From the Algebraic Connectivity of Hypergraphs to Fretsaw Preconditioning

Gil Shklarski and Sivan Toledo

School of Computer Science
Tel-Aviv University

Second International Workshop on Combinatorial Scientific Computing
June 21–23th, 2005 at CERFACS, Toulouse, France

The Big Picture

We are working toward support preconditioners for finite-element matrices

We viewed the characterization of the algebraic connectivity of finite-element hypergraphs as a way to explore this topic

We made some progress (definitions and some theorems) but did not yet prove the main result

However, we did find a way to construct support preconditioners for finite-element matrices (including both elliptic problems and elasticity)!

Outline of the Talk

The algebraic connectivity of hypergraphs

Minimally-rigid-substucture preconditioners don't work

Fretsaw Preconditioners

Cheeger Bounds for Laplacians

Let A be the Laplacian of a weigted undirected graph G

A models a network of resistors: $Ax = b$, $b =$ currents, $x =$ potentials

$\Phi = \min_{\text{cut } C}\{\frac{\text{capacity}(C)}{|S|} \text{ s.t. } C = (S, \bar{S}), |S| \leq \frac{n}{2}\}$

Cheeger's bounds: $\frac{\lambda_2}{2} \leq \Phi \leq \sqrt{2\Delta\lambda_2}$

The lower bound on λ_2 is the interesting one: if the graph does not have a small cut (small per vertex) then λ_2 is large

How is This Related To Preconditioning?

Many support preconditioners drop edges from the graph to create the preconditioner

By Cheeger's bound, if you do not make any cut much smaller, then λ_2 will not shrink too much (but the eigenvectors may differ; λ_n is bounded by 2Δ in any case)

We also thought that generalizing Cheeger's bound to finite-elements matrices will give us tools to do other types of spectral analyses (it did)

Cheeger's Bounds for Finite Elements: the Issues

The issues:

how to define the (hyper)graph

how to define cuts (not what you think)

how to define capacity(C)

how to define Φ and how to relate it to $\lambda_{\min} > 0$

Finite-Element Hypergraphs and their Cuts

Mesh nodes are **vertices** in the graph

Nodes' values are vectors (e.g. x-y-z forces or displacements)

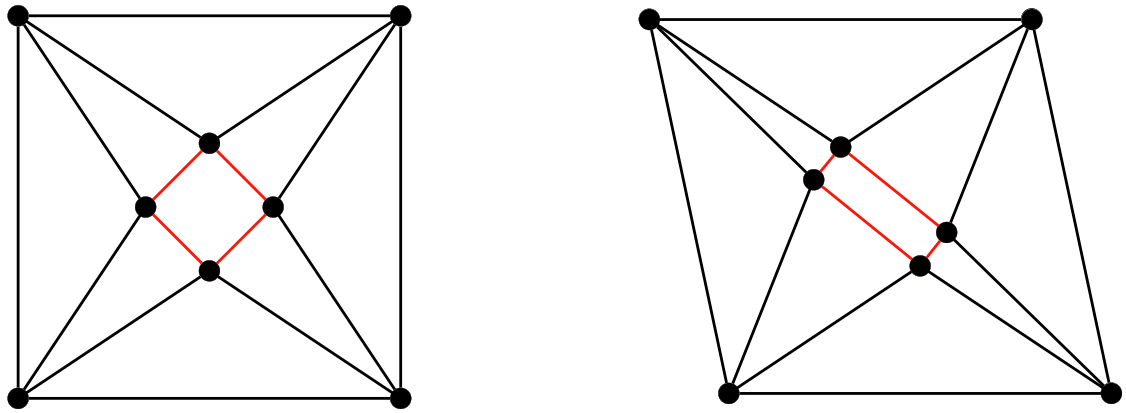
Each element e (element matrix A_e) corresponds to a **hyperedge**

A **cut** is a set of edges whose removal (exclusion from the global matrix's assembly) creates a new mechanism, a reduction in the rank of $A = \sum A_e$

The Capacity of a Cut

The minimal work possible under a unit vector of forces that is orthogonal to rigid-body motions (i.e., to $\mathrm{null}(A)$) and that strains only elements in the cut,

$$\lim_{\alpha \to \infty} \min_{x \perp \mathrm{null} A} \frac{x^T \left(\sum_{e \notin C} A_e + \sum_{e \in C} \alpha A_e \right) x}{x^T x}$$



Cheeger Bounds: Status

This generalizes perfectly the case of Laplacians

Note that cuts do not really cut; they add a mechanism

The upper bound on $\lambda_{\min}$ is easy

We still can't prove a lower bound...

An application for the lower bound: $\sigma(UU^T, VV^T) \leq \|V^+U\|_2^2 \leq \|V^+\|_2^2\|U\|_2^2 = \lambda_{\min}(VV^T)\|U\|_2^2$

The Next Idea: Cool but Flawed

Consider support preconditioners, like Vaidya's, that drop edges from a graph (nonzeros from a Laplacian)

We can try to generalize by dropping elements from a finite-element model (exclude A_e's from the assembly process)

A spanning-tree preconditioner is a minimally-connected substructure of a resistor network

Let's try to precondition elasticity problems with a minimally-rigid substructure

(To get good performance we may need to drop less, but let's explore the principle)

