1D DG method

Etienne PEILLON

15/05/2019

1 Problem

We consider the following problem:

$$\alpha u - \mu \Delta u = f \tag{1}$$

on the interval $\Omega =]a, b[$ with one of the three following boundary conditions:

- Dirichlet conditions: $u(a) = u_a^D$ and $u(b) = u_b^D$;
- Neumann conditions: $u'(a) = u_a^N$ and $u'(b) = u_b^N$;
- Mixed conditions: $u(a) + \gamma_a u'(a) = u_a^M$ and $u(b) + \gamma_b u'(b) = u_b^M$.

2 Discontinuous discretization

To discretize the problem we will follow some step:

1. weak formulation of the problem compatible with the method
2. writing of the basic matrix
3. implementation

The goal of this report is simple: give a little overview of the DG method with understandable but extendable notations.

2.1 Weak formulation

Let be a subdivision $a = x_0 < x_1 < \cdots < x_{K-1} < x_K = b$ of I. We write $C_n =]x_{n-1}, x_n[$, so that we have K cells, and $\mathcal{E}_h = \{C_n\}$ where $h = \max_n |C_n|$. We also note $h_n = |C_n| = x_n - x_{n-1}$.

We consider the broken Sobolev space, highly depending on the subdivision of the domain (see Riviere):

$$H^1(\mathcal{E}_h) := \{v \in L^2(\Omega) \quad that \quad \forall E \in \mathcal{E}_h, v|_E \in H^1(E)\} \tag{2}$$

Denote that $H^1(\Omega) \subset H^1(\mathcal{E}_h)$.

The strong idea of discontinuous Galerkin method is to approximate the problem, not with an approximation of $H^1(\Omega)$ as for continuous Galerkin method, but with an approximation of $H^1(\mathcal{E}_h)$.

Let be $v \in H^1(\mathcal{E}_h)$, and multiply 1 by it and then use Green formula (also known as integration by parts in 1D). We take care that Green formula works only on each cell:

$$\sum_{n=1}^{K} \int_{C_n} (\mu \nabla u \nabla v + \alpha u v) d\omega - \int_{\partial C_n} \mu \nabla u \cdot \mathbf{n} \; v \; d\gamma = \sum_{n=1}^{K} \int_{C_n} f \; v \; d\omega \tag{3}$$

Here exists different methods in the literature. In Riviere, terms are added to create a bi-linear symmetric coercive and continuous form on left side and a linear conituous form on the right side, so that you can prove existence and uniqueness with functional analysis theorems. The approach is very interesting in the theorical way, but complicated for just a simple implementation.

On the other hand, you can use the following approach, which is less general, but more understandable as first view of the subject:

Because $v \in H^1(\mathcal{E}_h)$, we assume $u \in H^1(\mathcal{E}_h)$ for the resolution of the problem, but we want equality of the solution on the common edges of each cell, wich means we have a bad-defined value at $u'(x_n)$ in our case, wich is the *numerical flux*.

A solution consists on assuming numerical flux equals one unique value on cell frontiers:

$$\widehat{\nabla u(x)} \cdot \mathbf{n} = \beta \frac{[\![u]\!]}{\overline{\Delta x}} + \overline{\nabla u(x)} \cdot \mathbf{n}$$

where

- $[\![u]\!] = u^-(x) - u^+(x)$, u^- is the interior value and u^+ is the exterior value,
- $\overline{\nabla u(x)} = \dfrac{\nabla u^+(x) + \nabla u^-(x)}{2}$ with the same policy,
- $\overline{\Delta x} = \dfrac{\Delta x^+ + \Delta x^-}{2}$ with the same policy.

TRUE section *Finally, because $v \in H^1(\mathcal{E}_h)$, we have this equality for each elements:*

$$\int_{C_n} (\mu \nabla u \nabla v + \alpha u v) d\omega - \int_{\partial C_n} \mu \nabla u \cdot \mathbf{n} \; v \; d\gamma = \int_{C_n} f \; v \; d\omega$$

2.2 $H^1(\mathcal{E}_h)$ discretization

$H^1(\mathcal{E}_h)$ discretization is quite similar with the discretization of $H^1(\Omega)$. Let be a cell $C_k \in \mathcal{E}_h$, then consider $(\phi_n^k)_{n=0,\dots,N}$ a basis of polynomials of degree N on C_k. We assume $\phi_n^k|_{C_k^c} = 0$.

We call $\mathcal{D}_N(\mathcal{E}_h) = \text{span}\{\phi_n^k, \; 0 \le n \le N, \; 1 \le k \le K\}$ the approximation of $H^1(\mathcal{E}_h)$. Then we have to choose the type of basis function. Here is some possibilities:

- monomial functions,
- lagrangian functions,
- legendrian functions.

Obviously, each basis function is the image of a basis function on a reference element composed with an affine map. We call E the physical element and $\hat{E}$ the reference element, and

$$F_E : E \longrightarrow \hat{E} \qquad \text{and} \qquad F_E^{-1} : \hat{E} \longrightarrow E$$

the affine maps between E and $\hat{E}$.

