Mesh adaptation via shape optimization

Zoubida Mghazli[1]

[1] Université Ibn Tofail, Kenitra, Maroc,

collaborators

A. Alla [1], M. Fortin[2], F. Hecht[3], M. Masmoudi[4]

[2] Université Laval, Québec , [3] Université Pierre et Marie Curie, Paris,
[4] Université Paul Sabatier, Toulouse

Wednesday, July 7, 2010

Workshop on A posteriori error estimates and mesh adaptivity for evolutionary and nonlinear problems

Introduction

The shape optimization problems are generally in the form

$$\min_{\Omega \in \mathcal{O}_{ad}} j(\Omega)$$

where

- $j(\Omega) = J(\Omega, u_\Omega)$ is a fonctional of the domain Ω
- $\Omega \in \mathcal{O}$, a set of admissible domains
- u_Ω solution of a PDE in Ω.

If we consider a functional $J(\cdot)$ which measures the error made in approaching the PDE by a finite element method built on a mesh $\mathcal{T}_h$, this raises the question of whether one can obtain an optimal mesh in the sense that minimizes the functional $J(\cdot)$

$$J(u_h) = J(u_h(\mathcal{T}_h)) = j((\mathcal{T}_h))$$

$$\min_{\mathcal{T}_h \in \mathcal{T}_{ad}} j((\mathcal{T}_h))$$

The mesh adaptation procedure is based on a number of local operations on the initial mesh:

- node displacement,
- edge refinement,
- node suppression,
- edge swapping.

This sequence is repeated a number of times until the mesh stabilizes.

The decision whether a given local operation has to be performed is based on the definition of the error on an element K and on its gradient. The local operations are used with two objectives: control the error level on each element so that the absolute error takes the desired value e_d everywhere. The second objective is to achieve equidistribution of this error by minimizing the error gradient.

If we fixe the number of nodes of a triangulation $\mathcal{T}_h$, we prove in the first part of this presentation that the optimal position of the nodes can be obtained by minimizing the approximation error using a hierarchical estimator. The node displacement procedure (also called R-adaptation)which is a crucial step in a global mesh adaptation procedure, is obtained by the derivative of the functional with respect to the triangulation $\mathcal{T}_h$.

The usual techniques for shape optimization exclude change topological domain and does not allow the creation of holes.

Topological optimization is concerned with the variation of a cost function with respect to a topological modification of a domain. The most simple way of modifying the topology consists in creating a small hole in the domain. In the case of structural shape optimization, creating a hole means simply removing some material.
For $\varepsilon \geq 0$ given let $\Omega_\varepsilon = \Omega \setminus \overline{(x_0 + \varepsilon\omega)}$ be the subset obtained by removing a small part $\overline{(x_0 + \varepsilon\omega)}$ from Ω, where $x_0 \in \Omega$ and $\omega \subset \mathbb{R}^n$ is a fixed open and bounded subset containing the origin.

$$\mathcal{O}_{ad} = \{\Omega_\varepsilon, \varepsilon \geq 0\}$$

Generally, the functional $j(\cdot)$ is not differentiable functional with respect to ε at $\varepsilon = 0$, An asymptotic expansion of the function $j(\cdot)$ is obtained in the following form:

(J. Céa, S. Garreau, PH. Guillaume, M. Mamsoudi, 2000)

$$j(\Omega_\varepsilon) = j(\Omega) + \rho(\varepsilon) g(x_0) + o(\rho(\varepsilon)).$$

$$\lim_{\varepsilon \longrightarrow 0} \rho(\varepsilon) = 0, \ \rho(\varepsilon) > 0.$$

The function $g(\cdot)$ is called the topological sensitivity or topological gradient. It can be used as a descent direction of the domain optimization process. Obviously, if we want to minimize $j(\cdot)$, the "best" place to create an infinitesimal hole is there where $g(x)$ is the most negative. Starting with this observation, topological optimization algorithms can then be constructed.

In the context of mesh adaptation, the edge refinement and node suppression ares local operations that can be seen as a change of mesh topology.

In the second part of this talk, we present some ideas to deal the mesh refinement as a topological optimization.

Outline

- The model problem
- The node displacement as a Shape optimization problem
- Refinement via an asymptotic expansion
- Conclusion and perspective

• Model problem

$u \in \mathcal{V} = H_0^1(\Omega)$ such that $a(u,v) = L(v)\ \forall v \in \mathcal{V}$

$$a(u,v) := \int_\Omega (\nabla u \cdot \nabla v + \alpha u v)\, dx, \quad L(v) := \int_\Omega f v dx.$$

$\mathcal{T}_h$ triangulation of Ω.

$$\mathcal{V}_h^k = \{v_h \in \mathcal{V}; v_{h|K} \in P_k(K), \forall\ K \in \mathcal{T}_h\} \cap H_0^1(\Omega)$$

$$u_h \in \mathcal{V}_h^k \text{ tel que } a(u_h, v_h) = L(v_h), \quad \forall v_h \in \mathcal{V}_h^k.$$

$e = u - u_h$ approximation error

Can we determine the "optimal" mesh in the sense that minimizes the approximation error??

The first ideas in this direction are due to M. C. Delfour, G. Payré, J.P. Zolesio, (1986). They considered the energy functional as a cost functional .

For two triangulations $\mathcal{T}_h$ and $\mathcal{T}_{h'}$ they showed, with the help of directional derivatives, that respective approximate solutions verify

$$J(u_h) = J(u) + \frac{1}{2}a(u_h - u, u_h - u)$$

$$\|u - u_h\|^2 - \|u - u_{h'}\|^2 = 2\,(J(u_h) - J(u_{h'}))$$

so minimizing the error becomes minimizing J .

Il seems that the "best" representation of the error is the hierarchical estimator.

Hierarchical estimator

The hierarchical method is based on a simple idea: given an approximation of order k, a better approximation, of order $k+1$ could be used to assess the accuracy of the solution. If now $\mathcal{V}_h^k \subset \mathcal{V}$ denotes the discrete space of piecewise continuous polynomials of degree k, the approximate solution is obtained by finding $u_h^k \in \mathcal{V}_h^k$ such that:

$$a(u_h^k, w_h^k) = L(w_h^k), \qquad \forall w_h^k \in \mathcal{V}_h^k.$$

The error $e^k = u - u_h^k$ is then the solution of:

$$a(e^k, w) = R_h(w), \qquad \forall w \in \mathcal{V}$$

where $R_h(w) = L(w) - a(u_h^k, w)$ is the residual.

An approximation of this error e^k can be obtained by considering the space $\mathcal{V}_h^{k+1}$ of piecewise continuous polynomials of degree $k+1$ and by solving:

$$a(e_h^{k+1}, w_h^{k+1}) = R(w_h^{k+1}), \qquad \forall w_h^{k+1} \in \mathcal{V}_h^{k+1}.$$

This would imply computing a full problem in the space $\mathcal{V}_h^{k+1}$, which would obviously be prohibitive.

The idea is then to use a hierarchical finite element basis of the space $\mathcal{V}_h^{k+1}$:

$$\mathcal{V}_h^{k+1} = \mathcal{V}_h^k \oplus E_h^{k+1}$$

so that $\mathcal{V}_h^{k+1}$ is the direct sum of $\mathcal{V}_h^k$ (polynomials of degree k) and a correction space E_h^{k+1} consisting of polynomials of degree $k+1$.

The error can then be approximated by solving the following problem: Find $\hat{\boldsymbol{e}}_h^{k+1} \in \boldsymbol{E}_h^{k+1}$ such that

$$\boldsymbol{a}(\hat{\boldsymbol{e}}_h^{k+1}, \bar{\boldsymbol{w}}_h^{k+1}) = \boldsymbol{R}(\bar{\boldsymbol{w}}_h^{k+1}), \qquad \forall \bar{\boldsymbol{w}}_h^{k+1} \in \boldsymbol{E}_h^{k+1}.$$

In the following we consider the case where a piecewise linear approximation which is corrected by a piecewise quadratic one, only the mid-edge nodes are computed, leaving unchanged the values at the vertices.

We drop the index $\boldsymbol{k+1}$ and write henceforth

$$\begin{cases} \boldsymbol{e}_h \in \boldsymbol{E}_h \\ \boldsymbol{a}(\boldsymbol{e}_h, \hat{\boldsymbol{w}}_h) = \boldsymbol{R}_h(\hat{\boldsymbol{w}}_h) \qquad \forall\, \hat{\boldsymbol{w}}_h \in \boldsymbol{E}_h. \end{cases}$$

This method has been analysed for a wide class of problems. The typical result is global. Under two principal hypothesis one can show an estimate of the following form

(R.E.Bank, R.K. Smith,1993): Under

(1) Saturation assumption

$$\exists \beta, \quad 0 < \beta < 1 \quad \text{tel que} \quad |||u - u_h^{(k+1)}||| \leq \beta \, |||u - u_h^{(k)}|||$$

(2) Cauchy-Binyakowski-Schwarz inegality (ou strong Cauchy-Schwarz inegality)

$$\begin{cases} \exists \gamma < 1 \quad \text{independant of } h, \quad \text{such that} \\ \forall v \in \mathcal{V}_h^k \quad \forall w \in E_h \;\; |a(v,w)| < \gamma \; |||v||| \; |||w|||, \end{cases}$$

We have $(1-\beta^2)(1-\gamma^2)|||u-u_h^{(k)}|||^2 \;\leq\; |||e_h|||^2 \;\leq\; |||u-u_h^{(k)}|||^2.$

Without saturation assumption (A. Agouzal, 2002)

$\mathcal{V}_b = \mathcal{V}_h \oplus \{b_T, T \in \mathcal{T}_h\}$

$$\|u - u_h\| \leq C_0 \|\epsilon_h\| + C \left(\sum_{K \in \mathcal{T}_h} h_K^2 \|f - f_K\|_{0,K}^2 \right)^{1/2}$$

$$\forall K \in \mathcal{T}_h, \ \|\epsilon_h\| \leq C_1 \|u - u_h\|_{1,\Delta(K)}.$$

ϵ_h local representation .

• The node displacement

(a Shape optimization problem)

A.Alla, M. Fortin, F. Hecht, Z.M.,: " *R-Adaptation de maillage par l'estimateur d'erreur hiérarchique*", ARIMA, Volume 5, p1-10, 2006.

$\longrightarrow$ For $\mathcal{T}_h$ given:

$$u_h = u_h(\mathcal{T}_h), \text{ and } e_h = e_h(\mathcal{T}_h)$$

Objective: **the best position nodes**.
$\longrightarrow$ minimize error approximation via the error estimator hierarchical

Let

$$\boldsymbol{J(e_h(\mathcal{T}_h)) := \frac{1}{2}\int_{\Omega_h} |\nabla e_h(\mathcal{T}_h)|^2 = \frac{1}{2}|||e_h(\mathcal{T}_h)|||^2}$$

the minimization problem is

$$\left\{\begin{array}{l} \text{Find the nodes position of } \mathcal{T}_h \text{ that minimizes } J(e_h(\mathcal{T}_h)), \\ \ \text{under the constraints} \\ \left\{\begin{array}{ll} a(e_h, \hat{v}_h) = L(\hat{v}_h) - a(u_h, \hat{v}_h) & \forall\ \hat{v}_h \in E_h \\ a(u_h, v_h) = L(v_h) \quad\quad \forall\ v_h \in \mathcal{V}_h. \end{array}\right. \end{array}\right. \tag{1}$$

$\mathcal{M}$: the set of nodes of $\mathcal{T}_h$
$\mathcal{T}_T$ the set of triangulations $\mathcal{T}_h(\mathcal{M})$ generated by $\mathcal{M}$ with the same topological table .
$m_T = \{\mathcal{M} : \mathcal{M} \subset \overline{\Omega} / \quad \mathcal{T}_h(\mathcal{M}) \; \in \mathcal{T}_T \; \},$

and let

$$j(\mathcal{M}) := J(e_h(\mathcal{T}_h(\mathcal{M}))$$

The shape optimization problem is:

$$\left\{ \begin{array}{l} \text{Find}\widehat{\mathcal{M}} \in m_T \text{ such that :} \\ j(\widehat{\mathcal{M}}) = \inf\limits_{\mathcal{M} \in m_T} j(\mathcal{M}), \quad with \\ \left\{ \begin{array}{l} a(e_h(\mathcal{T}_h(\mathcal{M})), \hat{v}_h) = L(\hat{v}_h) - a(u_h(\mathcal{T}_h(\mathcal{M})), \hat{v}_h) \quad \forall \; \hat{v}_h \in E_h \\ a(u_h(\mathcal{T}_h(\mathcal{M})), v_h) = L(v_h) \qquad \forall \; v_h \in \mathcal{V}_h \end{array} \right. \end{array} \right.$$

We assume that the last problem admits a unique solution. To calculate the derivative of the functional $j(\cdot)$ with respect to shape parameters which are the nodes of the mesh, we introduce the Lagrangian to transform the constrained problem to a problem without constraints.