Minimally-Rigid Substructure

For two-dimensional trusses, rigidity can be characterized combinatorially

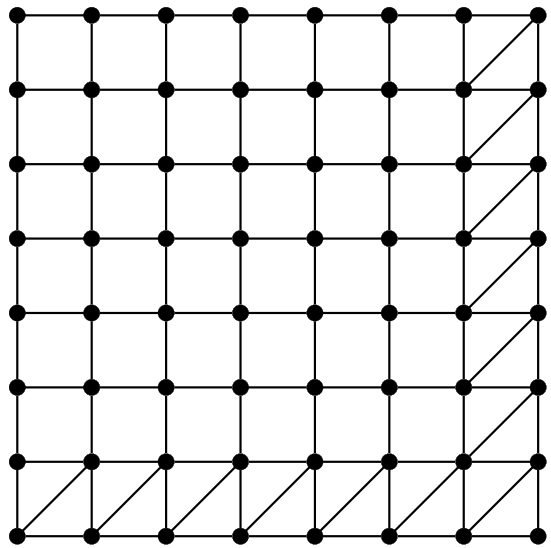
There are combinatorial algorithms to find a minimally-rigid substructure of a 2D truss (e.g., an $O(n^2)$ algorithm by Bruce Hendrickson)

Can we use this to constuct effective preconditioners? No, there is a serious flaw

Minimally-Rigid Structures can be Hard to Factor

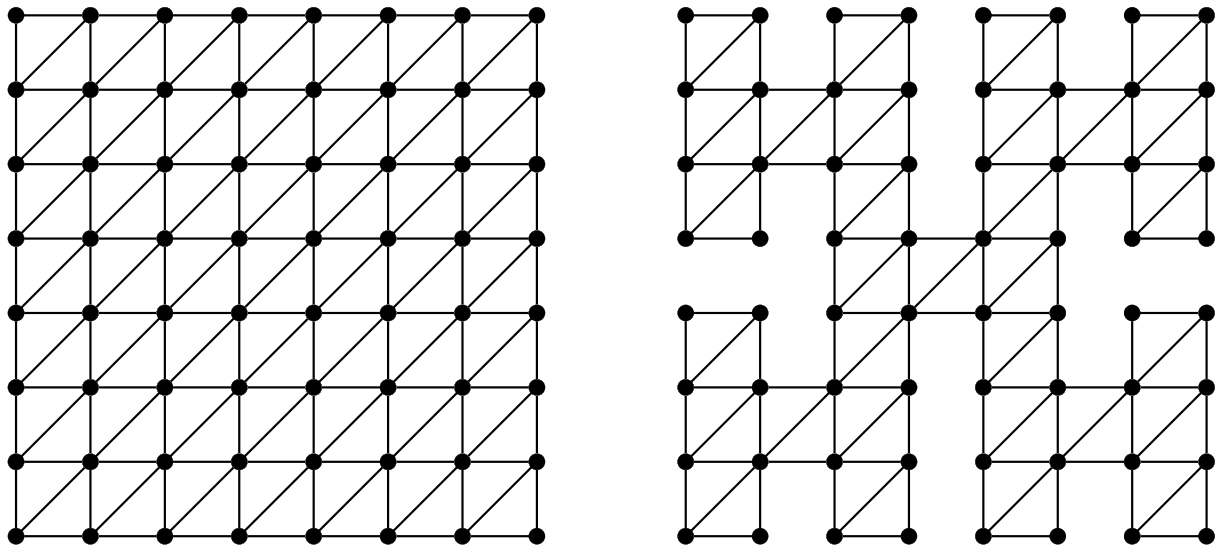
The minimally-connected subgraph of a Laplacian (a spanning tree) can always be factored with no fill in $O(n)$ time

But a minimally-rigid substructure can be as hard to factor as a generic planar truss (has $\Theta(\sqrt{n})$ separators)
We decided not to persue this for now



Dropping Elements, or...

In some cases, a rigid substructure can be a lot easier to factor



But this is not always true (at least for trusses; see the previous example), and when it does, we do not know how to find these substructure

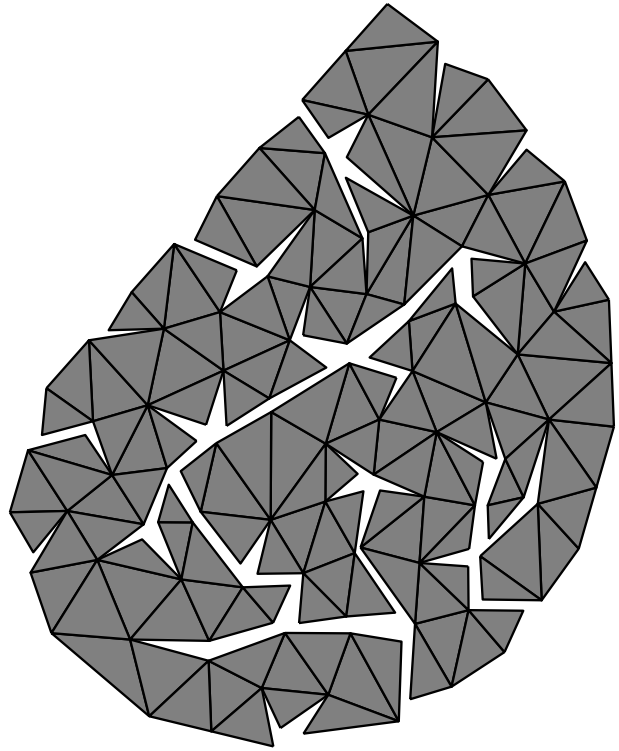
Cutting Slits: Fretsaw Preconditioning

We don't drop elements (cut away material)

We cut the "glue" that connects elements, but such that the structure remains rigid (we do not create new mechanisms)

This requires duplicating some nodes (augmenting the matrix with more rows/columns)

