Deep learning driven self-adaptive hp finite element method

Maciej Paszyński

Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
home.agh.edu.pl/paszynsk

R. Grzeszczuk (AGH) D. Pardo (BCAM) L. Demkowicz (UT)



Outline

- Self-adaptive *hp*-FEM algorithm
- Deep Neural Network Driven *hp*-FEM algorithm
- Numerical results

L shape domain model problem

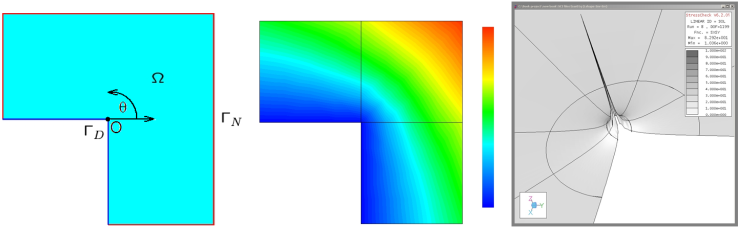


Figure: The L-shape domain model problem. Plot of $\|\nabla u\|_{L^2}$

$$\Delta u = 0 \text{ in } \Omega \qquad u = 0 \text{ on } \Gamma_D \qquad \frac{\partial u}{\partial n} = g \text{ on } \Gamma_N$$

$$b(u,v) = l(v) \forall v \in V \quad b(u,v) = \int_\Omega \nabla u \nabla v dx \quad l(v) = \int_{\Gamma_N} gvdS \quad (*)$$

$$V = \left\{ v \in L^2(\Omega) : \int_\Omega ||v||^2 + ||\nabla v||^2 < \infty : tr(v) = 0 \text{ on } \Gamma_D \right\}$$

2D *hp* finite element (1/5)

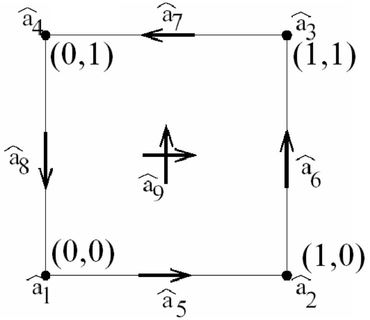
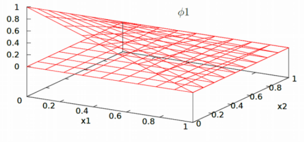


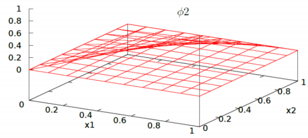
Figure: The 2D reference rectangular hp-finite element

The basis functions are constructed as tensor products of 1D hierarchical basis functions

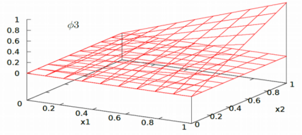
$$\hat{\chi}_1(\xi) = 1 - \xi \quad \hat{\chi}_2(\xi) = \xi \quad \hat{\chi}_l(\xi) = (1-\xi)\xi(2\xi - 1)^{l-3} \quad l = 3, 4, ..., p$$

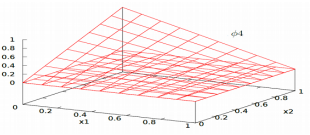
2D *hp* finite element (2/5)





$\hat{\varphi}_1(\xi_1, \xi_2) = \hat{\chi}_1(\xi_1)\, \hat{\chi}_1(\xi_2)$ $\hat{\varphi}_2(\xi_1, \xi_2) = \hat{\chi}_2(\xi_1)\, \hat{\chi}_1(\xi_2)$

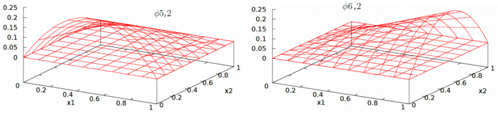




$\hat{\varphi}_3(\xi_1, \xi_2) = \hat{\chi}_2(\xi_1)\, \hat{\chi}_2(\xi_2)$ $\hat{\varphi}_4(\xi_1, \xi_2) = \hat{\chi}_1(\xi_1)\, \hat{\chi}_2(\xi_2)$

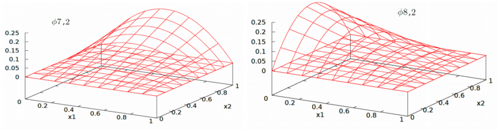
The *vertex shape functions* are:

$$\hat{\phi}_1(\xi_1, \xi_2) = \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2) \quad \hat{\phi}_2(\xi_1, \xi_2) = \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)$$
$$\hat{\phi}_3(\xi_1, \xi_2) = \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2) \quad \hat{\phi}_4(\xi_1, \xi_2) = \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)$$