The Lagrangian in $\boldsymbol{E}_h \times \boldsymbol{\mathcal{V}}_h \times \boldsymbol{E}_h \times \boldsymbol{\mathcal{V}}_h$ is given by

$$\begin{aligned}\mathcal{L}(\hat{v}_h, v_h, q_h, \mu_h) &= J(\hat{v}_h) + \ [a(\hat{v}_h, q_h) - L(q_h) + a(v_h, q_h)] \\ &\quad + [a(v_h, \mu_h) - L(\mu_h)].\end{aligned}$$

Dual problems are given by:

$$\left\{ \begin{array}{l} \text{Find } \ \boldsymbol{p_h} \in \boldsymbol{E_h}, \text{ such that} \\ \displaystyle\int_{\boldsymbol{\Omega_h}} \boldsymbol{\nabla p_h} \, \boldsymbol{\nabla \hat{v}_h} \, \boldsymbol{d\Omega_h} = - \int_{\boldsymbol{\Omega_h}} \boldsymbol{\nabla e_h} \, \boldsymbol{\nabla \hat{v}_h} \, \boldsymbol{d\Omega_h} \qquad \forall \boldsymbol{\hat{v}_h} \in \boldsymbol{E_h}. \end{array} \right.$$

$$\left\{ \begin{array}{l} \text{Find } \ \boldsymbol{\lambda_h} \in \boldsymbol{\mathcal{V}_h}, \text{ such that} \\ \displaystyle\int_{\boldsymbol{\Omega_h}} \boldsymbol{\nabla \lambda_h} \, \boldsymbol{\nabla v_h} \, \boldsymbol{d\Omega_h} = - \int_{\boldsymbol{\Omega_h}} \boldsymbol{\nabla p_h} \, \boldsymbol{\nabla v_h} \, \boldsymbol{d\Omega_h} \qquad \forall \boldsymbol{v_h} \in \boldsymbol{\mathcal{V}_h}, \end{array} \right.$$

where the notation $\boldsymbol{\Omega_h}$ means that the domain is meshed with the triangulation $\boldsymbol{\mathcal{T}_h}$.

To calculate the shape derivative of $j(\cdot)$ we use the techniques of Zolezio-Delfour-Sokolowski . The variation of the domain Ω is defined through a regular transformation $T_t : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ given by a velocity field V and such that

$$T_t(\Omega_h) = \Omega_h^t, \qquad T_t(\mathcal{M}) = \mathcal{M}^t$$

we note $e_t := e_{ht} = e_h^t \circ T_h^{-1} \in E_h^t$.

For $t > 0$:

$$j(\mathcal{M}^t) \;=\; \frac{1}{2}\int_{\Omega_h^t} |\nabla e_t|^2 \, d\Omega_h^t = \frac{1}{2}\int_{\Omega_h} \Big(A(t)\, \nabla e_h^t, \nabla e_h^t \Big) \; d\Omega_h$$

where

$A(t) = DT_t^{-1} \, J_t \, {}^*DT_t^{-1}$,

$J_t = det(DT_t)$

$A^{'}(0) = divV(0)\, I \;-\; (DV(0) \;+\; {}^*(DV(0)))$

Using this change of variable, the dual problems and the shape derivative of bilinear forms we obtain the explicit formula for the derivative of the fonctional

$$\begin{aligned} dj(\Omega_h; V) \; = \; & \frac{1}{2} \int_{\Omega_h} \left(A'(0)\, \nabla e_h, \nabla e_h \right) \, d\Omega_h \; - \; \int_{\Omega_h} div\,(f V(0)) \\ & (p_h + \lambda_h)\, d\Omega_h + \int_{\Omega_h} A'(0)\; \nabla u_h \nabla \lambda_h \, d\Omega_h \\ & + \int_{\Omega_h} A'(0)\; \nabla (u_h + e_h)\; \nabla p_h \, d\Omega_h \end{aligned}$$

We use the velocity fields, continuous and piecewise linear (which transforms the triangles into triangles) introduced by Delfour-Payré-Zolezio (1985) and defined from $\{\phi_i\}_i$ basis of $\mathcal{V}_h$.

We take successively $V_1 = (\phi_i, 0)$ and $V_2 = (0, \phi_j)$ to compute de partial derivative of J

Algorithm

Data: ε, initial mesh $\mathcal{T}_h^{(0)}$.

Step 1: computation of the approximate solution u_h.

Step 2: computation of the estimator e_h and the functional j.

Step 3: computation of the dual solutions p_h et λ_h.

Step 4: For every node do

- computation of $dj(\mathcal{M}, V_1)$ and $dj(\mathcal{M}, V_2)$ où $V_1 = (\phi_i, 0)$ et $V_2 = (0, \phi_i)$;

- calcul de $\|dj(\mathcal{M}, V)\| = ((dj(\mathcal{M}, V_1))^2 + (dj(\mathcal{M}, V_2))^2)^{\frac{1}{2}}$.

Etape 5: $test :=$ maximum of $\|dj(\mathcal{M}, V)\|$;

if $test > \varepsilon$

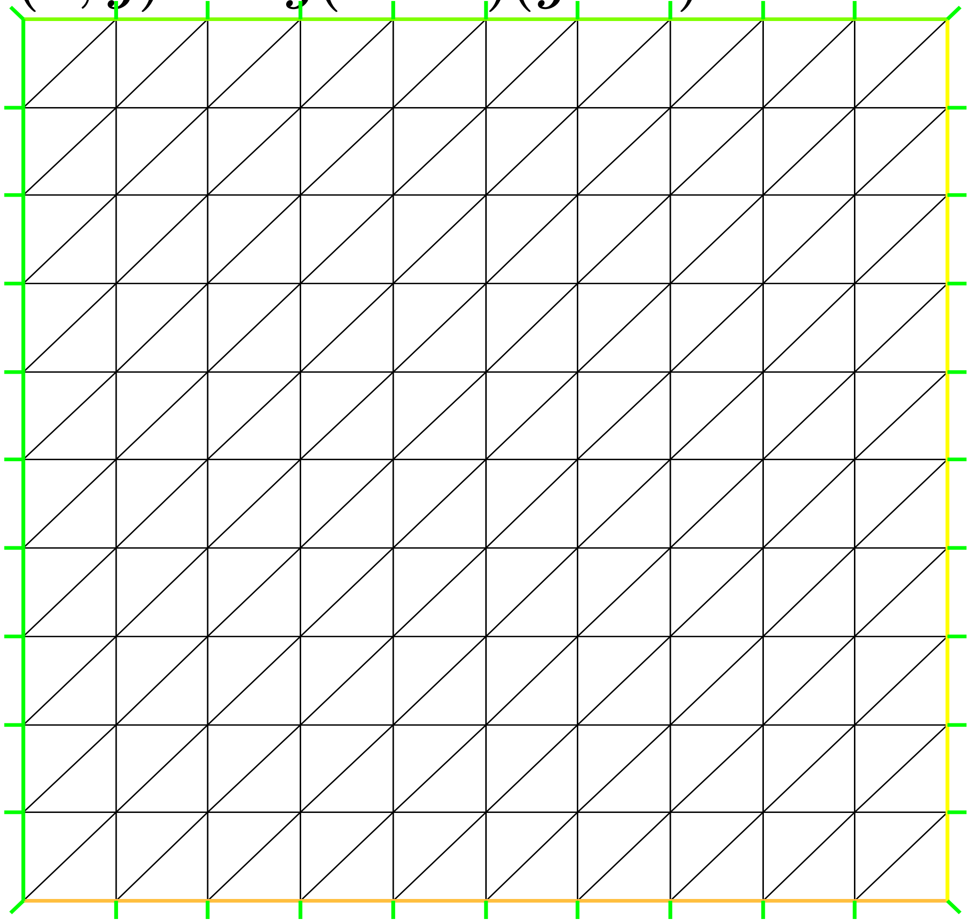
- displacement of the node in the direction of $dj(\mathcal{M}, V)$, (new triangulation $\mathcal{T}_h^{(n)}$);

-go to step 1.

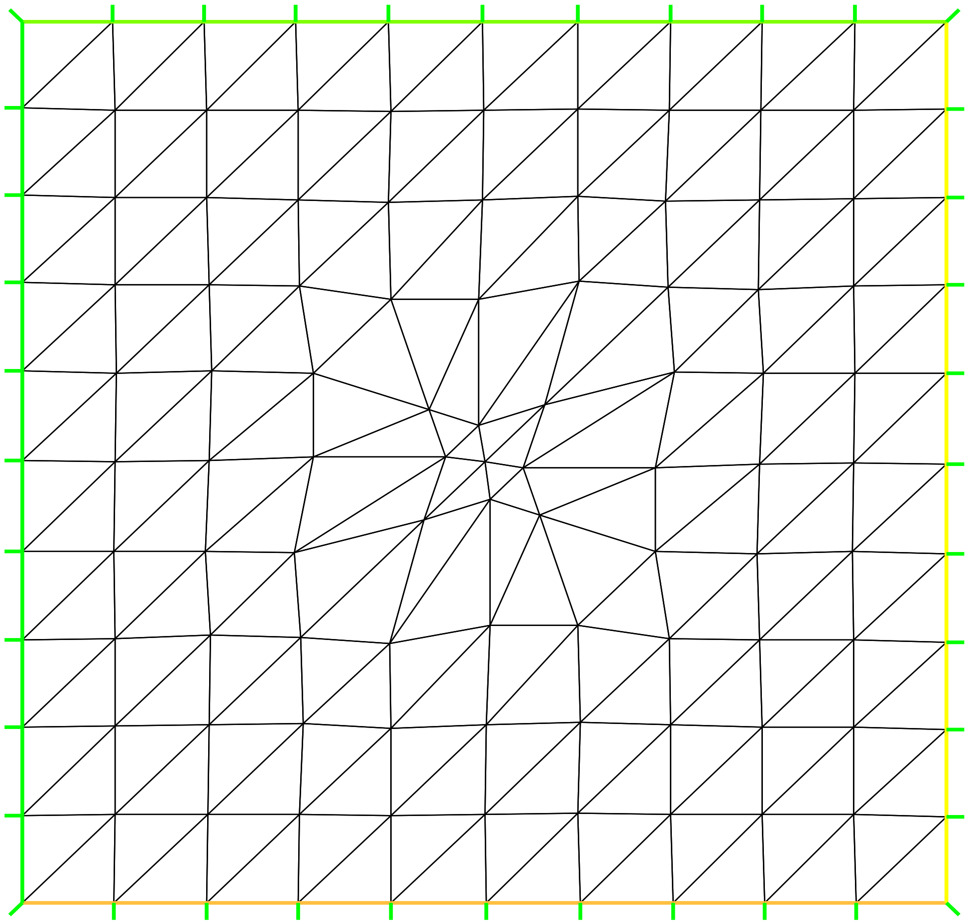
otherwise STOP.