Since those maps are affine, we have $|\det(\mathrm{D}F_E^{-1})| = |\det(\mathrm{D}F_E)|^{-1} = \dfrac{|E|}{|\hat{E}|}$.

We also express physical basis functions with reference basis functions with the formula

$$\phi_i^E = \sqrt{\frac{|\hat{E}|}{|E|}}\ \hat{\phi}_i \circ F_E$$

so that $\|\phi_i^E\|_{L^2(E)} = 1$ and $\|\hat{\phi}_i^E\|_{L^2(E)} = 1$.

For 1D case, the reference element will be the segment $\hat{E} = [-1, 1]$. Then the affine map will be:

$$\begin{array}{rccc} F_k^{-1}: & [-1,1] & \longrightarrow & [x_{k-1}, x_k] \\ & x & \longmapsto & \frac{h_k}{2}x + \frac{x_{k-1}+x_k}{2} \end{array} \quad \text{and} \quad \begin{array}{rccc} F_k: & [a,b] & \longrightarrow & [-1,1] \\ & y & \longmapsto & \frac{2}{h_k}\left(y - \frac{x_{k-1}+x_k}{2}\right) \end{array}$$

With this map, we now just have to describe the basis functions on the reference element to have every basis functions we need. So we define the basis functions on the reference element by $\hat{\phi}_n$ for $n = 0, \ldots, N$ and then we can write $\phi_n^k = \sqrt{\frac{2}{h_k}}\hat{\phi}_n \circ F_k$

2.3 discretization on each cell

In this part, we assume $u \in \mathcal{D}_N(\mathcal{E}_h)$ and it can be decomposed as:

$$u = \sum_{k=1}^{K} u^k = \sum_{k=1}^{K}\sum_{n=0}^{N} u_n^k \phi_n^k$$

where $\forall n, k,\ u_n^k \in \mathbb{R}$.

Then we can inject u in the equation 3. Then we define $U^k = (u_0^k, \ldots, u_N^k)^\top$ and $U = [U^1, \ldots, U^K]^\top$. The next idea is to project the equation 3 on $\mathcal{D}_N(\mathcal{E}_h)$.

2.3.1 right term

The right term of the equation is the easiest part of the discretization: we define $F = [F^1, \ldots, F^K]^\top$ and $F^k = (f_0^k, \ldots, f_N^k)^\top$ where $f_i^k = \int_{C_k} f\phi_i^k d\omega$.

2.3.2 discretization of the mass and stiffness term

Consider $\int_{C_E}(\mu\nabla u \cdot \nabla v + \alpha uv)d\omega$, and take $v = \phi_n^E$. Then we can discretize this integral by

$$\left(\alpha\mathbb{M}^E + \mu\mathbb{S}^E\right)U^E \tag{4}$$

where $\mathbb{M}_{i,j}^E = \int_{C_E}\phi_i^E\phi_j^E d\omega$ and $\mathbb{S}_{i,j}^E = \int_{C_E}\nabla\phi_i^E\nabla\phi_j^E d\omega$

Mass matrix: Firstly, because we have $\phi_n^E = \sqrt{\frac{|\hat{E}|}{|E|}}\hat{\phi}_n \circ F_E$,

$$\mathbb{M}_{i,j}^E = \int_{C_E}\phi_i^E\phi_j^E d\omega = \frac{|\hat{E}|}{|E|}\int_{\hat{E}}\hat{\phi}_i\hat{\phi}_j d\hat{\omega} \cdot |\det(\mathrm{D}F_E^{-1})| = \int_{\hat{E}}\hat{\phi}_i\hat{\phi}_j d\hat{\omega} = \widehat{\mathbb{M}}_{i,j}$$

Then we just have $\mathbb{M} = \mathrm{diag}(\widehat{\mathbb{M}}, \ldots, \widehat{\mathbb{M}})$.

Stiffness matrix: Secondly, we just derivate ϕ_i^E:

$$\mathrm{D}\phi_n^E = \sqrt{\frac{|\hat{E}|}{|E|}}\mathrm{D}\hat{\phi}_n(F_E) \circ \mathrm{D}F_E \qquad \text{and} \qquad \nabla\phi_n^E = \sqrt{\frac{|\hat{E}|}{|E|}}\mathrm{D}F_E{}^\top\nabla\hat{\phi}_n(F_E)$$

$$\begin{aligned}\mathbb{S}_{i,j}^E &= \int_{C_E}\nabla\phi_i^E \cdot \nabla\phi_j^E d\omega = \int_{\hat{E}}\nabla\hat{\phi}_i \cdot (\mathrm{D}F_E\mathrm{D}F_E{}^\top\nabla\hat{\phi}_j)d\hat{\omega} \\ &= \left(\mathrm{D}F_E{}^\top\nabla\hat{\phi}_i, \mathrm{D}F_E{}^\top\nabla\hat{\phi}_i\right)_{L^2(\hat{E})}\end{aligned}$$

Finally, we obtain $\mathbb{S} = \mathrm{diag}(\mathbb{S}^1, \ldots, \mathbb{S}^K)$.

