Discretization of the fractional Laplacian using finite element methods and a posteriori error estimation

Raphaël Bulle

University of Luxembourg
Université de Bourgogne Franche-Comté

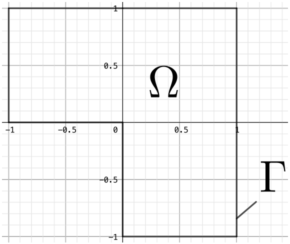
February 23, 2021

Table of contents

Table of contents

A finite element method

Example of elliptic PDE



Let $\gamma \in \mathbb{R}^{+,*}$ and $f \in L^2(\Omega)$. We are looking for u (with sufficient regularity) such that

$$\begin{aligned} u - \gamma \Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma, \end{aligned}$$

where $\Delta u(x_1, x_2) := (\partial^2_{xx} + \partial^2_{yy})u$.

A finite element method

Example of elliptic PDE

Instead of looking at the strong formulation: find u satisfying

$$\begin{aligned} u - \gamma \Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma, \end{aligned}$$

A finite element method

Example of elliptic PDE

Instead of looking at the strong formulation: find u satisfying

$$\begin{aligned} u - \gamma \Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma, \end{aligned}$$

we consider the weak formulation: find a function u in $H_0^1(\Omega)$ such that

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

where $H_0^1(\Omega)$ is the Sobolev space of functions v in $L^2(\Omega)$ vanishing on Γ and with $\partial_x v$ and $\partial_y v$ in $L^2(\Omega)$.

A finite element method

Discretization

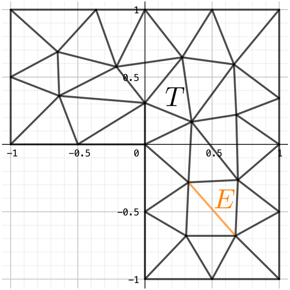
Goal: Compute a numerical approximation to u, solution to

$$\int_\Omega uv + \gamma \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \quad \forall v \in H_0^1(\Omega).$$

A finite element method

Discretization

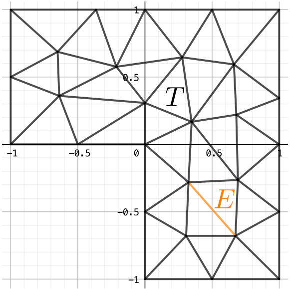
Let $\mathcal{T} = \{T\}$ be a mesh on Ω, of edges $\mathcal{E} = \{E\}$.



A finite element method

Discretization

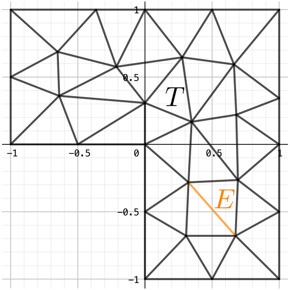
Let $\mathcal{T} = \{T\}$ be a mesh on Ω, of edges $\mathcal{E} = \{E\}$.
We consider $V^1 \subset H_0^1(\Omega)$ the space of continuous piecewise linear polynomial functions over $\mathcal{T}$, vanishing on Γ.



A finite element method

Discretization

Let $\mathcal{T} = \{T\}$ be a mesh on Ω, of edges $\mathcal{E} = \{E\}$.
We consider $V^1 \subset H_0^1(\Omega)$ the space of continuous piecewise linear polynomial functions over $\mathcal{T}$, vanishing on Γ.



Let $u_1 \in V^1$ be the solution to

$$\int_\Omega u_1 v_1 + \gamma \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$

A finite element method

Discretization

Original problem:

$$\int_\Omega uv + \gamma \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \quad \forall v \in H_0^1(\Omega).$$

A finite element method

Discretization

Original problem:

$$\int_\Omega uv + \gamma \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \quad \forall v \in H_0^1(\Omega).$$

Linear Lagrange finite element discretization:

$$\int_\Omega u_1 v_1 + \gamma \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$

A finite element method

Discretization

Original problem:

$$\int_\Omega uv + \gamma \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \quad \forall v \in H_0^1(\Omega).$$

Linear Lagrange finite element discretization:

$$\int_\Omega u_1 v_1 + \gamma \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$

We take $u_1 \approx u$.

A finite element method

Discretization

Let's try it out!

A finite element method

Discretization

Let's try it out!
We take $\gamma = 1$, $f = 1$ and solve

$$\int_\Omega u_1 v_1 + \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$

A finite element method

Discretization

Let's try it out!

We take $\gamma = 1$, $f = 1$ and solve

$$\int_\Omega u_1 v_1 + \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$

using

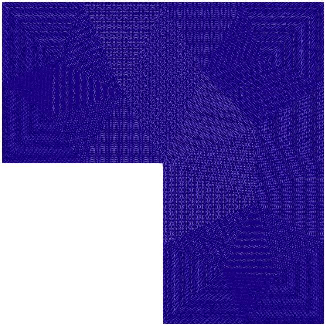
A finite element method

Discretization

Let's try it out!

We take $\gamma = 1$, $f = 1$ and solve

$$\int_\Omega u_1 v_1 + \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$



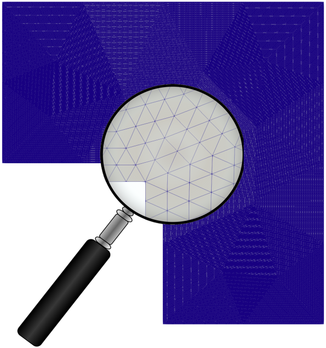
A finite element method

Discretization

Let's try it out!

We take $\gamma = 1$, $f = 1$ and solve

$$\int_\Omega u_1 v_1 + \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$



A finite element method

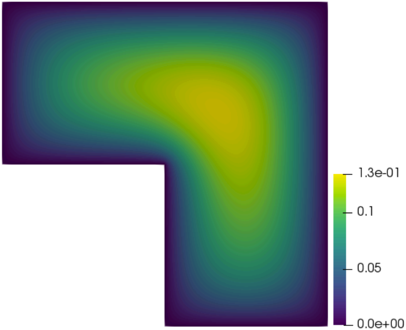
Discretization

Let's try it out!

We take $\gamma = 1$, $f = 1$ and solve

$$\int_\Omega u_1 v_1 + \int_\Omega \nabla u_1 \cdot \nabla v_1 = \int_\Omega f v_1 \quad \forall v_1 \in V^1.$$

(Linear system dimension: 66049.)



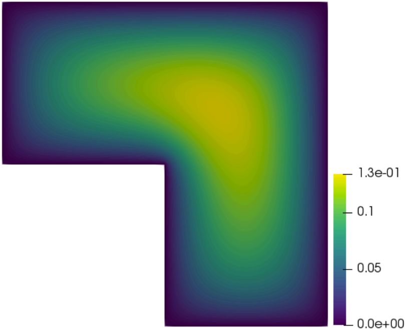


Table of contents

Error estimation

A priori error estimation

What can we say about the discretization error ?

Error estimation

A priori error estimation

What can we say about the discretization error ?
We quantify the error $e := u - u_1$ using the energy norm

$$\|e\|_\gamma := \left(\int_\Omega e^2 + \gamma \int_\Omega \nabla e \cdot \nabla e \right)^{1/2}.$$

Error estimation

A priori error estimation

What can we say about the discretization error ?