Numerical tests: A manufactured solution given by

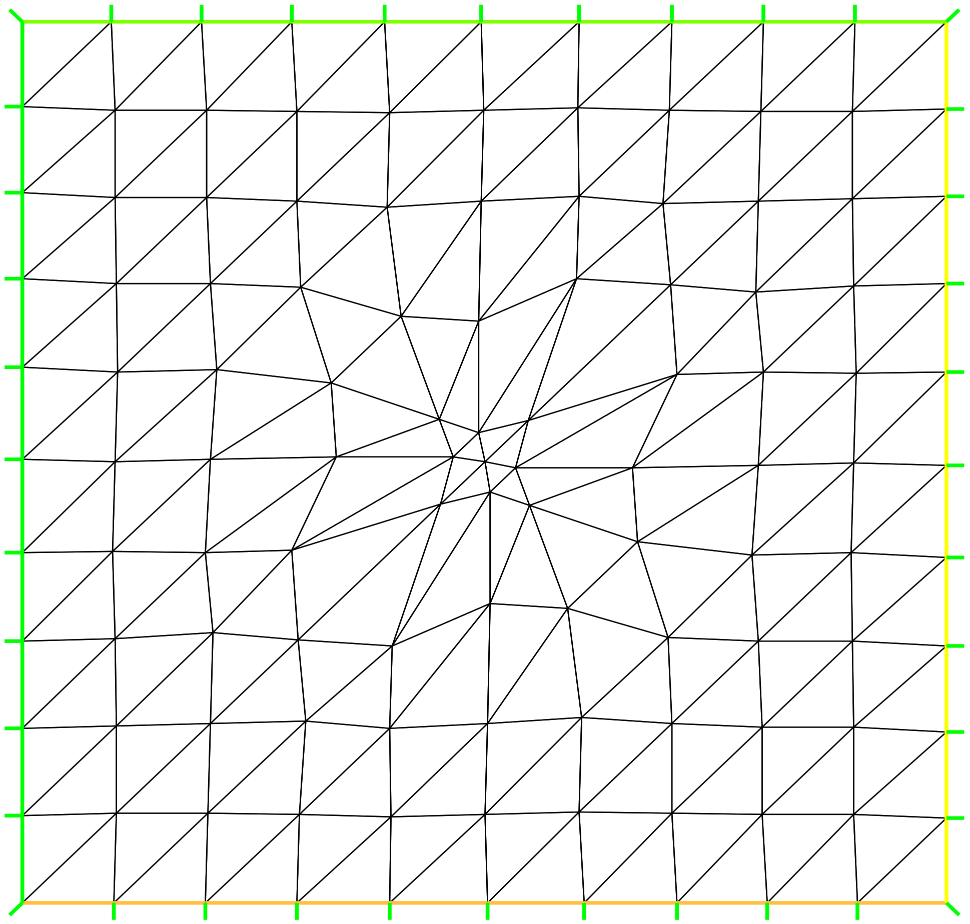
$$u(x,y) = xy(x-1)(y-1)e^{-50\left((x-1/2)^2+(y-1/2)^2\right)}$$



