Finite elements on polygonal discretizations: Concepts and applications

Sofie Leon
Cameron Talischi

Seminar at LCCV/UFAL
July 1st, 2013
We are from…

United States  Illinois  Champaign-Urbana

Total area of ~5000 km-sq. and population of ~210,000!
We are from…

University of Illinois at Urbana-Champaign

With ~42,000 students, ~3,000 faculty
Acknowledgements

Prof. Glaucio Paulino, UIUC
Daniel Spring, UIUC
Prof. Ivan Menezes, PUC-Rio
Prof. Marcio Carvalho, PUC-Rio
Prof. Waldemar Celes, PUC-Rio
Dr. Anderson Pereira, Tecgraf
Outline:

Motivation for polygonal discretizations

Mesh generation and adaptation

Formulation of polygonal finite elements

Dynamic fracture simulations
Motivation for polygonal discretizations

Mesh generation and adaptation

Formulation of polygonal finite elements

Dynamic fracture simulations
We compare the performance of polygonal elements with low order triangles and quads. The quantity of interest is the tip deflection at midspan. Cook’s Swept panel:

Elasticity BVP:

$$\text{div} \ (C \nabla_s u) = 0 \quad \text{in} \ \Omega$$

$$(C \nabla_s u) \cdot n = t \quad \text{on} \ \Gamma_t$$

$$u = g \quad \text{on} \ \Gamma_u$$

The quantity of interest is the tip deflection at midspan.
Comparison of performance 10 sample random and CVT meshes with regular quadrilateral partition

[Talischi, Paulino, Pereira, Menezes, 2010]
Roughly 2x DOFs are needed with the quad mesh for the same level of accuracy.

[Talischi, Paulino, Pereira, Menezes, 2010]
The factor is x1.6 for the triangulation (connecting centroid to the vertices) of CVT meshes.
We have observed improved accuracy for 3D elasticity problems when using polyhedral discretizations.

---

Random Voronoi

Centroidal Voronoi

\[ \sigma_{zz} \]

\[ \text{Centroidal Voronoi} \]

![Graph showing L2 error in displacements vs. Number of DOFs](image)

Random Voronoi

Hex. FEM

CVT

L2 error in displacements

Number of DOFs

\[ \text{[Gain, Talischi, Paulino, 2013, unpublished]} \]
Low-order mixed finite elements on polygonal meshes are naturally incompressible Stokes flow

\[
a(u_h, v_h) + b(p_h, v_h) = l(v_h), \quad \forall v_h \in V_h \subseteq H^1_0(\Omega)^2
\]

\[
b(q_h, u_h) = 0, \quad \forall q_h \in Q_h \subseteq L^2_0(\Omega)
\]

\[
a(u, v) = \int_\Omega \mu \nabla_s u : \nabla_s v \, dx, \quad b(p, v) = -\int_\Omega p \, \text{div} \, v \, dx
\]

Lid-driven cavity

[u = (1, 0)^T]

Computed velocity field

[Talischi, Pereira, Paulino, Menezes, Carvalho, 2013]
Low-order mixed finite elements on polygonal meshes are naturally incompressible Stokes flow

\begin{align*}
a(u_h, v_h) + b(p_h, v_h) &= l(v_h), & \forall v_h \in V_h \subseteq H^1_0(\Omega)^2 \\
b(q_h, u_h) &= 0, & \forall q_h \in Q_h \subseteq L^2_0(\Omega)
\end{align*}

Piece-wise constant

\begin{align*}
a(u, v) &= \int_\Omega \mu \nabla_s u : \nabla_s v \, dx, & b(p, v) &= -\int_\Omega p \, \text{div} \, v \, dx
\end{align*}

[Talischi, Pereira, Paulino, Menezes, Carvalho, 2013]
The Babuska-Brezzi conditions are shown to hold if every vertex of the mesh is incident on at most three edges.

\[
\tilde{\beta}_h := \inf q_h \in (Q_h^0)^\perp \sup v_h \in V_h \frac{b(q_h, v_h)}{\|q_h\|_Q \|v_h\|_V}
\]

\[Q_h^0 = \{ q_h \in Q_h : b(q_h, v_h) = 0, \forall v_h \in V_h \}\]

Space of pressure modes

Compute for a sequence of meshes

\[\tilde{\beta}_h\] remains bounded away from zero under mesh refinement for polygonal meshes

[Talischi, Pereira, Paulino, Menezes, Carvalho, 2013]
We again observe improved accuracy compared with polygonal discretizations.

