A Posteriori Estimates and Mesh Refinement

Andreia Chapouto John Stewart

MIGSAA

What is Numerical Analysis?
27 March 2018

Outline

Outline

A Posteriori Estimate

Cea's Lemma

$$\|u - u_h\|_{H^1} \leq c \inf_{v_h \in H_h} \|u - v_h\|_{H^1}.$$

A Posteriori Estimate

Cea's Lemma

$$\|u - u_h\|_{H^1} \leq c \inf_{v_h \in H_h} \|u - v_h\|_{H^1}.$$

A Posteriori Estimate

$$\underbrace{c_* \eta_K \leq}_{\text{efficiency}} \|u - u_h\|_K \underbrace{\leq c^* \eta_K}_{\text{reliability}}, \quad \forall K \in \mathcal{T}$$

Outline

Poisson's Equation

Original Problem: Find u such that

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned}$$

Poisson's Equation

Original Problem: Find u such that

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned}$$

Variational Problem: Find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv, \quad \forall v \in H_0^1(\Omega)$$

Poisson's Equation

Original Problem: Find u such that

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned}$$

Variational Problem: Find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv, \quad \forall v \in H_0^1(\Omega)$$

Finite Element Problem: Find $u_h \in H_h(\mathcal{T})$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega fv_h, \quad \forall v_h \in H_h(\mathcal{T}).$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla (u - u_h) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla \left(u - u_h\right) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla \left(u - u_h\right) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \leq \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \|\nabla v\|_{L^2(\Omega)} \|\nabla w\|_{L^2(\Omega)}$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla \left(u - u_h\right) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \leq \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \|\nabla v\|_{L^2(\Omega)} \|\nabla w\|_{L^2(\Omega)} \leq \|v\|_{H^1(\Omega)}$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla (u - u_h) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \leq \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \|\nabla v\|_{L^2(\Omega)} \|\nabla w\|_{L^2(\Omega)} \leq \|v\|_{H^1(\Omega)}$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla (u - u_h) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \leq \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \|\nabla v\|_{L^2(\Omega)} \|\nabla w\|_{L^2(\Omega)} \leq \|v\|_{H^1(\Omega)}$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \geq \int_\Omega \nabla v \cdot \frac{\nabla v}{\|v\|_{H_1(\Omega)}} = \frac{\|\nabla v\|^2_{L^2(\Omega)}}{\|v\|_{H^1(\Omega)}}$$

Residual

Subtracting $\int_\Omega \nabla u_h \cdot \nabla v$ from the variational problem

$$\int_\Omega \nabla (u - u_h) \cdot \nabla v = \int_\Omega fv - \int_\Omega \nabla u_h \cdot \nabla v =: \text{Residual}[u_h](v).$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \leq \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \|\nabla v\|_{L^2(\Omega)} \|\nabla w\|_{L^2(\Omega)} \leq \|v\|_{H^1(\Omega)}$$

$$\sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla v \cdot \nabla w \geq \int_\Omega \nabla v \cdot \frac{\nabla v}{\|v\|_{H_1(\Omega)}} = \frac{\|\nabla v\|^2_{L^2(\Omega)}}{\|v\|_{H^1(\Omega)}} \geq \frac{1}{1 + c_\Omega^2} \|v\|_{H^1(\Omega)}$$

Upper and Lower Bounds

$$\|\text{Residual}[u_h]\| := \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \left\{ \int_\Omega fw - \int_\Omega \nabla u_h \cdot \nabla w \right\} = \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla (u - u_h) \cdot \nabla w$$

Upper and Lower Bounds

$$\|\text{Residual}[u_h]\| := \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \left\{ \int_\Omega fw - \int_\Omega \nabla u_h \cdot \nabla w \right\} = \sup_{\substack{w \in H^1(\Omega) \\ \|w\|_{H^1(\Omega)}=1}} \int_\Omega \nabla (u - u_h) \cdot \nabla w$$

$$\|\text{Residual}[u_h]\| \leq \|u - u_h\|_{H^1(\Omega)} \leq (1 + c_\Omega^2) \|\text{Residual}[u_h]\|$$

Element Decomposition of Residual

$$\int_\Omega fw - \int_\Omega \nabla u_h \cdot \nabla w = \int_\Omega fw - \sum_{K \epsilon \mathcal{T}} \int_K \nabla u_h \cdot \nabla w$$

Element Decomposition of Residual

$$\int_\Omega fw - \int_\Omega \nabla u_h \cdot \nabla w = \int_\Omega fw - \sum_{K \epsilon \mathcal{T}} \int_K \nabla u_h \cdot \nabla w$$

$$= \int_\Omega fw + \sum_{K \epsilon \mathcal{T}} \left\{ \int_K \Delta u_h w - \int_{\partial K} n_K \cdot \nabla u_h \; w \right\}$$

Element Decomposition of Residual

$$\begin{aligned}\int_\Omega fw - \int_\Omega \nabla u_h \cdot \nabla w &= \int_\Omega fw - \sum_{K\epsilon\mathcal{T}} \int_K \nabla u_h \cdot \nabla w \\ &= \int_\Omega fw + \sum_{K\epsilon\mathcal{T}} \left\{ \int_K \Delta u_h w - \int_{\partial K} n_K \cdot \nabla u_h \, w \right\} \\ &= \sum_{K\epsilon\mathcal{T}} \int_K \left(f + \Delta u_h\right) w - \sum_{E\epsilon\varepsilon_\Omega} \int_E \mathbb{J}_E \left(n_e \cdot \nabla u_h\right) w,\end{aligned}$$