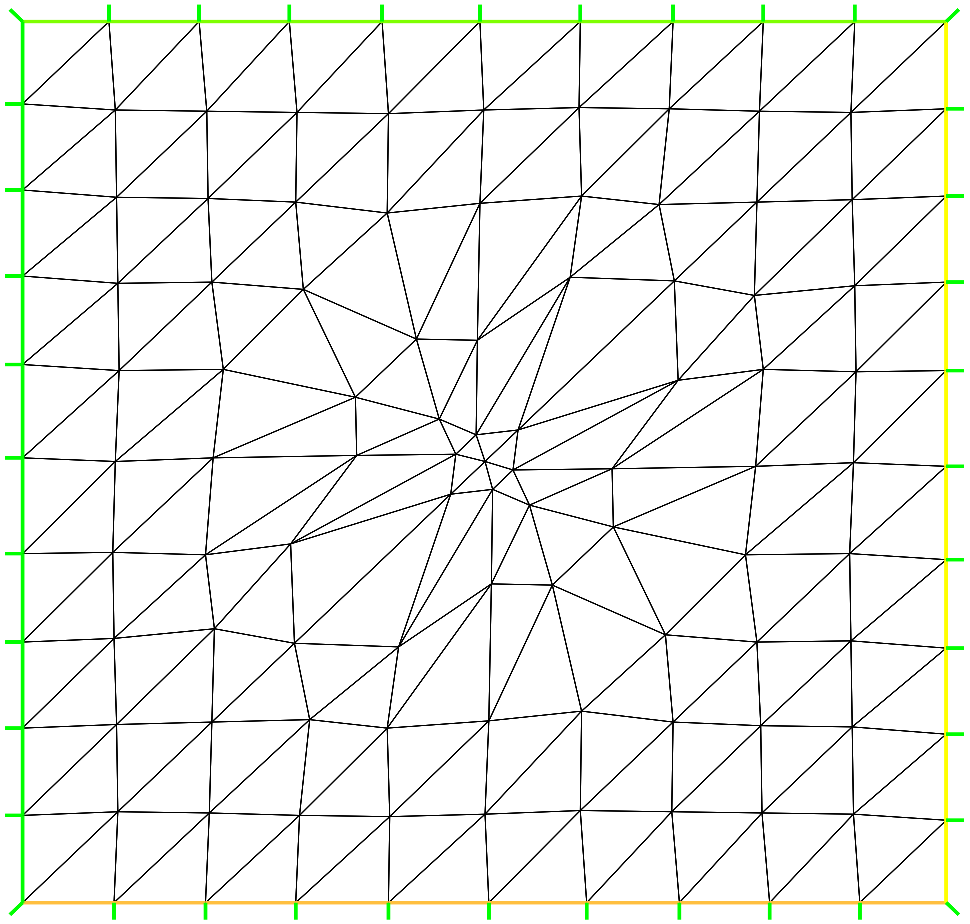
50 iterations



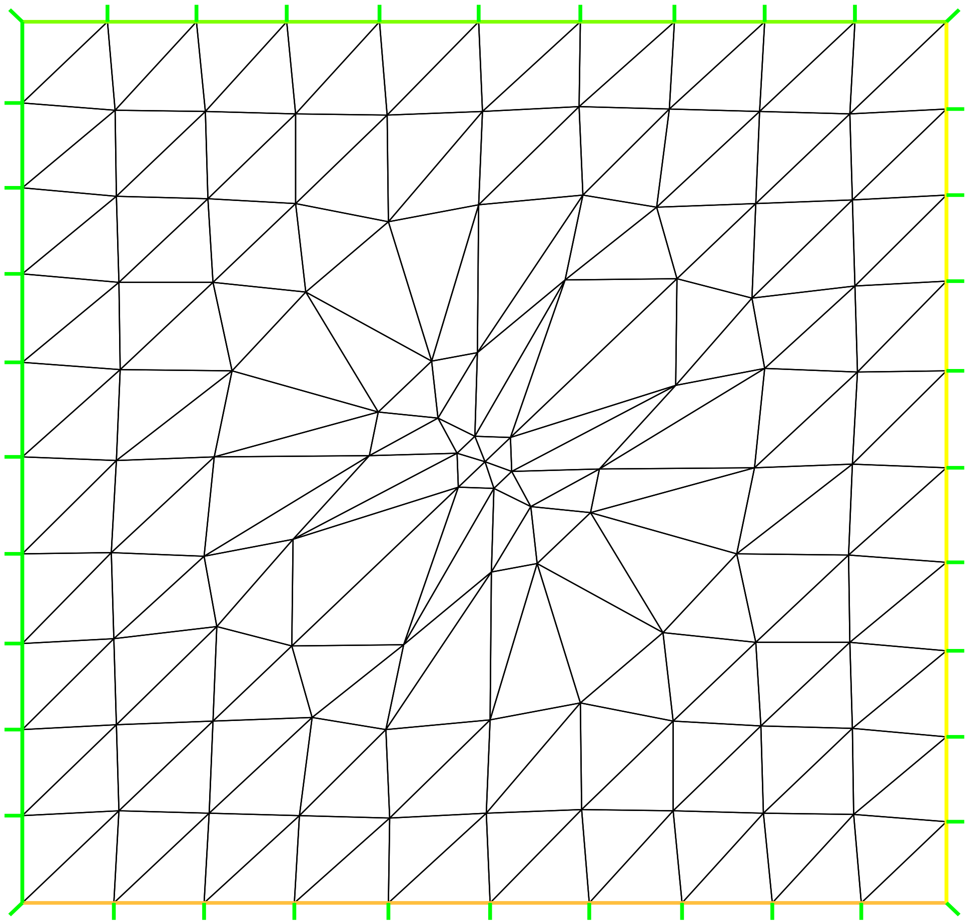
100 iterations



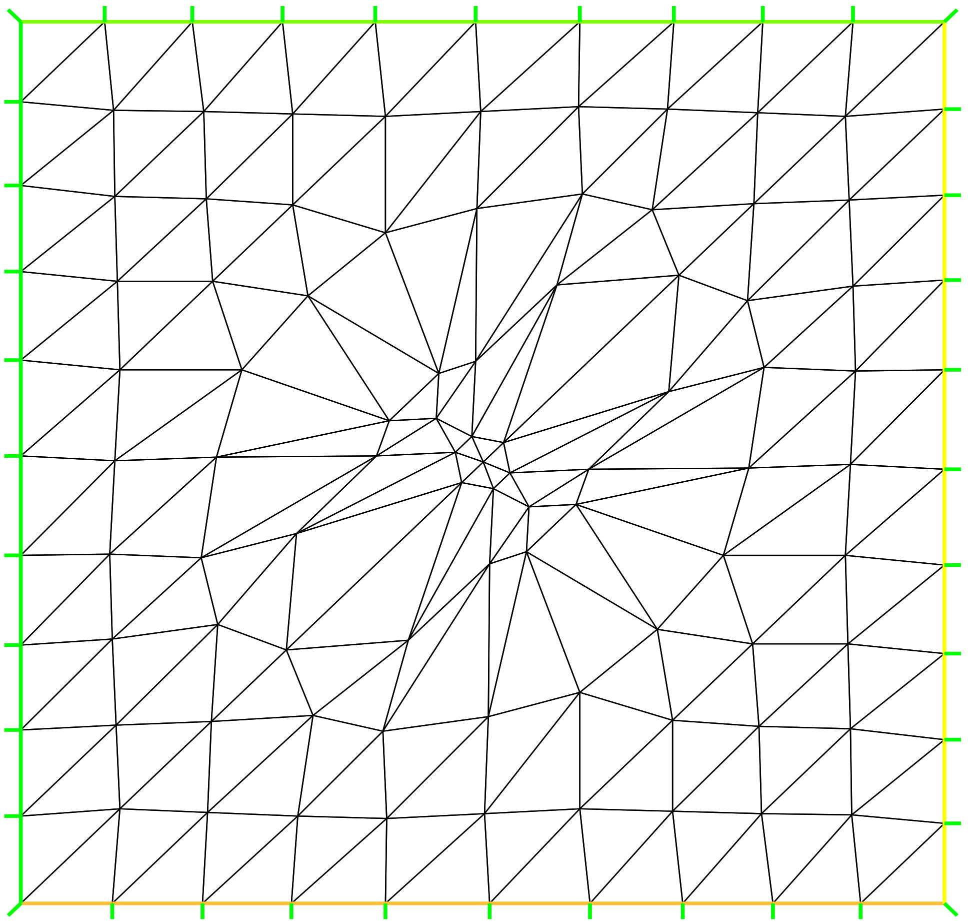
150 iterations



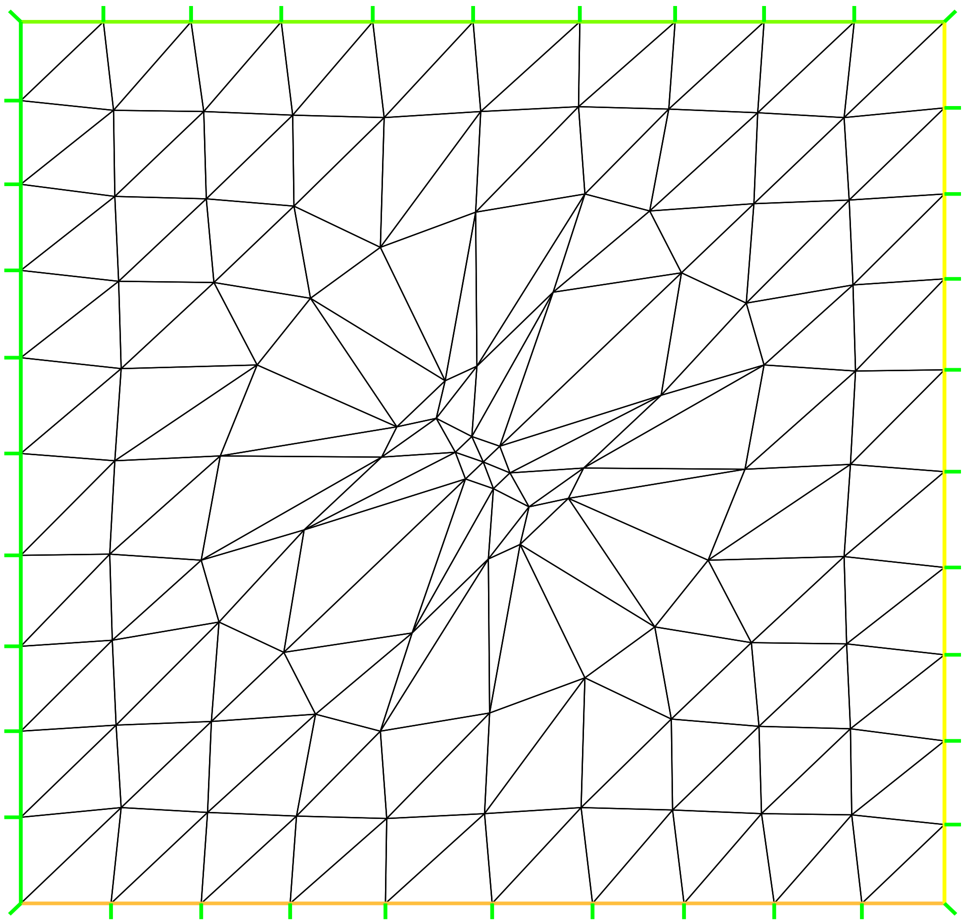
200 iterations



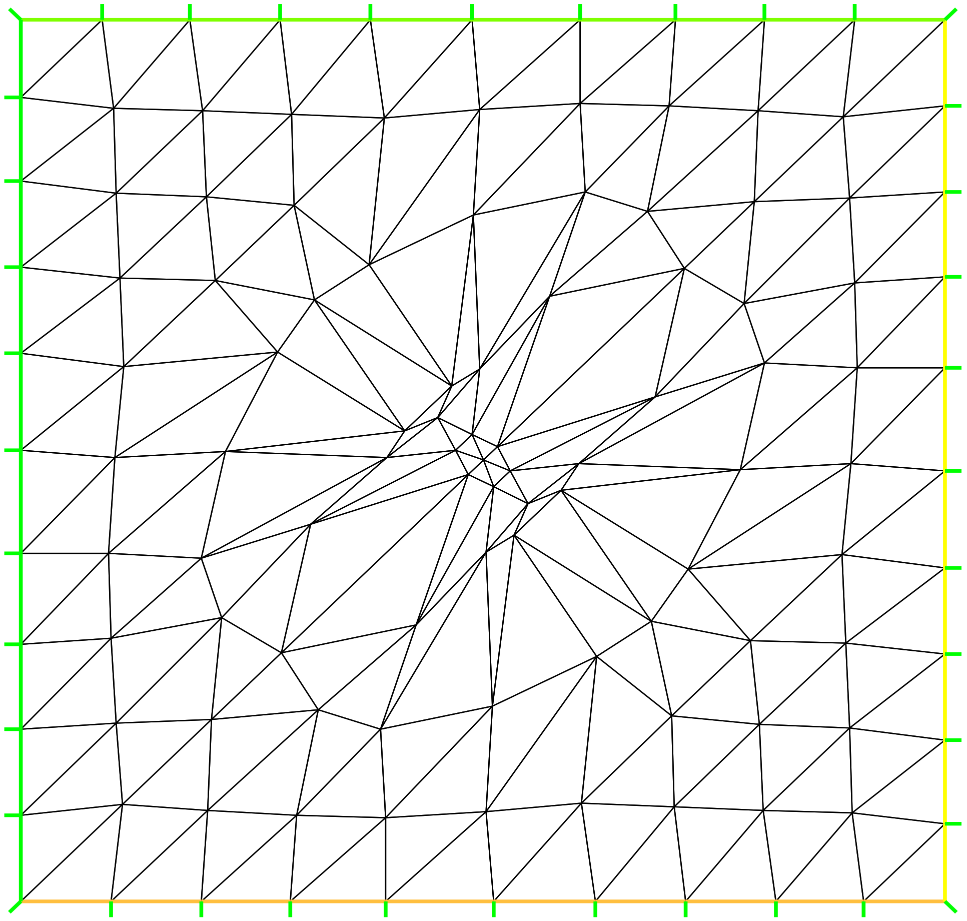
250 iterations



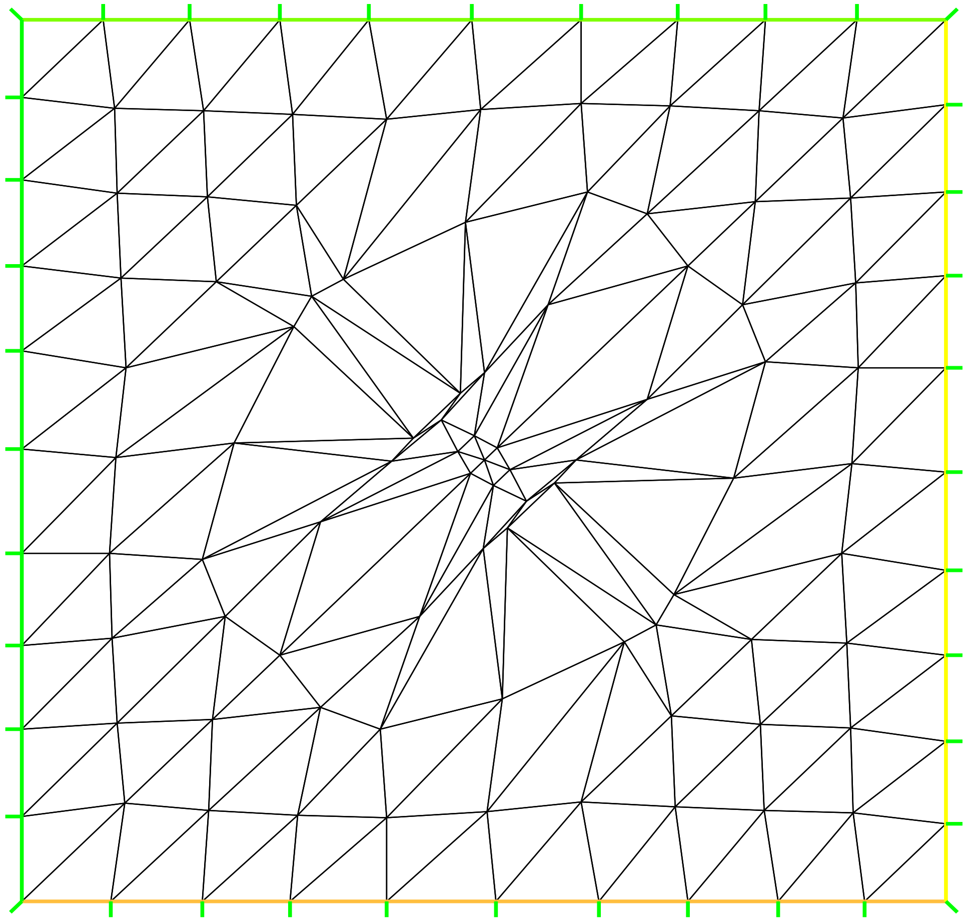
300 iterations



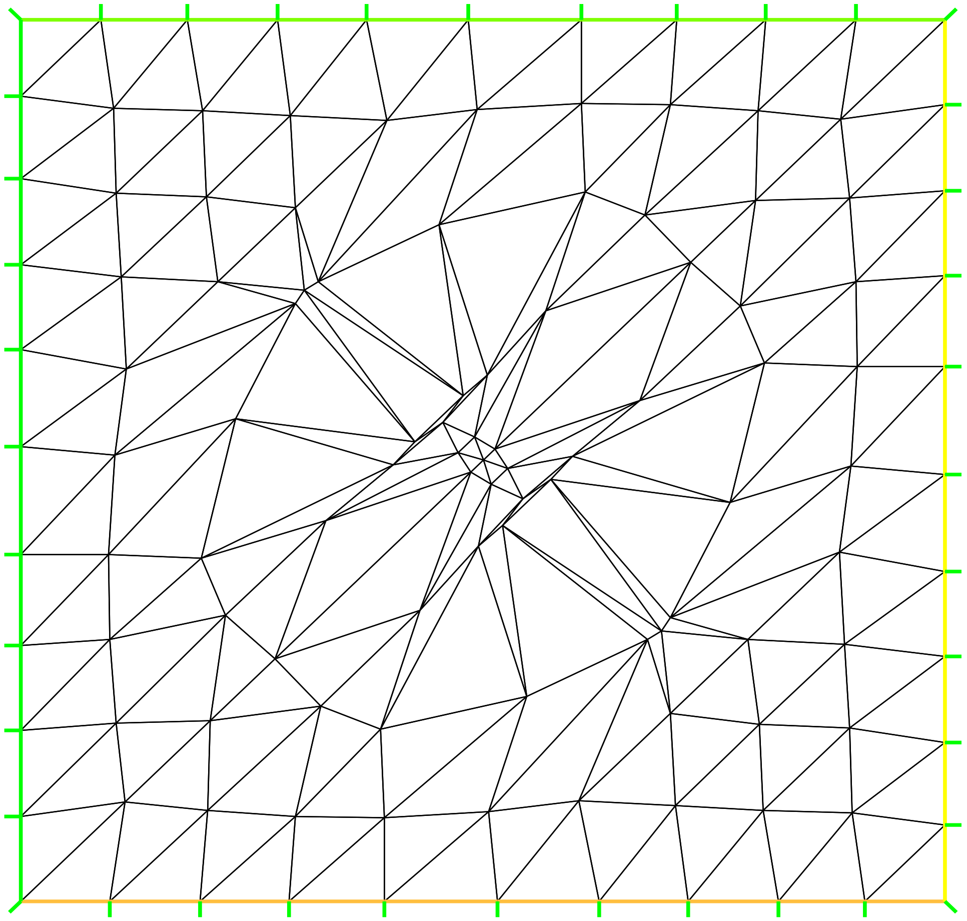
350 iterations



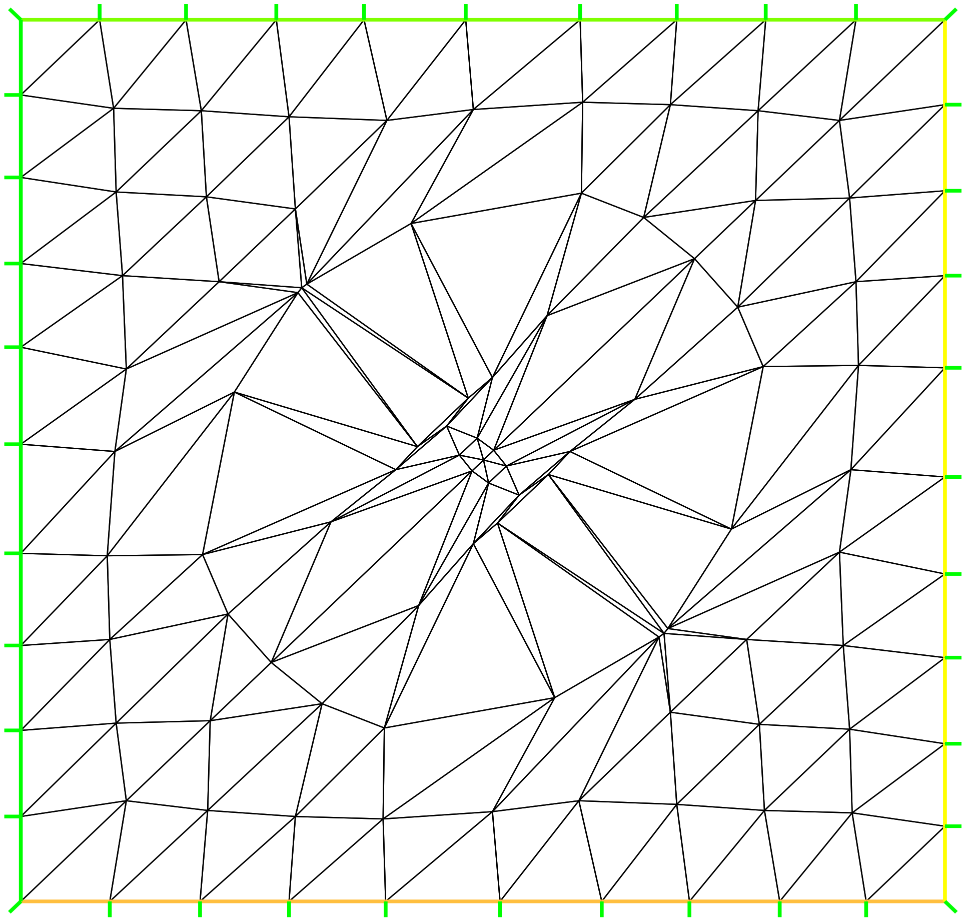
400 iterations



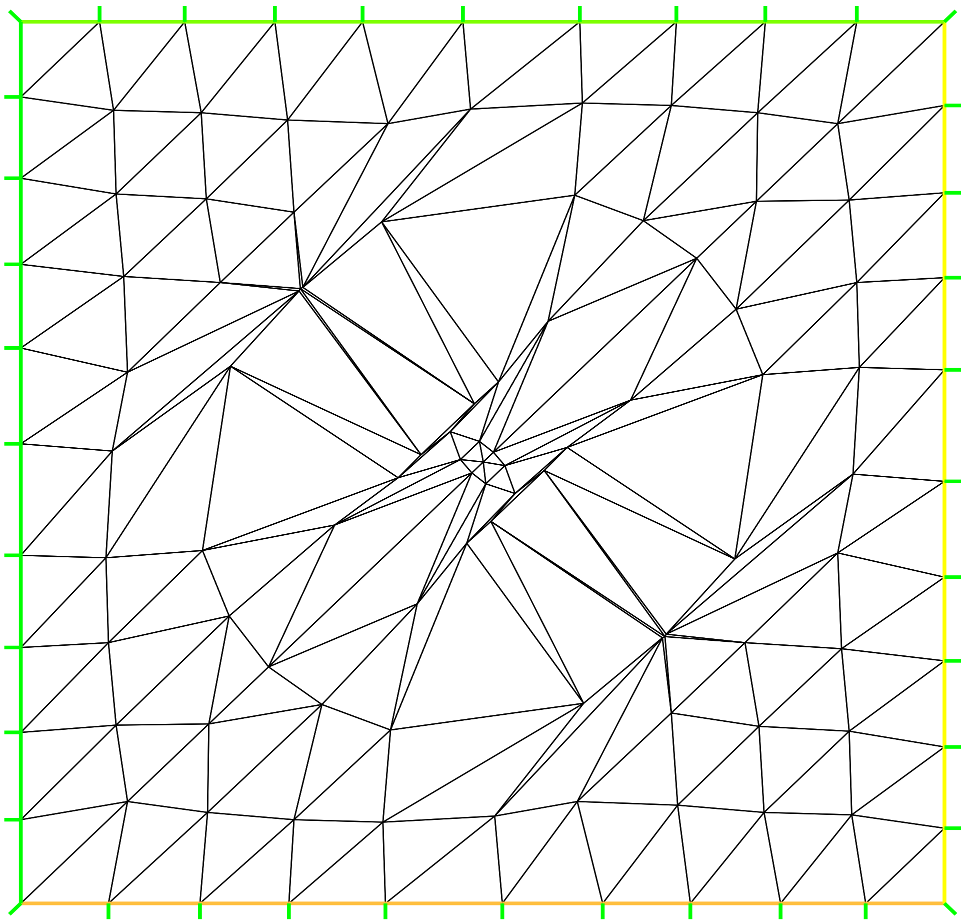
450 iterations



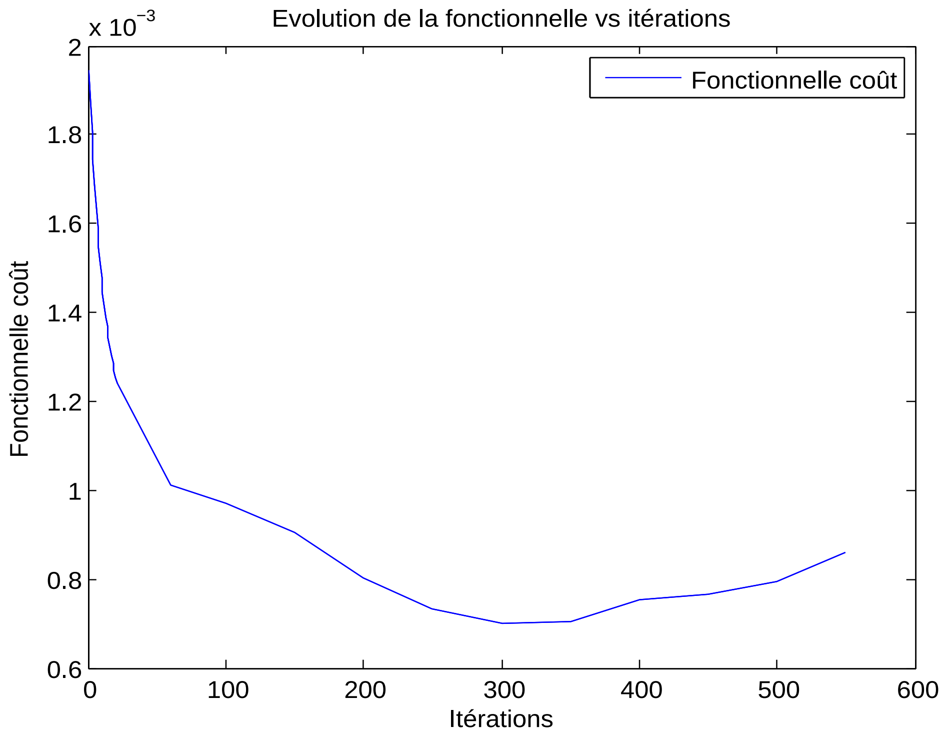
500 iterations



550 iterations



Cost functionnal



• Refinement via an asymptotic expansion

we just want to present some ideas on how to make the error functional minimization, as a shape optimization, when the mesh adaptation requires a discontinuity in the mesh topology by introducing new nodes.

Until now the development is not complete.

General framework of asymptotic expansion

• $\varepsilon \geq 0$ given, $\Omega_\varepsilon = \Omega \setminus \overline{(x_0 + \varepsilon\omega)}$, $x_0 \in \Omega$ et $\omega \subset \mathbb{R}^n$
• $\mathcal{V}$ Hilbert space,

$$u_\varepsilon \in \mathcal{V}, \quad a_\varepsilon(u_\varepsilon, v) = \ell_\varepsilon(v), \quad \forall v \in \mathcal{V}$$

• $a_\varepsilon(\cdot,\cdot)$ a bilinear form, symmetric and uniformly coercive on $\mathcal{V}$
• $\ell_\varepsilon(\cdot)$ a linear form uniformly continues on $\mathcal{V}$

Hypothesis:

- $\|a_\varepsilon - a_0 - f(\varepsilon)\delta_a\| = o(f(\varepsilon))$
- $\|\ell_\varepsilon - \ell_0 f(\varepsilon)\delta_\ell\| = o(f(\varepsilon))$
- J_0 differentiable with respect to u and

$$J_\varepsilon(v) - J_0(u) = DJ_0(v-u) + f(\varepsilon)\delta J + o(f(\varepsilon))$$

- p_0 solution of the dual problem :

$$a_0(w, p_0) = -DJ_0(u_0)w, \quad \forall w \in \mathcal{V}$$

then (M.Masmoudi) $j(\varepsilon) := J(u_\varepsilon)$ is such that

$$j(\varepsilon) - j(0) = f(\varepsilon)\delta j + o(f(\varepsilon))$$

$$\delta j = \delta a + \delta \ell + \delta J$$

Mesh adaptation .

We recall the expression of the fonctionnal to optimize

$$J(e_h) = \frac{1}{2} |||e_h|||^2 \quad \text{where} \begin{cases} e_h \in E_h, \text{ such that } : \\ a(e_h, \hat{w}_h) = R(u_h; \hat{w}_h) \ \forall \ \hat{w}_h \in E_h. \end{cases}$$

For $v \in H_0^1(\Omega)$ we have, by integration by part the following expression of the residual

$$\begin{aligned} R(u_h; v) &= L(v) - a(u_h, v) \\ &= \sum_{K \in \mathcal{T}_h} \int_K (f - \alpha u_h) v dx - \sum_{E \in \mathcal{E}_h^0} \int_E \left[\frac{\partial u_h}{\partial n_E} \right]_E v ds \end{aligned}$$