2.3.3 Flux discretization

In the same way than Armand's report, we define $\mathbb{F}$ as the matrix describing the flux part. We also define the bound term $\mathbb{F}^b$. Those are the most complicated terms, wich will be treated here with less generality. But without loss of generality, we can write, for any $E \in \mathcal{E}_h$:

$$\int_{\partial E} \widehat{\nabla u_E} \cdot \mathbf{n}_E \; v \; d\gamma = \sum_{d=1}^{D} \int_{\partial E^d} \widehat{\nabla u_E} \cdot \mathbf{n}_E^d \; v \; d\gamma$$

where $\left\{\partial E^d, \; d = 1, \cdot, D\right\}$ are the faces of E.

Then we have two cases:

- ∂E^d is the frontier between two elements,
- ∂E^d is a part of the frontier of the domain Ω.

Therefore, we have two treat those two cases distinctly.

If ∂E^d is the frontier between two elements: We can make this separation:

$$\begin{aligned}\widehat{\nabla u_E} \cdot \mathbf{n}_E &= \beta \frac{[\![u]\!]}{\Delta x} + \overline{\nabla u(x)} \cdot \mathbf{n} \\ &= \underbrace{\left[-\beta \frac{u^+}{\Delta x} + \frac{\nabla u(x)^+ \cdot \mathbf{n}}{2}\right]}_{\text{interior contribution}} + \underbrace{\left[\beta \frac{u^-}{\Delta x} + \frac{\nabla u(x)^- \cdot \mathbf{n}}{2}\right]}_{\text{exterior contribution}}\end{aligned}$$

Then, if E^d is the neighbor of E by ∂E^d, we have to compute, for all $i, j = 0, \ldots, N$:

$$\int_{\partial E^d} \left(-\beta \frac{\phi_j^E}{\Delta x} + \frac{1}{2} \nabla \phi_j^E \cdot \mathbf{n}_E\right) \phi_i^E \; d\gamma \quad \text{and} \quad \int_{\partial E^d} \left(\beta \frac{\phi_j^{E^d}}{\Delta x} + \frac{1}{2} \nabla \phi_j^{E^d} \cdot \mathbf{n}_E\right) \phi_i^E \; d\gamma$$

If ∂E^d is a part of the frontier of the domain Ω: We have to consider the bound condition. We assume here this is a Dirichlet condition $u|_{\partial\Omega} = g$. That gives us:

- $u^- = g$,
- $\nabla u(x)^+ = \nabla u(x)^-$ and $\overline{\nabla u(x)} = \nabla u(x)^+$,

and

$$\widehat{\nabla u_E} \cdot \mathbf{n}_E = \underbrace{\left[-\beta \frac{u^+}{\Delta x} + \nabla u(x)^+ \cdot \mathbf{n}\right]}_{\text{flux contribution}} + \underbrace{\left[\beta \frac{g}{\Delta x}\right]}_{\text{bound contribution}}$$

Then we have to compute, for all $i, j = 0, \ldots, N$:

$$\int_{\partial E^d} \left(-\beta \frac{\phi_j^E}{\Delta x} + \nabla \phi_j^E \cdot \mathbf{n}_E\right) \phi_i^E \; d\gamma \quad \text{and} \quad \int_{\partial E^d} \beta \frac{g}{\Delta x} \phi_i^E d\gamma.$$

We assemble all those computations in two matrix $\mathbb{F}$ and $\mathbb{F}^b$.

2.3.4 Linear system

At the end wa assembly all the matrices and we obtain a system of this shape:

$$\left(\alpha \mathbb{M} + \mu(\mathbb{S} - \mathbb{F})\right) U = F + \mu \mathbb{F}^b \tag{5}$$

Then, to obtain U, we just have to invert the matrix $\alpha\mathbb{M} + \mu(\mathbb{S} - \mathbb{F})$ wich is not easy with high dimension, even with iterative algorithm. An idea of efficient iterative algorithm could be biconjugate gradient algorithm.

2.4 The Legendre polynomials

The purpose of this part is to exploit Armand's results on the legendre polynomials for the construction of the differents matrices. So we define $\hat{\phi}_i = \sqrt{\frac{2i+1}{2}} P_i$ where P_i is the i-th Legendre's polynomial.

2.4.1 Mass matrix

By their definition, the legendre polynomials are orthogonals for the scalar product $\int_{-1}^{1} uvdx$. So, the reference mass matrix is just identity matrix.

2.4.2 Stiffness matrix

The stiffness matrix computation is more complicated because of the derivative. With the definition of basis function, we can already write:

$$\widehat{\mathbb{S}}_{i,j} = \frac{\sqrt{(2i+1)(2j+1)}}{2} \left(P'_i, P'_j\right)_{L^2(]-1,1[)}$$

But due to Armand, the majority part of the computation is already done: $\forall i \leq j \in \mathbb{N}$

- $\left(P'_{2i}, P'_{2j+1}\right)_{L^2(]-1,1[)} = 0$
- $\left(P'_{2i}, P'_{2j}\right)_{L^2(]-1,1[)} = 2i(2i+1)$
- $\left(P'_{2i+1}, P'_{2j+1}\right)_{L^2(]-1,1[)} = 2i(2i+3)+2$

As final result, we have $\mathbb{S}^k_{i,j} = \frac{2}{{h_k}^2} \sqrt{(2i+1)(2j+1)} \left(P'_i, P'_j\right)_{L^2(]-1,1[)} = \frac{2}{{h_k}^2} \mathbb{P}_{i,j}$. Note that result is consistent with Armand's result (pay attention that $\|P_0\| = 2$).