Element Decomposition of Residual

$$\begin{aligned}\int_\Omega fw - \int_\Omega \nabla u_h \cdot \nabla w &= \int_\Omega fw - \sum_{K\epsilon\mathcal{T}} \int_K \nabla u_h \cdot \nabla w \\ &= \int_\Omega fw + \sum_{K\epsilon\mathcal{T}} \left\{ \int_K \Delta u_h w - \int_{\partial K} n_K \cdot \nabla u_h \, w \right\} \\ &= \sum_{K\epsilon\mathcal{T}} \int_K \left(f + \Delta u_h\right) w - \sum_{E\epsilon\varepsilon_\Omega} \int_E \mathbb{J}_E \left(n_e \cdot \nabla u_h\right) w,\end{aligned}$$

with ε_Ω the edges with at least on vertex in the interior of Ω, n_E the unit exterior normal to edge E and $\mathbb{J}_E(\cdot)$ the jump across E in direction n_E,

$$\mathbb{J}_E(n_E \cdot \nabla u_h)(x) := n_E \cdot \left\{ \lim_{t\to 0^+} \nabla u_h(x + tn_E) - \lim_{t\to 0^-} \nabla u_h(x - tn_E) \right\}.$$

Residual Estimator

$$\eta_{R,K} := \left\{ h_K^2 \| f_K + \Delta u_h \|_K^2 + \frac{1}{2} \sum_{E \epsilon \varepsilon_K \cap \varepsilon_\Omega} h_E \| \mathbb{J}_E \left(n_E \cdot \nabla u_h \right) \|_E^2 \right\}^{1/2}$$

Residual Estimator

$$\eta_{R,K} := \left\{ h_K^2 \| f_K + \Delta u_h \|_K^2 + \frac{1}{2} \sum_{E \epsilon \varepsilon_K \cap \varepsilon_\Omega} h_E \| \mathbb{J}_E \left(n_E \cdot \nabla u_h \right) \|_E^2 \right\}^{1/2}$$

$$\| u - u_h \|_{H^1(\Omega)} \leq c^* \left\{ \sum_{K \epsilon \mathcal{T}} \eta_{R,K}^2 + \sum_{K \epsilon \mathcal{T}} h_K^2 \| f - f_K \|_K^2 \right\}^{1/2}$$

Residual Estimator

$$\eta_{R,K} := \left\{ h_K^2 \| f_K + \Delta u_h \|_K^2 + \frac{1}{2} \sum_{E \epsilon \varepsilon_K \cap \varepsilon_\Omega} h_E \| \mathbb{J}_E \left(n_E \cdot \nabla u_h \right) \|_E^2 \right\}^{1/2}$$

$$\| u - u_h \|_{H^1(\Omega)} \leq c^* \left\{ \sum_{K \epsilon \mathcal{T}} \eta_{R,K}^2 + \sum_{K \epsilon \mathcal{T}} h_K^2 \| f - f_K \|_K^2 \right\}^{1/2}$$

$$\eta_{R,K} \leq c_* \left\{ \| u - u_h \|_{H^1(\omega_K)}^2 + \sum_{\substack{K' \epsilon \mathcal{T} \\ \varepsilon_K \cap \varepsilon_{K'} \neq \emptyset}} h_{K'}^2 \| f - f_{K'} \|_{H^1(K')}^2 \right\}^{1/2}$$

Outline

Averaging Techniques

Consider an approximation Gu_h to ∇u_h such that

$$\|Gu_h - \nabla u_h\| \leq \beta \|\nabla u - \nabla u_h\|, \quad 0 < \beta < 1$$

Averaging Techniques

Consider an approximation Gu_h to ∇u_h such that

$$\|Gu_h - \nabla u_h\| \leq \beta \|\nabla u - \nabla u_h\|, \quad 0 < \beta < 1$$

$$\Downarrow$$

$$\frac{1}{1+\beta}\|Gu_h - \nabla u_h\| \leq \|\nabla u - \nabla u_h\| \leq \frac{1}{1-\beta}\|Gu_h - \nabla u_h\|$$

Averaging Techniques

Consider an approximation Gu_h to ∇u_h such that

$$\|Gu_h - \nabla u_h\| \leq \beta \|\nabla u - \nabla u_h\|, \quad 0 < \beta < 1$$

$$\Downarrow$$

$$\frac{1}{1+\beta}\|Gu_h - \nabla u_h\| \leq \|\nabla u - \nabla u_h\| \leq \frac{1}{1-\beta}\|Gu_h - \nabla u_h\|$$

$$Gu_h(x) = \sum_{\substack{K \in \mathcal{T} \\ x \in \mathcal{N}_K}} \frac{|K|}{|\omega_x|} \nabla u_h]_K$$

Averaging Techniques

Consider an approximation Gu_h to ∇u_h such that

$$\|Gu_h - \nabla u_h\| \leq \beta \|\nabla u - \nabla u_h\|, \quad 0 < \beta < 1$$

$$\Downarrow$$

$$\frac{1}{1+\beta}\|Gu_h - \nabla u_h\| \leq \|\nabla u - \nabla u_h\| \leq \frac{1}{1-\beta}\|Gu_h - \nabla u_h\|$$

$$Gu_h(x) = \sum_{\substack{K \in \mathcal{T} \\ x \in \mathcal{N}_K}} \frac{|K|}{|\omega_x|} \nabla u_h]_K \Rightarrow \begin{cases} \eta_{Z,K} = \|Gu_h - \nabla u_h\|_K \\ \eta_Z = \left\{ \sum_{K \in \mathcal{T}} \eta_{Z,K}^2 \right\}^{1/2} \end{cases}$$

Outline

Solution of Auxiliary Local Problems

We want to solve a local higher order finite element problem and compare the solutions.

Solution of Auxiliary Local Problems

We want to solve a local higher order finite element problem and compare the solutions.

- **Vertices:** For a fixed vertex x, find a solution $u_x \in \tilde{H}_h(\mathcal{T}) \supset H_h(\mathcal{T})$ corresponding to

$$\begin{cases} -\Delta\varphi = f & \text{in } \omega_x \\ \varphi = u_h & \text{on } \partial\omega_x \end{cases}$$

Solution of Auxiliary Local Problems

We want to solve a local higher order finite element problem and compare the solutions.

- **Vertices:** For a fixed vertex x, find a solution $u_x \in \tilde{H}_h(\mathcal{T}) \supset H_h(\mathcal{T})$ corresponding to

$$\begin{cases} -\Delta\varphi = f & \text{in } \omega_x \\ \varphi = u_h & \text{on } \partial\omega_x \end{cases} \Rightarrow \eta_{D,x} = \|\nabla(u_x - u_h)\|_{\omega_x}$$

Solution of Auxiliary Local Problems

We want to solve a local higher order finite element problem and compare the solutions.