We introduce a perturbation which is performed in each element as follow

$$\begin{aligned} R_{c}(u_h; w_h) &= \sum_{K\in\mathcal{T}_h} \int_K (f - \alpha u_h - c_1) w_h dx \\ &\quad - \sum_{E\in\mathcal{E}_h} \int_E \Big(\Big[\frac{\partial u_h}{\partial n_E} \Big]_E - c_2 \Big) w_h ds \in \mathbb{R} \end{aligned}$$

where, For $1 < p < 2$,

$$c = (c_1, c_2) \in \mathbb{G} := \prod_{K\in\mathcal{T}_h} L^p(K) \times \prod_{E\in\mathcal{E}_h} L^p(E)$$

$$\begin{cases} \text{Find } e_h^c \in E_h, \quad \text{solution of} \\ a(e_h^c, w_h) = R_c(u_h; w_h) \ \forall w_h \in E_h \end{cases}$$

Cost functional: $\mathcal{J}(c) = J(e_h^c)$

The Lagrangien: $\mathcal{L}(c, e_h, w_h) := J(e_h) + a(e_h, w_h) - R_c(u_h; w_h)$

Dual problem

$$\begin{cases} p_h^c \in E_h \\ a(\psi_h; p_h^c) = -DJ(e_h^c) \cdot \psi \ \forall \psi \in E_h \end{cases}$$

$\mathcal{J}$ is differentiable and for $\delta c \in \prod_{K \in \mathcal{T}_h} L^p(K) \times \prod_{E \in \mathcal{E}_h} L^p(E)$

$$\mathcal{J}'(c) \cdot \delta c = \sum_{K \in \mathcal{T}_h} \int_K \delta c_1 \cdot p_h^c dx + \sum_{E \in \mathcal{E}_h} \int_E \delta c_2 \cdot p_h^c ds$$

For $K \in \mathcal{T}_h$ and $E \in \mathcal{E}_h^0$, let $\delta c^K = (\delta c_K, 0)$ and $\delta c^E = (0, \delta c_E)$ where

$$\delta c_K = \begin{cases} (f - \alpha u_h) & \text{in } K \\ 0 & \text{elsewhere} \end{cases}, \quad \delta c_E = \begin{cases} \left[\dfrac{\partial u_h}{\partial n_E}\right]_E & \text{on } E \\ 0 & \text{elsewhere} \end{cases}$$

so

$$\mathcal{J}'(0) \cdot \delta c^K = \int_K (f - \alpha u_h) p_h^0 \, dx$$

$$\mathcal{J}'(0) \cdot \delta c^E = \int_E \left[\frac{\partial u_h}{\partial n_E}\right]_E p_h^0 \, ds$$

The perturbation related to adding a new node can be presented as follows: x_K^ε is the new node in K, ε is the distance to vertex.

$$\mathcal{M} \longrightarrow \mathcal{T}_h \longrightarrow u_h$$

$$\mathcal{M} \oplus \{x_K^\varepsilon\} \longrightarrow \mathcal{T}_h^\varepsilon \longrightarrow u_h^\varepsilon = u_h + \delta u_h$$

For $K \in \mathcal{T}_h$ $\quad f - \alpha u_h^\varepsilon = f - \alpha u_h - \alpha \delta u_h$

For $E \in \mathcal{E}_h^0$, $\quad \left[\frac{\partial u_h^\varepsilon}{\partial n_E}\right]_E = \left[\frac{\partial u_h}{\partial n_E}\right]_E + \left[\frac{\partial \delta u_h}{\partial n_E}\right]_E$

$$j(\varepsilon) = J(e_h(\mathcal{T}_h^\varepsilon))$$

$\mathcal{T}_h$: triangulation associated with the set $\mathcal{M}$
$\mathcal{T}_h^\varepsilon$: triangulation associated with the set $\mathcal{M} \oplus \{x^\varepsilon\}$.