We quantify the error $e := u - u_1$ using the energy norm

$$\|e\|_\gamma := \left(\int_\Omega e^2 + \gamma \int_\Omega \nabla e \cdot \nabla e\right)^{1/2}.$$

A priori error estimation

Let Ω be an open subset of $\mathbb{R}^2$ of polygonal boundary and let $\{\mathcal{T}_h\}_{h>0}$ be a shape-regular family of conformal meshes of Ω. Then,

$$\lim_{h \to 0} \|e\|_\gamma = 0.$$

Moreover, if $u \in H^2(\Omega)$ there exists c_γ such that

$$\|e\|_\gamma \leqslant c_\gamma h |u|_{H^2}.$$

$H^2(\Omega) := \left\{v \in L^2(\Omega),\ \partial^\alpha u \in L^2(\Omega),\ \alpha \in \mathbb{N}^2,\ |\alpha| \leqslant 2\right\}$ is an Hilbert space on which we define the semi-norm, $|u|^2_{H^2} := \|\partial^2_{xx} u\|^2_{L^2} + \|\partial^2_{yy} u\|^2_{L^2} + \|\partial^2_{xy} u\|^2_{L^2}$.

Error estimation

A priori error estimation

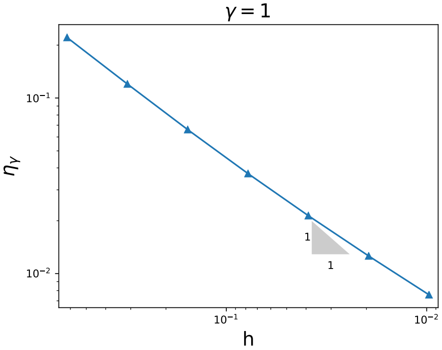
What can we say about the discretization error ?

We quantify the error $e := u - u_1$ using the energy norm

$$\|e\|_\gamma := \left(\int_\Omega e^2 + \gamma \int_\Omega \nabla e \cdot \nabla e\right)^{1/2}.$$

A priori error estimation

$$\|e\|_\gamma \leqslant c_\gamma h^1 |u|_{H^2}.$$

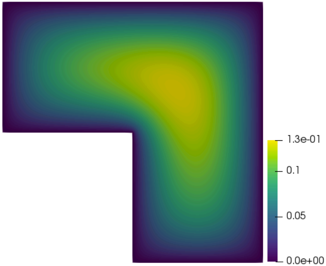


Error estimation

What happened ?

A priori error estimation

$$\|e\|_\gamma \leqslant c_\gamma h|u|_{H^2}.$$



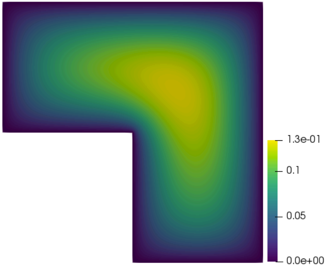
Error estimation

What happened ?

A priori error estimation

$$\|e\|_\gamma \leqslant c_\gamma h |u|_{H^2}.$$

The solution u does not belong to $H^2(\Omega)$!
∇u admits a singularity in the reentrant corner of Ω [Grisvard, 1986].



Error estimation

A posteriori error estimation

- How to deal with solutions having local features ?

Error estimation

A posteriori error estimation

- How to deal with solutions having local features ?
- How to quantify the discretization error ?

Error estimation

A posteriori error estimation

- How to deal with solutions having local features ?
- How to quantify the discretization error ?
- How to choose the size of the mesh to reach a certain tolerance ?

Error estimation

A posteriori error estimation

- How to deal with solutions having local features ?
- How to quantify the discretization error ?
- How to choose the size of the mesh to reach a certain tolerance ?

A priori error estimation	A posteriori error estimation
$\|e\|_\gamma \leqslant \widetilde{C}(u)$	$\|e\|_\gamma \approx \eta$
$\widetilde{C}(u)$ is unknown.	η is known.

Error estimation

A posteriori error estimation

Let η be an a posteriori error estimator. We would like η to be:

- computable from problem data (f, boundary conditions data...) and u_1 only,

Error estimation

A posteriori error estimation

Let η be an a posteriori error estimator. We would like η to be:

- computable from problem data (f, boundary conditions data...) and u_1 only,
- reliable i.e. there exists a constant C only depending on the mesh regularity such that

$$\|e\|_\gamma \leqslant C\eta \quad \text{with } C \text{ close to } 1,$$

Error estimation

A posteriori error estimation

Let η be an a posteriori error estimator. We would like η to be:

- computable from problem data (f, boundary conditions data...) and u_1 only,
- reliable i.e. there exists a constant C only depending on the mesh regularity such that

$$\|e\|_\gamma \leqslant C\eta \quad \text{with } C \text{ close to } 1,$$

- efficient i.e. there exists a constant c only depending on the mesh regularity such that

$$c\eta \leqslant \|e\|_\gamma \quad \text{with } c \text{ close to } 1,$$

Error estimation

A posteriori error estimation

Let η be an a posteriori error estimator. We would like η to be:

- computable from problem data (f, boundary conditions data...) and u_1 only,
- reliable i.e. there exists a constant C only depending on the mesh regularity such that

$$\|e\|_\gamma \leqslant C\eta \quad \text{with } C \text{ close to } 1,$$

- efficient i.e. there exists a constant c only depending on the mesh regularity such that

$$c\eta \leqslant \|e\|_\gamma \quad \text{with } c \text{ close to } 1,$$

- local i.e.

$$\eta = \sum_{T\in\mathcal{T}} \eta_T,$$

Error estimation

A posteriori error estimation

Let η be an a posteriori error estimator. We would like η to be:

- computable from problem data (f, boundary conditions data...) and u_1 only,
- reliable i.e. there exists a constant C only depending on the mesh regularity such that

$$\|e\|_\gamma \leqslant C\eta \quad \text{with } C \text{ close to } 1,$$

- efficient i.e. there exists a constant c only depending on the mesh regularity such that

$$c\eta \leqslant \|e\|_\gamma \quad \text{with } c \text{ close to } 1,$$

- local i.e.

$$\eta = \sum_{T \in \mathcal{T}} \eta_T,$$

- cheap to compute, ideally much less expensive than computing u_1.

Error estimation

A posteriori error estimation

Let $e := u - u_1$, we can show that e is solution to the local problem:

$$\int_T e v_T + \gamma \int_T \nabla e \cdot \nabla v_T = \int_T r_{\gamma,T} v_T + \sum_{E \in \partial T} \int_E J_{\gamma,E} v_T \quad \forall v_T \in H_0^1(T),$$

with $r_{\gamma,T} := (f - u_1 + \gamma \Delta u_1)_{|T}$ and $J_{\gamma,E} := \gamma \left[\!\!\left[\frac{\partial u_1}{\partial n} \right]\!\!\right]_E$.

Error estimation

A posteriori error estimation

Let $e := u - u_1$, we can show that e is solution to the local problem:

$$\int_T e v_T + \gamma \int_T \nabla e \cdot \nabla v_T = \int_T r_{\gamma,T} v_T + \sum_{E \in \partial T} \int_E J_{\gamma,E} v_T \quad \forall v_T \in H_0^1(T),$$

with $r_{\gamma,T} := (f - u_1 + \gamma \Delta u_1)_{|T}$ and $J_{\gamma,E} := \gamma \left[\!\!\left[\frac{\partial u_1}{\partial n} \right]\!\!\right]_E$.

We consider $V^{\mathrm{bw}}(T) \subset V^2(T)$ a particular local finite element space.

Error estimation

A posteriori error estimation

Let $e := u - u_1$, we can show that e is solution to the local problem:

$$\int_T e v_T + \gamma \int_T \nabla e \cdot \nabla v_T = \int_T r_{\gamma,T} v_T + \sum_{E \in \partial T} \int_E J_{\gamma,E} v_T \quad \forall v_T \in H_0^1(T),$$

with $r_{\gamma,T} := (f - u_1 + \gamma \Delta u_1)_{|T}$ and $J_{\gamma,E} := \gamma \left[\!\left[\frac{\partial u_1}{\partial n} \right]\!\right]_E$.