2.4.3 Stiffness matrix

We need here to evaluate the basis functions on each frontier. The biggest difficulty is to evaluate effiently every quantity.

2.5 Results

2.6 Results with Dirichlet bound conditions

Here is the order table for the function $u(x) = \sin(5\pi x)$ on $[0,1]$:

	L^1	L^2	L^∞
N = 0	0.9983	0.9973	0.9999
N = 1	1.9980	1.9981	1.9975
N = 2	2.0186	2.0177	2.0188
N = 3	3.9858	3.9864	3.9769
N = 4	4.0496	4.0436	4.0397
N = 5	5.9529	5.9575	5.9269
N = 6	6,1840	6,1768	6,2155
N = 7	7.8522	7.8686	7.7576

Table 1: Convergence order with sine

2.7 Result with Neumann bound conditions

Here is the order table for the function $u(x) = \cos(5\frac{\pi}{2}x)$ on $[0,1]$:

Table 2: Order table for neumann condition

Polynomial degree	L^1	L^2	L^∞
0	1.0000	1.0000	0.9999
1	0.9999	0.9999	1.0000
2	2.0135	2.0345	2.1319
3	3.0000	2.9999	2.9999
4	4.6029	4.5676	4.4408
5	4.9913	4.9911	4.9627
6	6.9988	6.9866	7.0267
7	6.9621	6.9570	6.8508
8	8.9691	8.9660	8.9740
9	10.1759	10.1484	10.1337

3 New problem

The problem we want to compute now is the following:

$$\alpha u - \mu \Delta u = f + \zeta(g-u)\delta_\Gamma \quad \text{on } \Omega_S$$

where $\Gamma = \partial\Omega$, Ω is the previous domain and $\Omega_S \supset \Omega$ and δ_Γ the Dirac function.

We consider a discretization $\mathcal{E}_h$ of Ω_S and we follw the same previous steps. Let be $E \in \mathcal{E}_h$. We have:

$$\int_E (\mu \nabla u \nabla v + \alpha u v) d\omega - \int_{\partial E} \mu \nabla u \cdot \mathbf{n}\ v\ d\gamma = \int_E f\ v\ d\omega + \int_{E\cap\Gamma} \zeta(g-u) v d\gamma$$

We have here three cases:

- if $E \cap \Gamma = \emptyset$ and $\partial E \cap \partial\Omega_S = \emptyset$ then the previous construction is valid here;
- if $\partial E \cap \partial \Omega_S \neq \emptyset$ then a part of the bound has to be treated separatly;
- if $E \cap \Gamma \neq \emptyset$ then the last term of the previous equation has to be discretized.

3.1 if $\partial E \cap \partial\Omega_S \neq \emptyset$

We do not know any thing on the bound so we assume that $u^+ = u^-$ and $\nabla u^+ = \nabla u^-$. Then we have $[\![u]\!] = 0$, $\overline{\nabla u} = \nabla u^+$ and

$$\widehat{\nabla u} \cdot \mathbf{n} = \nabla u^+ \cdot \mathbf{n}.$$

The following computation is, for all $i, j = 0, \ldots, N$:

$$\int_{\partial E^b} \nabla \phi_j^E \cdot \mathbf{n}\ \phi_i^E d\gamma$$

3.2 if $E \cap \Gamma \neq \emptyset$

We separate the integral in two terms: one is being part of the matrix and the other one is being par of the right side term:

$$\int_{E\cap\Gamma} \phi_j^E\ \phi_i^E\ d\gamma \quad \text{and} \quad \int_{E\cap\Gamma} g\ \phi_i^E\ d\gamma$$

Table 3: Convergence order for $\sin(5\pi x)$ on $[0, 1]$ with simulation on $[-1, 2]$ and Dirichlet conditions.

Polynomial degree $\backslash\Delta x$	7.5000e-01	3.7500e-01	1.8750e-01	9.3750e-02	4.6875e-02	2.3438e-02	1.1719e-02	5.8594e-03	2.9297e-03	order
0	2.5771e-01	1.2337e-01	6.2111e-02	3.0447e-02	1.5508e-02	7.6018e-03	3.8763e-03	1.9003e-03	9.6872e-04	1
1	2.8726e-02	7.1678e-03	1.7770e-03	4.4329e-04	1.1078e-04	2.7682e-05	6.9217e-06	1.7305e-06	4.3271e-07	2
2	1.8808e-02	4.4547e-03	1.1571e-03	2.8455e-04	7.1772e-05	1.7861e-05	4.4737e-06	1.1166e-06	2.7912e-07	2
3	7.0969e-04	4.5592e-05	4.2294e-06	1.2078e-06	4.5735e-07	1.6258e-07	5.8284e-08	2.0619e-08	7.2893e-09	1.5
4	7.2828e-05	5.9566e-06	7.7374e-07	2.8381e-07	1.0197e-07	3.6105e-08	1.2872e-08	4.5556e-09	1.6030e-09	1.5
5	1.1295e-05	5.0285e-06	1.8062e-06	6.6673e-07	2.3669e-07	8.4565e-08	2.9932e-08	1.0592e-08	3.7397e-09	1.5

Table 4: Convergence order for $\sin(5\pi x)$ on $[0, 1]$ with simulation on $[-1, 2]$ and Neumann conditions.