- **Vertices:** For a fixed vertex x, find a solution $u_x \in \tilde{H}_h(\mathcal{T}) \supset H_h(\mathcal{T})$ corresponding to

$$\begin{cases} -\Delta\varphi = f & \text{in } \omega_x \\ \varphi = u_h & \text{on } \partial\omega_x \end{cases} \Rightarrow \eta_{D,x} = \|\nabla(u_x - u_h)\|_{\omega_x}$$

- **Elements:** For a fixed element K, find a solution $u_K \in \tilde{H}_h(\mathcal{T}) \supset H_h(\mathcal{T})$ corresponding to

$$\begin{cases} -\Delta\varphi = f & \text{in } \omega_K \\ \varphi = u_h & \text{on } \partial\omega_K \end{cases}$$

Solution of Auxiliary Local Problems

We want to solve a local higher order finite element problem and compare the solutions.

- **Vertices:** For a fixed vertex x, find a solution $u_x \in \tilde{H}_h(\mathcal{T}) \supset H_h(\mathcal{T})$ corresponding to

$$\begin{cases} -\Delta\varphi = f & \text{in } \omega_x \\ \varphi = u_h & \text{on } \partial\omega_x \end{cases} \Rightarrow \eta_{D,x} = \|\nabla(u_x - u_h)\|_{\omega_x}$$

- **Elements:** For a fixed element K, find a solution $u_K \in \tilde{H}_h(\mathcal{T}) \supset H_h(\mathcal{T})$ corresponding to

$$\begin{cases} -\Delta\varphi = f & \text{in } \omega_K \\ \varphi = u_h & \text{on } \partial\omega_K \end{cases} \Rightarrow \eta_{D,K} = \|\nabla(u_K - u_h)\|_{\omega_K}$$

Outline

Hierarchical Base Error Estimates

Consider a more accurate finite element space Y_h, such that $H_h(\mathcal{T}) \subset Y_h \subset H_0^1(\Omega)$.

Hierarchical Base Error Estimates

Consider a more accurate finite element space Y_h, such that $H_h(\mathcal{T}) \subset Y_h \subset H_0^1(\Omega)$.

- Find $w_h \in Y_h$ such that

$$\int_\Omega \nabla w_h \cdot \nabla v_h = \int_\Omega f v_h, \quad \forall v_h \in Y_h$$

Hierarchical Base Error Estimates

Consider a more accurate finite element space Y_h, such that $H_h(\mathcal{T}) \subset Y_h \subset H_0^1(\Omega)$.

- Find $w_h \in Y_h$ such that

$$\int_\Omega \nabla w_h \cdot \nabla v_h = \int_\Omega f v_h, \quad \forall v_h \in Y_h \Rightarrow \eta_H = \|\nabla(w_h - u_h)\|$$

Hierarchical Base Error Estimates

Consider a more accurate finite element space Y_h, such that $H_h(\mathcal{T}) \subset Y_h \subset H_0^1(\Omega)$.

- Find $w_h \in Y_h$ such that

$$\int_\Omega \nabla w_h \cdot \nabla v_h = \int_\Omega f v_h, \quad \forall v_h \in Y_h \Rightarrow \eta_H = \|\nabla(w_h - u_h)\|$$

- Let $Y_h = H_h(\mathcal{T}) \oplus Z_h$ and find $z_h \in Z_h$ such that

$$\int_\Omega \nabla z_h \cdot \nabla \xi_h = \int_\Omega f \xi_h - \int_\Omega \nabla u_h \cdot \nabla \xi_h, \quad \forall \xi_h \in Z_h$$

Hierarchical Base Error Estimates

Consider a more accurate finite element space Y_h, such that $H_h(\mathcal{T}) \subset Y_h \subset H_0^1(\Omega)$.

- Find $w_h \in Y_h$ such that

$$\int_\Omega \nabla w_h \cdot \nabla v_h = \int_\Omega f v_h, \quad \forall v_h \in Y_h \Rightarrow \eta_H = \|\nabla(w_h - u_h)\|$$

- Let $Y_h = H_h(\mathcal{T}) \oplus Z_h$ and find $z_h \in Z_h$ such that

$$\int_\Omega \nabla z_h \cdot \nabla \xi_h = \int_\Omega f \xi_h - \int_\Omega \nabla u_h \cdot \nabla \xi_h, \quad \forall \xi_h \in Z_h \Rightarrow \eta_H = \|\nabla z_h\|$$

Outline

H(div)-Lifting

Find a vector field ρ_h such that

$$\begin{cases} -\mathrm{div}(\rho_h) = f & \text{on every } K \in \mathcal{T} \\ \mathbb{J}_E(n_E \cdot \rho_h) = -\mathbb{J}_E(n_E \cdot \nabla u_h) & \text{on } E \in \varepsilon_\Omega \end{cases}$$

H(div)-Lifting

Find a vector field ρ_h such that

$$\begin{cases} -\mathrm{div}(\rho_h) = f & \text{on every } K \in \mathcal{T} \\ \mathbb{J}_E(n_E \cdot \rho_h) = -\mathbb{J}_E(n_E \cdot \nabla u_h) & \text{on } E \in \varepsilon_\Omega \end{cases}$$

$$\Updownarrow$$

Find $\rho = \rho_h + \nabla u_h \in H(\mathrm{div}; \Omega)$ such that

$$-\mathrm{div}\rho = f \quad \text{in } \Omega$$

H(div)-Lifting

Find a vector field ρ_h such that

$$\begin{cases} -\mathrm{div}(\rho_h) = f & \text{on every } K \in \mathcal{T} \\ \mathbb{J}_E(n_E \cdot \rho_h) = -\mathbb{J}_E(n_E \cdot \nabla u_h) & \text{on } E \in \varepsilon_\Omega \end{cases}$$

$$\Updownarrow$$

Find $\rho = \rho_h + \nabla u_h \in H(\mathrm{div}; \Omega)$ such that

$$-\mathrm{div}\rho = f \quad \text{in } \Omega$$

$$\eta_{H(\mathrm{div})} = \|\rho_h\|$$

Outline

Asymptotic Exactness

Let $\sigma_\eta := \frac{\text{estimated error}}{\text{actual error}}$ be the **efficiency index**, with $h = \max_{K \epsilon \mathcal{T}} h_K$ the mesh size.