Question:

$$j(\varepsilon) = j(0) + f(\varepsilon) g(x^\varepsilon) + \circ(f(\varepsilon))?$$

Conjecture:

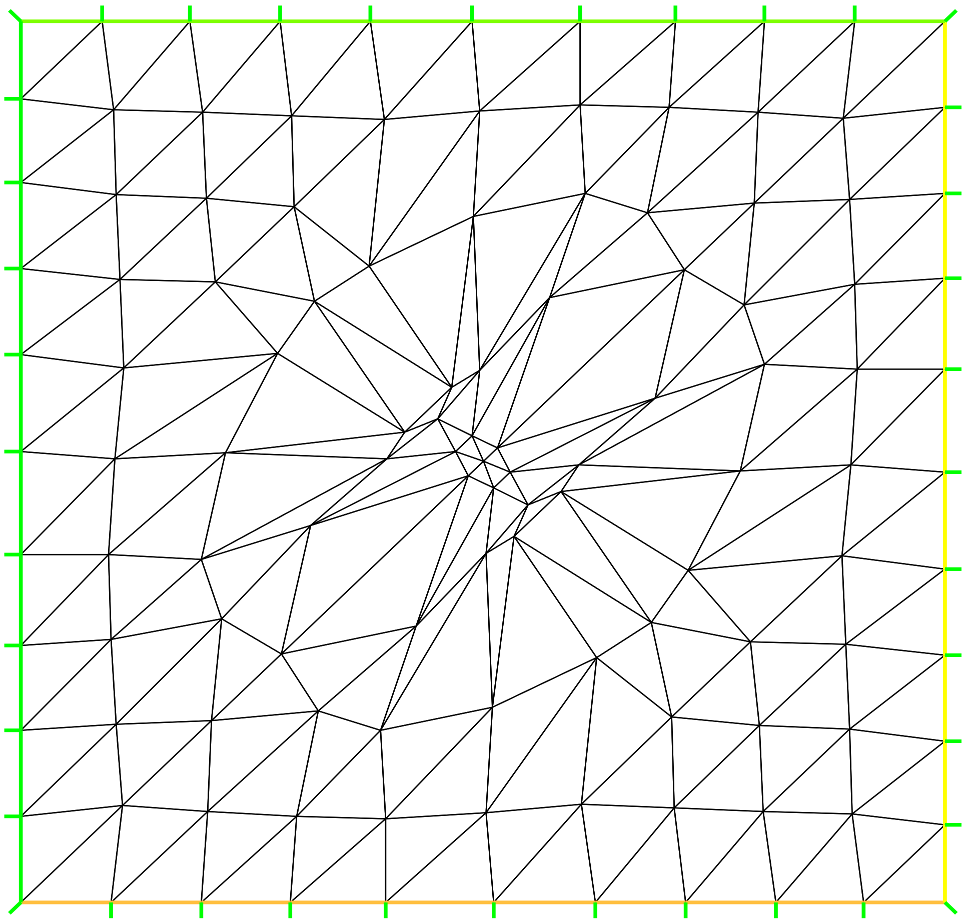
$$g(x^\varepsilon) := \begin{cases} \int_K (f - \alpha u_h) p_h^0 dx & \text{if } x^\varepsilon \in K \\ \int_E \left[\frac{\partial u_h}{\partial n_E}\right]_E p_h^0 ds & \text{if } x^\varepsilon \in E \end{cases}$$

or

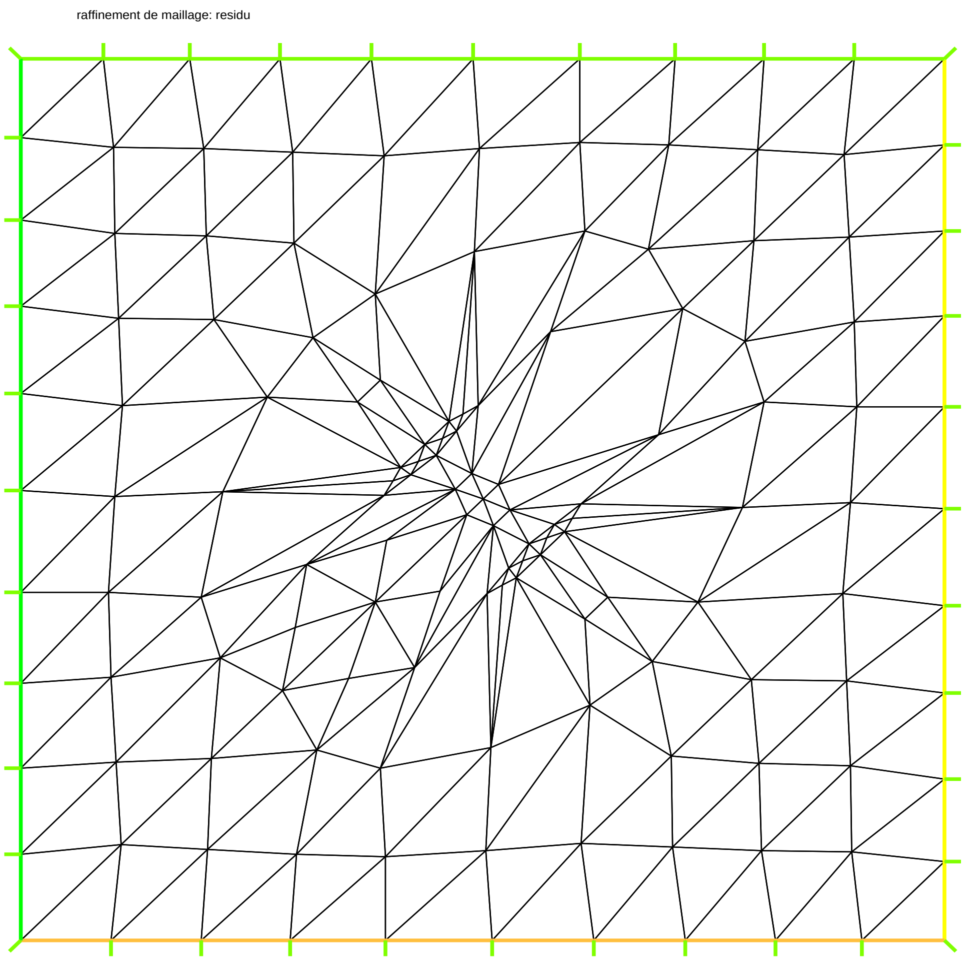
$$g(x^\varepsilon) := \int_K e_h p_h^0 dx$$

We test the three cases. It seems that the last one is the best.

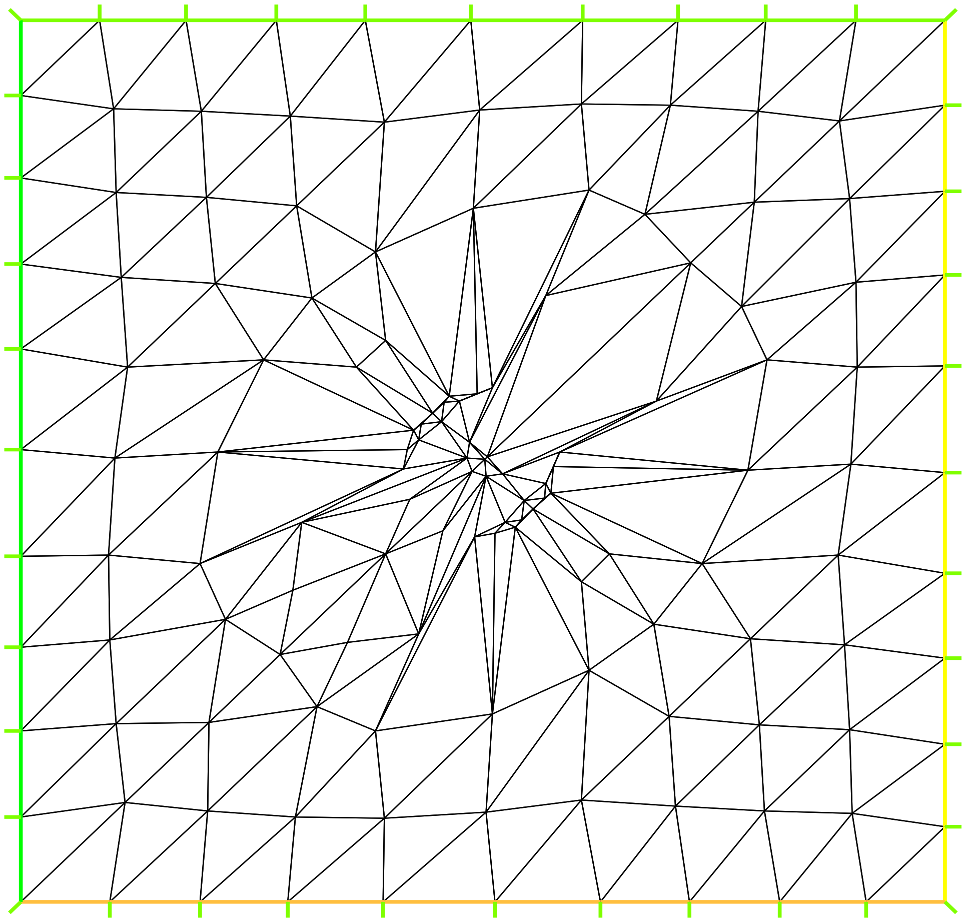
:Dispalcement 1 (350 iterations)



Refinement with the first one



Displacement (131 iterations)

