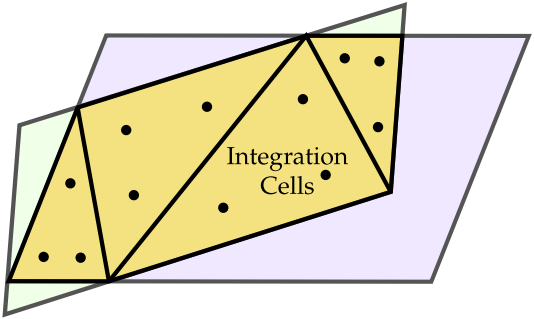
- Coplanar facets $\implies$ can directly find their intersection
- Very difficult to do robustly, especially for non-convex facets
- We use an algorithm by Greiner and Hormann 1998 with robustness improvements by Foster, Hormann, and Popa 2019

Triangulate the Clipped Polygon



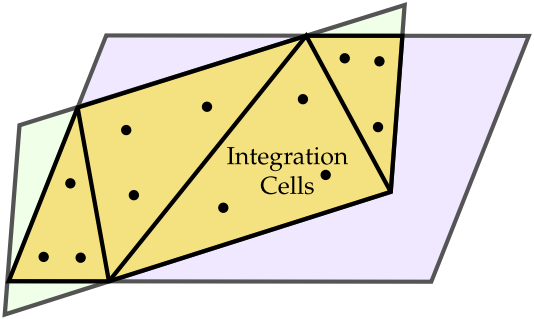
- Partition the clipped element into triangles

Triangulate the Clipped Polygon



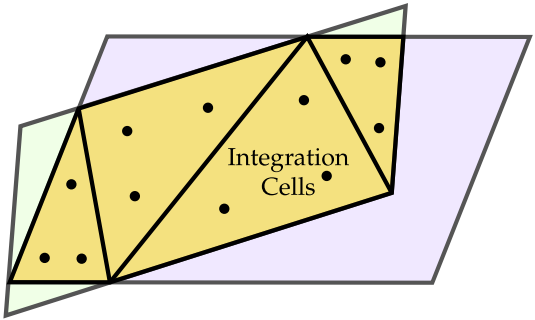
- Partition the clipped element into triangles
  - We use a deterministic linear-time ear cut method by Livesu et al. 2022

Triangulate the Clipped Polygon



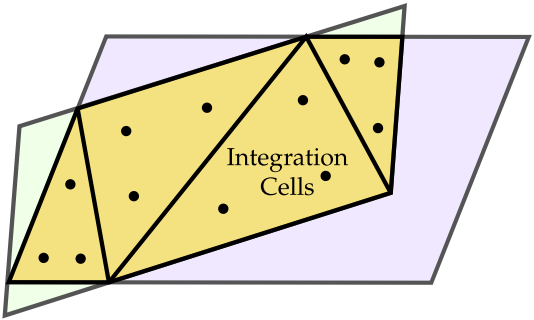
- Partition the clipped element into triangles
  - We use a deterministic linear-time ear cut method by Livesu et al. 2022
- Assign quadrature points to the triangles using your preferred method

Triangulate the Clipped Polygon



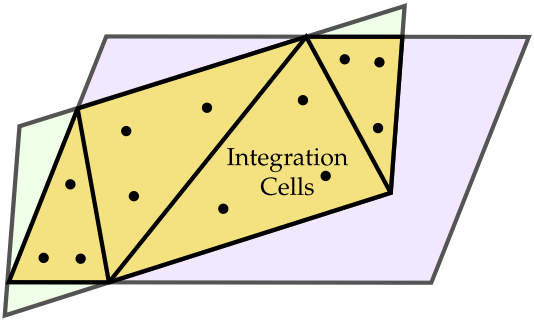
- Partition the clipped element into triangles
  - We use a deterministic linear-time ear cut method by Livesu et al. 2022
- Assign quadrature points to the triangles using your preferred method
  - We use symmetric quadrature rules by Jaskowiec and Sukumar 2021