Singular solution on L-shaped domain

\[ \| p - p_h \|_{L^2} \]

Number of DOFs

[Talischi, Pereira, Paulino, Menezes, Carvalho, 2013]
Motivation for polygonal discretizations

Mesh generation and adaptation

Formulation of polygonal finite elements

Dynamic fracture simulations
Motivation for polygonal discretizations

Mesh generation and adaptation

Formulation of polygonal finite elements

Dynamic fracture simulations
A polygonal discretization can be obtained from the Voronoi diagram of a given set of seeds and their reflections.

Recall the definition of Voronoi tessellation associated with $P$:

\[
\mathcal{T}(P; \Omega) = \{V_y \cap \Omega : y \in P\}
\]

\[
V_y = \{x \in \mathbb{R}^d : \|x - y\| < \|x - z\|, \forall z \in P \setminus \{y\}\}
\]
A polygonal discretization can be obtained from the Voronoi diagram of a given set of seeds and their reflections.

Set of seeds placed inside the domain
A polygonal discretization can be obtained from the Voronoi diagram of a given set of seeds and their reflections.

Reflections of the seeds about the boundary $R_\Omega(P)$
A polygonal discretization can be obtained from the Voronoi diagram of a given set of seeds and their reflections.

The mesh consists of the Voronoi cells associated with $P$:

$$\mathcal{M}_\Omega(P) := \left\{ V_y \in \mathcal{T}(P \cup R_\Omega(P); \mathbb{R}^d) : y \in P \right\}$$

[Talischi, Paulino, Pereira, Menezes, 2012]
We use Lloyd’s algorithm to obtain a using uniform point set from a random set.
We use Lloyd’s algorithm to obtain a using uniform point set from a random set

Each iteration consists of replacing each seed by the centroid of its cell:

\[ P_{i+1} = L(P_i) \]

Lloyd’s map:

\[ L_y(P) = \frac{\int_{V_y(P) \cap \Omega} x \mu(x) dx}{\int_{V_y(P) \cap \Omega} \mu(x) dx} \]

Prescribed density function

[Talischi, Paulino, Pereira, Menezes, 2012]
We use Lloyd’s algorithm to obtain a uniform point set from a random set:

Each iteration consists of replacing each seed by the centroid of its cell:

\[
P_{i+1} = L(P_i)
\]

Lloyd’s map:

\[
L_y(P) = \frac{\int_{V_y(P) \cap \Omega} x \mu(x) \, dx}{\int_{V_y(P) \cap \Omega} \mu(x) \, dx}
\]

Prescribed density function

[Talischi, Paulino, Pereira, Menezes, 2012]
We use Lloyd’s algorithm to obtain a using uniform point set from a random set

Each iteration consists of replacing each seed by the centroid of its cell:

\[ P_{i+1} = L(P_i) \]

The Lloyd’s algorithm produces a Centroidal Voronoi Tessellation (CVT)

\[ y = \frac{\int_{V_y \cap \Omega} x \mu(x) dx}{\int_{V_y \cap \Omega} \mu(x) dx}, \quad \forall y \in P \]

[Talischi, Paulino, Pereira, Menezes, 2012]
The Lloyd’s algorithm is a descent algorithm for minimizing the following energy functional:

\[ \mathcal{E}(P; \Omega) = \sum_{y \in P} \int_{V_y(P) \cap \Omega} \mu(x) \|x - y\|^2 \, dx \]

Minimizers of this functional are “stable” CVTs.
Comparison of mesh quality

Random seeds

Grayscale plot of c.o.v. edge length

Element areas

\[ \mu = 3.7 \times 10^{-3} \]
\[ \sigma = 1.6 \times 10^{-3} \]

Interior angles

\[ \mu = 120^\circ \]
\[ \sigma = 26.5^\circ \]
Comparison of mesh quality

Quasi-random seeds

Grayscale plot of c.o.v. edge length

Element areas

\[ \mu = 3.7 \times 10^{-3} \]
\[ \sigma = 0.6 \times 10^{-3} \]

Interior angles

\[ \mu = 120^\circ \]
\[ \sigma = 16.8^\circ \]
Comparison of mesh quality

CVT mesh

Element areas
\[ \mu = 3.7 \times 10^{-3} \]
\[ \sigma = 0.2 \times 10^{-3} \]

Interior angles
\[ \mu = 120^\circ \]
\[ \sigma = 7.7^\circ \]

Grayscale plot of c.o.v. edge length
One can construct graded meshes with prescribed element size using non-constant density function.
The Voronoi meshing algorithms can be extended to three-dimensions

[Duarte, Celes, Menezes, Paulino, 2013]

[Ebeida, Mitchell, 2012]
PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab

Cameron Talischi · Glaucio H. Paulino · Anderson Pereira · Ivan F. M. Menezes

The domain SDF encodes the information necessary for seed placement and reflection
For fracture simulation applications, crack patterns on structured grids are biased by the mesh.