We consider $V^{\mathrm{bw}}(T) \subset V^2(T)$ a particular local finite element space.
For a cell T in $\mathcal{T}$, we define $e_T^{\mathrm{bw}} \in V^{\mathrm{bw}}(T)$ the solution to

$$\int_T e_T^{\mathrm{bw}} v_T + \gamma \int_T \nabla e_T^{\mathrm{bw}} \cdot \nabla v_T = \int_T r_{\gamma,T} v_T + \sum_{E \in \partial T} \int_E J_{\gamma,E} v_T \quad \forall v_T \in V^{\mathrm{bw}}(T),$$

Error estimation

A posteriori error estimation

The local Bank-Weiser a posteriori error estimator [Bank, Weiser, 1985] is defined by:

$$\eta_T := \|e_T^{\mathrm{bw}}\|_\gamma,$$

Error estimation

A posteriori error estimation

The local Bank-Weiser a posteriori error estimator [Bank, Weiser, 1985] is defined by:

$$\eta_T := \|e_T^{\mathrm{bw}}\|_\gamma,$$

and the global estimator by:

$$\eta_{\mathrm{bw}}^2 := \sum_{T \in \mathcal{T}} \eta_T^2.$$

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\text{bw}}$):

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],
  - ✓ without the assumption, for linear finite elements in 1D and 2D [Nochetto, 1993],

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],
  - ✓ without the assumption, for linear finite elements in 1D and 2D [Nochetto, 1993],
  - ✓ without the assumption, for linear finite elements in 3D [B., Chouly, Hale, Lozinski, 2019],

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],
  - ✓ without the assumption, for linear finite elements in 1D and 2D [Nochetto, 1993],
  - ✓ without the assumption, for linear finite elements in 3D [B., Chouly, Hale, Lozinski, 2019],
  - ✗ still an open problem for higher order finite elements.

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],
  - ✓ without the assumption, for linear finite elements in 1D and 2D [Nochetto, 1993],
  - ✓ without the assumption, for linear finite elements in 3D [B., Chouly, Hale, Lozinski, 2019],
  - ✗ still an open problem for higher order finite elements.
- efficient ($c\eta_{\mathrm{bw}} \leqslant \|e\|_\gamma$): ✓ [Bank, Weiser, 1985],

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],
  - ✓ without the assumption, for linear finite elements in 1D and 2D [Nochetto, 1993],
  - ✓ without the assumption, for linear finite elements in 3D [B., Chouly, Hale, Lozinski, 2019],
  - ✗ still an open problem for higher order finite elements.
- efficient ($c\eta_{\mathrm{bw}} \leqslant \|e\|_\gamma$): ✓ [Bank, Weiser, 1985],
- local: ✓

Error estimation

A posteriori error estimation

Is the Bank-Weiser estimator a good estimator ?

- computable from data and u_1 only: ✓
- reliable ($\|e\|_\gamma \leqslant C\eta_{\mathrm{bw}}$):
  - ✓ under a constraining assumption on the solution u [Bank, Weiser, 1985],
  - ✓ without the assumption, for linear finite elements in 1D and 2D [Nochetto, 1993],
  - ✓ without the assumption, for linear finite elements in 3D [B., Chouly, Hale, Lozinski, 2019],
  - ✗ still an open problem for higher order finite elements.
- efficient ($c\eta_{\mathrm{bw}} \leqslant \|e\|_\gamma$): ✓ [Bank, Weiser, 1985],
- local: ✓
- cheap: ✓ [Bordas, B., Chouly, Hale, Lozinski, 2020].

Table of contents

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,
4. compute the global estimator η_{bw} and check if $\eta_{\mathrm{bw}} \leqslant \varepsilon$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,
4. compute the global estimator η_{bw} and check if $\eta_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u_1^l.

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,
4. compute the global estimator η_{bw} and check if $\eta_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u_1^l.
   - ✗ continue,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,
4. compute the global estimator η_{bw} and check if $\eta_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u_1^l.
   - ✗ continue,
5. mark the cells we need to refine (e.g. each cell T for which $\eta_T \geqslant 0.9 \max_{T \in \mathcal{T}_l} \eta_T$),

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,
4. compute the global estimator η_{bw} and check if $\eta_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u_1^l.
   - ✗ continue,
5. mark the cells we need to refine (e.g. each cell T for which $\eta_T \geqslant 0.9 \max_{T \in \mathcal{T}_l} \eta_T$),
6. refine the mesh $\mathcal{T}_l$ into a new mesh $\mathcal{T}_{l+1}$,

Numerical results

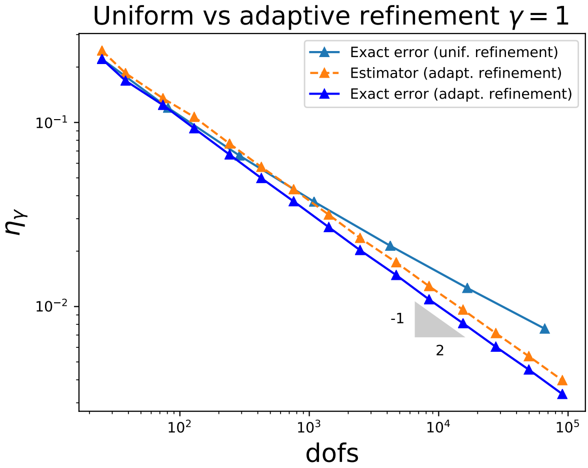
Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε and pick an initial (coarse) mesh $\mathcal{T}_l$ $(l = 0)$,
2. solve the PDE to get u_1^l on $\mathcal{T}_l$,
3. Loop over the cells T of $\mathcal{T}_l$:
   - solve the BW equation on T,
   - compute the local estimator η_T,
4. compute the global estimator η_{bw} and check if $\eta_{\text{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u_1^l.
   - ✗ continue,
5. mark the cells we need to refine (e.g. each cell T for which $\eta_T \geqslant 0.9 \max_{T \in \mathcal{T}_l} \eta_T$),
6. refine the mesh $\mathcal{T}_l$ into a new mesh $\mathcal{T}_{l+1}$,
7. go back to 2. replacing l by $l + 1$.

Numerical results

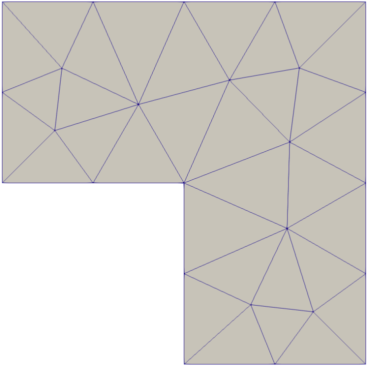
Let's try it out!



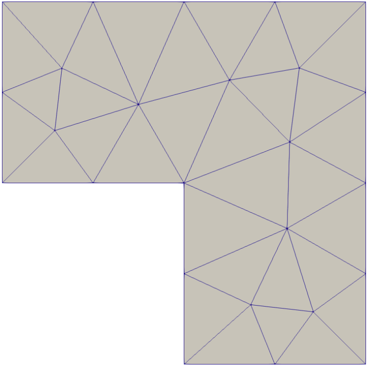
Numerical results

Let's try it out!

Uniform refinement:
Initial mesh



Adaptive refinement:
Initial mesh



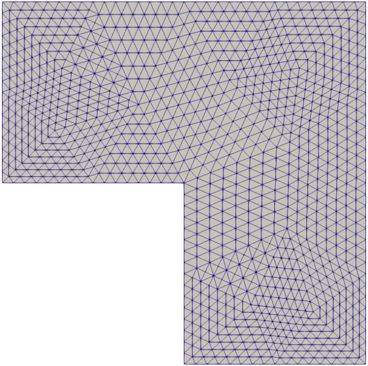
Exact error ≈ 0.2210
Linear system dim. $= 25$

Exact error ≈ 0.2210
Linear system dim. $= 25$

Numerical results

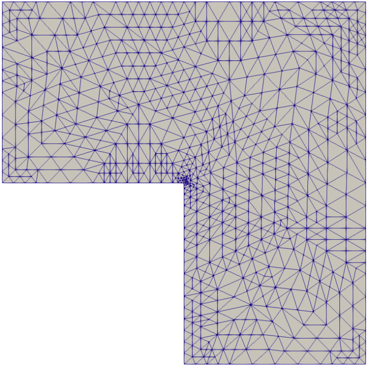
Let's try it out!