Works for any type of solid element (and triangularized truss) for both elliptic problems and elasticity (and probably more)

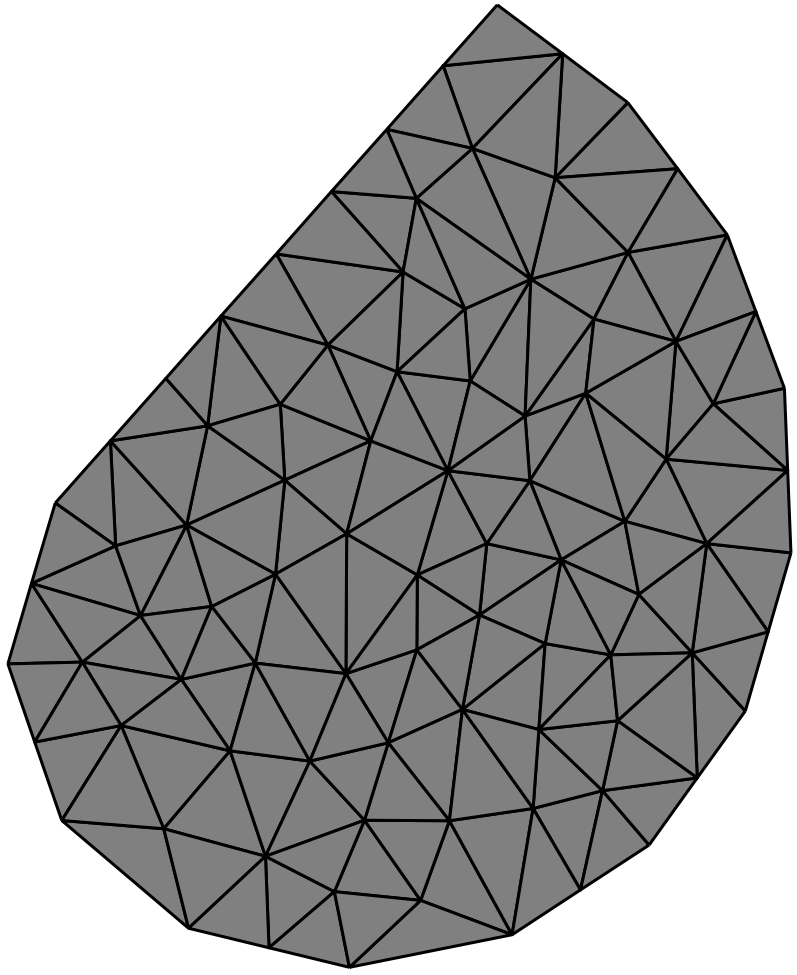


Fretsaw Preconditioning

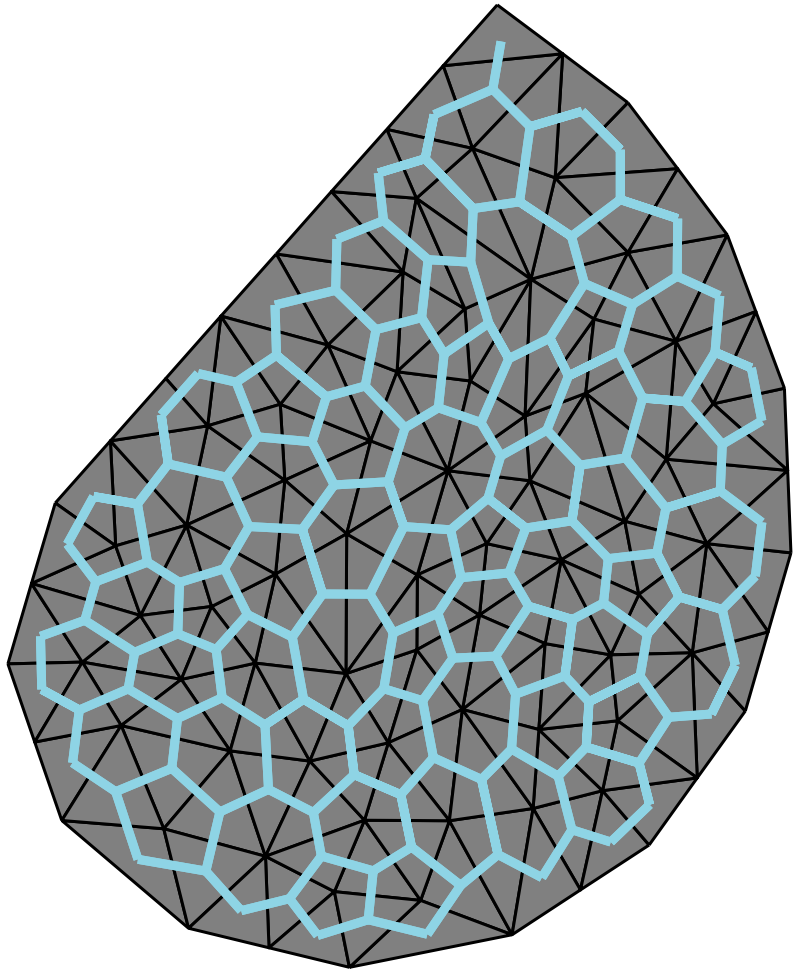
1. Compute the dual (simple) graph H of the finite-element hypergraph; adjacency is determined by a local path condition
2. Compute a good spanning tree T of H (maximum-weight, low stretch, etc); or even augment the tree as in Vaidya or Spielman-Teng
3. Determine node duplications; another local condition
4. Cut in-between elements whenever there is no edge in T (shift element indices to new duplicate rows/columns)
5. Assemble the preconditioner

Get Ready for the Animation!