Asymptotic Exactness

Let $\sigma_\eta := \frac{\text{estimated error}}{\text{actual error}}$ be the **efficiency index**, with $h = \max_{K \epsilon \mathcal{T}} h_K$ the mesh size.

An estimator η is said to be

- **efficient** if σ_η and $(\sigma_\eta)^{-1}$ are bounded for all mesh sizes h.
- **asymptotically exact** if $\sigma_\eta \to 1$ as $h \to 0$.

Outline

Mesh Refinement

A partition $\mathcal{T}$ must satisfy

Mesh Refinement

A partition $\mathcal{T}$ must satisfy

- $\Omega = \bigcup K, K \in \mathcal{T}$

Mesh Refinement

A partition $\mathcal{T}$ must satisfy

- $\Omega = \bigcup K, K \in \mathcal{T}$
- Affine equivalence: Each $K \in \mathcal{T}$ is a triangle

Mesh Refinement

A partition $\mathcal{T}$ must satisfy

- $\Omega = \bigcup K, K \in \mathcal{T}$
- Affine equivalence: Each $K \in \mathcal{T}$ is a triangle
- Admissibility: For any two elements in $\mathcal{T}$ they are either disjoint, share a vertex or share a complete edge.

Mesh Refinement

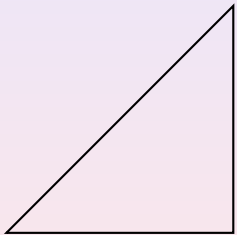
A partition $\mathcal{T}$ must satisfy

- $\Omega = \bigcup K, K \in \mathcal{T}$
- Affine equivalence: Each $K \in \mathcal{T}$ is a triangle
- Admissibility: For any two elements in $\mathcal{T}$ they are either disjoint, share a vertex or share a complete edge.
- Shape-regularity: For any element the the ratio of its diameter and the diameter of the largest inscribed ball is uniformly bounded.

Outline

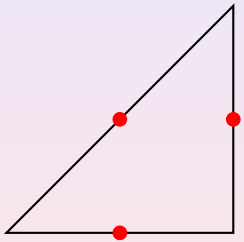
Uniform Refinement

We apply a regular refinement to every element in our partition by connecting the midpoints of each side of a triangle



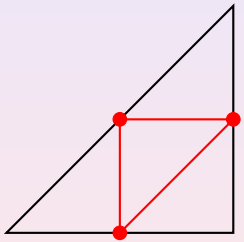
Uniform Refinement

We apply a regular refinement to every element in our partition by connecting the midpoints of each side of a triangle



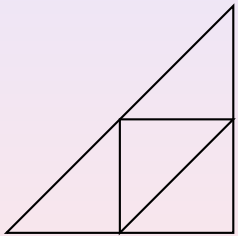
Uniform Refinement

We apply a regular refinement to every element in our partition by connecting the midpoints of each side of a triangle



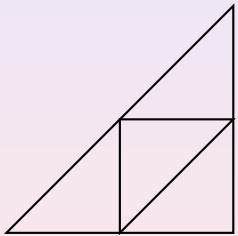
Uniform Refinement

We apply a regular refinement to every element in our partition by connecting the midpoints of each side of a triangle



Uniform Refinement

We apply a regular refinement to every element in our partition by connecting the midpoints of each side of a triangle



Simple but inefficient

Outline

Adaptive Refinement

- Uniform refinement requires significant resources to reduce error.

Adaptive Refinement

- Uniform refinement requires significant resources to reduce error.
- Rather than refine the whole mesh focus instead on areas of the mesh which contribute most to the error.

Adaptive Refinement

- Uniform refinement requires significant resources to reduce error.
- Rather than refine the whole mesh focus instead on areas of the mesh which contribute most to the error.
- We can use local error estimators, such as the residual a posteriori estimator, to find areas to refine.

General Algorithm

Require: PDE data, tolerance ϵ
Provide: Approximate solution to PDE with error less than ϵ
Construct an initial admissible partition $\mathcal{T}_0$;
for *k=0,1,...* **do**
  Solve the discrete problem corresponding to $\mathcal{T}_k$;
  foreach $K \in \mathcal{T}_k$ **do**
    Compute an estimate η_K of the error on K;
  end
  $\eta \leftarrow \{\sum_{K \in \mathcal{T}_k} \eta_K^2\}^{1/2}$;
  if $\eta \leq \epsilon$ **then**
    stop;
  end
  Based on $(\eta_K)_K$ determine a set $\widetilde{\mathcal{T}}_k$ of elements to be refined;
  Based on $\widetilde{\mathcal{T}}_{k \in \mathcal{T}}$ determine an admissible refinement $\mathcal{T}_{k+1}$ of $\mathcal{T}_k$;
end

Marking Algorithms

Maximum Strategy

Require: Partition $\mathcal{T}$, error estimates $(n_K)_{K\epsilon\mathcal{T}}$, threshold $\theta \in (0,1)$
Provide: Subset $\widetilde{\mathcal{T}}$ of marked elements that should be refined

$\widetilde{\mathcal{T}} \leftarrow \emptyset$;
$\eta \leftarrow \max_{K\epsilon\mathcal{T}} \eta_K$;
foreach $K \in \mathcal{T}$ **do**
  if $\eta_K \geq \theta\eta$ **then**
    $\widetilde{\mathcal{T}} \leftarrow \widetilde{\mathcal{T}} \cup \{K\}$;
  end
end

Equilibration Strategy

Require: Partition $\mathcal{T}$, error estimates $(n_K)_{K\epsilon\mathcal{T}}$, threshold $\theta \in (0,1)$
Provide: Subset $\widetilde{\mathcal{T}}$ of marked elements that should be refined

$\widetilde{\mathcal{T}} \leftarrow \emptyset$, $\Sigma \leftarrow 0$, $\Phi \leftarrow \sum_{K\epsilon\mathcal{T}_k} \eta_K^2$;
$\eta \leftarrow \max_{K\epsilon\mathcal{T}} \eta_K$;
while $\Sigma < \theta\Phi$ **do**
  $\eta \leftarrow \max_{K\epsilon\mathcal{T}\setminus\widetilde{\mathcal{T}}} \eta_K$;
  foreach $K \in \mathcal{T}\setminus\widetilde{\mathcal{T}}$ **do**
    if $\eta_K = \eta$ **then**
      $\widetilde{\mathcal{T}} \leftarrow \widetilde{\mathcal{T}} \cup \{K\}$, $\Sigma \leftarrow \Sigma + \eta_K^2$;
    end
  end
end

Marking Algorithms

- Both strategies yield similar results.