$\hat{\varphi}_{5,2}(\xi_1, \xi_2) = \hat{\chi}_3(\xi_1)\,\hat{\chi}_1(\xi_2)$

$\hat{\varphi}_{6,2}(\xi_1, \xi_2) = \hat{\chi}_2(\xi_1)\,\hat{\chi}_3(\xi_2)$

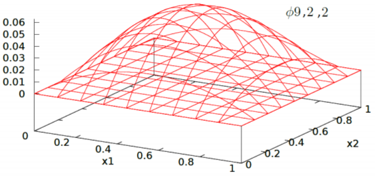


$\hat{\varphi}_{7,2}(\xi_1, \xi_2) = \hat{\chi}_3(\xi_1)\,\hat{\chi}_2(\xi_2)$

$\hat{\varphi}_{8,2}(\xi_1, \xi_2) = \hat{\chi}_1(\xi_1)\,\hat{\chi}_3(\xi_2)$

The *edge shape functions* are defined as:

$$\hat{\phi}_{5,j}(\xi_1, \xi_2) = \hat{\chi}_{2+j}(\hat{\xi}_1)\hat{\chi}_1(\hat{\xi}_2), j = 1, \ldots, p_1 - 1$$
$$\hat{\phi}_{6,j}(\xi_1, \xi_2) = \hat{\chi}_2(\hat{\xi}_1)\hat{\chi}_{2+j}(\hat{\xi}_2), j = 1, \ldots, p_2 - 1$$
$$\hat{\phi}_{7,j}(\xi_1, \xi_2) = \hat{\chi}_{2+j}(\hat{\xi}_1)\hat{\chi}_2(\hat{\xi}_2), j = 1, \ldots, p_3 - 1$$
$$\hat{\phi}_{8,j}(\xi_1, \xi_2) = \hat{\chi}_1(\hat{\xi}_1)\hat{\chi}_{2+j}(\hat{\xi}_2), j = 1, \ldots, p_4 - 1$$



$$\hat{\varphi}_{9,2\cdot 2\cdot}(\xi_1,\xi_2)=\hat{\chi}_{3}(\xi_1)\,\hat{\chi}_{3}(\xi_2)$$

The *interior shape functoins* are defined as

$$\hat{\phi}_{9,i,j} = \hat{\xi}_{2+j}(\chi_1)\hat{\xi}_{2+j}(\chi_2) i = 1, \ldots, p_h - 1, j = 1, \ldots, p_v - 1$$

Definition

The reference 2D *hp* finite element is a triple $(\hat{K}, X(\hat{K}), \Pi_p)$ defined

1. Geometry $\hat{K} = [0,1]^2$
2. Selection of nodes. There are four vertex nodes $\hat{a}_1$, $\hat{a}_2$, $\hat{a}_3$, $\hat{a}_4$, four edge nodes $\hat{a}_5$, $\hat{a}_6$, $\hat{a}_7$, $\hat{a}_8$, and one edge node $\hat{a}_9$ selected
3. Definition of element shape functions $X(\hat{K})$

   $$X(\hat{K}) = \text{span}\left\{\hat{\phi}_j \in \mathcal{Q}^{(p_h,p_v)}(\hat{K}), j = 1, \ldots, (p_h+1)(p_v+1)\right\}$$

   where $\mathcal{Q}^{(p_h,p_v)}$ are polynomials of order p_h with respect to ξ_2 over $\hat{K} = (0,1)^2$. With each of the element edges, a possibly different order of approximation $p_i, i = 1, \ldots, 4$ is associated, under the assumption that $p_1, p_3 \leq p_h$ and $p_2, p_4 \leq p_v$
4. Definition of the projection based interpolation operator $\Pi_p : H^1(0,1) \to X(\hat{K})$, given a function $u \in H^1(\hat{K})$, it computes its projection-based interpolant $\Pi_p u \in X(\hat{K})$

Coarse mesh and approximation space

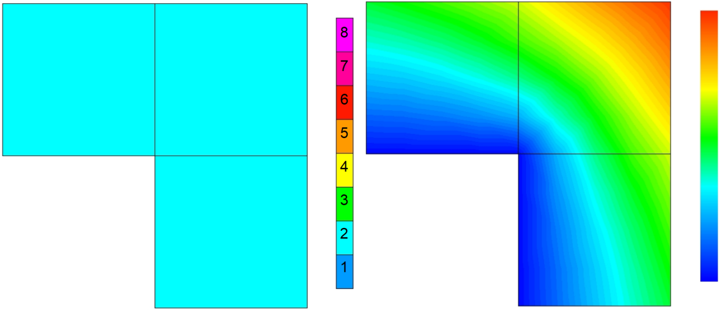


Figure: The coarse mesh and the solution from the coarse mesh approximation space.