Uniform refinement: 3rd mesh



Exact error ≈ 0.0371
Linear system dim. $= 1089$

Adaptive refinement: 6th mesh

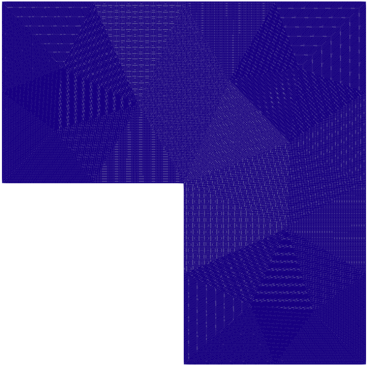


Exact error ≈ 0.0372
Linear system dim. $= 757$

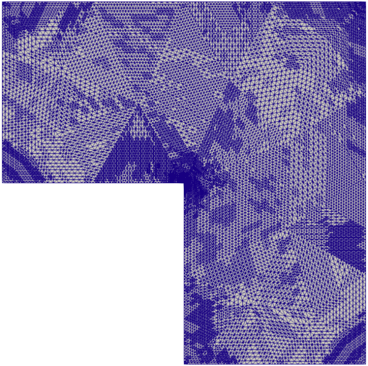
Numerical results

Let's try it out!

Uniform refinement:
6th mesh



Adaptive refinement:
11th mesh



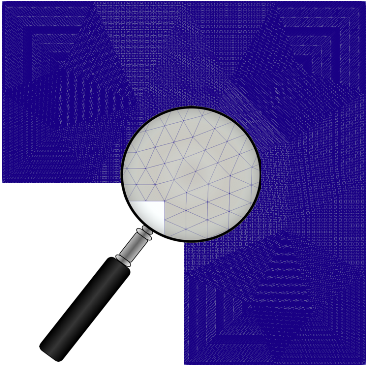
Exact error ≈ 0.0076
Linear system dim. $= 66049$

Exact error ≈ 0.0081
Linear system dim. $= 15429$

Numerical results

Let's try it out!

Uniform refinement:
6th mesh



Exact error ≈ 0.0076
Linear system dim. $= 66049$

Adaptive refinement:
11th mesh

Exact error ≈ 0.0081
Linear system dim. $= 15429$

Table of contents

Application to a fractional Laplacian problem

The spectral fractional Laplacian

Let $\alpha \in (0, 2)$ and $f \in L^2(\Omega)$, we are looking for $u \in L^2(\Omega)$ (with sufficient regularity) such that

$$\begin{aligned} (-\Delta)^{\alpha/2} u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma. \end{aligned}$$

Application to a fractional Laplacian problem

The spectral fractional Laplacian

Let $\alpha \in (0,2)$ and $f \in L^2(\Omega)$, we are looking for $u \in L^2(\Omega)$ (with sufficient regularity) such that

$$\begin{aligned} (-\Delta)^{\alpha/2} u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma. \end{aligned}$$

How is the function u defined ?

Let $\mathcal{L}$ be the Laplace-Dirichlet operator on Ω such that $\mathcal{L} w = f$ if w is the solution of

$$\begin{aligned} -\Delta w &= f \quad \text{in } \Omega, \\ w &= 0 \quad \text{on } \Gamma. \end{aligned}$$

Application to a fractional Laplacian problem

The spectral fractional Laplacian

Let $\alpha \in (0, 2)$ and $f \in L^2(\Omega)$, we are looking for $u \in L^2(\Omega)$ (with sufficient regularity) such that

$$\begin{aligned} (-\Delta)^{\alpha/2} u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma. \end{aligned}$$

How is the function u defined ?

Let $\mathcal{L}$ be the Laplace-Dirichlet operator on Ω such that $\mathcal{L}w = f$ if w is the solution of

$$\begin{aligned} -\Delta w &= f \quad \text{in } \Omega, \\ w &= 0 \quad \text{on } \Gamma. \end{aligned}$$

We consider the weak formulation: w in $H_0^1(\Omega)$ is solution to

$$\int_\Omega \nabla w \cdot \nabla v = \int_\Omega f v \quad \forall v \in H_0^1(\Omega).$$

Application to a fractional Laplacian problem

The spectral fractional Laplacian

The eigenfunctions $\{\psi_j\}_{j=1}^{\infty}$ of $\mathcal{L}$ form a basis of $L^2(\Omega)$.

$$f = \sum_{j=1}^{\infty} f_j \psi_j,$$

with $f_j := \int_\Omega f \psi_j$ for $j = 1, \cdots, \infty$.

Application to a fractional Laplacian problem

The spectral fractional Laplacian

The eigenfunctions $\{\psi_j\}_{j=1}^{\infty}$ of $\mathcal{L}$ form a basis of $L^2(\Omega)$.

$$f = \sum_{j=1}^{\infty} f_j \psi_j,$$

with $f_j := \int_\Omega f\psi_j$ for $j = 1, \cdots, \infty$.
If $\{\lambda_j\}_{j=1}^{\infty}$ are the corresponding eigenvalues, we define the solution u as follow:

$$u = (-\Delta)^{-\alpha/2} f = \mathcal{L}^{-\alpha/2} f := \sum_{j=1}^{\infty} \lambda_j^{-\alpha/2} f_j \psi_j.$$

Table of contents

Application to a fractional Laplacian problem

How to compute the solution numerically ?

There are many ways to compute a numercial approximation to the solution u e.g.

Application to a fractional Laplacian problem

How to compute the solution numerically ?

There are many ways to compute a numercial approximation to the solution u e.g.

- using spectral method [Song, Xu, Karniadakis, 2017],

Application to a fractional Laplacian problem

How to compute the solution numerically ?

There are many ways to compute a numercial approximation to the solution u e.g.

- using spectral method [Song, Xu, Karniadakis, 2017],
- using Euler's reflection formula [Bonito, Pasciak, 2013],

Application to a fractional Laplacian problem

How to compute the solution numerically ?

There are many ways to compute a numercial approximation to the solution u e.g.

- using spectral method [Song, Xu, Karniadakis, 2017],
- using Euler's reflection formula [Bonito, Pasciak, 2013],
- using Cauchy's integral formula [Gavrilyuk, Hackbusch, Khoromskij, 2004] [Bonito, Lei, Pasciak, 2016],

Application to a fractional Laplacian problem

How to compute the solution numerically ?

There are many ways to compute a numercial approximation to the solution u e.g.

- using spectral method [Song, Xu, Karniadakis, 2017],
- using Euler's reflection formula [Bonito, Pasciak, 2013],
- using Cauchy's integral formula
  [Gavrilyuk, Hackbusch, Khoromskij, 2004] [Bonito, Lei, Pasciak, 2016],
- using polynomial interpolation...

Application to a fractional Laplacian problem

How to compute the solution numerically ?

There are many ways to compute a numercial approximation to the solution u e.g.

- using spectral method [Song, Xu, Karniadakis, 2017],
- using Euler's reflection formula [Bonito, Pasciak, 2013],
- using Cauchy's integral formula [Gavrilyuk, Hackbusch, Khoromskij, 2004] [Bonito, Lei, Pasciak, 2016],
- using polynomial interpolation...

How to compute the solution numerically ?