Triangulate the Clipped Polygon



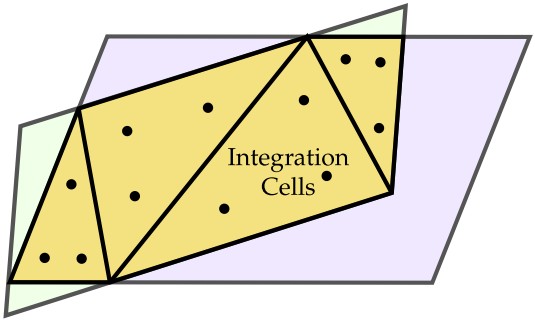
- Partition the clipped element into triangles
  - We use a deterministic linear-time ear cut method by Livesu et al. 2022
- Assign quadrature points to the triangles using your preferred method
  - We use symmetric quadrature rules by Jaskowiec and Sukumar 2021
- Assign integration weights $w_i \det J$, where

Triangulate the Clipped Polygon



- Partition the clipped element into triangles
  - We use a deterministic linear-time ear cut method by Livesu et al. 2022
- Assign quadrature points to the triangles using your preferred method
  - We use symmetric quadrature rules by Jaskowiec and Sukumar 2021
- Assign integration weights $w_i \det J$, where
  - w_i is the quadrature weight

Triangulate the Clipped Polygon

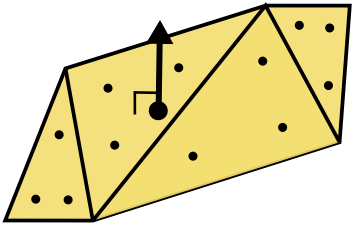


- Partition the clipped element into triangles
  - We use a deterministic linear-time ear cut method by Livesu et al. 2022
- Assign quadrature points to the triangles using your preferred method
  - We use symmetric quadrature rules by Jaskowiec and Sukumar 2021
- Assign integration weights $w_i \det J$, where
  - w_i is the quadrature weight
  - $\det J = \frac{\partial \boldsymbol{x}}{\partial \xi_1} \times \frac{\partial \boldsymbol{x}}{\partial \xi_2}$ is the Jacobian of the reference element mapping

Mapping Back to the Mortar Surface

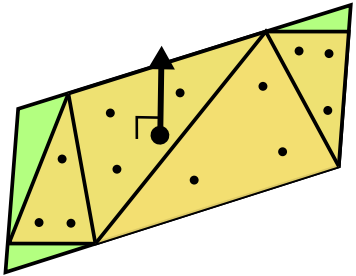
- Reverse the projection for point p:

Mapping Back to the Mortar Surface



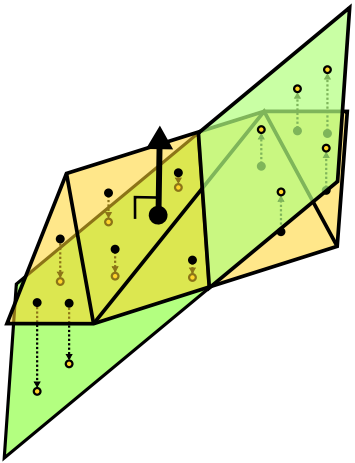
- Reverse the projection for point p:
  1. Ref-to-real: interpolate $\boldsymbol{\xi}_p^{\text{tri}} \mapsto \boldsymbol{x}_p^{\text{proj}}$ with triangle geometry

Mapping Back to the Mortar Surface



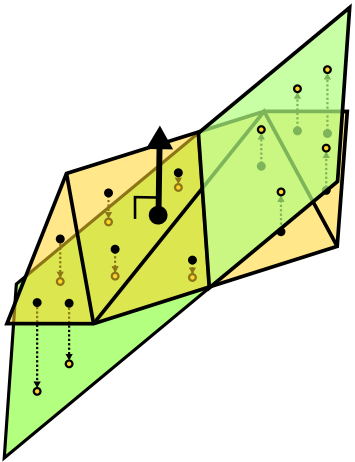
- Reverse the projection for point p:
  1. Ref-to-real: interpolate $\boldsymbol{\xi}_p^{\text{tri}} \mapsto \boldsymbol{x}_p^{\text{proj}}$ with triangle geometry
  2. Real-to-ref: $\boldsymbol{x}_p^{\text{proj}} \mapsto \boldsymbol{\xi}_p^{\text{mortar}}$ via local Newton solve with **projected** mortar nodal coordinates and Jacobian