Definition

Coarse mesh problem: Find $\{u^i_{hp}\}_{i=1}^{N_{hp}}$ coefficients (*dofs*) of approximate solution $V \supset V_{hp} \ni u_{hp} = \sum_{i=1}^{N_{hp}} u^i_{hp} e^i_{hp}$ fulfilling (*).

Coarse mesh and approximation space

Definition

The *initial coarse mesh* is obtained by partitioning the domain Ω into a finite set $(K, X(K), \Pi_p) \in T_{hp}$ of *hp* finite elements and selecting arbitrary polynomial orders of approximation.

Definition

The *coarse mesh approximation space* is defined as

$$V_{hp} = \mathrm{span}\{e^j_{hp} : \forall K \in T_{hp}|_K, \forall \phi_k \in X(K), \exists! e^i_{hp} : e^i_{hp}|_K = \phi_k\}$$

where e^i_{hp} is a global basis function (element basis of V_{hp}), ϕ_k is a shape function and $(k, K) \to i(k_1, K)$ is the mapping over the coarse mesh assigning global number $i(k, K)$ of *dofs* (basis functions) related with shape function k from element K

Remark

The approximation space $V_{hp} \subset V$ with basis $\{e^i_{hp}\}_{i=1}^{N_{hp}}$ is constructed by gluing together element-local shape functions.

Fine mesh and approximation space

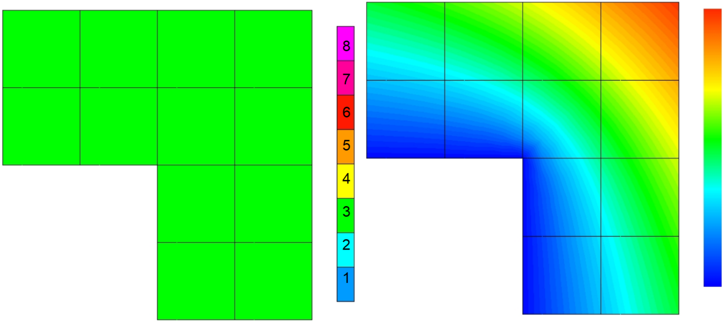


Figure: Fine mesh and solution from the fine mesh approximation space.

Definition

Fine mesh problem: Find $\{u^i_{\frac{h}{2},p+1}\}_{i=1}^{N_{\frac{h}{2},p+1}}$ coefficients (*dofs*) of approximate solution $V \supset V_{\frac{h}{2},p+1} \ni u_{\frac{h}{2},p+1} = \sum_{i=1}^{N_{\frac{h}{2},p+1}} u^i_{\frac{h}{2},p+1} e^i_{\frac{h}{2},p+1}$ fulfilling (*).

Fine mesh and approximation space

Definition

The *fine mesh* is obtained by breaking each element from the coarse mesh $(K, X(K), \Pi_p) \in T_{\frac{h}{2},p+1}$ into 4 elements (in 2D) and increasing the polynomial orders of approximation by one.

Definition

The *fine mesh approximation space* is defined as

$$V_{\frac{h}{2},p+1} = \text{span}$$

$$\{e^j_{\frac{h}{2},p+1} : \forall K \in T_{\frac{h}{2},p+1}|_K, \forall \phi_k \in X(K), \exists! e^i_{\frac{h}{2},p+1} : e^i_{\frac{h}{2},p+1}|_K = \phi_k\}$$

where $e^i_{\frac{h}{2},p+1}$ is a basis function (element basis of $V_{\frac{h}{2},p+1}$), ϕ_k is a shape function, $(k, K) \to i(k_1, K)$ is the mapping over the fine mesh assigning global number $i(k, K)$ of *dofs* (basis function) related to shape function k from element K.