4k structured mesh is commonly used in fracture simulation.

Crack patterns on the structured 4k mesh are limited by the element facets.

[Zhang et al, 2007]
Mesh adaptivity operators are introduced to improve fracture patterns.

![Mesh refinement](image)

**Original 4k mesh**

**Nodal perturbation**

**Edge swap**

[Paulino et al, 2010; Park et al 2012]
Even with adaptive operators, structured meshes exhibit bias for crack propagation.

FEM distance measured with Dijkstra’s algorithm

Error in crack length is dependent on angle of propagation

[Leon, Spring, Paulino 2013, submitted]
CVT meshes provide an alternative to structured meshes that reduces mesh bias

Crack length studies show the CVT meshes are isotropic. However, error is significantly higher than the 4k mesh.

[Leon, Spring, Paulino 2013, submitted]
High error is explained by the lack of available crack directions in a CVT mesh.

Typically 3 possible directions

[Leon, Spring, Paulino 2013, submitted]
We introduce element splitting to provide more directions for the crack to propagate.

[Leon et al, 2013]
To avoid poorly shaped elements, we limit the available nodes for element splitting. Choose the node that minimizes the difference in areas.

[Leon et al, 2013]
Element splitting decreases the error in crack length

[Leon, Spring, Paulino 2013, submitted]
Element splitting decreases the error in crack length

Element splitting preserves isotropy and reduced error significantly

[Leon, Spring, Paulino 2013, submitted]
We also perform studies on crack angle because it is a quantity of interest in fracture simulation.

[Leon, Spring, Paulino 2013, submitted]
Crack angle deviation is significantly lower with polygonal meshes compared to 4K

No splitting/swapping

With splitting/swapping

[Leon, Spring, Paulino 2013, submitted]
Hausdorff distances are also lower for polygonal meshes compared to 4k

[Leon, Spring, Paulino 2013, submitted]
Simple refinement strategies can be implemented on polygonal element meshes
In fracture simulation, crack tips are tracked and elements within a given radius are refined.
In the quad refinement scheme, “hanging nodes” are handled naturally.
The error in crack length decreases when the mesh is refined.

We want to take advantage of the splitting scheme, in which the error in crack length was between 3-5%.
Element splitting plus quad refinement increases the number of potential crack directions
When element splitting is combined with quad refinement, the crack length error is very low.

All of the refinement schemes preserve isotropy.
Quad refinement results in lower error for crack length studies

1,700 CVT elements
1,700 CVT element refined ~10,000 quads
~10,000 CVT elements

CVT element meshes that are refined with quads have lower error than meshes with an equivalent number of CVT elements.
Error in Hausdorff distance with refinement is nearly as low as a mesh of fine polygons

1,700 CVT elements
1,700 CVT element refined ~10,000 quads
~10,000 CVT elements

Since the proposed refinement scheme will be applied adaptively, we will gain the benefit of a smaller Hausdorff distance associated with using a fine mesh without needing to refine the entire domain.
Motivation for polygonal discretizations

Mesh generation and adaptation

Formulation of polygonal finite elements

Dynamic fracture simulations
For the sake of concreteness, we consider steady state diffusion as a model problem

**Strong form:**

\[- \text{div} (K \nabla u) = f \quad \text{in} \; \Omega \]

\[u = g \quad \text{on} \; \partial \Omega \]

For simplicity, we assume that the diffusion tensor is \textit{constant}. 

55
For the sake of concreteness, we consider steady state diffusion as a model problem.

**Strong form:**

\[- \text{div} (\mathbf{K} \nabla u) = f \quad \text{in} \quad \Omega\]
\[u = g \quad \text{on} \quad \partial \Omega\]

For simplicity, we assume that the diffusion tensor is **constant**.

**Weak form:**

\[
\begin{aligned}
\text{Find } u \in H^1_g(\Omega) \text{ s.t. } \\
 a(u, v) = \ell(v), \quad \forall v \in H^1_0(\Omega)
\end{aligned}
\]

\[
\ell(v) := \int_\Omega f v \, d\mathbf{x}
\]

\[
a(u, v) := \int_\Omega \nabla u \cdot \mathbf{K} \nabla v \, d\mathbf{x}
\]
For the sake of concreteness, we consider steady state diffusion as a model problem.