Polynomial degree $\backslash\Delta x$	7.5000e-01	3.7500e-01	1.8750e-01	9.3750e-02	4.6875e-02	2.3438e-02	1.1719e-02	5.8594e-03	2.9297e-03
0	6.3688e-01	1.6169e+00	7.7916e-01	3.3372e-01	1.5706e-01	7.7195e-02	3.9184e-02	1.8830e-02	1.0209e-02
1	3.1313e+00	1.6167e+00	9.8497e-01	2.2751e-01	1.0665e-01	4.2794e-02	1.6363e-02	1.0167e-02	1.0163e-02
2	2.9299e+00	8.3396e-01	2.6811e-01	1.0072e-01	4.9207e-02	1.0082e-02	1.0151e-02	1.0167e-02	1.0169e-02
3	2.0346e+00	6.5900e-01	1.0057e-01	6.0847e-02	1.6653e-02	1.0183e-02	1.0177e-02	1.0171e-02	1.0169e-02
4	1.5252e+00	2.8842e-01	9.6649e-02	4.1397e-02	1.8250e-02	1.0179e-02	1.0174e-02	1.0170e-02	1.0169e-02
5	9.9681e-01	1.2989e-01	6.1748e-02	2.3232e-02	1.0182e-02	1.0175e-02	1.0171e-02	1.0169e-02	1.0168e-02

4 Armand's code

We check the Armand's code order with the optimal given β:

Polynomial order	0	1	2	3	4	5	6	7	8	9
β	1	2	4	6	10	14	20	26	34	42

and more generally, $\beta = \left\lfloor \frac{k^2}{2} \right\rfloor + 2$.
Here are the error on the solution, with $\alpha = 1$, $\mu = 1$, on $[0,1]^2$ and

$$u(x,y) = 100 + \sin(\pi x)\sin(\pi y) \quad , \quad f(x,y) = 100 + (1 + 2\pi^2)\sin(\pi x)\sin(\pi y).$$

Table 5: Armand's code orders

Δx		1/8	1/16		1/32		1/64	
degree		error	error	order	error	order	error	order
0	L^∞	0.3486	0.1856	0.91	0.09536	0.96	0.04846	0.98
	L^2	0.1943	0.08563	1.18	0.06093	0.49	0.02723	1.16
1	L^∞	2.712513e-02	6.633549e-03	2.03	1.634076e-03	2.02	4.051566e-04	2.01
	L^2	3.296159e-03	7.564157e-04	2.12	2.925713e-04	1.37	3.014780e-05	3.28
2	L^∞	3.351381e-03	8.123021e-04	2.04	1.977707e-04	2.04	4.865215e-05	2.02
	L^2	1.836942e-03	4.378152e-04	2.07	1.030663e-04	2.09	2.454416e-05	2.07
3	L^∞	2.398702e-05	1.915027e-06	3.65	8.438633e-08	4.50	7.234192e-09	3.54
	L^2	1.469243e-05	6.776494e-07	4.44	3.901747e-08	4.12	7.215828e-10	5.76
4	L^∞	1.526775e-06	8.971116e-08	4.09	5.448967e-09	4.04	3.346514e-10	4.03
	L^2	1.007513e-06	4.510937e-08	4.48	2.814988e-09	4.00	1.736744e-10	4.02
Δx		1/4	1/8		1/16		1/32	
5	L^∞	5.785312e-07	1.014580e-08	5.83	1.065530e-10	6.57	3.936407e-12	4.76
	L^2	2.959251e-07	3.494762e-09	6.40	4.885784e-11	6.16	1.341874e-12	5.19
Δx		1	1/2		1/4		1/8	
10	L^∞	1.516672e-08	6.025402e-12	11.30	3.225864e-12	0.90		
	L^2	5.529469e-10	3.070581e-12	7.49	1.443178e-12	1.09		
15	L^∞	3.252719e+06	6.494147e-06	38.87	1.283135e-07	5.66	2.117289e-09	5.92
	L^2	4.685713e+05	4.112094e-07	40.05	5.119214e-09	6.33	3.478102e-11	7.20
20	L^∞	Integral problem...						
	L^2							

This table has been established with the optimal values of β

5 New approach

We continue to simulate on a domain larger then the real domain, but we also truncate every integral. So if we note $\Omega = \cup \mathcal{C}$ the simulqtion domain and Ω^- the real domain, the integrals are expressed by $\int_{\Omega^- \cap \mathcal{C}} \cdot$.

There is no change for the middle elements but for the bounds element we have to take care. That means the mass matrix, the stiffness matrix, the flu matrix and the bound vector wil not have exactly the same expression.