hp-adaptive finite element method

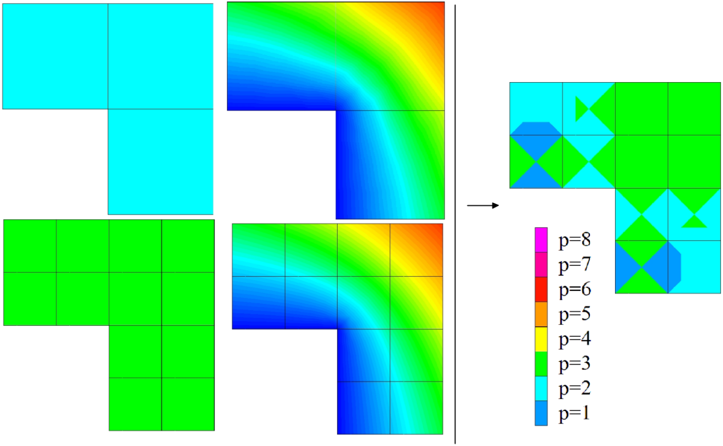


Figure: Self-adaptive *hp* finite element method algorithm

hp-adaptive finite element method

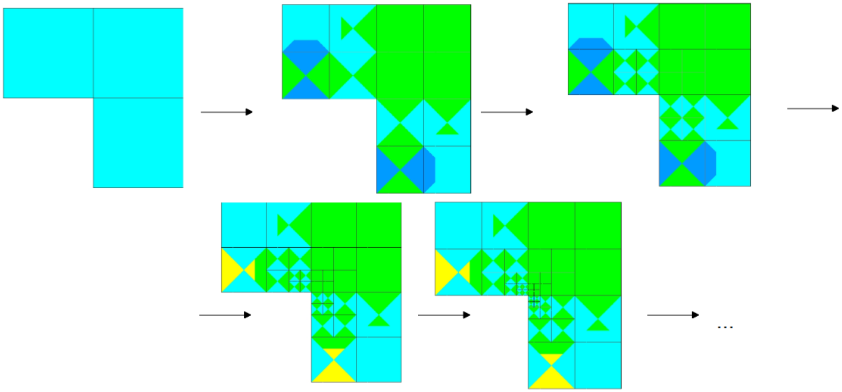


Figure: Sequence of *hp* adaptive meshes generated by the self-adaptive *hp* finite element method algorithm

hp-adaptive finite element method

Input: Initial mesh, PDE, boundary conditions, error
Output: Optimal mesh
coarse mesh = initial mesh
Solve the coarse mesh problem
Generate fine mesh
Solve the fine mesh problem
if *maximum relative error* $<$ *accuracy* **then**
    return *fine mesh solution*
end
Select optimal refinements for every *hp* finite element from the coarse mesh (**Call Algorithm 2**)
Perform all required *h* refinements
Perform all required *p* refinements
coarse mesh = actual mesh
goto 2

Algorithm 1: Self-adaptive *hp*-FEM algorithm

Optimal approximation space over an element

Definition

Let $V_{hp} \subset V_{\frac{h}{2},p+1} \subset V$ be the coarse and fine mesh approximation spaces. Let T_{hp} represents the coarse mesh elements. Let $u_{hp} \in V_{hp}$ and $u_{\frac{h}{2},p+1} \in V_{\frac{h}{2},p+1}$ be the coarse and fine mesh problem solutions, respectively. The approximation space V^K_{opt} is called the optimal approximation space over an element $K \in T_{hp}$, if the projection based interpolant w_{opt} of $u_{\frac{h}{2},p+1} \in V_{\frac{h}{2},p+1}$ into V^K_{opt} over element K realizes the following maximum

$$\frac{\left|u_{\frac{h}{2},p+1} - u_{hp}\right|_{H^1(K)} - \left|u_{\frac{h}{2},p+1} - w_{opt}\right|_{H^1(K)}}{\Delta \mathrm{nrdof}(V_{hp}, V^K_{opt}, K)} =$$

$$= \max_{V_{hp} \subseteq V_w \subseteq V_{\frac{h}{2},p+1}} \frac{\left|u_{\frac{h}{2},p+1} - u_{hp}\right|_{H^1(K)} - \left|u_{\frac{h}{2},p+1} - w\right|_{H^1(K)}}{\Delta \mathrm{nrdof}(V_{hp}, V^K_{opt}, K)}$$

where w is the projection-based interpolant of $u_{\frac{h}{2},p+1} \in V_{\frac{h}{2},p+1}$ into V_w over element K, and $\Delta nrdof(V, X, K) = dim V\Big|_K - dim X\Big|_K$

Selection of optimal refinements

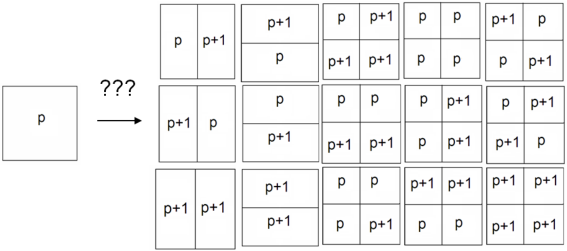


Figure: Selection of optimal refinements

Selection of the optimal refinements

Input: Element K, coarse mesh solution $u_{hp} \in V_{hp}$, fine mesh solution $u_{\frac{h}{2},p+1} \in V_{\frac{h}{2},p+1}$