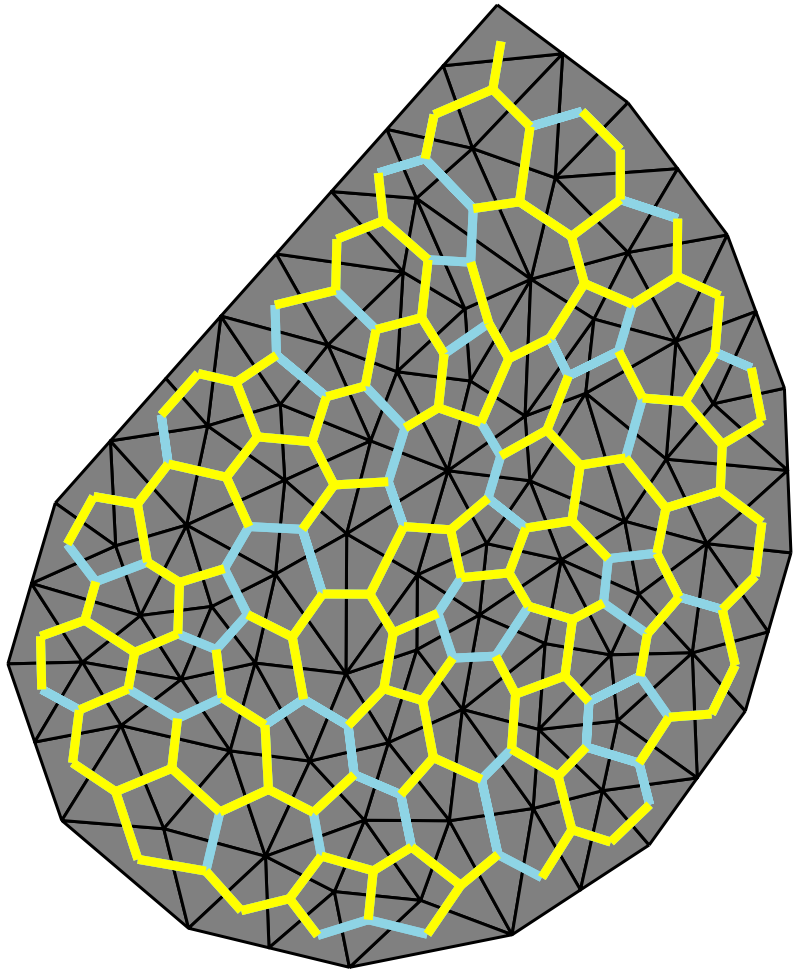
The Original Mesh



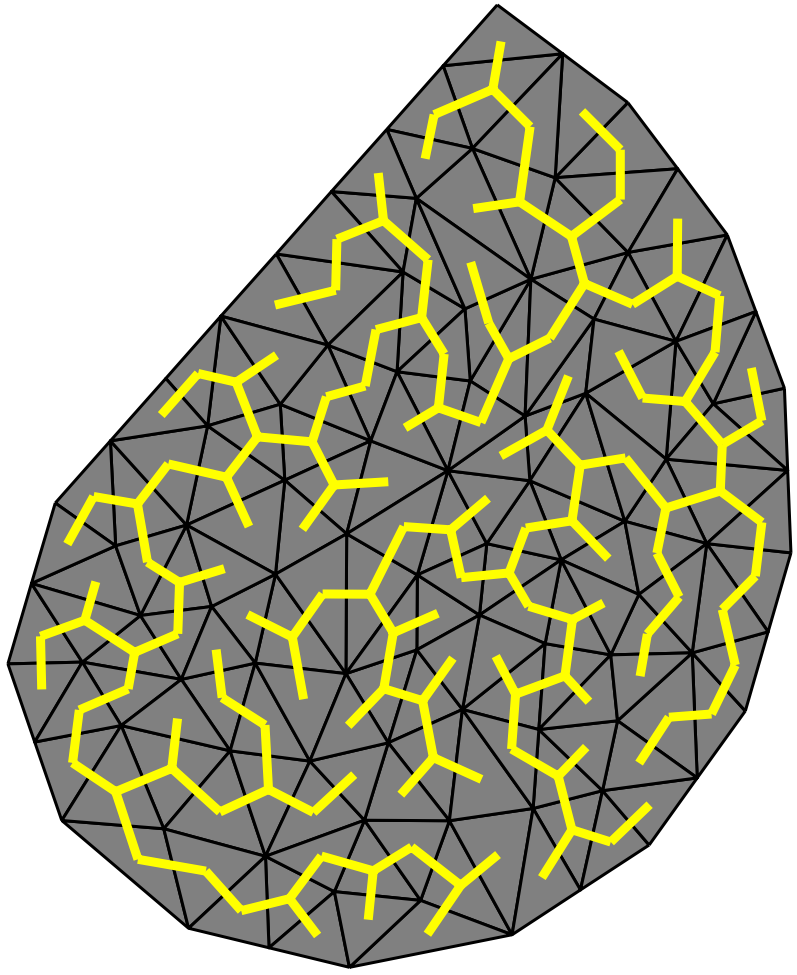
The Dual Graph of the Mesh



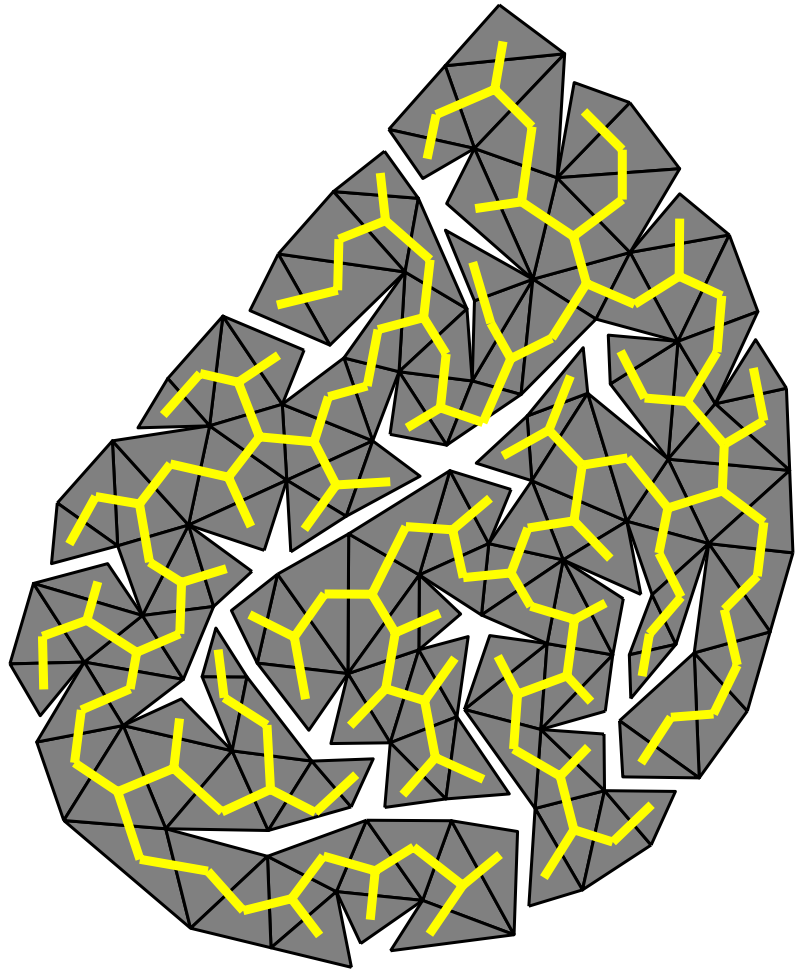
A Spanning Tree of the Dual Graph



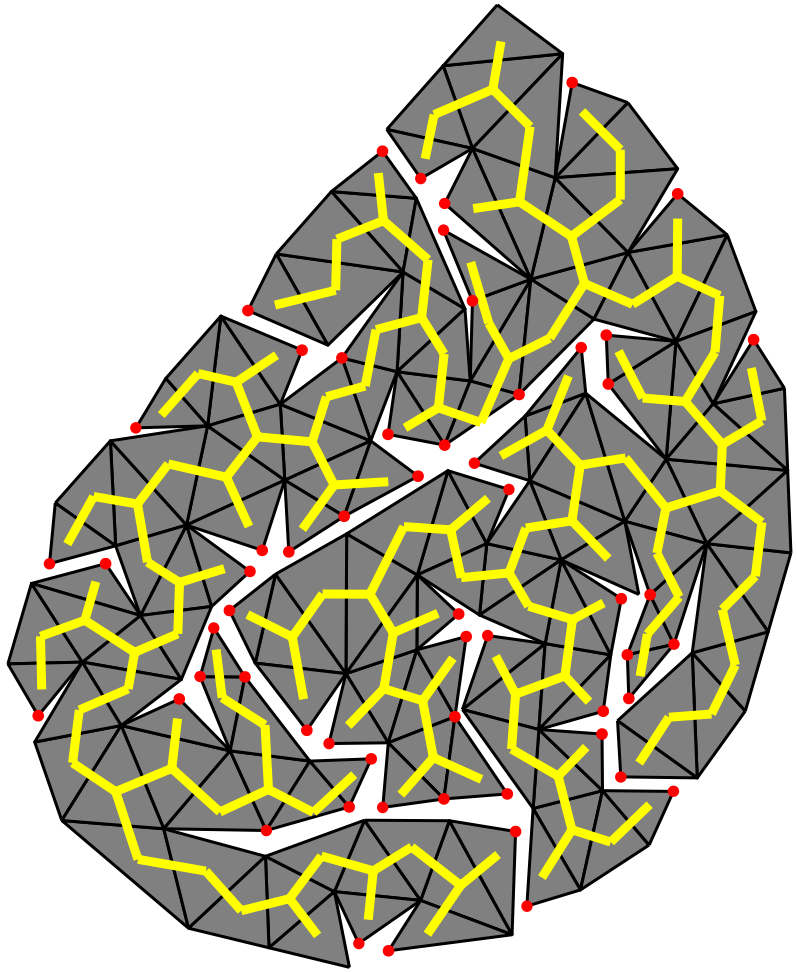
A Spanning Tree of the Dual Graph



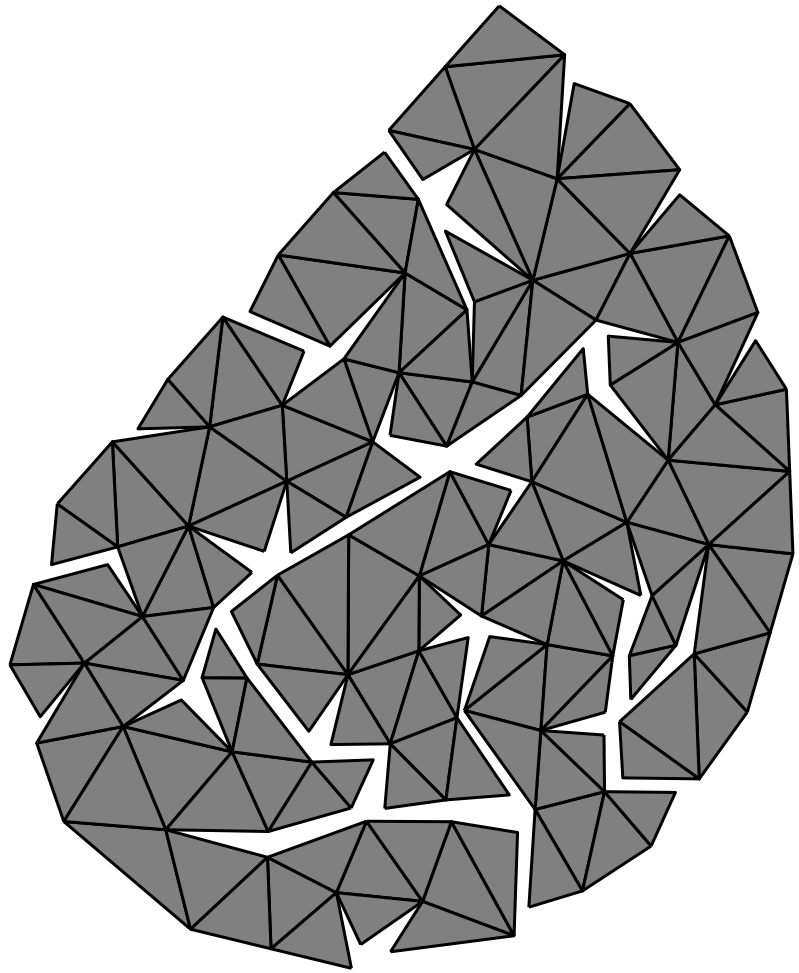