**Galerkin approximation:**

\[
\begin{align*}
\text{Find } u_h \in V_{h,g} \text{ s.t. } \\
a(u_h, v) = \ell(v), \quad \forall v \in V_{h,0}
\end{align*}
\]

**Conformity:**

\[
\begin{align*}
V_{h,g} &\subseteq H_{g}^1(\Omega) \\
V_{h,0} &\subseteq H_0^1(\Omega)
\end{align*}
\]
For the sake of concreteness, we consider steady state diffusion as a model problem.

### Galerkin approximation:

\[
\begin{align*}
\begin{cases}
\text{Find } u_h \in V_{h,g} \text{ s.t.} \\
a(u_h, v) &= \ell(v), \quad \forall v \in V_{h,0}
\end{cases}
\end{align*}
\]

### Conformity:

\[
V_{h,g} \subseteq H_g^1(\Omega) \\
V_{h,0} \subseteq H_0^1(\Omega)
\]

### Finite element space:

\[
V_h = \{ v_h \in C^0(\overline{\Omega}) : v_h|_E \in \nabla(E), \forall E \}
\]

Continuity across element boundaries

Local element space with approximation capabilities
Central to the construction of usual FE interpolants are the so-called Generalized Barycentric Coordinates.

Set of functions $\varphi_1, \ldots, \varphi_n$ on $E$ s.t.:

$$\varphi_i(x) > 0 \quad \forall x \in E^\circ$$

$$p(x) = \sum_{i=1}^{n} p(x_i) \varphi_i(x) \quad \forall p \in \mathbb{P}_1(E)$$

Space of first-order polynomials
Central to the construction of usual FE interpolants are the so-called Generalized Barycentric Coordinates.

Set of functions \( \varphi_1, \ldots, \varphi_n \) on \( E \) s.t.:

\[
\varphi_i(x) > 0 \quad \forall x \in E^\circ
\]

\[
p(x) = \sum_{i=1}^{n} p(x_i) \varphi_i(x) \quad \forall p \in \mathbb{P}_1(E)
\]

We have a continuous extension to the boundary s.t.:

\[
\varphi_i(x_j) = \delta_{ij}
\]

\( \varphi_i \) linear on \( \partial E \)

Space of first-order polynomials

Derived properties
We give some examples of generalized barycentric coordinates:

**Harmonic coordinates:**

\[-\Delta \varphi_i = 0 \quad \text{in } E\]

\[
\begin{cases}
\varphi_i(x_j) = \delta_{ij} \\
\varphi_i \text{ linear on } \partial E
\end{cases}
\]

For practical implementation in 2D and 3D using BEM, we refer to the work of S. Weiber and co-workers.
We give some examples of generalized barycentric coordinates:

Discrete harmonic [Pinkall et al., 1993]:

\[ \varphi_i(x) := w^*(x) \]

where:

\[ w^* = \arg\min_{w \in S_i(T_x)} \int_{\Omega} |\nabla w|^2 \, dx \]

Space of piecewise linear functions with prescribed BC

Analytical expressions are available!
We give some examples of generalized barycentric coordinates:

Mean value [Floater, 2003]:

\[ \varphi_i(x) := w^*(x) \]

where \( w^* \in S_i(T_x) \) such that:

\[ w^*(x) = \frac{1}{2\pi r} \int_{\partial B(x,r)} w^* ds \]

Sufficiently small neighborhood of \( x \)

Analytical expressions are available!
We give some examples of generalized barycentric coordinates:

Wachspress [1975]:

\[ \varphi_i(x) = \frac{w_i(x)}{\sum_{j=1}^{n} w_j(x)} \]

where:

\[ w_i(x) = A(x_{i-1}, x_i, x_{i+1}) \prod_{j \neq i-1, i} A(x, x_j, x_{j+1}) \]

Typical Wachspress functions on a regular hexagon:
We can define (conforming) local finite element spaces using any one of the available barycentric coordinates:

“Linear” elements:

\[ \mathcal{V}_1(E) = \text{span} \{ \varphi_1, \ldots, \varphi_n \} \supseteq \mathbb{P}_1(E) \]
We can define (conforming) local finite element spaces using any one of the available barycentric coordinates:

"Linear" elements:

\[ \mathcal{V}_1(E) = \text{span} \{ \varphi_1, \ldots, \varphi_n \} \supseteq \mathbb{P}_1(E) \]

"Quadratic" serendipity elements:

\[ \mathcal{V}_2(E) = \text{span} \{ \psi_1, \ldots, \psi_{2n} \} \supseteq \mathbb{P}_2(E) \]

where:

\[ \psi_i(x) = \sum_{a=1}^{n} \sum_{b=1}^{n} c_{i}^{ab} \varphi_a(x) \varphi_b(x) \]

and \( c_{i}^{ab} \) chosen such that:

\[ p(x) = \sum_{i=1}^{n} \left[ p(x_i) \psi_i(x) + p(\hat{x}_i) \psi_{i+n}(x) \right] \quad \forall p \in \mathbb{P}_2(E) \]
Evaluation of weak form integrals require quadrature and thus a departure from classical Galerkin framework.