Mapping Back to the Mortar Surface



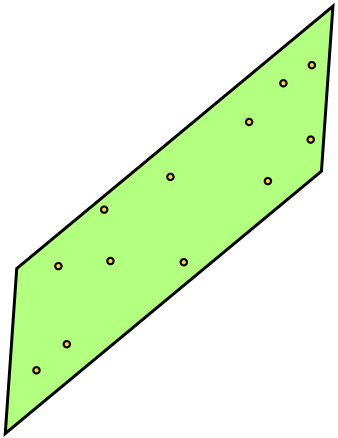
- Reverse the projection for point p:
  1. Ref-to-real: interpolate $\boldsymbol{\xi}_p^{\text{tri}} \mapsto \boldsymbol{x}_p^{\text{proj}}$ with triangle geometry
  2. Real-to-ref: $\boldsymbol{x}_p^{\text{proj}} \mapsto \boldsymbol{\xi}_p^{\text{mortar}}$ via local Newton solve with **projected** mortar nodal coordinates and Jacobian
  3. Ref-to-real: interpolate $\boldsymbol{\xi}_p^{\text{mortar}} \mapsto \boldsymbol{x}_p^{\text{real, mortar}}$ with the **true** mortar facet geometry

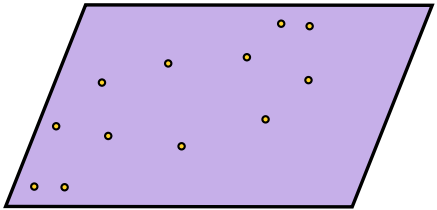
Mapping Back to the Mortar Surface



- Reverse the projection for point p:
  1. Ref-to-real: interpolate $\boldsymbol{\xi}_p^{\text{tri}} \mapsto \boldsymbol{x}_p^{\text{proj}}$ with triangle geometry
  2. Real-to-ref: $\boldsymbol{x}_p^{\text{proj}} \mapsto \boldsymbol{\xi}_p^{\text{mortar}}$ via local Newton solve with **projected** mortar nodal coordinates and Jacobian
  3. Ref-to-real: interpolate $\boldsymbol{\xi}_p^{\text{mortar}} \mapsto \boldsymbol{x}_p^{\text{real, mortar}}$ with the **true** mortar facet geometry
- That is, the reference coordinates for each point with respect to the mortar face are computed in the projected space, then remain constant

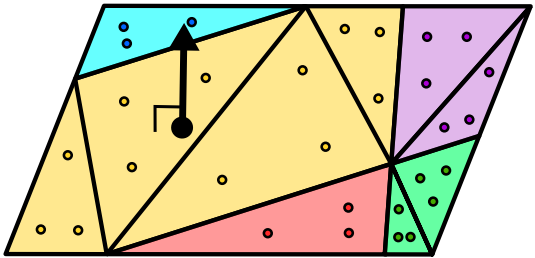
Resulting Points on Each Side





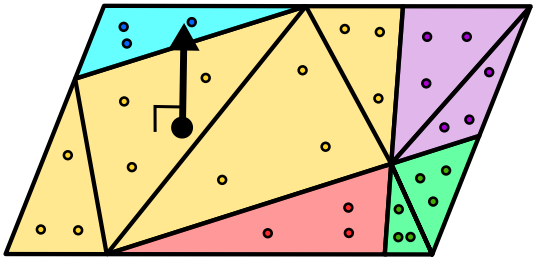
Each side has the same points, just with different coordinates!