Euler's reflection formula

$$\frac{\pi}{\sin(\pi\theta)} = \int_0^{+\infty} t^{\theta-1}(1+t)^{-1}\,\mathrm{d}t \quad \forall\theta \in (0,1).$$

How to compute the solution numerically ?

Euler's reflection formula

$$\frac{\pi}{\sin(\pi\theta)} = \int_0^{+\infty} t^{\theta-1}(1+t)^{-1}\ \mathrm{d}t \quad \forall \theta \in (0,1).$$

Some tweaks lead to

$$s^{\theta-1} = c_\theta \int_0^{+\infty} t^{\theta-1}(s+t)^{-1}\ \mathrm{d}t \quad \forall s > 0 \text{ and } \forall \theta \in (0,1),$$

with $c_\theta = \frac{\sin(\pi\theta)}{\pi}$.

How to compute the solution numerically ?

Euler's reflection formula

$$\frac{\pi}{\sin(\pi\theta)} = \int_0^{+\infty} t^{\theta-1}(1+t)^{-1}\,\mathrm{d}t \quad \forall\theta \in (0,1).$$

Some tweaks lead to

$$s^{\theta-1} = c_\theta \int_0^{+\infty} t^{\theta-1}(s+t)^{-1}\,\mathrm{d}t \quad \forall s > 0 \text{ and } \forall\theta \in (0,1),$$

with $c_\theta = \frac{\sin(\pi\theta)}{\pi}$.

Then, for $\theta - 1 = -\alpha/2 \in (-1,0)$ and $s = \lambda_j$, $j \in [\![1,+\infty[\![$,

$$\lambda_j^{-\alpha/2} = c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j+t)^{-1}\,\mathrm{d}t.$$

How to compute the solution numerically ?

Euler's reflection formula

$$\lambda_j^{-\alpha/2} = c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1} \, \mathrm{d}t.$$

Then,

$$\mathcal{L}^{-\alpha/2} f = \sum_{j=1}^{\infty} \lambda_j^{-\alpha/2} f_j \psi_j$$

How to compute the solution numerically ?

Euler's reflection formula

$$\lambda_j^{-\alpha/2} = c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1}\,\mathrm{d}t.$$

Then,

$$\begin{aligned}\mathcal{L}^{-\alpha/2} f &= \sum_{j=1}^{\infty} \lambda_j^{-\alpha/2} f_j \psi_j \\ &= \sum_{j=1}^{\infty} c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1} f_j \psi_j\,\mathrm{d}t\end{aligned}$$

How to compute the solution numerically ?

Euler's reflection formula

$$\lambda_j^{-\alpha/2} = c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1} \, \mathrm{d}t.$$

Then,

$$\begin{aligned}
\mathcal{L}^{-\alpha/2} f &= \sum_{j=1}^{\infty} \lambda_j^{-\alpha/2} f_j \psi_j \\
&= \sum_{j=1}^{\infty} c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1} f_j \psi_j \, \mathrm{d}t \\
&= c_\alpha \int_0^{+\infty} t^{-\alpha/2} \sum_{j=1}^{\infty} (\lambda_j + t)^{-1} f_j \psi_j \, \mathrm{d}t
\end{aligned}$$

How to compute the solution numerically ?

Euler's reflection formula

$$\lambda_j^{-\alpha/2} = c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1} \, \mathrm{d}t.$$

Then,

$$\begin{aligned}
\mathcal{L}^{-\alpha/2} f &= \sum_{j=1}^{\infty} \lambda_j^{-\alpha/2} f_j \psi_j \\
&= \sum_{j=1}^{\infty} c_\alpha \int_0^{+\infty} t^{-\alpha/2}(\lambda_j + t)^{-1} f_j \psi_j \, \mathrm{d}t \\
&= c_\alpha \int_0^{+\infty} t^{-\alpha/2} \sum_{j=1}^{\infty} (\lambda_j + t)^{-1} f_j \psi_j \, \mathrm{d}t \\
&= c_\alpha \int_0^{+\infty} t^{-\alpha/2} (\mathcal{L} + t \, \mathrm{Id})^{-1} f \, \mathrm{d}t.
\end{aligned}$$

How to compute the solution numerically ?

Euler's reflection formula

$$\mathcal{L}^{-\alpha/2} f = c_\alpha \int_0^{+\infty} t^{-\alpha/2} (\mathcal{L} + t\,\mathrm{Id})^{-1} f \;\mathrm{d}t,$$

How to compute the solution numerically ?

Euler's reflection formula

$$\mathcal{L}^{-\alpha/2} f = c_\alpha \int_0^{+\infty} t^{-\alpha/2} (\mathcal{L} + t\,\mathrm{Id})^{-1} f \;\mathrm{d}t,$$

and with a nice change of variable,

$$\mathcal{L}^{-\alpha/2} f = c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} \left(\mathrm{Id} + \mathrm{e}^{2y}\,\mathcal{L}\right)^{-1} f \;\mathrm{d}y \quad \text{for } \alpha \in (0,2).$$

with $c_\alpha := \frac{2\sin(\pi\alpha/2)}{\pi}$.

How to compute the solution numerically ?

Euler's reflection formula

$$u = \mathcal{L}^{-\alpha/2} f = c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} \left(\mathrm{Id} + \mathrm{e}^{2y}\,\mathcal{L}\right)^{-1} f \,\mathrm{d}y \quad \text{for } \alpha \in (0,2).$$

How to compute the solution numerically ?

Euler's reflection formula

$$u = \mathcal{L}^{-\alpha/2} f = c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} \left(\mathrm{Id} + \mathrm{e}^{2y}\,\mathcal{L}\right)^{-1} f \,\mathrm{d}y \quad \text{for } \alpha \in (0,2).$$

Let us denote $u_y := (\mathrm{Id} + \mathrm{e}^{2y}\,\mathcal{L})^{-1} f$. This function is solution to the following problem (in weak formulation)

$$\int_\Omega u_y v + \mathrm{e}^{2y} \int_\Omega \nabla u_y \cdot \nabla v = \int_\Omega f v \quad \forall v \in H_0^1(\Omega).$$

How to compute the solution numerically ?

Euler's reflection formula

$$u = \mathcal{L}^{-\alpha/2} f = c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} \left(\mathrm{Id} + \mathrm{e}^{2y}\,\mathcal{L}\right)^{-1} f \,\mathrm{d}y \quad \text{for } \alpha \in (0,2).$$

Let us denote $u_y := (\mathrm{Id} + \mathrm{e}^{2y}\,\mathcal{L})^{-1} f$. This function is solution to the following problem (in weak formulation)

$$\int_\Omega u_y v + \mathrm{e}^{2y} \int_\Omega \nabla u_y \cdot \nabla v = \int_\Omega f v \quad \forall v \in H_0^1(\Omega).$$

We want to discretize the above integral into a finite sum involving computable terms.

How to compute the solution numerically ?

Euler's reflection formula

$$c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \,\mathrm{d}y = u \approx u_1^N := c_\alpha \sum_{l=-N}^{N} \omega_l\, \mathrm{e}^{\alpha y_l}\, u_{y_l,1}.$$

How to compute the solution numerically ?

Euler's reflection formula

$$c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \,\mathrm{d}y = u \approx u_1^N := c_\alpha \sum_{l=-N}^{N} \omega_l\, \mathrm{e}^{\alpha y_l}\, u_{y_l,1}.$$

1. Using FEM, we discretize the function u_y into $u_{y,1}$ solution to

$$\int_\Omega u_{y,1} v_{y,1} + \mathrm{e}^{2y} \int_\Omega \nabla u_{y,1} \cdot \nabla v_{y,1} = \int_\Omega f v_{y,1} \quad \forall v_{y,1} \in V^1.$$

How to compute the solution numerically ?