Approximate problem with quadrature:

\[
\begin{cases}
\text{Find } u_h \in V_{h,g} \text{ s.t.} \\
a_h(u_h, v) = l_h(v), \quad \forall v \in V_{h,0}
\end{cases}
\]

where the \textit{discrete} bilinear form is defined by:

\[
a_h(u, v) = \sum_E a_h^E(u, v) = \sum_E \int_E \nabla u \cdot K \nabla v \, dx
\]

Numerically evaluated integral
Evaluation of weak form integrals require quadrature and thus a departure from classical Galerkin framework.

**Approximate problem with quadrature:**

\[
\begin{align*}
\text{Find } u_h & \in V_{h,0} \text{ s.t.} \\
a_h(u_h, \nu) & = l_h(\nu), \quad \forall \nu \in V_{h,0}
\end{align*}
\]

where the *discrete* bilinear form is defined by:

\[
a_h(u, \nu) = \sum_{E} a^E_h(u, \nu) \\
= \sum_{E} \int_{E} \nabla u \cdot \mathbf{K} \nabla \nu \, dx
\]

Numerically evaluated integral

Triangular subdivision

Quadrilateral subdivision
The celebrate patch test provides a means to assess the consistency of the approximation of the bilinear form

\( a_h^E(p, v) = a^E(p, v), \quad \forall v \in \mathbb{V}_m(E), p \in \mathbb{P}_m(E) \)

Patch test at the element level:

We perform a global patch test on a regular sequence of meshes for both linear and quadratic elements:

2\textsuperscript{nd} level mesh
The celebrated patch test provides a means to assess the consistency of the approximation of the bilinear form

**Patch test at the element level:**

\[ a_h^E(p, v) = a^E(p, v), \quad \forall v \in \mathbb{V}_m(E), p \in \mathbb{P}_m(E) \]

**Exact solution:**

\[ u(x) = 2x_1 - x_2 + 4 \]

**Error measures:**

\[ \epsilon_0(h) = \frac{\|u - u_h\|_0}{\|u\|_0} \]

\[ \epsilon_1(h) = \frac{|u - u_h|_1}{|u|_1} \]
The persistent errors in the patch test under mesh refinement also that FE approximation are NOT convergent!

Linear elements (same meshes)

Exact solution:
\[ u(x) = \sin(x_1) \exp(x_2) \]

Error measures:
\[ \varepsilon_0(h) = \frac{\|u - u_h\|_0}{\|u\|_0} \]
\[ \varepsilon_1(h) = \frac{|u - u_h|_1}{|u|_1} \]
The persistent errors in the patch test under mesh refinement also indicate that FE approximation are NOT convergent!

**Exact solution:**
\[ u(x) = \sin(x_1) \exp(x_2) \]

**Error measures:**
\[ \epsilon_0(h) = \frac{\|u - u_h\|_0}{\|u\|_0} \]
\[ \epsilon_1(h) = \frac{|u - u_h|_1}{|u|_1} \]

**Quadratic elements**
(same meshes)
The persistent errors in the patch test under mesh refinement also that FE approximation are NOT convergent!

Variation of Strang’s lemma [Beirão da Veiga et al., 2013]:

\[ |u - u_h|_1 \leq C \left( |u - \mathcal{I}_h u|_1 + |u - \tau_h u|_h + \sup_{v_h} \sum_{E} \frac{|a_h^E(\tau_h u, v_h) - a^E(\tau_h u, v_h)|}{|v_h|_1} \right) \]

Interpolation error

Poly. consistency error
We restore polynomial consistency at the level element using concepts from the Virtual Element Method [Brezzi and co-workers]

Let us define a map

\[ \Pi : \mathbb{V}_m(E) \rightarrow \mathbb{P}_m(E) \]

such that

\[ \Pi p = p, \quad \forall p \in \mathbb{P}_m(E) \]

Projection map! Leaves poly. fncs. intact
We restore polynomial consistency at the level element using concepts from the Virtual Element Method [Brezzi and co-workers]