Computing the Integral Over the Surface



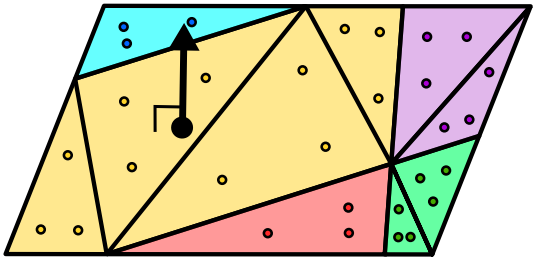
- Repeat this process with each overlapping mortar facet (labeled by color)

Computing the Integral Over the Surface



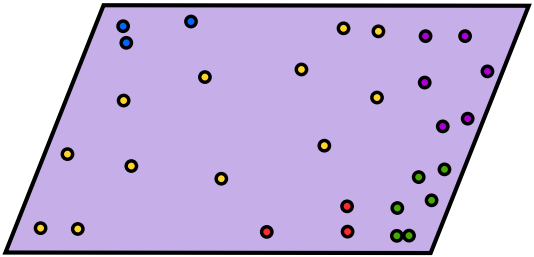
- Repeat this process with each overlapping mortar facet (labeled by color)
- Each non-mortar facet is decomposed into blocks of triangular elements, each corresponding to a unique mortar facet

Computing the Integral Over the Surface



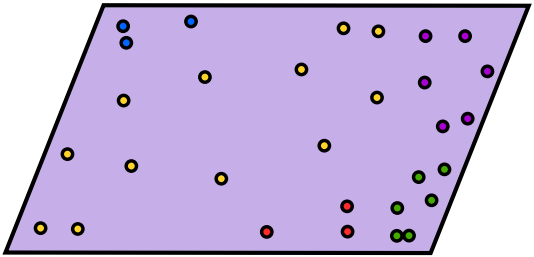
- Repeat this process with each overlapping mortar facet (labeled by color)
- Each non-mortar facet is decomposed into blocks of triangular elements, each corresponding to a unique mortar facet
- Traditional methods would form contact mass matrices (dense in general) from these triangles

Computing the Integral Over the Surface



- Instead, we forget about the triangles and only store the points

Computing the Integral Over the Surface



- Instead, we forget about the triangles and only store the points
- Integrals can be decomposed as:

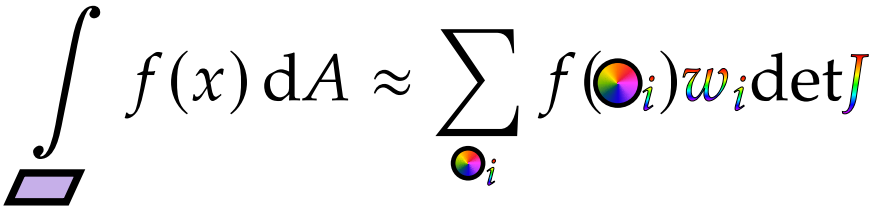
$$\int f(x)\,\mathrm{d}A \approx \sum_{\bullet_i} f(\bullet_i) w_i \det J + \sum_{\bullet_i} f(\bullet_i) w_i \det J + \sum_{\bullet_i} f(\bullet_i) w_i \det J + \sum_{\bullet_i} f(\bullet_i) w_i \det J + \sum_{\bullet_i} f(\bullet_i) w_i \det J$$

Numerical Contact Evaluation

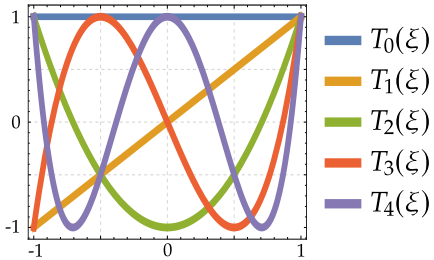
libCEED At-Points Operators

Efficient Evaluation at Arbitrary Quadrature Points

How can we actually compute this efficiently?



Efficient Evaluation at Arbitrary Quadrature Points[1]



$$T_0(\xi) = 1$$
$$T_1(\xi) = \xi$$
$$T_n(\xi) = 2\xi T_{n-1}(\xi) - T_{n-2}(\xi)$$