Euler's reflection formula

$$c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \, \mathrm{d}y = u \approx u_1^N := c_\alpha \sum_{l=-N}^{N} \omega_l \, \mathrm{e}^{\alpha y_l}\, u_{y_l,1}.$$

1. Using FEM, we discretize the function u_y into $u_{y,1}$ solution to

$$\int_\Omega u_{y,1} v_{y,1} + \mathrm{e}^{2y} \int_\Omega \nabla u_{y,1} \cdot \nabla v_{y,1} = \int_\Omega f v_{y,1} \quad \forall v_{y,1} \in V^1.$$

2. We discretize the integral using a simple rectangle quadrature rule of weights $\omega_l = \frac{1}{\sqrt{N}}$ and points $y_l = \frac{l}{\sqrt{N}}$ for any $l \in [\![-N, \cdots, N]\!]$, where N is a user chosen parameter.

Table of contents

A posteriori error estimation

Euler's reflection formula

We would like to quantify the approximation error:

$$\|e\|_{L^2} := \|u - u_1^N\|_{L^2} := \left\| c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - \frac{c_\alpha}{\sqrt{N}} \sum_{l=-N}^{N} \mathrm{e}^{\alpha y_l}\, u_{y_l,1} \right\|_{L^2}.$$

A posteriori error estimation

Euler's reflection formula

We would like to quantify the approximation error:

$$\|e\|_{L^2} := \|u - u_1^N\|_{L^2} := \left\| c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - \frac{c_\alpha}{\sqrt{N}} \sum_{l=-N}^{N} \mathrm{e}^{\alpha y_l}\, u_{y_l,1} \right\|_{L^2}.$$

To do so, we introduce

$$u_1 := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y.$$

A posteriori error estimation

Euler's reflection formula

We would like to quantify the approximation error:

$$\|e\|_{L^2} := \|u - u_1^N\|_{L^2} := \left\| c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - \frac{c_\alpha}{\sqrt{N}} \sum_{l=-N}^{N} \mathrm{e}^{\alpha y_l}\, u_{y_l,1} \right\|_{L^2}.$$

To do so, we introduce

$$u_1 := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y.$$

We have,

$$\|u - u_1^N\|_{L^2}$$

A posteriori error estimation

Euler's reflection formula

We would like to quantify the approximation error:

$$\|e\|_{L^2} := \|u - u_1^N\|_{L^2} := \left\| c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - \frac{c_\alpha}{\sqrt{N}} \sum_{l=-N}^{N} \mathrm{e}^{\alpha y_l}\, u_{y_l,1} \right\|_{L^2}.$$

To do so, we introduce

$$u_1 := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y.$$

We have,

$$\|u - u_1^N\|_{L^2} = \|u - u_1 + u_1 - u_1^N\|_{L^2}$$

A posteriori error estimation

Euler's reflection formula

We would like to quantify the approximation error:

$$\|e\|_{L^2} := \|u - u_1^N\|_{L^2} := \left\| c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - \frac{c_\alpha}{\sqrt{N}} \sum_{l=-N}^{N} \mathrm{e}^{\alpha y_l}\, u_{y_l,1} \right\|_{L^2}.$$

To do so, we introduce

$$u_1 := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y.$$

We have,

$$\begin{aligned} \|u - u_1^N\|_{L^2} &= \|u - u_1 + u_1 - u_1^N\|_{L^2} \\ &\leqslant \underbrace{\|u - u_1\|_{L^2}}_{\text{FE error}} + \underbrace{\|u_1 - u_1^N\|_{L^2}}_{\text{quadrature error}}. \end{aligned}$$

A posteriori error estimation

Finite element error

We want to quantify the finite element error $\|u - u_1\|_{L^2}$.

A posteriori error estimation

Finite element error

We want to quantify the finite element error $\|u - u_1\|_{L^2}$.

$$u - u_1 = c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y$$

A posteriori error estimation

Finite element error

We want to quantify the finite element error $\|u - u_1\|_{L^2}$.

$$\begin{aligned} u - u_1 &= c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y \\ &= c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} (u_y - u_{y,1}) \mathrm{d}y. \end{aligned}$$

A posteriori error estimation

Finite element error

We want to quantify the finite element error $\|u - u_1\|_{L^2}$.

$$u - u_1 = c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_y \mathrm{d}y - c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, u_{y,1} \mathrm{d}y$$
$$= c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} (u_y - u_{y,1}) \mathrm{d}y.$$

Idea: We already know an estimator for the error $u_y - u_{y,1}$.
Can we use it to estimate $u - u_1$?

A posteriori error estimation

Finite element error

For a fixed y in $\mathbb{R}$ and for a cell T of the mesh, we compute $e^{\mathrm{bw}}_{y,T} \in V^{\mathrm{bw}}(T)$ the solution to

$$\int_T e^{\mathrm{bw}}_{y,T} v_T + \mathrm{e}^{2y} \int_T \nabla e^{\mathrm{bw}}_{y,T} \cdot \nabla v_T = \int_T r_{y,T} v_T + \sum_{E \in \partial T} \int_E J_{y,E} v_T \quad \forall v_T \in V^{\mathrm{bw}}(T).$$

A posteriori error estimation

Finite element error

For a fixed y in $\mathbb{R}$ and for a cell T of the mesh, we compute $e^{\mathrm{bw}}_{y,T} \in V^{\mathrm{bw}}(T)$ the solution to

$$\int_T e^{\mathrm{bw}}_{y,T} v_T + \mathrm{e}^{2y} \int_T \nabla e^{\mathrm{bw}}_{y,T} \cdot \nabla v_T = \int_T r_{y,T} v_T + \sum_{E \in \partial T} \int_E J_{y,E} v_T \quad \forall v_T \in V^{\mathrm{bw}}(T).$$

We compute

$$e^{\mathrm{bw}}_T := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, e^{\mathrm{bw}}_{y,T} \mathrm{d}y,$$

A posteriori error estimation

Finite element error

For a fixed y in $\mathbb{R}$ and for a cell T of the mesh, we compute $e_{y,T}^{\mathrm{bw}} \in V^{\mathrm{bw}}(T)$ the solution to

$$\int_T e_{y,T}^{\mathrm{bw}} v_T + \mathrm{e}^{2y} \int_T \nabla e_{y,T}^{\mathrm{bw}} \cdot \nabla v_T = \int_T r_{y,T} v_T + \sum_{E \in \partial T} \int_E J_{y,E} v_T \quad \forall v_T \in V^{\mathrm{bw}}(T).$$

We compute

$$e_T^{\mathrm{bw}} := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} \, e_{y,T}^{\mathrm{bw}} \mathrm{d}y,$$

$$\eta_{\mathrm{bw},T} := \| e_T^{\mathrm{bw}} \|_{L^2} \quad \text{and} \quad \eta_{\mathrm{bw}}^2 := \sum_{T \in \mathcal{T}} \eta_{\mathrm{bw},T}^2.$$

A posteriori error estimation

Finite element error

For a fixed y in $\mathbb{R}$ and for a cell T of the mesh, we compute $e_{y,T}^{\mathrm{bw}} \in V^{\mathrm{bw}}(T)$ the solution to

$$\int_T e_{y,T}^{\mathrm{bw}} v_T + \mathrm{e}^{2y} \int_T \nabla e_{y,T}^{\mathrm{bw}} \cdot \nabla v_T = \int_T r_{y,T} v_T + \sum_{E \in \partial T} \int_E J_{y,E} v_T \quad \forall v_T \in V^{\mathrm{bw}}(T).$$

We compute

$$e_T^{\mathrm{bw}} := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y} \, e_{y,T}^{\mathrm{bw}} \mathrm{d}y,$$

$$\eta_{\mathrm{bw},T} := \| e_T^{\mathrm{bw}} \|_{L^2} \quad \text{and} \quad \eta_{\mathrm{bw}}^2 := \sum_{T \in \mathcal{T}} \eta_{\mathrm{bw},T}^2.$$