Let us define a map

\[ \Pi : \mathcal{V}_m(E) \rightarrow \mathbb{P}_m(E) \]

such that

\[ \Pi p = p, \quad \forall p \in \mathbb{P}_m(E) \]

and

\[ a^E(u - \Pi u, p) = 0, \quad \forall p \in \mathbb{P}_m(E) \]

“Remainder” is energetically orthogonal to polynomials

Leaves poly. fncs. intact
We restore polynomial consistency at the level element using concepts from the Virtual Element Method [Brezzi and co-workers]

Let us define a map

\[ \Pi : \nabla_m(E) \rightarrow P_m(E) \]

such that

\[ \Pi p = p, \quad \forall p \in P_m(E) \]

and

\[ a^E(u - \Pi u, p) = 0, \quad \forall p \in P_m(E) \]

Or equivalently:

\[ a^E(\Pi u, p) = a^E(u, p), \quad \forall p \in P_m(E) \]
We restore polynomial consistency at the level element using concepts from the Virtual Element Method [Brezzi and co-workers]

Let us define a map

\[ \Pi : \mathbb{V}_m(E) \rightarrow \mathbb{P}_m(E) \]

such that

\[ \Pi p = p, \quad \forall p \in \mathbb{P}_m(E) \]

and

\[ a^E(u - \Pi u, p) = 0, \quad \forall p \in \mathbb{P}_m(E) \]

Or equivalently:

\[ \Pi u = \arg \min_{p \in \mathbb{P}_m(E)} a^E(u - p, u - p) \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Using the *kinematic* decompositions:

\[ u = \Pi u + (u - \Pi u) \]
\[ \nu = \Pi \nu + (\nu - \Pi \nu) \]

we get:

\[ a^E(u, \nu) = a^E(\Pi u + (u - \Pi u), \Pi \nu + (\nu - \Pi \nu)) \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Using the *kinematic* decompositions:

\[ u = \Pi u + (u - \Pi u) \]
\[ v = \Pi v + (v - \Pi v) \]

we get:

\[ a^E(u, v) = a^E(\Pi u + (u - \Pi u), \Pi v + (v - \Pi v)) \]
\[ = a^E(\Pi u, \Pi v) + a^E(\Pi u, v - \Pi v) \]
\[ + a^E(u - \Pi u, \Pi v) + a^E(u - \Pi u, v - \Pi v) \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Using the *kinematic* decompositions:

\[ u = \Pi u + (u - \Pi u) \]
\[ v = \Pi v + (v - \Pi v) \]

we get:

\[ a^E(u, v) = a^E(\Pi u + (u - \Pi u), \Pi v + (v - \Pi v)) \]
\[ = a^E(\Pi u, \Pi v) + a^E(\Pi u, v - \Pi v) \]
\[ + a^E(u - \Pi u, \Pi v) + a^E(u - \Pi u, v - \Pi v) \]
\[ = a^E(\Pi u, \Pi v) + a^E(u - \Pi u, v - \Pi v) \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Thus we have the identity:

\[ a^E(u, v) = a^E(\Pi u, \Pi v) + a^E(u - \Pi u, v - \Pi v) \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Thus we have the identity:

\[ a^E(u, v) = a^E(\Pi u, \Pi v) + a^E(u - \Pi u, v - \Pi v) \]

We apply quadrature to the alternative representation of the exact bilinear form:

\[ a^E_h(u, v) := a^E(\Pi u, \Pi v) + \int_E \nabla (u - \Pi u) \cdot K \nabla (v - \Pi v) \, dx \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Thus we have the identity:

\[ a^E(u, \nu) = a^E(\Pi u, \Pi \nu) + a^E(u - \Pi u, \nu - \Pi \nu) \]

We apply quadrature to the alternative representation of the exact bilinear form:

\[ a^E_h(u, \nu) := a^E(\Pi u, \Pi \nu) + \int_E \nabla (u - \Pi u) \cdot K \nabla (\nu - \Pi \nu) \, dx \]

The resulting element PASSES the patch test!

\[ a^E_h(p, \nu) = a^E(\Pi p, \Pi \nu) = a^E(p, \Pi \nu) = a^E(p, \nu) \]
The projection can be used to decompose the bilinear form into polynomial and “non-polynomial” parts

Thus we have the identity:

\[ a^E(u, v) = a^E(\Pi u, \Pi v) + a^E(u - \Pi u, v - \Pi v) \]

We apply quadrature to the alternative representation of the exact bilinear form:

\[ a_h^E(u, v) := a^E(\Pi u, \Pi v) + \int_E \nabla (u - \Pi u) \cdot K \nabla (v - \Pi v) \, dx \]