Marking Algorithms

- Both strategies yield similar results.
- For the maximum strategy larger θ marks fewer elements.

Marking Algorithms

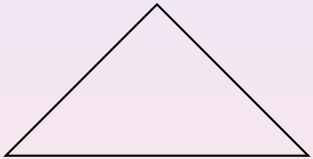
- Both strategies yield similar results.
- For the maximum strategy larger θ marks fewer elements.
- For the equilibration strategy smaller θ marks fewer elements.

Marking Algorithms

- Both strategies yield similar results.
- For the maximum strategy larger θ marks fewer elements.
- For the equilibration strategy smaller θ marks fewer elements.
- $\theta \approx 0.5$ is a good choice for both.

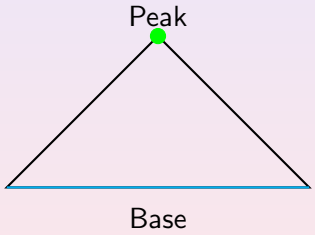
Element Refinement

We will refine an element via bisection by adding a new edge connecting its peak to the mid point of its base.



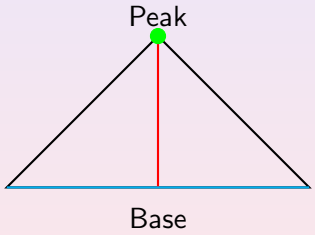
Element Refinement

We will refine an element via bisection by adding a new edge connecting its peak to the mid point of its base.



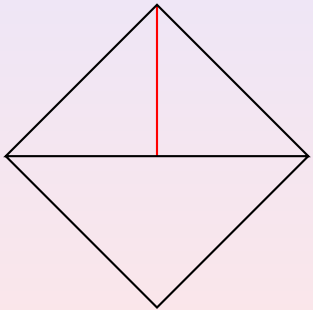
Element Refinement

We will refine an element via bisection by adding a new edge connecting its peak to the mid point of its base.



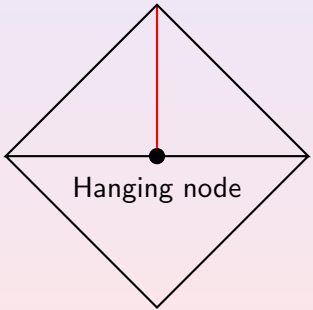
Element Refinement

If the base of the element did not lie on the boundary then we will also need to refine the neighbouring element to ensure the resulting partition is admissible.



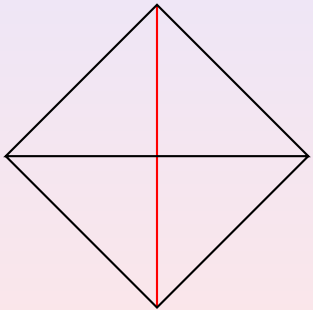
Element Refinement

If the base of the element did not lie on the boundary then we will also need to refine the neighbouring element to ensure the resulting partition is admissible.



Element Refinement

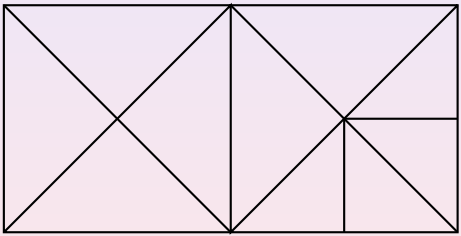
If the base of the element did not lie on the boundary then we will also need to refine the neighbouring element to ensure the resulting partition is admissible.



Marked Edge Bisection

Edge Marking Algorithm

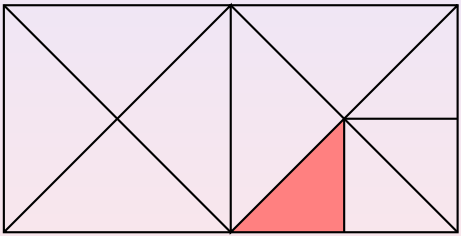
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

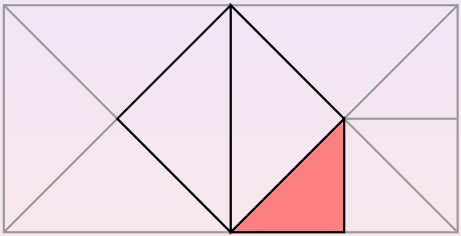
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

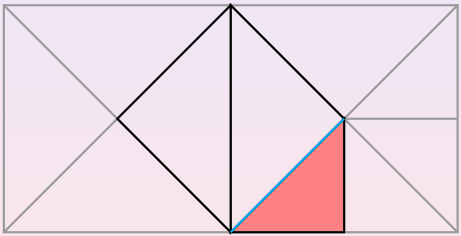
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

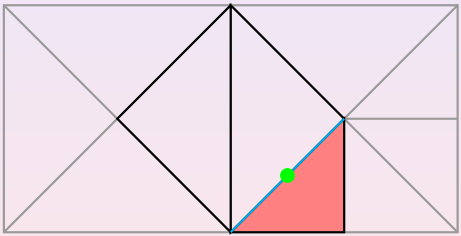
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

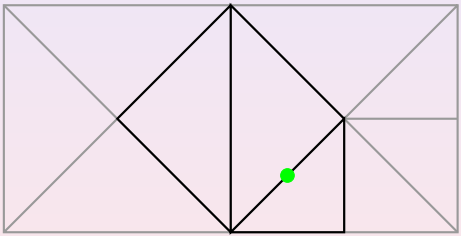
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