Output: Optimal refinement V^K_{opt} for element K

for *coarse mesh elements* $K \in T_{hp}$ **do**

  for *approximation space* $V_{opt} \in K$ **do**

    $rate_{max} = 0$

    Compute the projection based interpolant $w|_K$ of $u_{\frac{h}{2},p+1}|_K$

    Compute the error decrease rate

$$rate(w) = \frac{\left|u_{\frac{h}{2},p+1} - u_{hp}\right|_{H^1(K)} - \left|u_{\frac{h}{2},p+1} - w\right|_{H^1(K)}}{\Delta \mathrm{nrdof}(V_{hp}, V^K_{opt}, K)}$$

    if $rate(w) > rate_{max}$ **then**

      $rate_{max} = rate(w)$

      Select V^K_{opt} corresponding to $rate_{max}$ as the optimal refinement for element K

    end

  end

end

Select orders on edges = MIN (orders from neighboring interiors)

Algorithm 2: Selection of optimal refinements over K

Our goal is to replace this Algorithm 2 with a Deep Neural Network.

Solution to L-shape domain problem with 0.001 accuracy

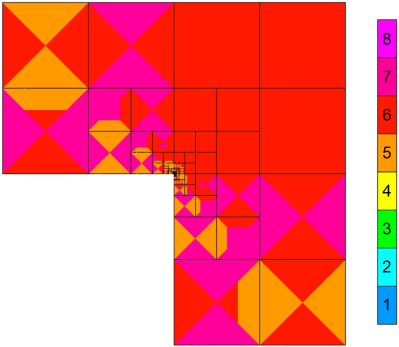


Figure: Distribution of polynomial orders over the mesh generated by self-adaptive *hp*-FEM delivering solution with 0.001 accuracy.

Solution to L-shape domain problem with 0.001 accuracy

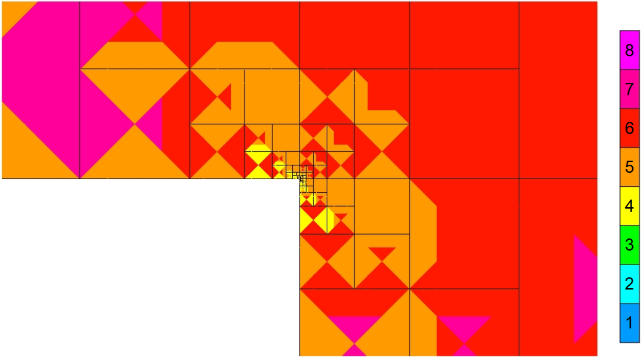


Figure: Zoom 10 X.

Solution to L-shape domain problem with 0.001 accuracy

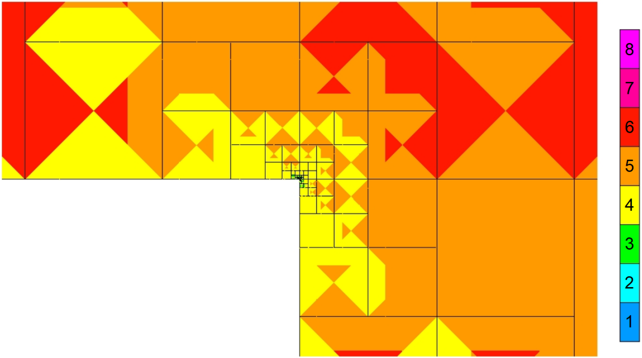


Figure: Zoom 100 X.

Solution to L-shape domain problem with 0.001 accuracy

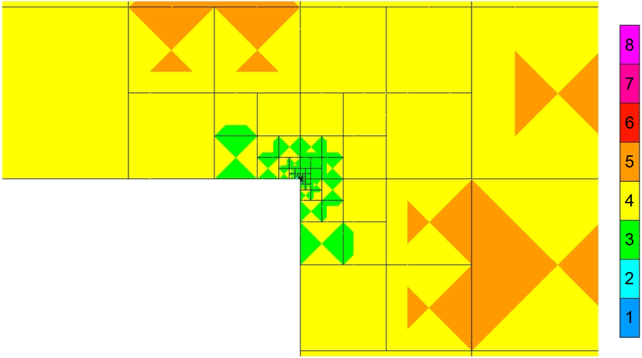


Figure: Zoom 1000 X.