A posteriori error estimation

Finite element error

For a fixed y in $\mathbb{R}$ and for a cell T of the mesh, we compute $e_{y,T}^{\mathrm{bw}} \in V^{\mathrm{bw}}(T)$ the solution to

$$\int_T e_{y,T}^{\mathrm{bw}} v_T + \mathrm{e}^{2y} \int_T \nabla e_{y,T}^{\mathrm{bw}} \cdot \nabla v_T = \int_T r_{y,T} v_T + \sum_{E \in \partial T} \int_E J_{y,E} v_T \quad \forall v_T \in V^{\mathrm{bw}}(T).$$

We compute

$$e_T^{\mathrm{bw}} := c_\alpha \int_{-\infty}^{+\infty} \mathrm{e}^{\alpha y}\, e_{y,T}^{\mathrm{bw}} \mathrm{d}y,$$

$$\eta_{\mathrm{bw},T} := \|e_T^{\mathrm{bw}}\|_{L^2} \quad \text{and} \quad \eta_{\mathrm{bw}}^2 := \sum_{T \in \mathcal{T}} \eta_{\mathrm{bw},T}^2.$$

$$\|u - u_1\|_{L^2} \approx \eta_{\mathrm{bw}}?$$

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ $(j = 0)$ and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ $(j = 0)$ and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u_{y_l,1}^{j}$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ $(j = 0)$ and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ $(j = 0)$ and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},
6. check if $\eta^j_{\mathrm{bw}} \leqslant \varepsilon$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ $(j = 0)$ and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},
6. check if $\eta^j_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u^j_1.

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ $(j = 0)$ and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},
6. check if $\eta^j_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u^j_1.
   - ✗ continue,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},
6. check if $\eta^j_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u^j_1.
   - ✗ continue,
7. mark the cells we need to refine (e.g. each cell T for which $\eta^j_{\mathrm{bw},T} \geqslant 0.9 \max_{T\in\mathcal{T}_j} \eta^j_{\mathrm{bw},T}$),

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},
6. check if $\eta^j_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u^j_1.
   - ✗ continue,
7. mark the cells we need to refine (e.g. each cell T for which $\eta^j_{\mathrm{bw},T} \geqslant 0.9 \max_{T\in\mathcal{T}_j} \eta^j_{\mathrm{bw},T}$),
8. refine the mesh $\mathcal{T}_j$ into $\mathcal{T}_{j+1}$,

Numerical results

Let's try it out!

Adaptive refinement algorithm:

1. fix a tolerance ε, pick an initial (coarse) mesh $\mathcal{T}_j$ ($j = 0$) and pick a very fine quadrature rule $\{y_l\}_{l=-N}^{N}$ (take N large),
2. for each quadrature point y_l:
   - solve the PDE on $\mathcal{T}_j$ to get $u^j_{y_l,1}$,
   - for each cell T of $\mathcal{T}_j$, solve the BW equation on T to compute $e^{\mathrm{bw},j}_{y_l,T}$,
3. sum the functions $u^j_{y_l,1}$ into the quadrature rule to get u^j_1,
4. for each cell T of $\mathcal{T}_j$:
   - sum (over l) the functions $e^{\mathrm{bw},j}_{y_l,T}$ into the quadrature rule to get $e^{\mathrm{bw},j}_T$,
   - compute the local contributions of the estimator $\eta^j_{\mathrm{bw},T} := \|e^{\mathrm{bw},j}_T\|_{L^2}$,
5. sum (over T) the local contributions $\eta^j_{\mathrm{bw},T}$ to get η^j_{bw},
6. check if $\eta^j_{\mathrm{bw}} \leqslant \varepsilon$,
   - ✓ stop the algorithm and return u^j_1.
   - ✗ continue,
7. mark the cells we need to refine (e.g. each cell T for which $\eta^j_{\mathrm{bw},T} \geqslant 0.9 \max_{T\in\mathcal{T}_j} \eta^j_{\mathrm{bw},T}$),
8. refine the mesh $\mathcal{T}_j$ into $\mathcal{T}_{j+1}$,
9. go back to 2. replacing j by $j + 1$.

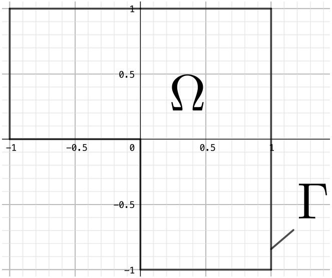
Numerical results

Let's try it out!

Taking $f = 1$, we solve

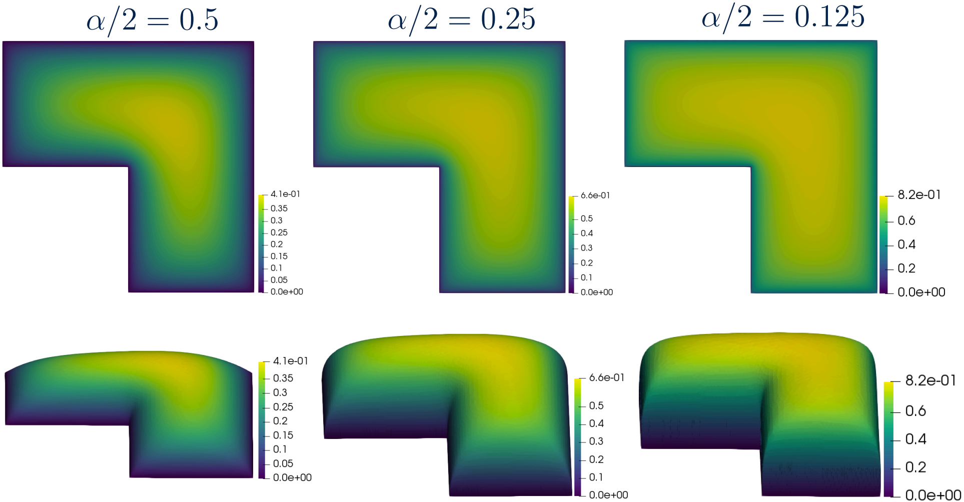
$$(-\Delta)^{\alpha/2} u = f \quad \text{in } \Omega$$
$$u = 0 \quad \text{on } \Gamma.$$

using finite elements and Euler's reflection formula and we estimate the error using the Bank-Weiser estimator.



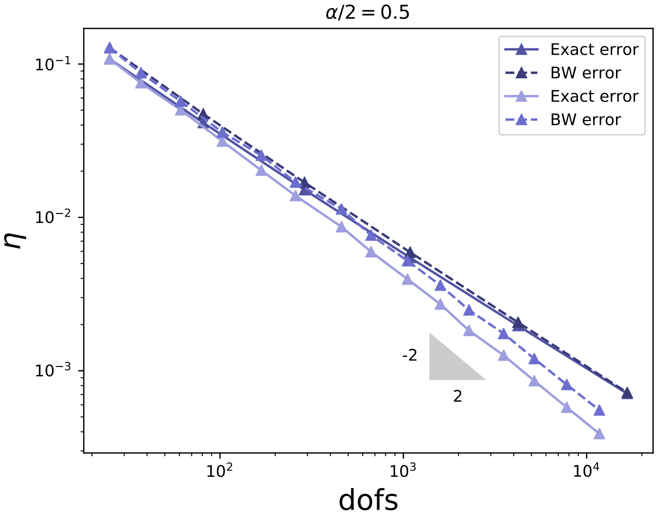
Numerical results

Let's try it out!