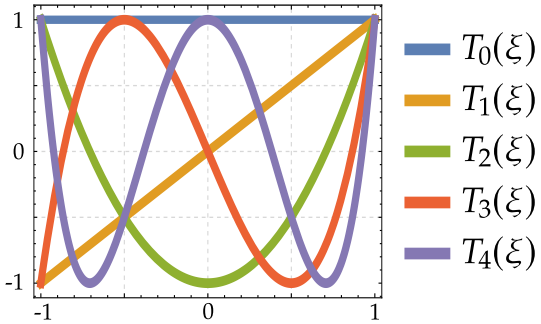
Core Idea Map function coefficients into a stable evaluation basis (Chebyshev) and contract against basis functions evaluated at point coordinates. In 1D:

$$u_i = \sum_{j=0}^{Q} (u_c^e)_j T_j(\bullet_i)$$

where $u_c^e = CN\mathcal{E}^e u$, N is the standard FEM interpolation matrix, and C maps function values at Gauss points to Chebyshev coefficients

[1] Zachary R. Atkins et al. (May 2026). **"Matrix-free finite element methods with arbitrary quadrature point locations".** en. In: *Computational Science and Engineering* 3.1, p. 3. ISSN: 2948-1597. DOI: 10.1007/s44207-026-00010-1.

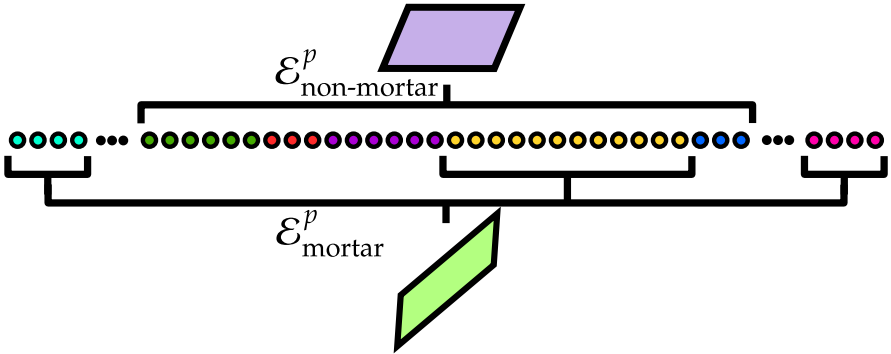
Efficient Evaluation at Arbitrary Quadrature Points



$$T_0(\xi) = 1$$
$$T_1(\xi) = \xi$$
$$T_n(\xi) = 2\xi T_{n-1}(\xi) - T_{n-2}(\xi)$$

- Higher dimension bases are applied efficiently via tensor products of CN
- Efficient Chebyshev polynomial evaluation means that for bases of order $p-1$ and a maximum of N_p points per facet, total cost is $O(p^2 \cdot N_p)$ (standard 3D FEM is $O(p^4)$)

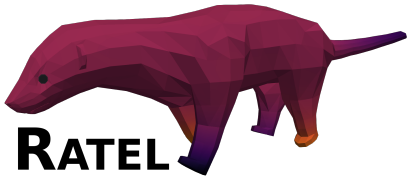
Segmented Point Restrictions



Index the same vector of values at quadrature points, e.g. gaps or contact force, with two different restrictions — mapping between sides is simply changing the element restriction