Solution to L-shape domain problem with 0.001 accuracy

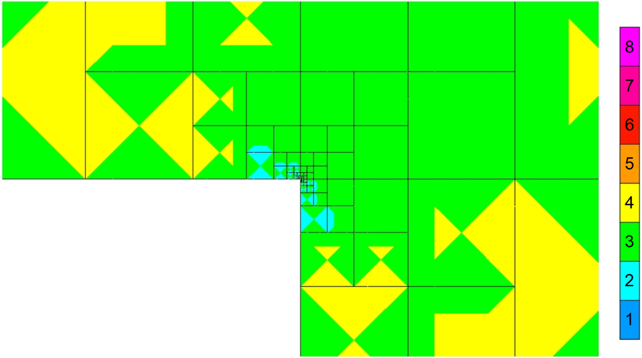


Figure: Zoom 10,000 X.

Solution to L-shape domain problem with 0.001 accuracy

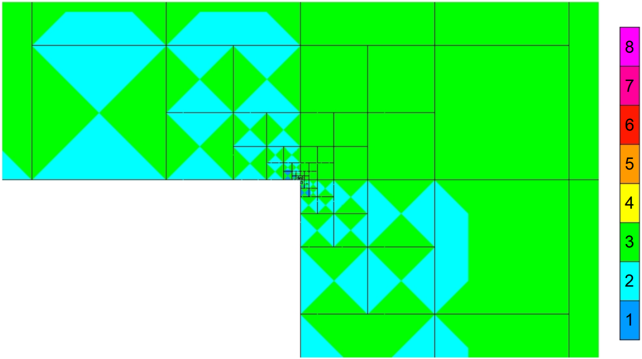


Figure: Zoom 100,000 X.

Solution to L-shape domain problem with 0.001 accuracy

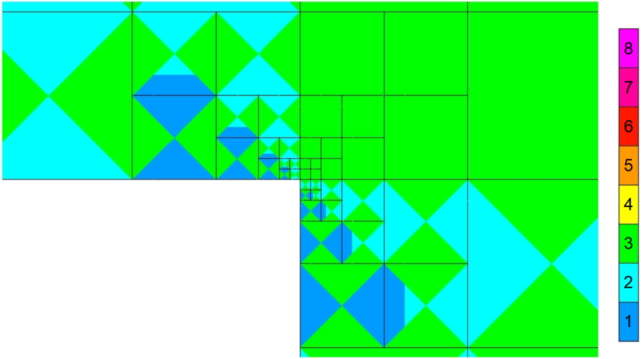


Figure: Zoom 1,000,000 X.

Deterministic and DNN driven *hp*-FEM

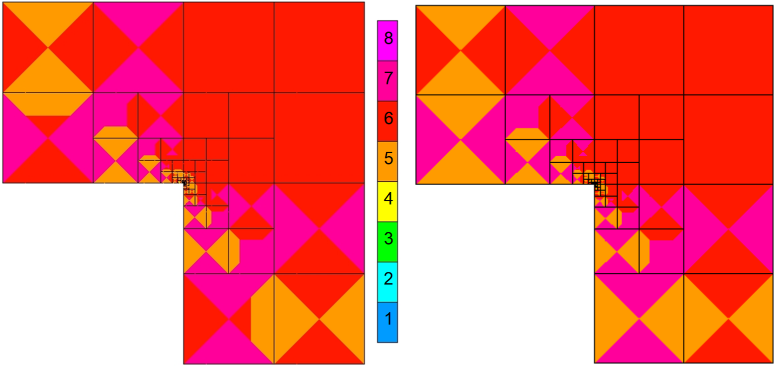


Figure: The mesh provided by the deterministic *hp*-FEM (left panel) and by the deep learning-driven *hp*-FEM (right panel) algorithms.

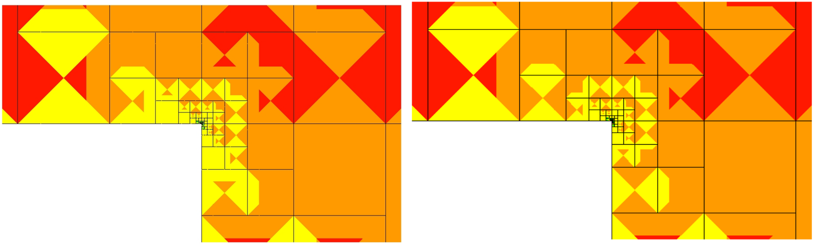


Figure: The mesh provided by the deterministic *hp*-FEM (left panel) and by the deep learning-driven *hp*-FEM (right panel) algorithms. Zoom 100 X