We Cut the Slits



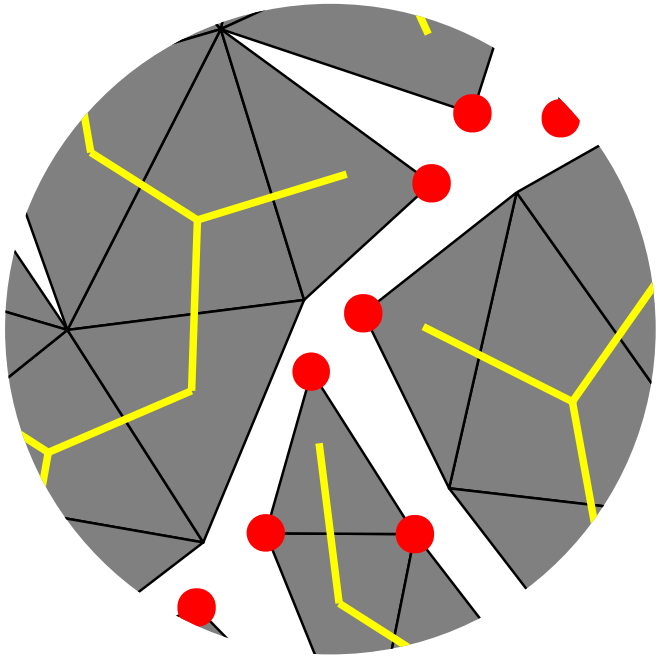
This Requires Duplication of Some Nodes

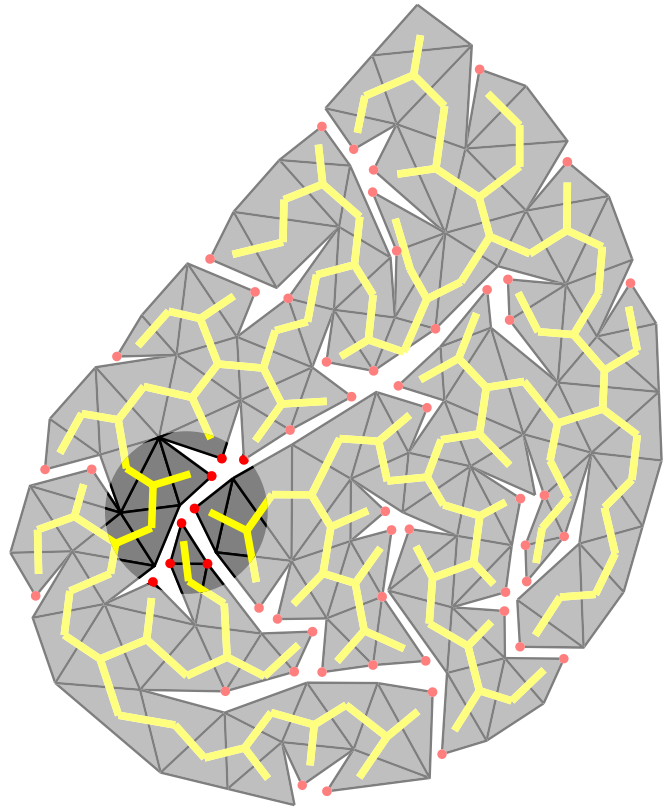


The Preconditioner Ready for Assembly



The Details of Node Duplication





Factoring and Applying the Preconditioner: Schur Complement Preconditioning

Fill: None

How? We order the rows/columns from a free node in a leaf element, recursively

To precondition, augment r with zeros, solve, restrict to original nodes to form z

This is equivalent to preconditioning with the Schur complement of the preconditioners, after eliminating the duplicate nodes (Gremban/Miller style)