Numerical Results

Software and Hardware

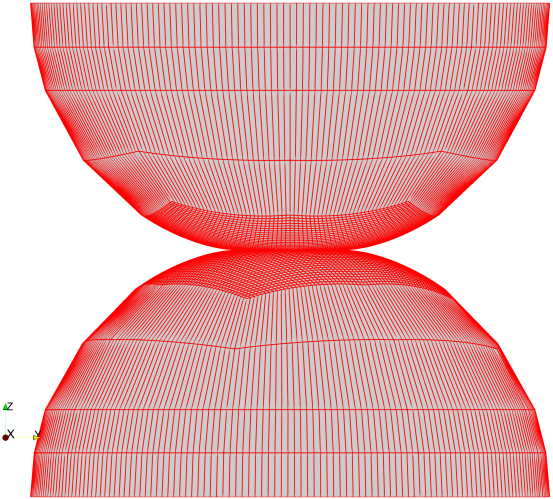


libCEED

PETSc

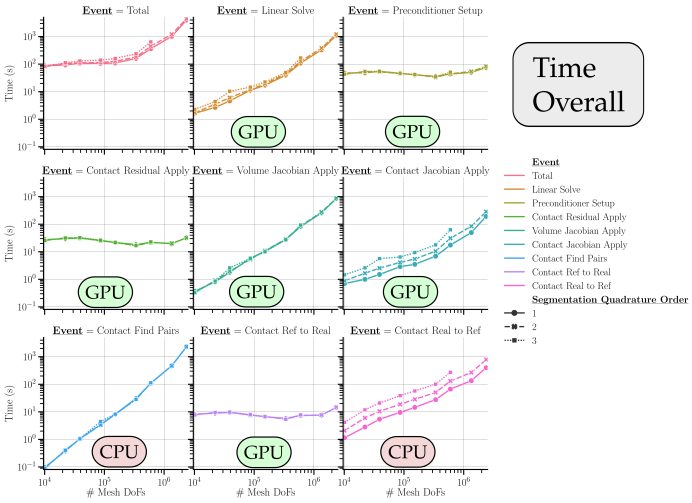
- Implemented in Ratel, our matrix-free solid mechanics code built on libCEED and PETSc
- GPU: ROCm/HIP 7.1.2 on AMD Radeon VII (*gfx906:sramecc+:xnack-*, MI50) with 16 GB HBM2 memory (CUDA also supported)
- CPU: AMD Ryzen 7 3700X, 64 GB DDR4 RAM
- Solvers:
  - libCEED */gpu/hip/gen* backend
  - Newton's method with backtracking line search
  - BiCGStab(2) linear solver with Jacobi preconditioning

Test Problem



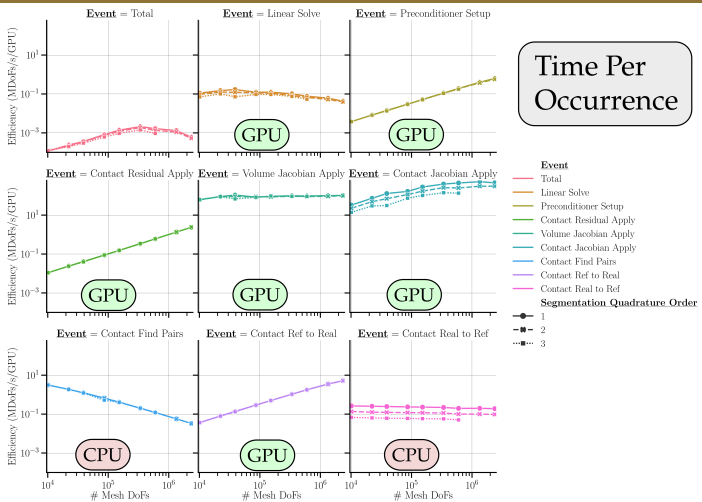
- Two half-spheres of radius 1 in contact, displacement of 0.1 on top surface
- Meshes are rotated $30°$ to increase clipping difficulty
- Only the area near contact surface is refined
  - Total DoFs between 1×10^4 and 2.4×10^6
  - Total contact points between 2×10^3 and 1.1×10^6

Wall Time



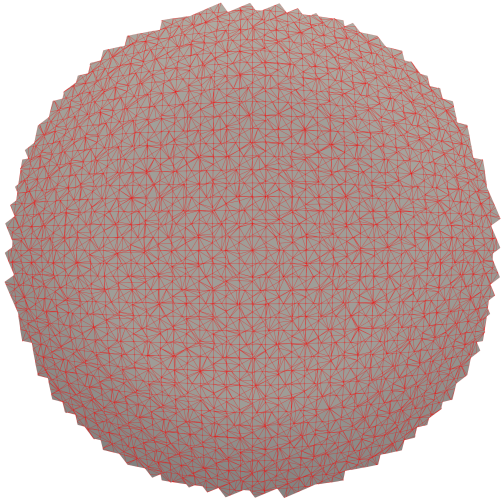
- All log-log, same scale
- Overall wall time — slightly misleading due to more linear solver iterations at larger problem sizes (Jacobi preconditioning)
- Excluding CPU bound kernels, highest costs are diagonal assembly and (extremely efficient, see Brown et al. 2022) volume Jacobian application

Efficiency



- Higher is better
- All log-log, same scale
- Positive slope means larger problems use less time relative to problem size ($t \in \omega(\#\text{DoFs})$)
- Negative slope means time grows super-linear with DoFs ($t \in o(\#\text{DoFs})$)
- Primary bottleneck is all-to-all contact pair search — not the focus of this talk