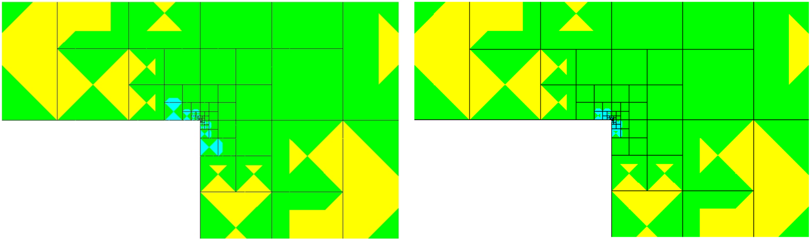


Figure: The mesh provided by the deterministic *hp*-FEM (left panel) and by the deep learning-driven *hp*-FEM (right panel) algorithms. Zoom 10,000 X

Deterministic and DNN driven *hp*-FEM

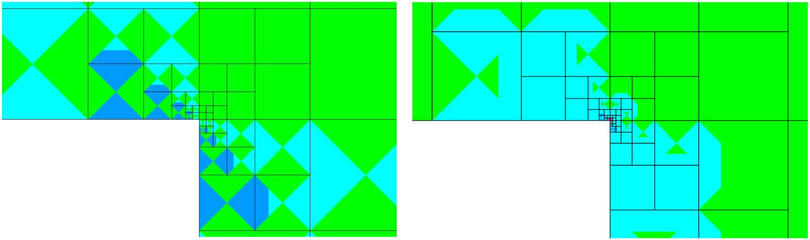


Figure: The mesh provided by the deterministic *hp*-FEM (left panel) and by the deep learning-driven *hp*-FEM (right panel) algorithms. Zoom 1,000,000 X

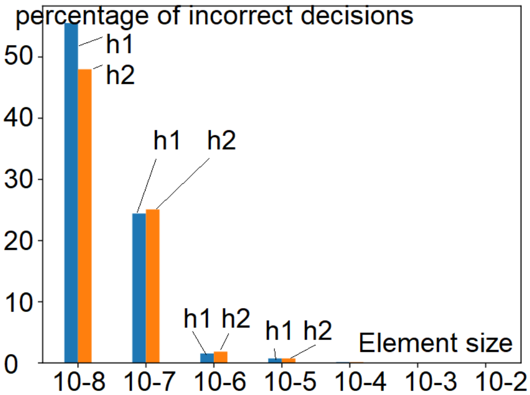


Figure: The sizes (horizontal h1 / vertical h2 directions) from 10^{-2} (right) down to 10^{-8} (left) of the elements where MPL network made incorrect decisions during verification.

Dataset

Input variables:

- coarse mesh solution $u_{hp} \in V_{hp}$ for element K,
- the element sizes and coordinates,
- the norm of the fine mesh solution over element K,
- the maximum norm of the fine mesh solution over elements

Output variables:

Optimal refinement V_{opt}^{K} for element K

Dataset

Construction of dataset:

- Executing of Algorithm 1 + Algorithm 2 for the model L-shape domain problem.
- 50 iteration of the *hp*-adaptivity generating over 10,000 deterministic element refinements, resulting in 10,000 samples.
- Repeat this operation for rotated boundary Neuman conditions. We obtain a total of 100,000 samples.
- We randomly select 90% of the samples for training and use the remaining 10% as a test set.
- One-hot encoding the categorical variables
- Supersampling of underrepresented *nref* classes.

Feed forward DNN with fully connected layers

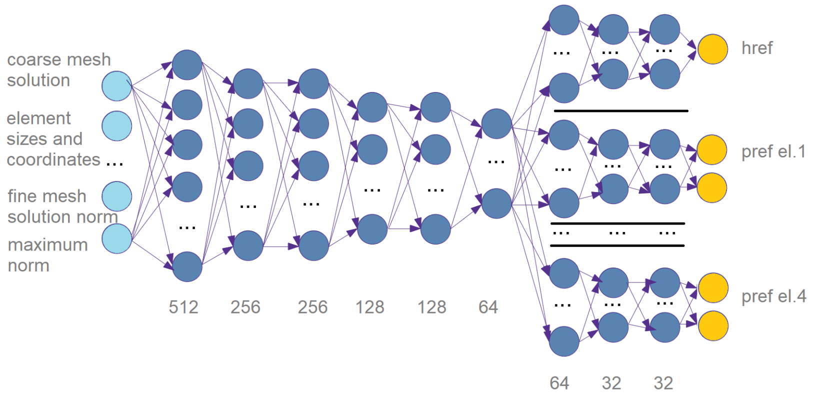


Figure: Feed-forward DNN with fully connected layers

Feed forward DNN with fully connected layers

Input	coarse mesh solution, element data, norm, max norm
Layer 1	512 nodes
Layer 2	256 nodes
Layer 3	256 nodes
Layer 4	128 nodes
Layer 5	128 nodes
Layer 6	64 nodes