In fact only a crude approximation of the second term is needed, thus significantly reducing the burden of numerical integration. In VEM,

\[ a_h^E(u, v) := a^E(\Pi u, \Pi v) + s^E(u - \Pi u, v - \Pi v) \]
Not only we recover optimal convergence rates, but the absolute errors are very close to Galerkin errors (<3%)

Exact solution:
\[ u(\mathbf{x}) = \sin(x_1) \exp(x_2) \]

Error measures:
\[ \epsilon_0(h) = \frac{\|u - u_h\|_0}{\|u\|_0} \]
\[ \epsilon_1(h) = \frac{\|u - u_h\|_1}{\|u\|_1} \]

Linear elements (same meshes)
Not only we recover optimal convergence rates, but the absolute errors are very close to Galerkin errors (<3%).

Exact solution:
\[ u(x) = \sin(x_1) \exp(x_2) \]

Error measures:
\[
\begin{align*}
\epsilon_0(h) &= \frac{\|u - u_h\|_0}{\|u\|_0} \\
\epsilon_1(h) &= \frac{|u - u_h|_1}{|u|_1}
\end{align*}
\]

Quadratic elements
(same meshes)
The cost of constructing the projection map is a small portion of overall construction of the stiffness matrix

Explicit expression for linear element:

\[ \Pi u = \arg\min_{p \in P_1(E)} a^E(u - p, u - p) \]
The cost of constructing the projection map is a small portion of overall construction of the stiffness matrix

Explicit expression for linear element:

\[ \Pi u = \arg\min_{p \in \mathbb{P}_1(E)} a^E(u - p, u - p) \]

\[ \implies \nabla (\Pi u) = \arg\min_{q \in [\mathbb{P}_0(E)]^2} \int_E |q - \nabla u|^2 \, dx \]

\[ = \frac{1}{|E|} \int_E \nabla u \, dx \]

\[ = \frac{1}{|E|} \int_{\partial E} u \, nds \]

Depends only on geometry of the element and boundary values!
The cost of constructing the projection map is a small portion of overall construction of the stiffness matrix

Explicit expression for linear element:

\[ \Pi u = \arg\min_{p \in \mathbb{P}_1(E)} a^E(u - p, u - p) \]

\[ \implies \nabla (\Pi u) = \arg\min_{q \in [\mathbb{P}_0(E)]^2} \int_E |q - \nabla u|^2 \, dx \]

\[ = \frac{1}{|E|} \int_E \nabla u \, dx \]

\[ = \frac{1}{|E|} \int_{\partial E} u n \, ds \]

Completing the definition by assigning an appropriate constant:

\[ (\Pi u)(x) := \bar{u} + \left( \frac{1}{|E|} \int_{\partial E} u n \, ds \right) \cdot (x - \bar{x}) \]

Depends only on geometry of the element and boundary values!
Outline:

- Motivation for polygonal discretizations
- Mesh generation and adaptation
- Formulation of polygonal finite elements
- Dynamic fracture simulations
Outline:

Motivation for polygonal discretizations

Mesh generation and adaptation

Formulation of polygonal finite elements

Dynamic fracture simulations
Cohesive elements capture the highly nonlinear behavior in the zone ahead of a crack tip.

When the size of the nonlinear zone ahead of a crack tip is not negligible, for example in ductile or quasi-brittle materials, the LEFM may not be appropriate.

[Park, 2009]
The traction-separation relation is characterized by the PPR potential-based cohesive zone model

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

Gradients of potential give traction-separation

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

\[
T_n(\Delta_n, \Delta_t) = -\alpha \frac{\Gamma_n}{\delta_n} \left(1 - \frac{\Delta_n}{\delta_n}\right)^{-1} \left[ \Gamma_t \left(1 - \frac{|\Delta_t|}{\delta_t}\right)^\beta + \langle \phi_t - \phi_n \rangle \right]
\]
Computational implementation of the PPR potential-based cohesive model in ABAQUS: Educational perspective

Kyoungsoo Park\textsuperscript{a,\*}, Glaucio H. Paulino\textsuperscript{b}

\textsuperscript{a} School of Civil & Environmental Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, South Korea.
\textsuperscript{b} Department of Civil & Environmental Engineering, University of Illinois at Urbana–Champaign, Urbana, Illinois, USA.