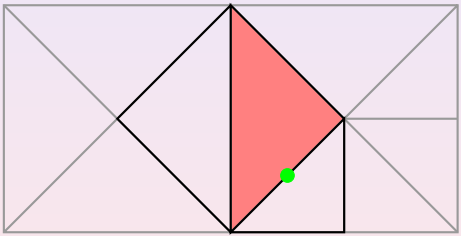
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

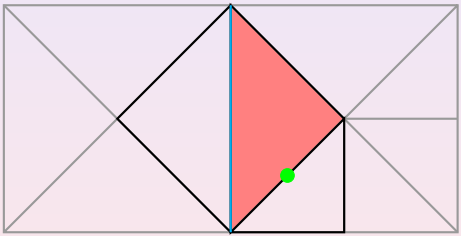
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

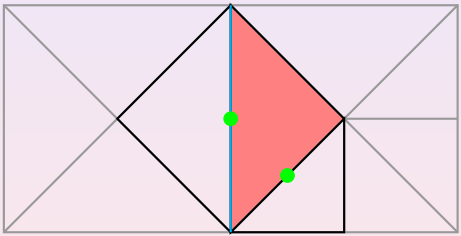
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

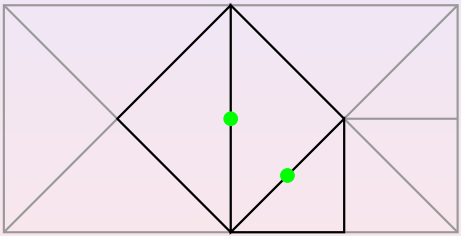
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

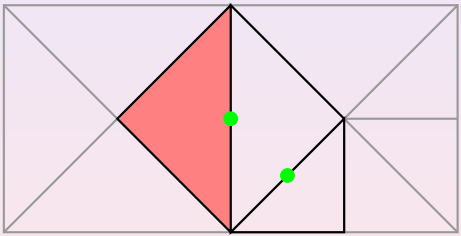
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

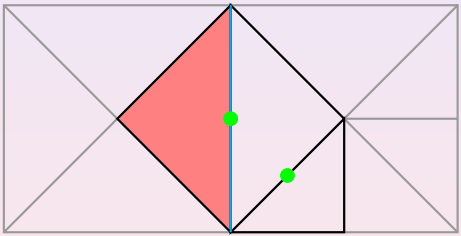
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

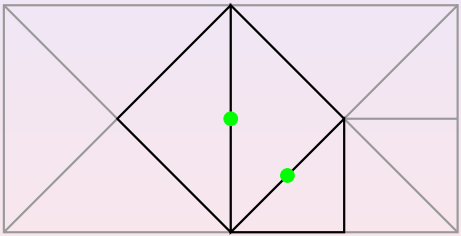
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Edge Marking Algorithm

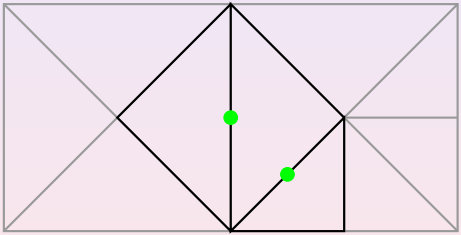
Require: Partition $\mathcal{T}$, marked elements $\widetilde{\mathcal{T}}$
Provide: Subset $\widetilde{E}$ of marked edges
foreach $K \in \widetilde{\mathcal{T}}$ **do**
  $\widetilde{K} \leftarrow K$;
  repeat
    $E_{\widetilde{K}} \leftarrow$ base of $\widetilde{K}$;
    if $E_{\widetilde{K}} \notin \widetilde{E}$ **then**
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{E_{\widetilde{K}}\}$;
      if $\widetilde{K}$ *has neighbour with edge* $E_{\widetilde{K}}$ **then**
        $\widetilde{K} \leftarrow$ neighbour with edge $E_{\widetilde{K}}$;
      else
        stop;
      end
    else
      stop;
    end
  until *stopped*;
end



Marked Edge Bisection

Bisection Algorithm

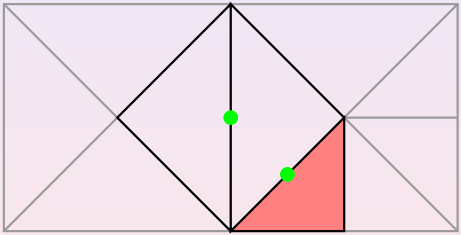
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

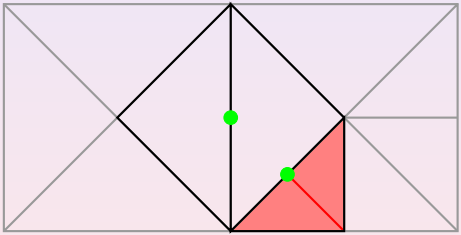
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

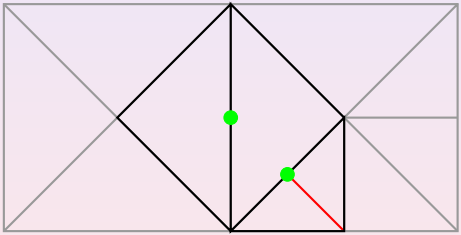
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

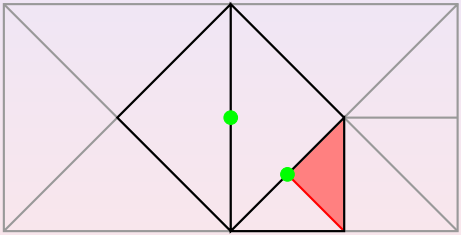
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

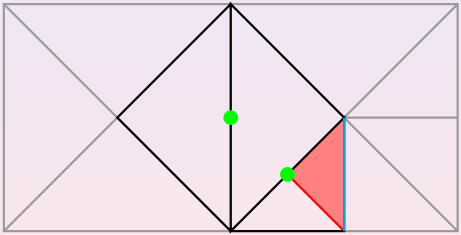
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