Branch 1		Branch 2		...	Branch 6	
Layer 7	64 nodes	Layer 7	64 nodes	...	Layer 7	64 nodes
Layer 8	32 nodes	Layer 8	32 nodes	...	Layer 8	32 nodes
Layer 9	32 nodes	Layer 9	32 nodes	...	Layer 9	32 nodes
Output	h-ref	Output	p-ref el.1	...	Output	p-ref el.4

- After 8 layers, the DNN splits into 6 branches, 4 layers each:
- the first branch decides about the optimal *nref* parameter - *h*

Feed-forward DNN with 12 fully-connected layers

- Experiments have shown that further expanding of the network makes it prone to overfitting.
- Splitting the network into branches assures sufficient parameter freedom for each variable.
- This approach also simplifies the model: there is no need to train a DNN for each variable.
- ReLU as activation function,
- softmax as double precision to integer converter (as final activation function in *h*-ref branch)

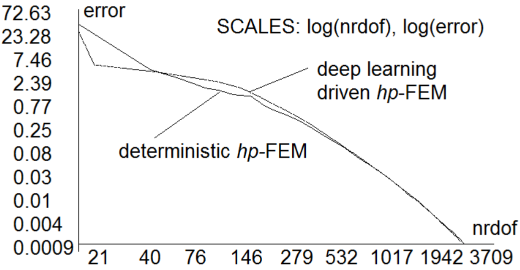


Figure: The comparison of deterministic and DNN *hp*-FEM on original L-shape domain.

DNN driven hp-FEM with modified b.c. (1/3)

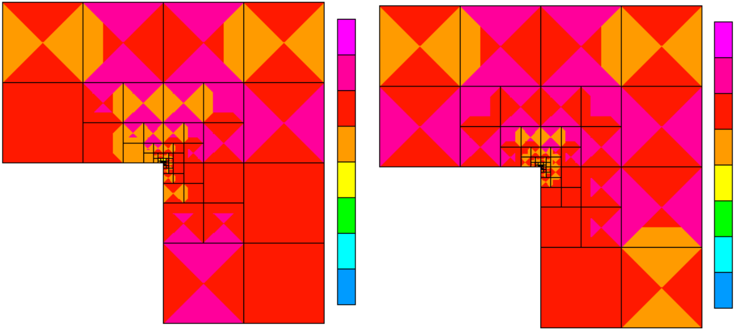


Figure: The deterministic and DNN hp-FEM algorithms for the L-shape with modified boundary condition. Final mesh Zoom X1

DNN driven hp-FEM with modified b.c. (2/3)

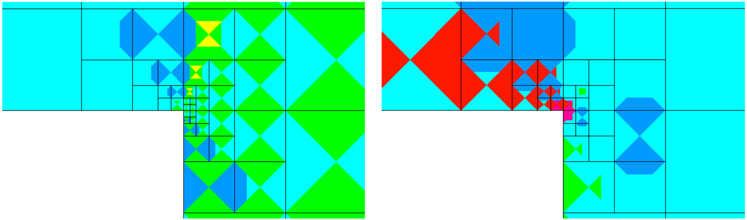


Figure: The deterministic and DNN hp-FEM algorithms for the L-shape with modified boundary condition. Zoom X100,000

DNN driven hp-FEM with modified b.c. (3/3)

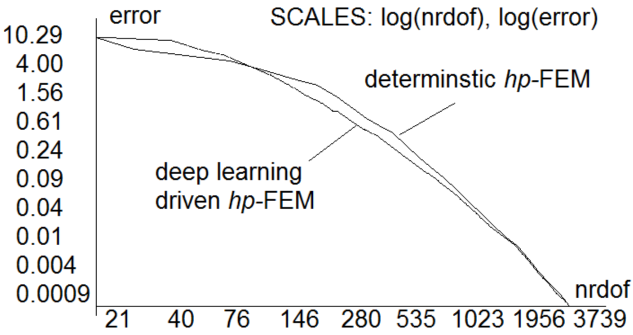


Figure: The convergence for deterministic and DNN hp-FEM algorithms for the L-shape with modified boundary condition.

Conclusions

- Deep Neural Networks can be employ to select optimal refinements over coarse mesh elements
- Self-adaptive *hp*-FEM algorithm still delivers exponential convergence if 10 percent of the decisions are wrong
- For a fixed singularity location, the element coarse mesh solution, element parameters and norms of the fine mesh solution over element are enough to teach DNN
- For a future work we plan to perform training based on coarse and fine mesh solution element data without providing element location