Conclusions



- libCEED at-points matrix-free operators provide excellent performance for evaluating contact constraints across non-conforming boundaries
  - Efficiency improves as problem sizes increase
- Future work:
  - Parallel/$O(n \log n)$ contact pair search
  - Lagrange multipliers for contact forces & field-split preconditioning
  - GPU offloading of contact pair evaluation and real-to-ref projections

References

- Atkins, Zachary R. et al. (May 2026). **"Matrix-free finite element methods with arbitrary quadrature point locations".** en. In: *Computational Science and Engineering* 3.1, p. 3. ISSN: 2948-1597. DOI: 10.1007/s44207-026-00010-1.
- Brown, Jed et al. (2022). ***Performance Portable Solid Mechanics via Matrix-Free p-Multigrid.*** arXiv: 2204.01722 [cs.MS]. URL: https://arxiv.org/abs/2204.01722.
- Foster, Erich L, Kai Hormann, and Romeo Traian Popa (Dec. 2019). **"Clipping simple polygons with degenerate intersections".** In: *Computers & Graphics: X* 2, p. 100007. ISSN: 2590-1486. DOI: 10.1016/j.cagx.2019.100007.
- Fries, Thomas-Peter and Ted Belytschko (Oct. 2010). **"The extended/generalized finite element method: An overview of the method and its applications".** en. In: *International Journal for Numerical Methods in Engineering* 84.3, pp. 253–304. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.2914.
- Greiner, Günther and Kai Hormann (Apr. 1998). **"Efficient clipping of arbitrary polygons".** In: *ACM Trans. Graph.* 17.2, pp. 71–83. ISSN: 0730-0301. DOI: 10.1145/274363.274364.

Jaskowiec, Jan and N. Sukumar (2021). **“High-order symmetric cubature rules for tetrahedra and pyramids”.** In: *International Journal for Numerical Methods in Engineering* 122.1, pp. 148–171. DOI: *10.1002/nme.6528*.

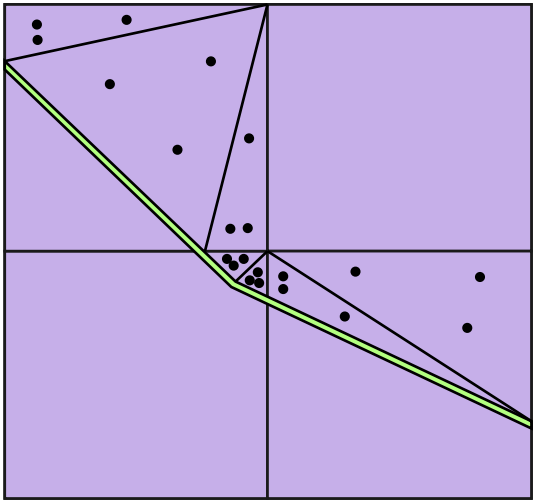
Livesu, Marco et al. (Dec. 2022). **“Deterministic Linear Time Constrained Triangulation Using Simplified Earcut”.** en. In: *IEEE Transactions on Visualization and Computer Graphics* 28.12, pp. 5172–5177. ISSN: 1077-2626, 1941-0506, 2160-9306. DOI: *10.1109/TVCG.2021.3070046*.

Puso, Michael A. and Tod A. Laursen (Feb. 2004). **“A mortar segment-to-segment contact method for large deformation solid mechanics”.** en. In: *Computer Methods in Applied Mechanics and Engineering* 193.6–8, pp. 601–629. ISSN: 00457825. DOI: *10.1016/j.cma.2003.10.010*.

Thank You

Questions?