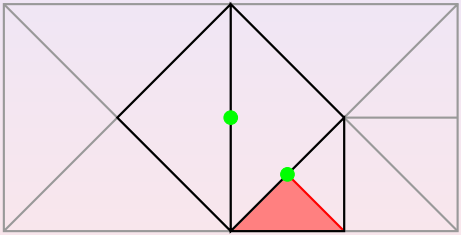
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

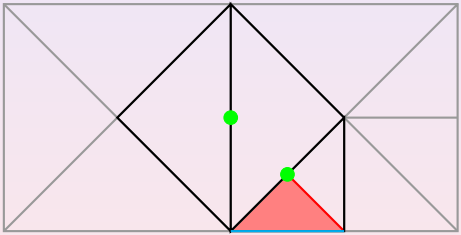
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

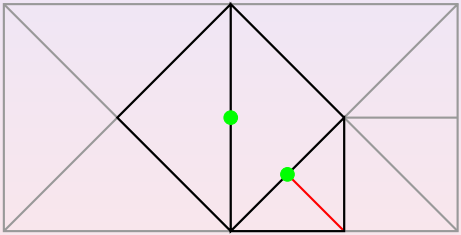
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

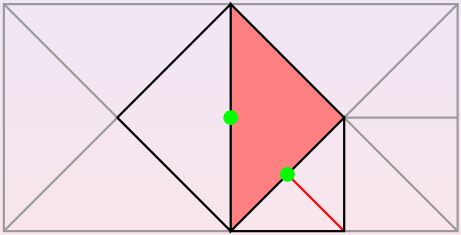
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

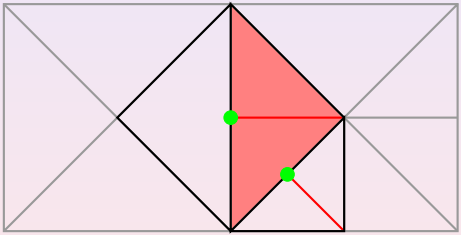
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

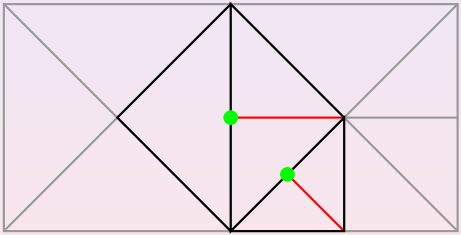
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

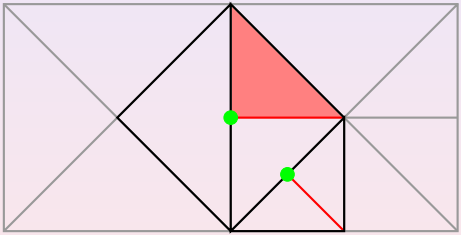
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

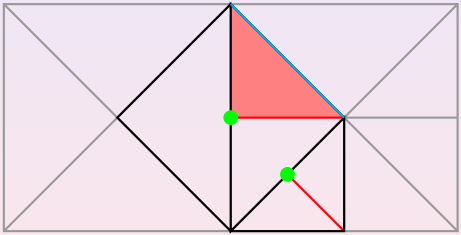
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

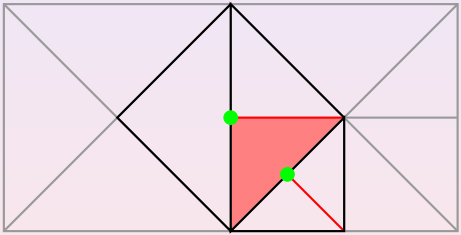
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

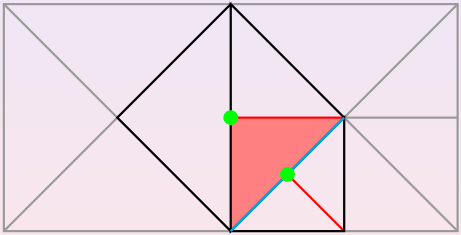
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

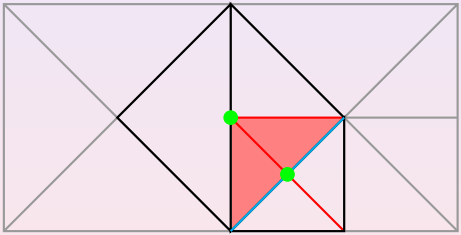
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

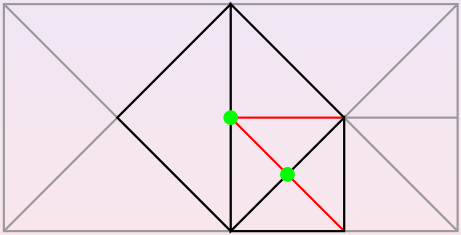
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

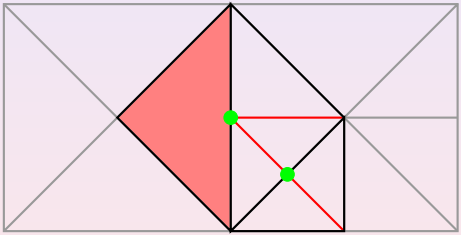
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

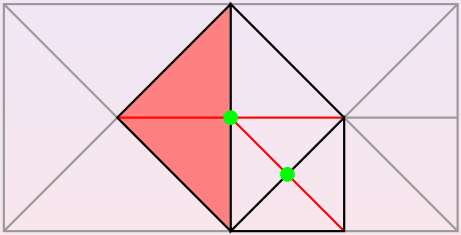
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

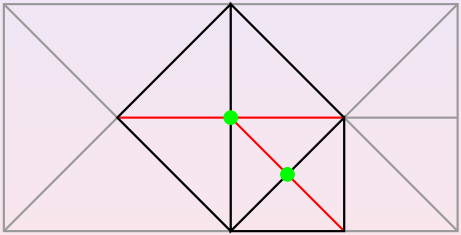
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

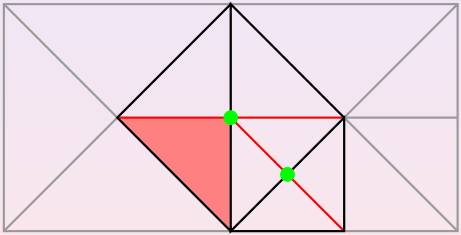
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