We also take care about the proportion of the ratio $\frac{|\mathcal{C}^-|}{|\mathcal{C}|}$.

Table 6: matrix condition number cut

precision \ polynomial degree		0	1	2	3	4	5
Number of element	Δx						
1	1.0000e+00	0.9000	0.9000	0.9000	0.6561	0.7290	0.8100
2	5.0000e-01	0.0000	0.5314	0.6561	0.7290	0.8100	0.9000
4	2.5000e-01	0.0000	0.2059	0.4783	0.6561	0.7290	0.8100
8	1.2500e-01	0.0000	0.0985	0.3487	0.5905	0.6561	0.7290
16	6.2500e-02	0.0000	0.0424	0.2542	0.4783	0.5905	0.7290
32	3.1250e-02	0.0000	0.0182	0.1668	0.4305	0.5314	0.6561
64	1.5625e-02	0.0000	0.0087	0.1216	0.3487	0.4783	0.5905
128	7.8125e-03	0.0000	0.0042	0.0798	0.2824	0.4305	0.5314
256	3.9062e-03	0.0001	0.0020	0.0523	0.2288	0.3487	0.4783
512	1.9531e-03	0.0027	0.0011	0.0343	0.1853	0.3138	0.4305
1024	9.7656e-04	0.1094	0.0005	0.0225	0.1351	0.2542	0.3874

We observe that the thinner is the mesh, the better is the condition number of the matrix and the the higher is the polynimial degree, the worst is the condition number. This table has been made without correcting Δx in the estimate flux formula.

The corrected Δx is still the same: the mean of the domain of interest of the two cells.

Table 7: matrix condition number cut

precision \polynomial degree		0	1	2	3	4	5
Number of element	Δx						
1	1.0000	0.7290	0.9000	0.4305	0.6561	0.7290	0.6561
2	0.5000	0.0985	0.5314	0.6561	0.7290	0.8100	0.8100
4	0.2500	0.4783	0.1853	0.4783	0.6561	0.7290	0.8100
8	0.1250	0.4783	0.0718	0.3487	0.5314	0.6561	0.7290
16	0.0625	0.4783	0.0278	0.2288	0.4783	0.5905	0.6561
32	0.0312	0.4783	0.0108	0.1668	0.3874	0.5314	0.6561
64	0.0156	0.4783	0.0046	0.1094	0.3138	0.4783	0.5905
128	0.0078	0.4783	0.2824	0.0718	0.2542	0.3874	0.5314
256	0.0039	0.4783	0.2824	0.0471	0.2059	0.3487	0.4783
512	0.0020	0.4783	0.2824	0.0343	0.1501	0.2824	0.4305
1024	0.0010	0.4783	0.2824	0.0225	0.1216	0.2542	0.3874

To eliminate this phenomena, we are going to watch diffents approaches:

- normalization of basis functions;

- reducing the cells, by bounding box, and to simulate it with the one dimension the bounding box will be firstly all the cell and secondly 120% of the cell.

The test are made on the function $u(x) = sin(5\pi x)$ reshaped. We put an interface on the last cell and its position vary.

Here are the graphs of the error and the condition number by the ratio of interest cell on total cell.

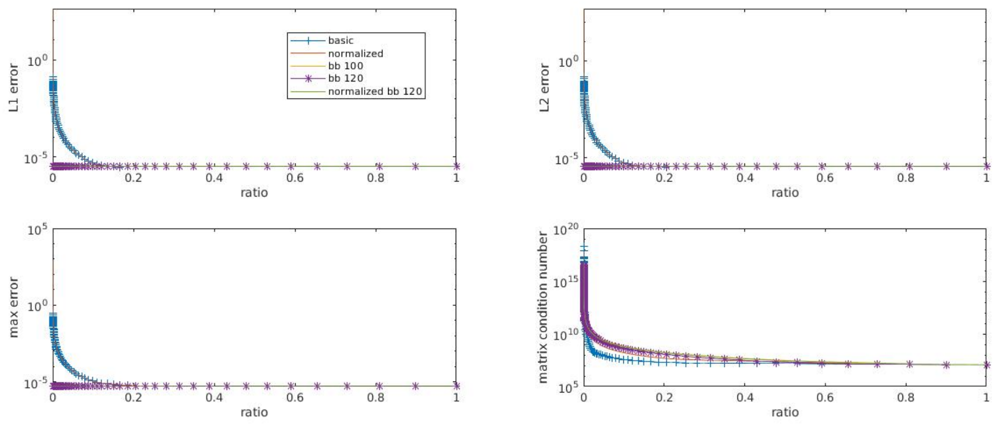


Figure 1: error and condition number by the interface position, polynomial degree 2, number of element 1024