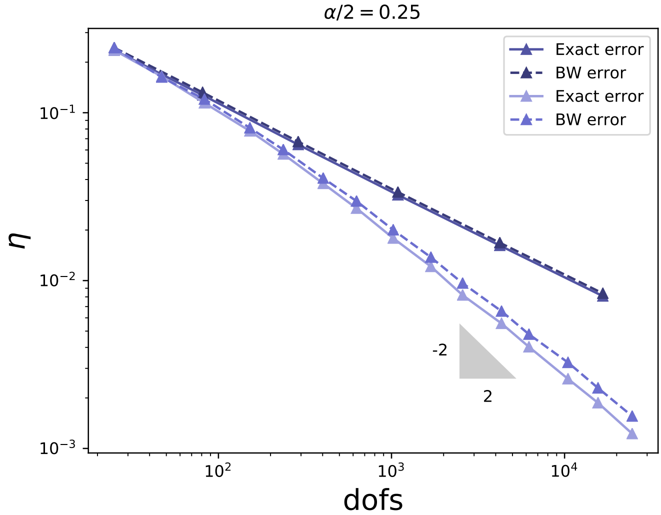
Numerical results

Let's try it out!



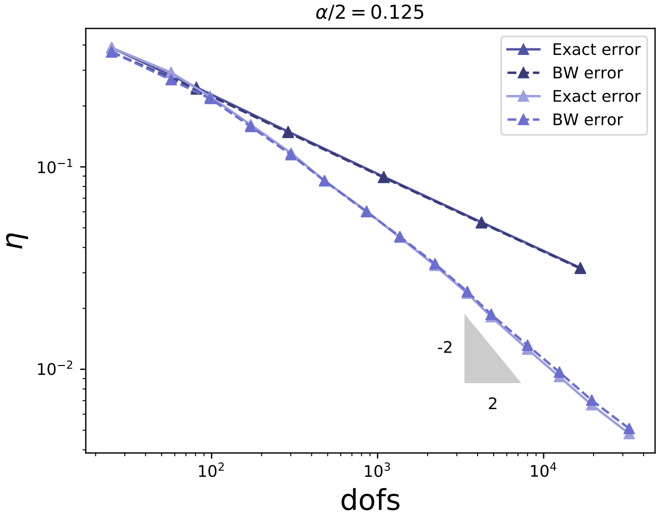
Numerical results

Let's try it out!



Numerical results

Let's try it out!

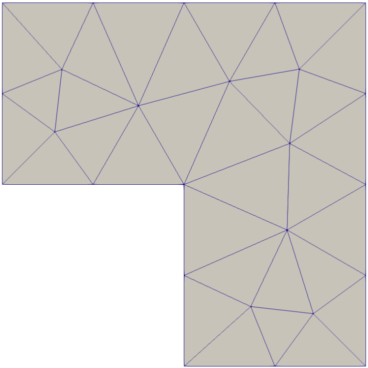


Numerical results

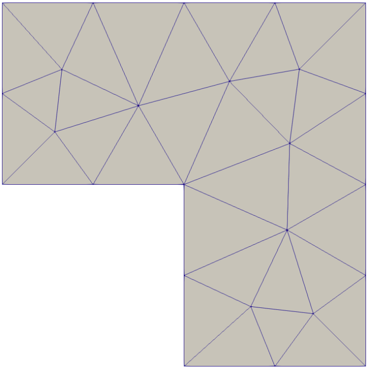
Let's try it out!

$\alpha/2 = 0.5$

Uniform refinement:
Initial mesh



Adaptive refinement:
Initial mesh



Exact error ≈ 0.1079
Linear system dim. $= 25$

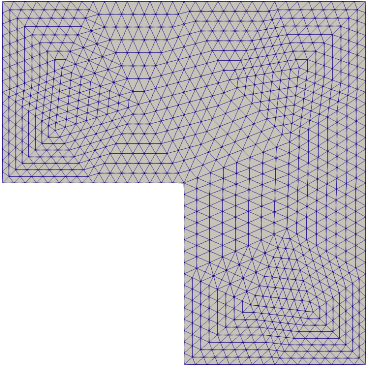
Exact error ≈ 0.1079
Linear system dim. $= 25$

Numerical results

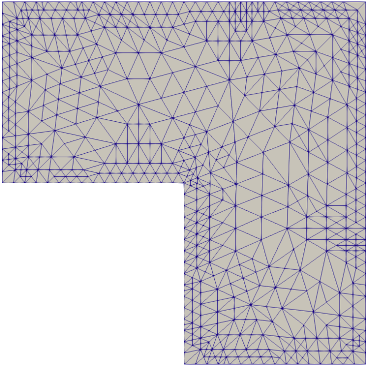
Let's try it out!

$$\alpha/2 = 0.5$$

Uniform refinement:
3rd mesh



Adaptive refinement:
7th mesh



Exact error ≈ 0.0055
Linear system dim. $= 1089$

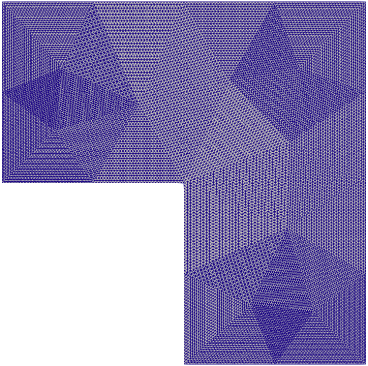
Exact error ≈ 0.0060
Linear system dim. $= 667$

Numerical results

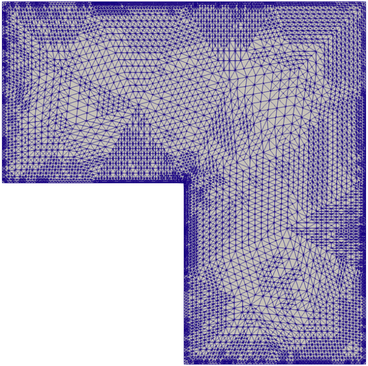
Let's try it out!

$\alpha/2 = 0.5$

Uniform refinement:
5th mesh



Adaptive refinement:
13th mesh



Exact error ≈ 0.0007
Linear system dim. $= 16641$

Exact error ≈ 0.0006
Linear system dim. $= 7791$

Table of contents

Challenges

- Computational science/engineering:

Challenges

- Computational science/engineering:
  - Compare with other methods.

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).
  - Apply it to a real-world problem.

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).
  - Apply it to a real-world problem.
- Mathematics:

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).
  - Apply it to a real-world problem.
- Mathematics:
  - Justify the reliability of the estimator ($\|e\|_{L^2} \leqslant C\eta_{\mathrm{bw}}$).

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).
  - Apply it to a real-world problem.
- Mathematics:
  - Justify the reliability of the estimator ($\|e\|_{L^2} \leqslant C\eta_{\mathrm{bw}}$).
  - Bound the finite element error in L^2 norm without extra regularity assumption on the solution (even for non-fractional problems).

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).
  - Apply it to a real-world problem.
- Mathematics:
  - Justify the reliability of the estimator ($\|e\|_{L^2} \leqslant C\eta_{\mathrm{bw}}$).
  - Bound the finite element error in L^2 norm without extra regularity assumption on the solution (even for non-fractional problems).
  - Prove the convergence of the adaptive scheme.

Challenges

- Computational science/engineering:
  - Compare with other methods.
  - Add a posteriori error estimation on the quadrature scheme.
  - Write a parallel code.
  - Extend the method to more complicated problems (e.g. time dependent PDEs).
  - Apply it to a real-world problem.
- Mathematics:
  - Justify the reliability of the estimator ($\|e\|_{L^2} \leqslant C\eta_{\mathrm{bw}}$).
  - Bound the finite element error in L^2 norm without extra regularity assumption on the solution (even for non-fractional problems).
  - Prove the convergence of the adaptive scheme.
- Become famous and get a permanent job.

Thank you for your attention!