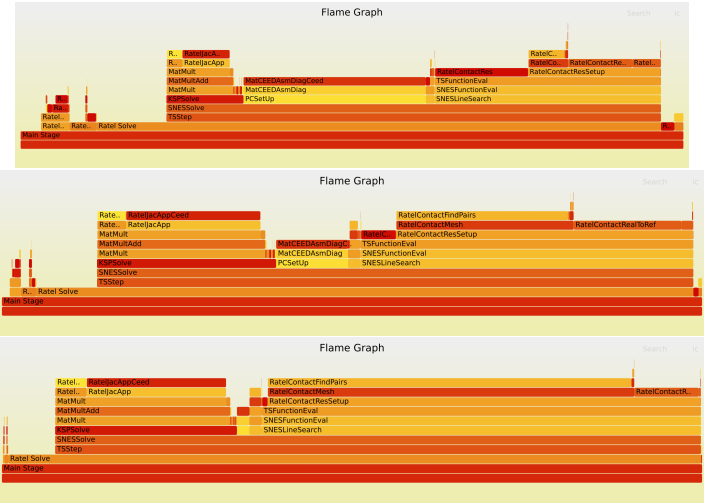
Aside: Similar Methods & Future Applications



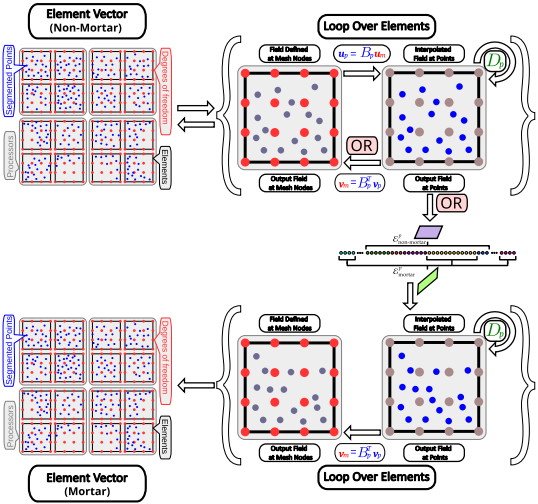
Generalized FEM (GFEM)

GFEM methods (Fries and Belytschko 2010) also use cut elements, and the parallelization strategy described later could be applied for them as well

Flame Graphs: 32^2, 64^2, & 128^2 Face Elements



Rough Contact Operator Diagram



Full Details on Basis Applications

$$\sum_e \mathcal{E}^T \left[(\mathbf{CN})^T \mathbf{W}_p^e \Lambda \left(g_0 \left(u_p^e, \nabla u_p^e \right) \right) \right]$$

with $u_p^e = \mathbf{N}_p^e \mathbf{CN} \mathcal{E}^e u$, $\nabla u_p^e = \{\mathbf{D}_{p,i}^e \mathbf{CN} \mathcal{E}^e u\}_{i=0}^{d-1}$, $\mathbf{CN} = (CN) \otimes (CN) \otimes (CN)$ and

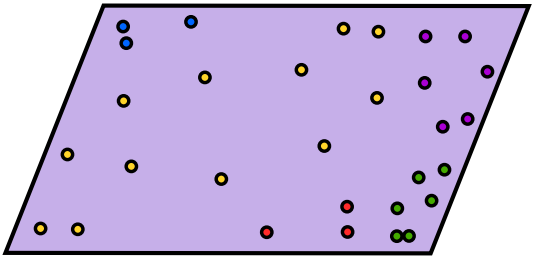
$$\mathbf{N}_p^e = N_{p,0}^e \otimes N_{p,1}^e,$$

$$\mathbf{D}_{p,0}^e = D_{p,0}^e \otimes N_{p,1}^e, \quad \mathbf{D}_{p,1}^e = N_{p,0}^e \otimes D_{p,1}^e$$

$$(N_{p,d}^e)_{ij} = T_j \left((\boldsymbol{\xi}_i^e)_d \right), \quad d = 0, 1$$

$$(D_{p,d}^e)_{ij} = \frac{\mathrm{d}T_j}{\mathrm{d}\xi} \left((\boldsymbol{\xi}_i^e)_d \right), \quad d = 0, 1$$

Computing the Integral Over the Surface



$$\int f(x)\,\mathrm{d}A \approx \sum_{i} f(\bullet_i) w_i \det J + \sum_{i} f(\bullet_i) w_i \det J + \sum_{i} f(\bullet_i) w_i \det J + \sum_{i} f(\bullet_i) w_i \det J + \sum_{i} f(\bullet_i) w_i \det J$$