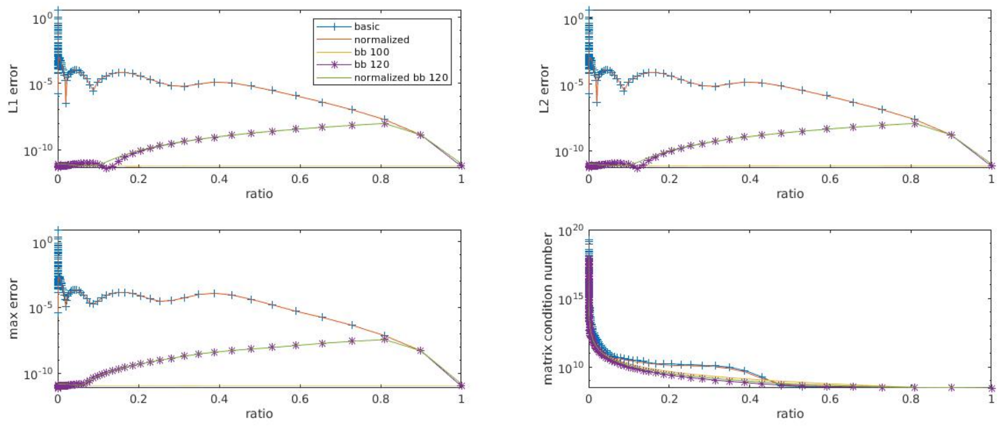
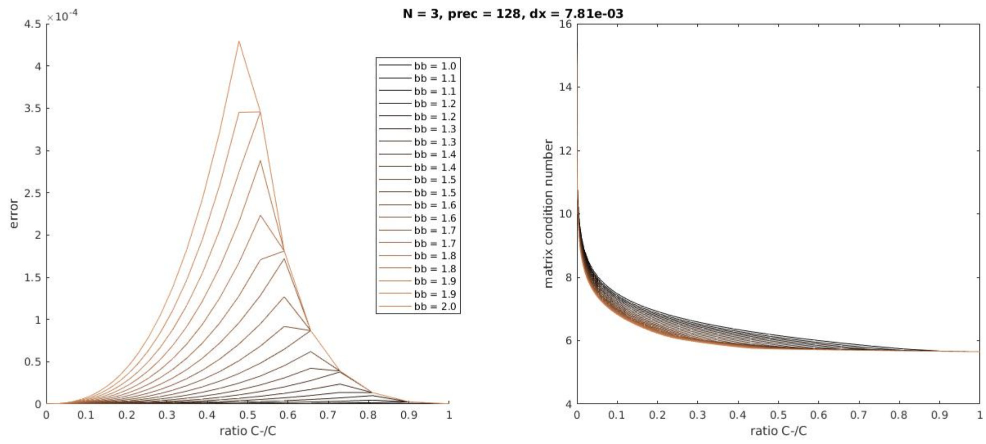


Figure 2: error and condition number by the interface position, polynomial degree 6, number of element 1024

We observe that every condition numbers increases in the same similar way. But we can see a big difference between the basic approach / normalized approach and the bounding box approach wich seems have better error results.

However, the methods does not seem to be high order method. For example, with 7-order polynomials and a bounding box of 120%, the method order is 3.5 average. There is no differences between the normalized and the basic method unless the higher matrix condition number in normalized method.

We also observed the effect of the size of the bounding box on the error and the conditionning number: we observe that larger the bounding box is, higher the error is, but lower the conditioning number is.


N = 3, prec = 128, dx = 7.81e-03
error
ratio C-/C
matrix condition number
bb = 1.0
bb = 1.1
bb = 1.1
bb = 1.2
bb = 1.2
bb = 1.3
bb = 1.3
bb = 1.4
bb = 1.4
bb = 1.5
bb = 1.5
bb = 1.6
bb = 1.6
bb = 1.7
bb = 1.7
bb = 1.8
bb = 1.8
bb = 1.9
bb = 1.9
bb = 2.0

6 Coming back on the new approach

We still look on the ratio $\dfrac{|\mathcal{C}^-|}{|\mathcal{C}|}$, but also on the mesh. We compare two subdivision generation method on an interval with an interface and a bounding box.

The interval of simulation is $]0,1[$ and the interval of interest is $]0,0.7[$. We create meshes with a bounding box of 120% on the edges. For example, if the original edge is $[0.5,1]$, then the bounding box process transforms this edge in $[0.5,0.94]$.

Here is a description of the two mesh generation processes:

(a) We create a subdivision of $[0,1]$, then we prune the cells totaly outside of the interest interval. Finally, we compute the bounding box process on the cells containing the interface.

(b) We create a subdivision of $[0,1]$, then we alternatively we prune the outside cells, we compute the bounding box process and either we divide all the cells into two cells and we repeat the previous process, or we stop the process because we have the precision needed.

The two processes seems the same but it gives really differents results, as for the generated meshes than for the convergences results.

The following tables are two examples of the generated meshes:

Table 8: Meshes generated by process (a) with an interface at 0.7 and a 120%-bounding box.

initial interval	[0,1]						
after bounding box	[0,0.84]						
dx	0.8400						
initial interval	[0,0.5]				[0.5,1]		
after bounding box					[0.5,0.74]		
dx	0.5000				0.2400		
initial interval	[0,0.25]		[0.25,0.5]		[0.5,0.75]		pruned cell
after bounding box					[0.5,0.74]		
dx	0.25		0.25		0.24		
initial interval	[0,0.125]	[0.125,0.25]	[0.25,0.375]	[0.375,0.5]	[0.5,0.625]	[0.625,0.75]	[0.75,1]
after bounding box						[0.625,0.715]	
dx	0.1250	0.1250	0.1250	0.1250	0.1250	0.0900	

Table 9: Meshes generated by process (b) with an interface at 0.7 and a 120%-bounding box.