Fretsaw for Finite-Elements vs. Vaidya for Laplacians

Fretsaw augments the system by cutting infinitesimal slits in the "material"; Vaidya cuts away material

In both cases one extreme generalized eigenvalue is bounded by 1; the preconditioner is weaker

In both cases the bound on the other extremal eigenvalue can be computed by path analysis

In both cases the construction selects a subgraph of a (simple) graph

When the selected subgraph is a tree, no fill

We can apply the fretsaw framework to Laplacians, but there may be no benefit to this

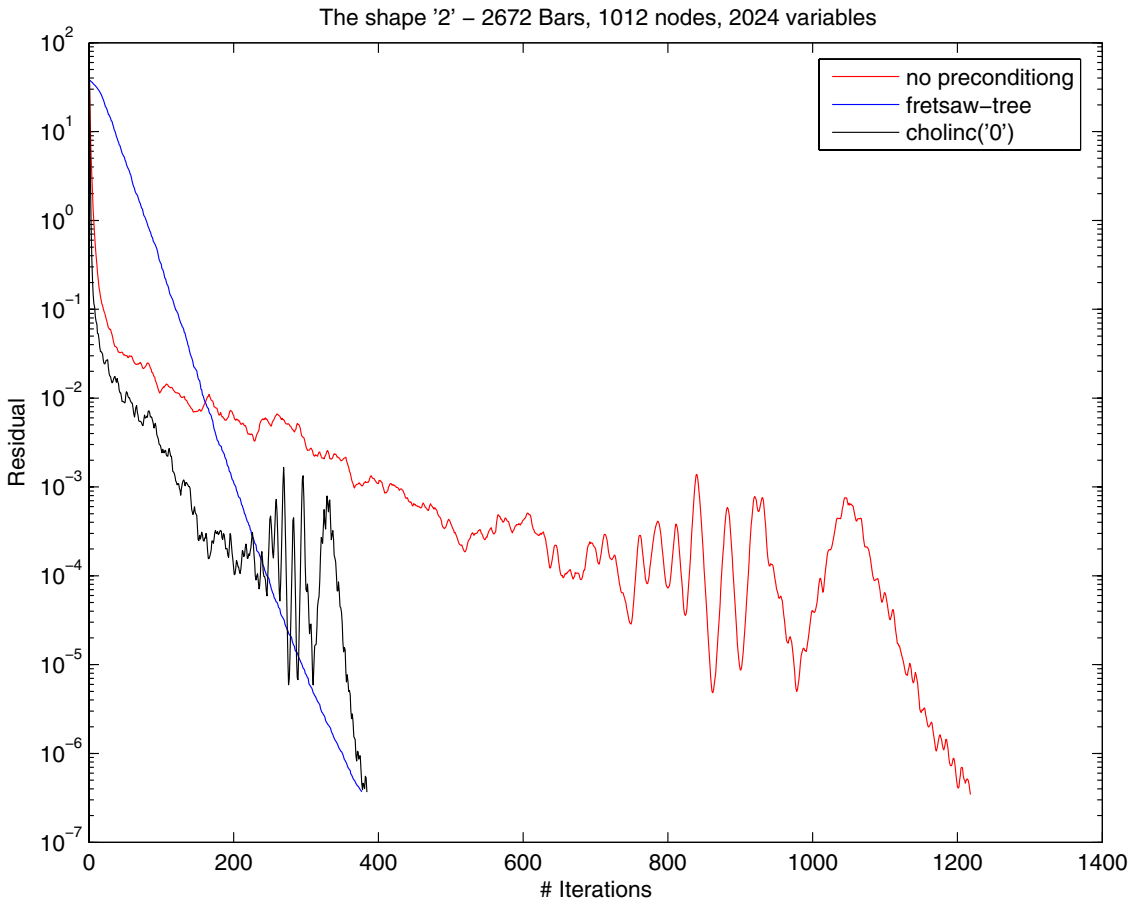
Fretsaw vs. Gremban/Miller's Support Trees