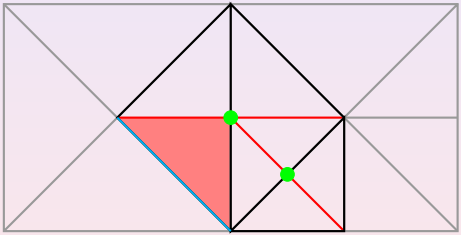
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

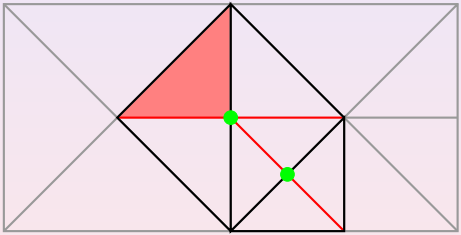
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

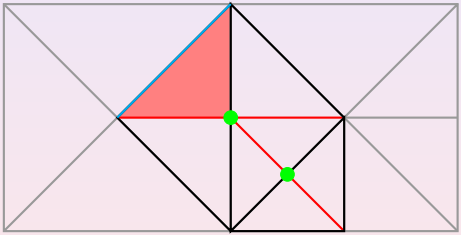
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

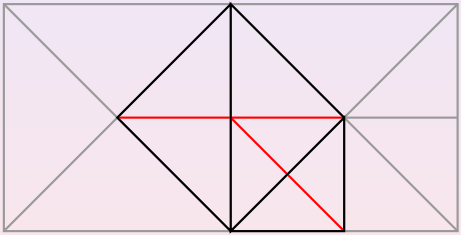
Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Marked Edge Bisection

Bisection Algorithm

Require: Partition $\mathcal{T}_k$, marked edges $\widetilde{E}$
Provide: Refined partition $\mathcal{T}_{k+1}$
$R \leftarrow \emptyset, N \leftarrow \emptyset$;
foreach $K \in \mathcal{T}_k$ **do**
  $E_k \leftarrow$ base of K;
  if $E_k \in \widetilde{E}$ **then**
    $K_L, K_R \leftarrow$ bisection K;
    $R \leftarrow R \cup \{K\}$;
    if *base* $K_L \in \widetilde{E}$ **then** [Also for K_R]
      $K_{L1}, K_{L2} \leftarrow$ bisection K_L;
      $N \leftarrow N \cup \{K_{L1}\} \cup \{K_{L2}\}$;
    else
      $N \leftarrow N \cup \{K_L\}$;
    end
  end
end
$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \backslash R \cup N$;



Outline

Example

We consider the Poisson equation

$$-\Delta u = 0 \text{ in } \Omega,$$
$$u = g \text{ on } \delta\Omega,$$

where $\Omega = (1,-1)^2 \backslash (0,1) \times (-1,0)$ and g is chosen such that the exact solution in polar co-ordinates is given by

$$u = r^{\frac{2}{3}} \sin\left(\frac{3}{2}\pi\theta\right)$$

Comparison

- We first obtain a solution numerically using uniform refinement where we refine until we run out of memory.

Comparison

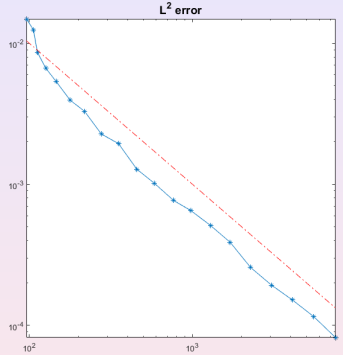
- We first obtain a solution numerically using uniform refinement where we refine until we run out of memory.
- We then obtain another numerical solution this time using adaptive refinement, stopping once we reach have approximately the same relative error as in the uniform case.

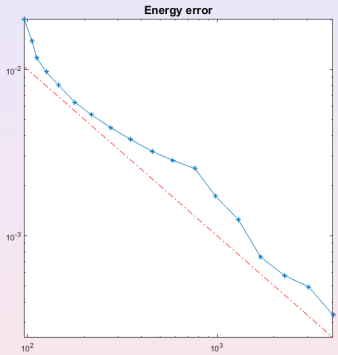
Comparison

- We first obtain a solution numerically using uniform refinement where we refine until we run out of memory.
- We then obtain another numerical solution this time using adaptive refinement, stopping once we reach have approximately the same relative error as in the uniform case.

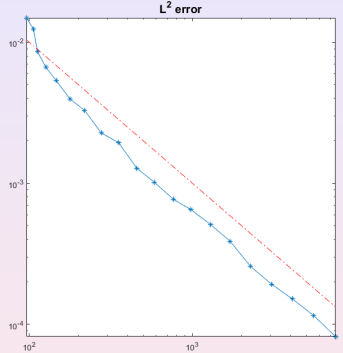
	Uniform	Adaptive
Unknowns	2945	718
Elements	6144	1508
Error(%)	1.3	1.5

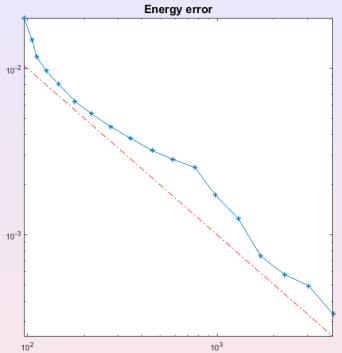
Adaptive Error





Adaptive Error





It can be shown that

$$\|\nabla(u - u_i)\| \leq \beta^i \|\nabla(u - u_0)\|$$

where u_i is the numerical solution corresponding to $\mathcal{T}_i$ and $\beta \in (0, 1)$.

Outline

Coarsening

To obtain an optimal partition we may also want to coarsen the mesh

Coarsening

To obtain an optimal partition we may also want to coarsen the mesh

- For time dependent problems critical regions may move through space in the course of time

Coarsening

To obtain an optimal partition we may also want to coarsen the mesh

- For time dependent problems critical regions may move through space in the course of time
- For elliptic problems it may also be beneficial to coarsen in regions where the error is small.