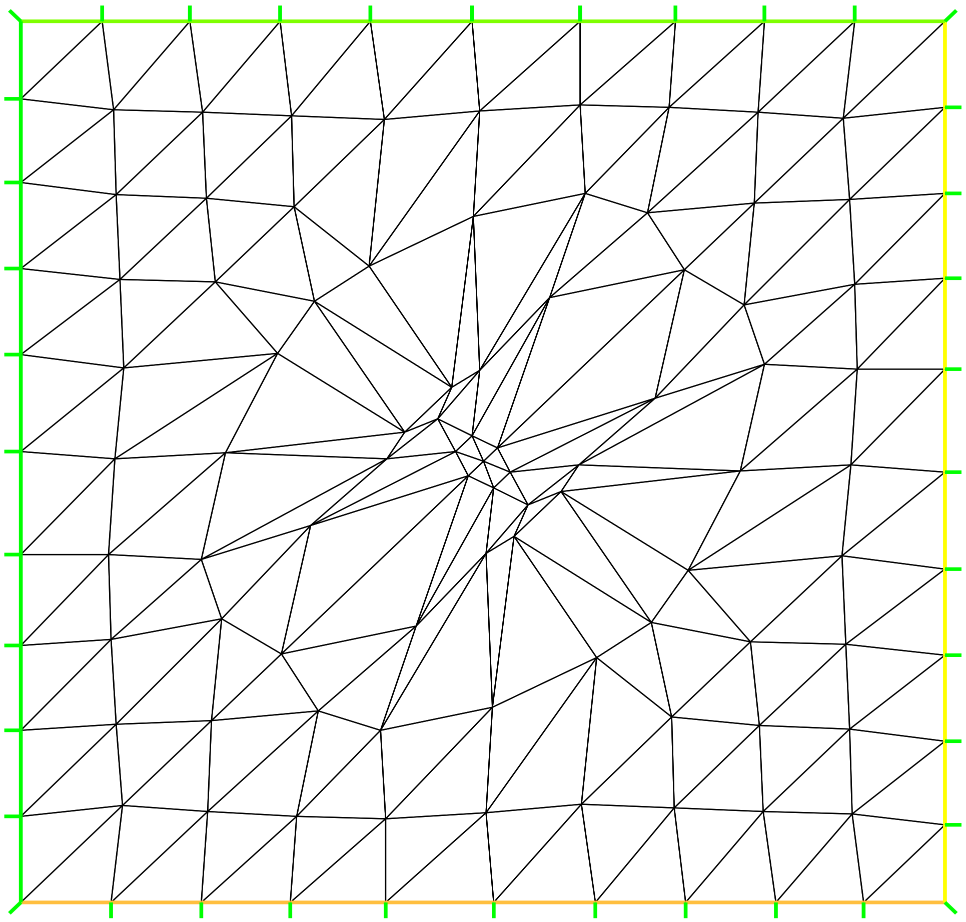


If we take

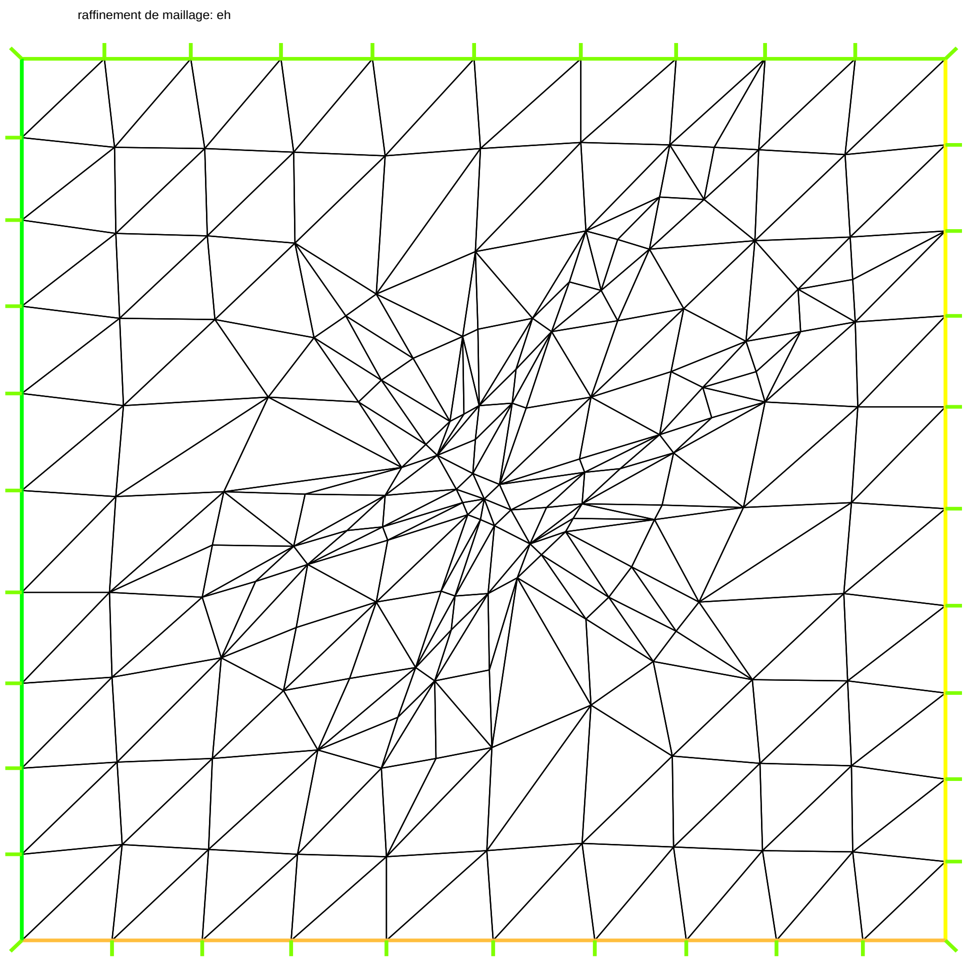
$$\boldsymbol{g(x^{\varepsilon}) := \int_K e_h p_h^0 dx}$$

we have

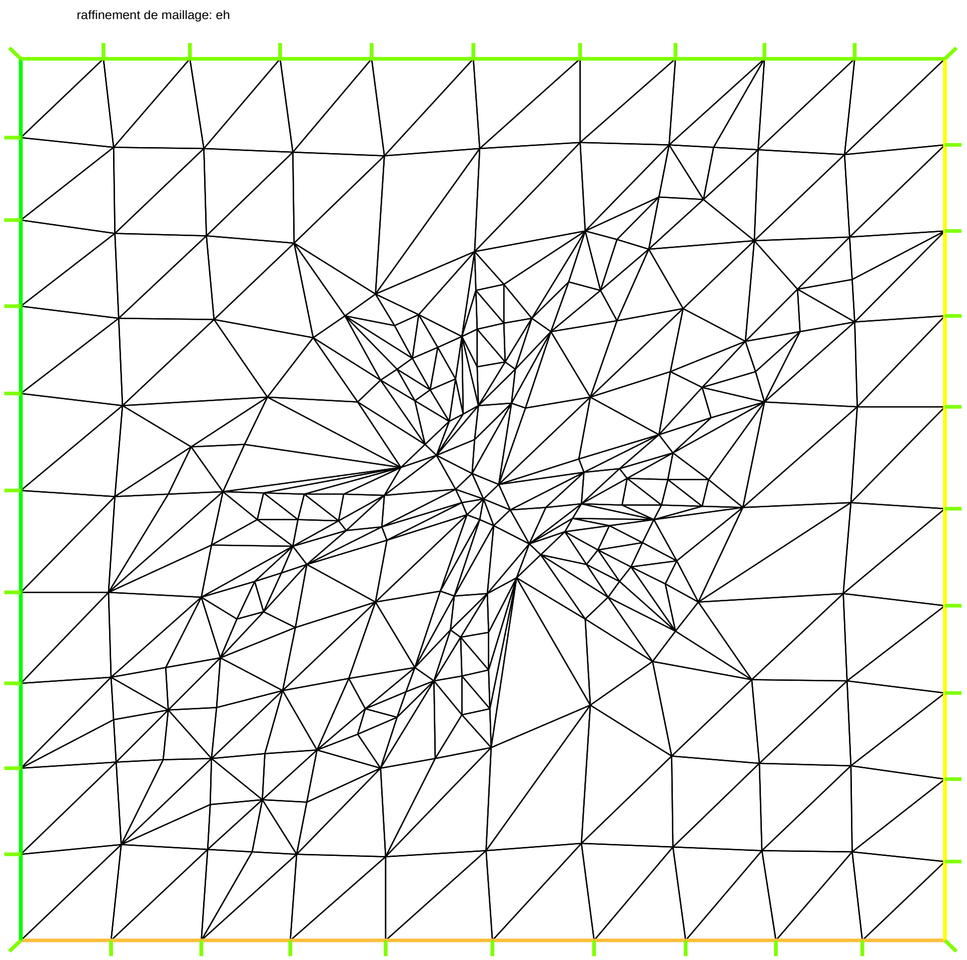
:Dispalcement 1 (350 iterations)