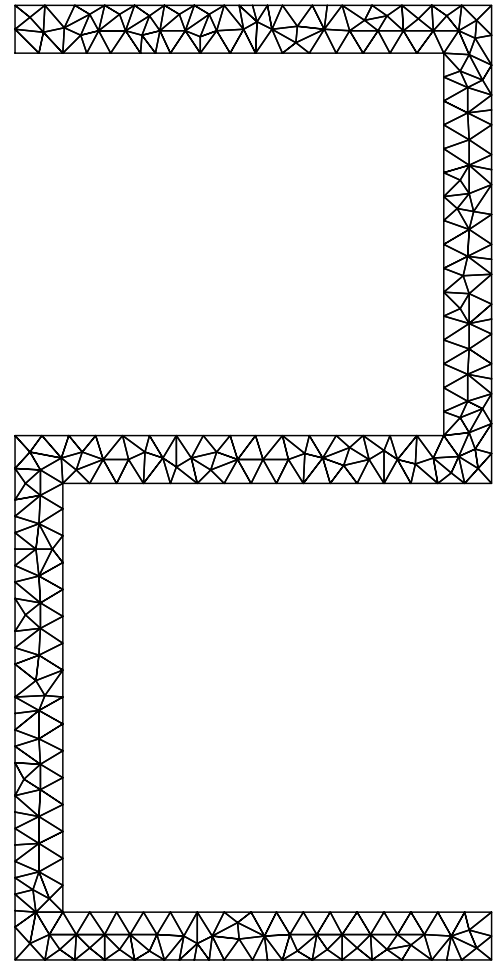
Almost no connection

Except that we use Gremban/Miller's augmented-preconditioner framework

But the preconditioners are completely different

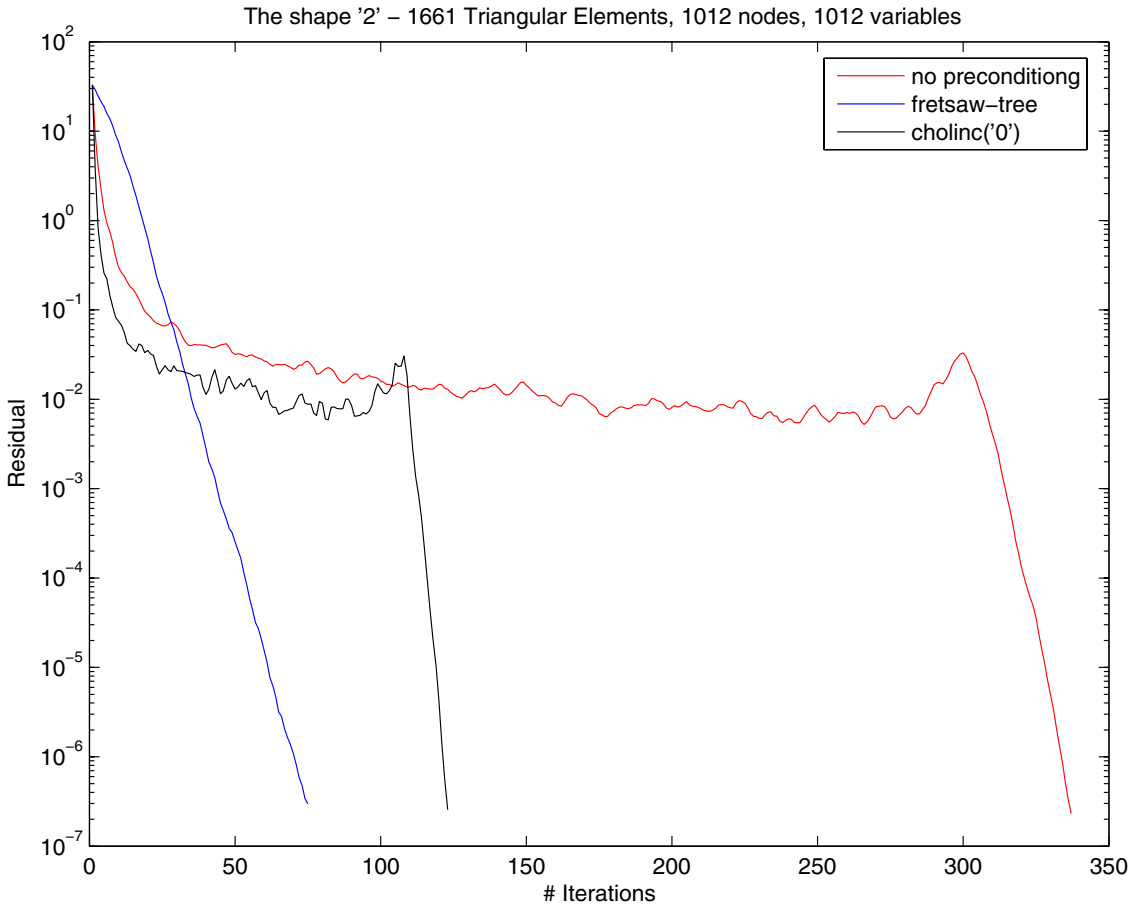
Experimental Results: Elasticity

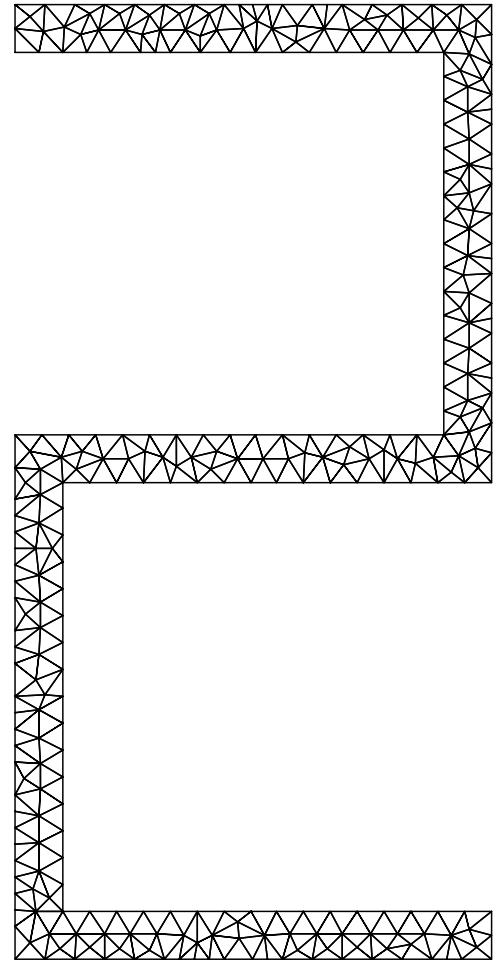




(figure is coarser than in the experiment)

Experimental Results: Poisson





Different element matrices, dual adjacency rule

Discussion and Conclusions

A class of support preconditioners for finite-elements

Preconditioning in a larger space by dropping elements in the dual graph

Relatively easy to apply to different problems

Most of the diagonally-dominant constructions apply

The problem specific aspect: defining adjacency in the dual graph and assigning edge weights (we're still working on the path lemmas)

That's It