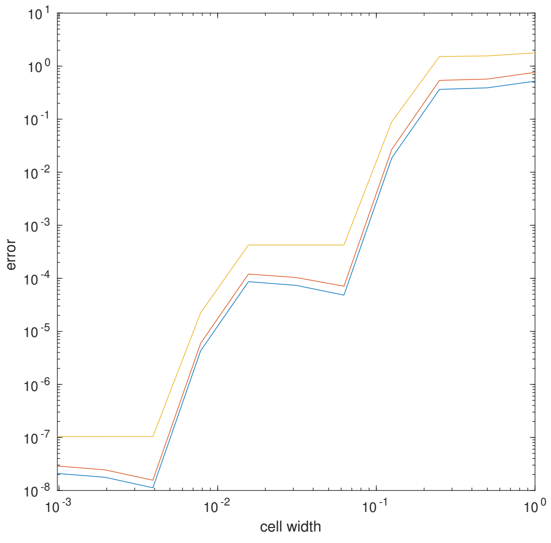
initial interval	[0,1]							
after bounding box	[0,0.84]							
dx	0.84							
initial interval	[0,0.42]				[0.42,0.84]			
after bounding box					[0.42,0.756]			
dx	0.42				0.336			
initial interval	[0,0.21]		[0.21,0.42]		[0.42,0.588]		[0.588,0.756]	
after bounding box							[0.588,0.7224]	
dx	0.21		0.21		0.168		0.1344	
initial interval	[0,0.105]	[0.105,0.21]	[0.21,0.315]	[0.315,0.42]	[0.42,0.504]	[0.504,0.588]	[0.588,0.6552]	[0.6552,0.756]
after bounding box								[0.6552,0.709]
dx	0.105	0.105	0.105	0.105	0.084	0.084	0.0672	0.0538

We observe two major differences between the two methods:

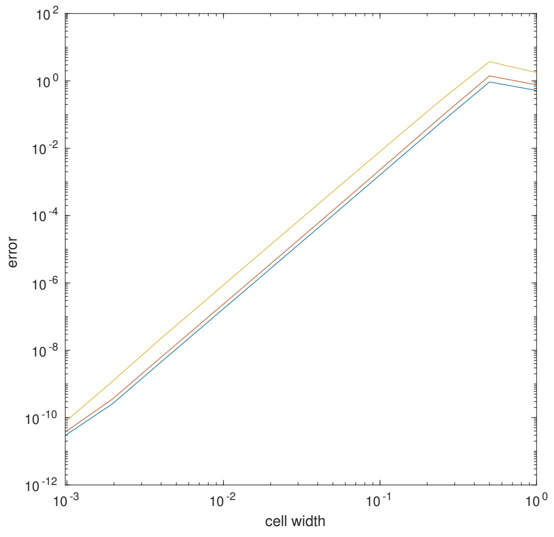
- The second method has no pruned cell, and therefore, the number of cell is always 2^{depth}.
- The cell widths has not the same monotony properties: let be D the depth, dx_i^D for the width of the initial interval, and dx_b^D the cell width after the bounding box process. The important width we want to control is dx_b. Then, with the process (a), $dx_b^D \leq \frac{1}{2} dx_i^{D-1}$ whereas the process (b) gives $dx_b^D \leq \frac{1}{2} dx_b^{D-1}$ which is better.

We can easily very bad cases where we have actualy no control, for example if we put the interface at $0.5 + \epsilon$. Then the final cell will always be $[0.5, 0.5 + 1.2\epsilon]$, wich is really bad because we cannot make any refinment in the interface region.

The results are obvious:



(a) convergence result with the process (a), with an average order of 3.75.



(b) convergence result with the process (a), with an average order of 3.96.

Figure 3: Convergence comparison

In 1D, the processes seems quite similar, but in 2D, the differences are obvious:

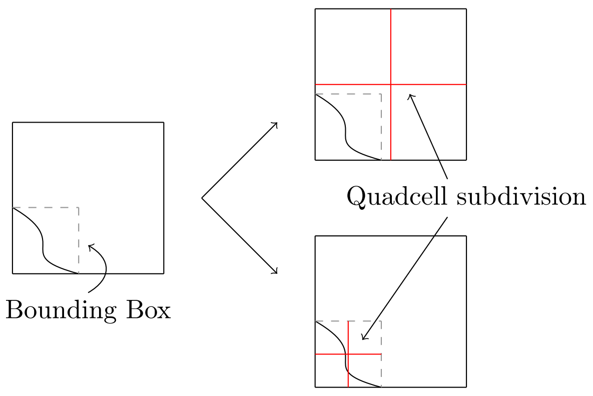


Figure 4: The two meshes generation method

Table 10: Convergence table

polynomial order	0	1	2	3	4	5	6	7	8	9	10
convergence order	1.13	2.19	2.06	3.97	4.01	3.87	3.96	3.96	3.98	3.96	4.01