\textbf{ARTICLE INFO}

\textbf{Article history:}
Received 2 August 2011
Received in revised form 9 February 2012

\textbf{ABSTRACT}

A potential-based cohesive zone model implemented in the commercial software, e.g. ABAQUS, is described. This cohesive zone modeling approach is based on the non-local approach, which allows for the prediction of crack localization and propagation within a specified area. The implementation is demonstrated through a web-based tutorial on the NLS++ code, based on the Applied Mechanics Reviews journal paper:


\textbf{Educational Perspective of the PPR Potential Based Cohesive Model}

Welcome to the web-based tutorial on the NLS++ code, based on the Applied Mechanics Reviews journal paper:


\textbf{ForTRAN UEL and Resources}

The flowchart to the left shows the major steps of the PPR UEL, which is available below. The subroutine is entered and exited from the ABAQUS finite element routine at the green ellipses.

Download ABAQUS UEL (zip format)

Additional PPR Resources:
\begin{itemize}
  \item Instructions for implementing the PPR cohesive model in ABAQUS
  \item Nomenclature (equivalence between cohesive model parameters and ABAQUS material properties)
\end{itemize}
Dynamic fracture simulation with extrinsic model uses an explicit time integration scheme

1. Update displacements
2. Step = check cohesive step? yes no
3. Compute stress at nodes
4. Check facets for insertion of cohesive elements yes no
5. Insert cohesive elements
6. Compute internal force vector
7. Compute cohesive force vector
8. Update velocity
9. Update acceleration
10. Update energy
11. Apply boundary conditions
12. Cohesive element inserted? yes no
13. Refine about crack tips
14. Update element matrices and nodal mass
15. Transfer nodal quantities
16. Print output

Begin time step
Next time step
Dynamic fracture simulations with polygonal elements led to unrealistic results

Expected result contains complete fragments

[Paulino et al. 2010]
Dynamic fracture simulations with polygonal elements led to unrealistic results

Expected result contains complete fragments

[Paulino et al. 2010]
Even as the mesh is refined, the crack patterns do not converge to the expected result.
When element splitting is enabled, we obtain the desired fracture pattern.

Mesh refinement
Compact Compression Specimen (CCS) investigated with polygonal elements and splitting
Polygonal elements with splitting provide excellent results for CCS test
Polygonal elements with splitting provide excellent results for CCS test
Quadrilateral refinement plus splitting is superior than individual schemes

Full test specimen

Polygonal elements only

Refinement + element splitting

[Sharon and Fineberg, 1996]
Some concluding thoughts

• The availability of polygonal (and polyhedral) finite elements can lead to capabilities in mesh generation and adaptation

• Polygonal and polyhedral discretizations can exhibit improved accuracy compared to their simplicial and cubical counterparts

• These attributed are essential ingredients of their successful use in dynamics fracture simulations
Thank you for having us!

Maceio!
Thank you for having us!

Friends!
Thank you for having us!

Friends!

Questions?
Back up slides
A cohesive zone element is a set of two facets that can separate from each other.

The displacement calculated in the FEM analysis, the then cohesive traction is calculated from the cohesive constitutive relation.
CZ elements may be inserted a priori (intrinsic) or when/where they are needed (extrinsic)

Intrinsic approach is appropriate for scenarios where you know where the crack will propagate, e.g. material interfaces, but are not well suited for scenarios when the crack direction is unknown.

Mesh topology does not change in an intrinsic scheme, but constantly changes in an extrinsic scheme.
The intrinsic traction-separation relation is given by the PPR potential-based cohesive zone model

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

Gradients of the potential give the traction

Summary of User Inputs
Shape of softening: \( \alpha, \beta \)
Initial slope: \( \lambda_n, \lambda_t \)
Fracture energies: \( \Phi_n, \Phi_t \)
Cohesive strengths: \( \sigma, \tau \)
The **extrinsic** traction-separation relation is given by the PPR potential-based cohesive zone model

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

**Summary of User Inputs**

| Shape of softening: \( \alpha, \beta \) |
| Fracture energies: \( \Phi_n, \Phi_t \) |
| Cohesive strengths: \( \sigma, \tau \) |
Many models exist in the literature, and PPR parameters can be easily tuned to recover them.

**PPR**

- $\alpha = \beta = 5$
- $\delta_n = 0.13$
- $\delta_t = 0.1$
- $\Phi_n = \Phi_t = 352.3 \text{ J/m}^2$
- $\sigma = 324 \text{ MPa}$
- $\tau = 755.4 \text{ MPa}$

**Xu and Needleman 1994**

- $r = 0$
- $\delta_{cn} = 0.4 \mu m$
- $\delta_{ct} = 0.4 \mu m$
- $\Phi_n = \Phi_t = 352.3 \text{ J/m}^2$
- $\sigma = 324 \text{ MPa}$
- $\tau = 755.4 \text{ MPa}$