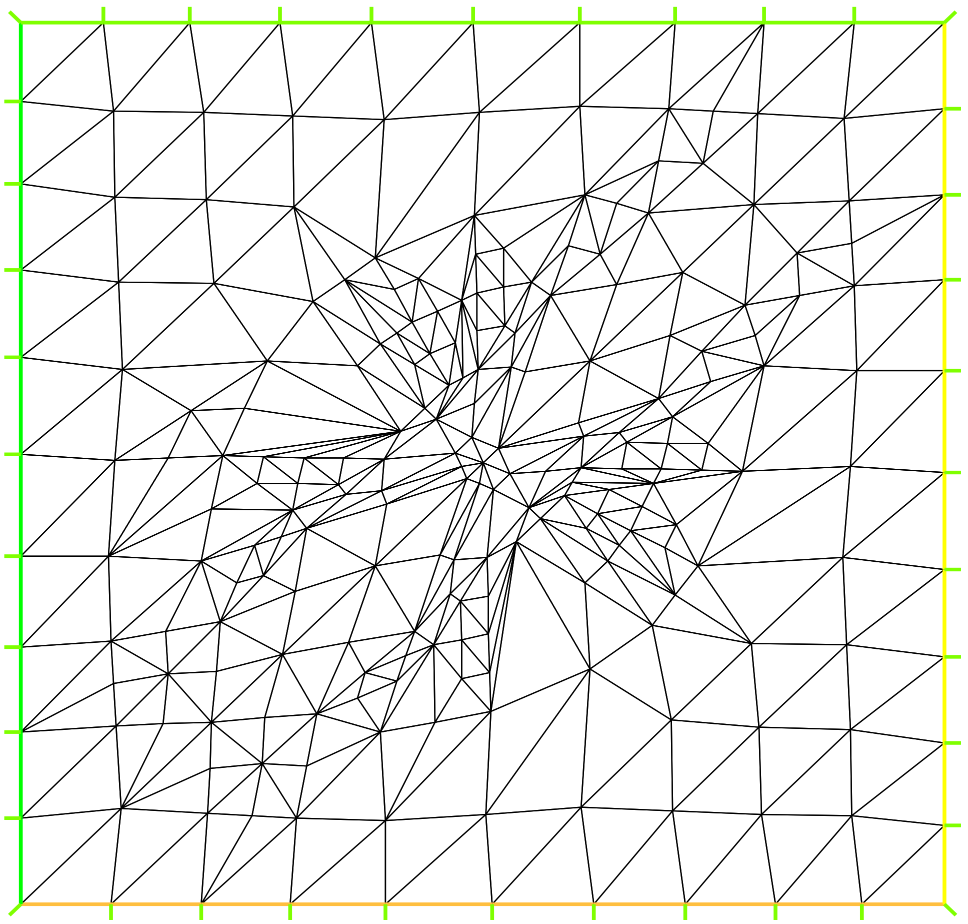
First Refinement



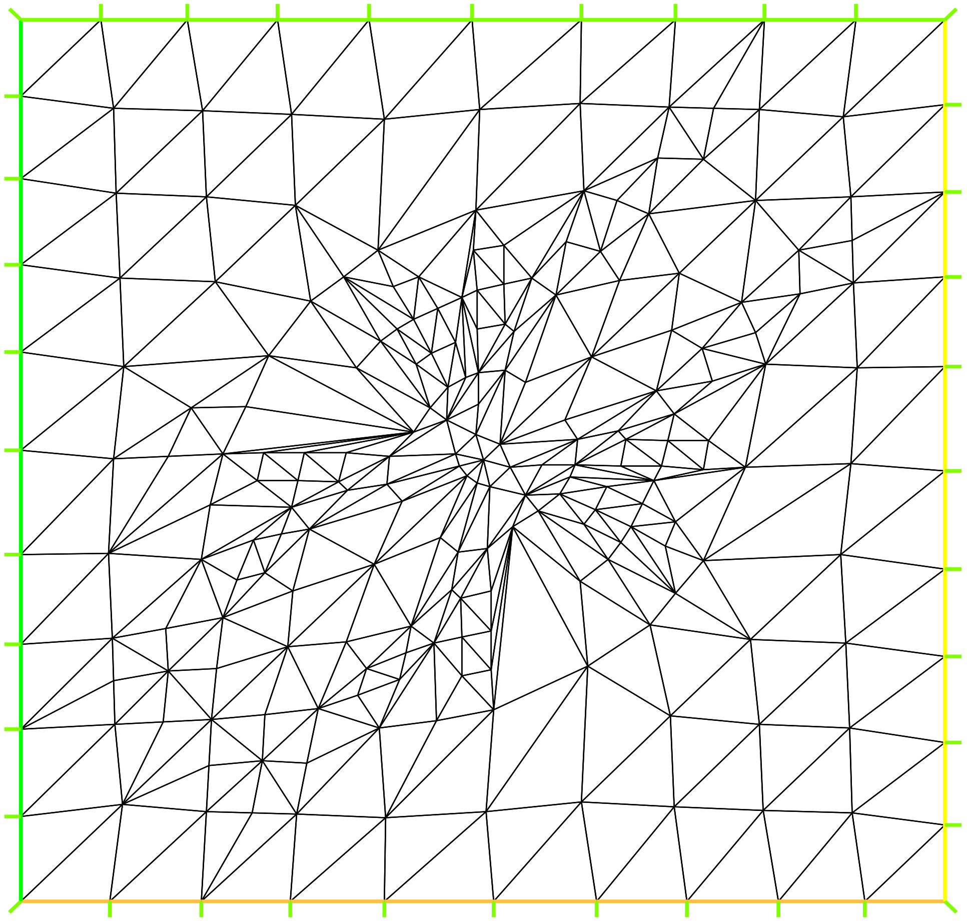
second Refinement



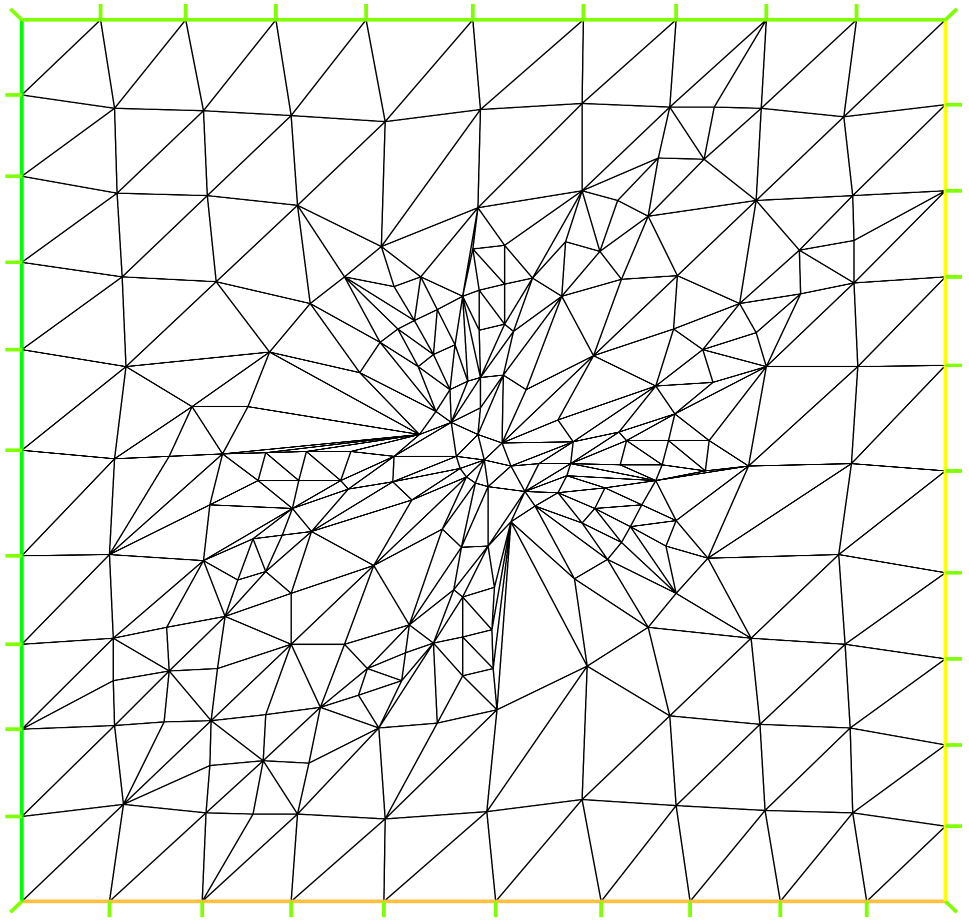
Displacement



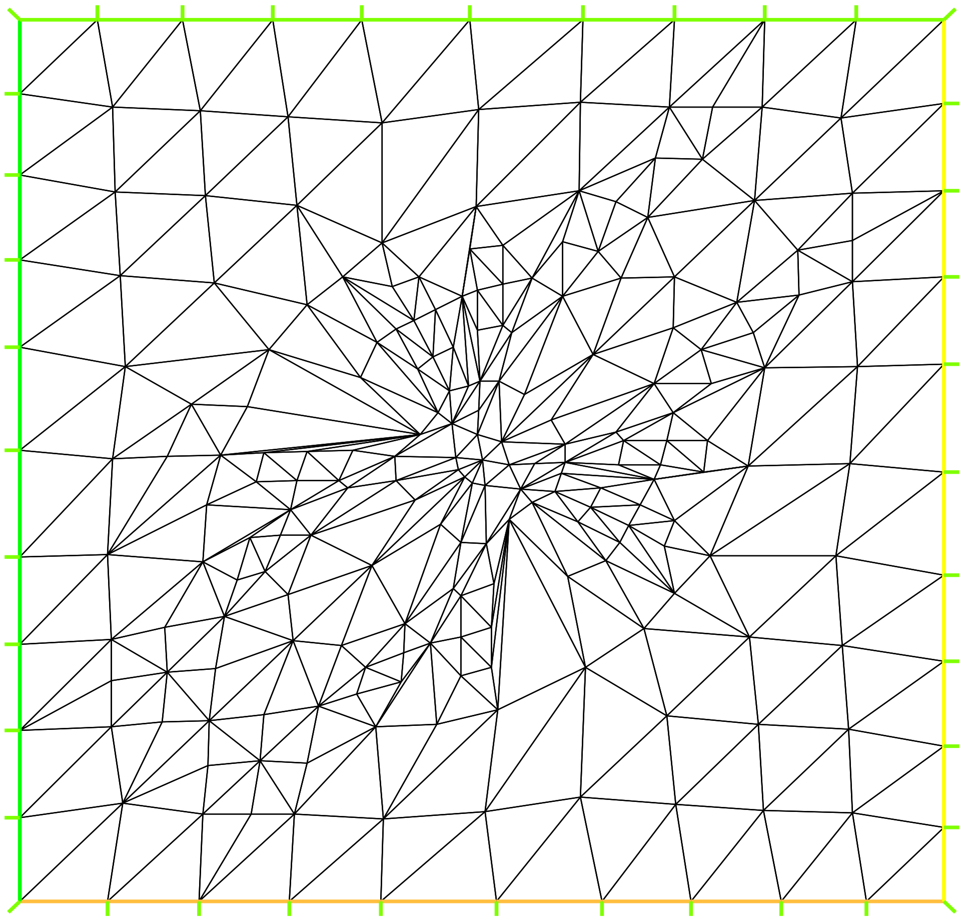
Iteration 50



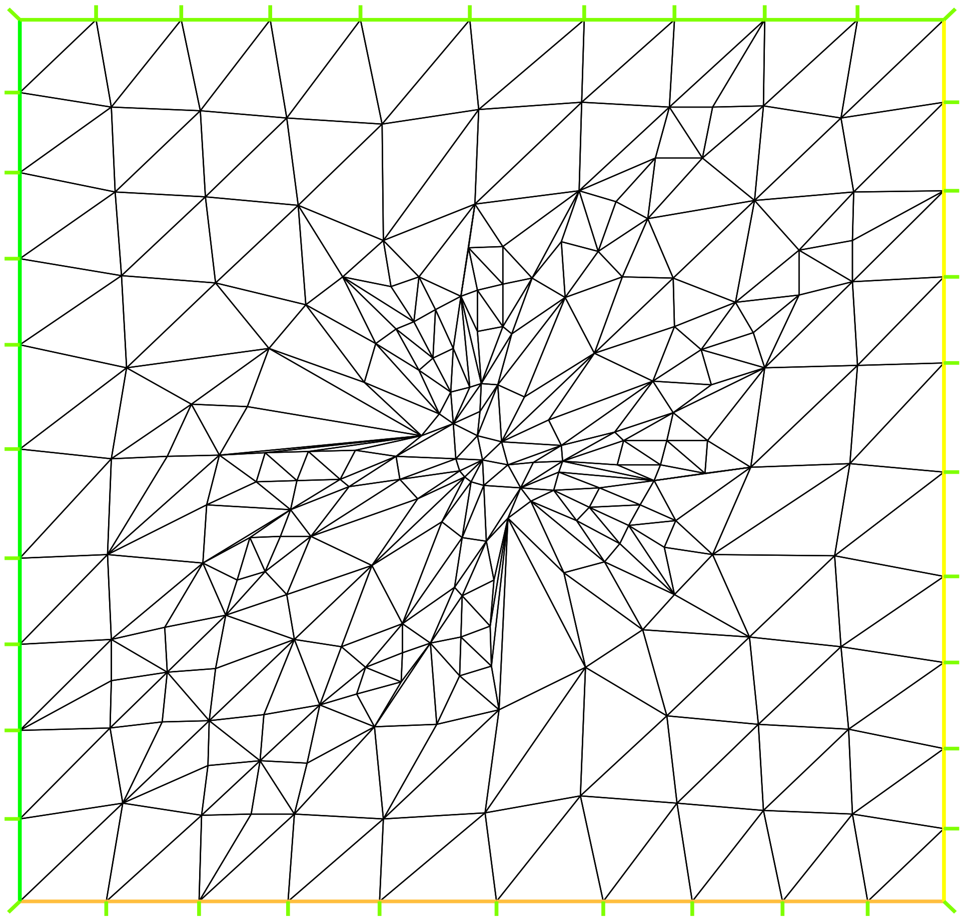
Iteration 100



Iteration 150



Iteration 193



One dimensionnal Case

For $f \in L^2(]0,1[$ and $\alpha \in \mathbb{R}$ let $u \in H_0^1(]0,1[)$ solution of

$$\int_0^1 (u'v' + \alpha uv)dx = \int_0^1 fv dx.$$

Let $\mathcal{T}_h = \{K_i\}_{1 \le i \le N}$, et $K_i = [x_i, x_{i+1}]$.

$$\mathcal{V}_h = \{v_h \in C^0([0,1]/v_h|_K \in P_1(K)\, \forall K \in \mathcal{T}_h, v_h(0) = v_h(1) = 0\}$$

$e_h \in E_h$:

$$\int_0^1 (e_h' w_h' + \alpha e_h w_h)dx = R(u_h; w_h) \;\forall w_h \in E_h$$

$$\begin{aligned} E_h = \; & \{w_h \in C^0([0,1]) \,/\, w_h \in P_2(K) \,\forall K \in \mathcal{T}_h, \\ & w_h(x_i) = 0,\; 1 \le i \le N\} \end{aligned}$$

$$\begin{aligned} R(u_h; w_h) \; & := \sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} (f - \alpha u_h) w_h dx + \sum_{i=1}^{N} [u_h']_i w_h(x_i) \\ & = \sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} (f - \alpha u_h) w_h dx \end{aligned}$$

For $K_i = [x_i, x_{i+1}]$ let $x_K^\varepsilon = x_0^i$ middle of K_i

$$g(x_K^\varepsilon) = (f - \alpha u_h)((x_0^i)) q_h^0(x_0^i)$$

Adaptation Strategy

Initial mesh$\mathcal{T}_h^{0,0}$

Step 1: Adaptation 1:
$\eta_K := h_K \|f - f_K\|_{0,K} \longrightarrow \mathcal{T}_h^{0,1}$.

Step 2: $\left\{ \begin{array}{l} -\ \text{Computation of } u_h,\ e_h \\ -\ \text{Displacement 1} \ \longrightarrow \mathcal{T}_h^{0,2} = \mathcal{T}_h^{0}. \end{array} \right.$

Step 3: Computation of $g(x_0^i)$ for $i = 1, \ldots, N$

Step 4: In K_n where $g(x_0^n)$ is the most negative, add a node $m_n^\varepsilon = x_n + \frac{h_n}{2} + \varepsilon$, and $-\frac{h_n}{2} < \varepsilon < \frac{h_n}{2}$.

step 5: local update

$(\varphi_i)_{1\leq i\leq N}$ base functions of $\mathcal{V}_h$ associated to $\mathcal{T}_h^0 = (x_i)_{1\leq i\leq N}$

$(\tilde{\varphi}_i)_{1\leq i\leq N+1}$ new base functions:

$$\varphi_i = \tilde{\varphi}_i \text{ pour } i \neq n-1, n, n+1$$

Computation of the perturbed solution

$\tilde{u}_h^\varepsilon \in C^0([0,1]$

$$\forall K \neq K_n \qquad \tilde{u}_h^\varepsilon|_K = u_h|_K$$

$$\text{Sur } K_n \qquad \begin{cases} \tilde{u}_h^\varepsilon(x_n) = u_h(x_n) \\ \tilde{u}_h^\varepsilon|_{[x_n,m_n^\varepsilon]} \in P_1([x_n,m_n^\varepsilon]) \\ \tilde{u}_h^\varepsilon|_{[m_n^\varepsilon,x_{n+1}]} \in P_1([m_n^\varepsilon,x_{n+1}]) \\ \tilde{u}_h^\varepsilon(x_{n+1}) = u_h(x_{n+1}). \end{cases}$$

$\tilde{u}_h^\varepsilon(m_n^\varepsilon)$ solution of local problem (where $\varphi_{m_n^\varepsilon}$ is the base function associeted to m_n^ε)

$$\int_{x_n}^{x_{n+1}} \left((\tilde{u}_h^\varepsilon)'(\varphi_{m_n^\varepsilon})' + \alpha \tilde{u}_h^\varepsilon \varphi_{m_n^\varepsilon}\right) dx = \int_{x_n}^{x_{n+1}} f \varphi_{m_n^\varepsilon} dx$$

Etape 6: globale update by local problems

Computation of $\tilde{u}_h^\varepsilon(x_j)$ (for $j \neq n+1$)

$$\int_{x_{j-1}}^{x_{j+1}} \left((\tilde{u}_h^\varepsilon)'(\tilde{\varphi}_j)' + \tilde{u}_h^\varepsilon \tilde{\varphi}_j\right) dx = \int_{x_{j-1}}^{x_{j+1}} f \tilde{\varphi}_j dx.$$

Step 7 Computation of the new estimator

$$a(\tilde{e}_h^\varepsilon, \tilde{w}_h) = R(\tilde{u}_h^\varepsilon; \tilde{w}_h),\ \forall \tilde{w}_h \in \tilde{E}_h.$$

Step 8 Computation of the new $g(\tilde{x}_0^i), \quad i = 1, \dots, N+1$

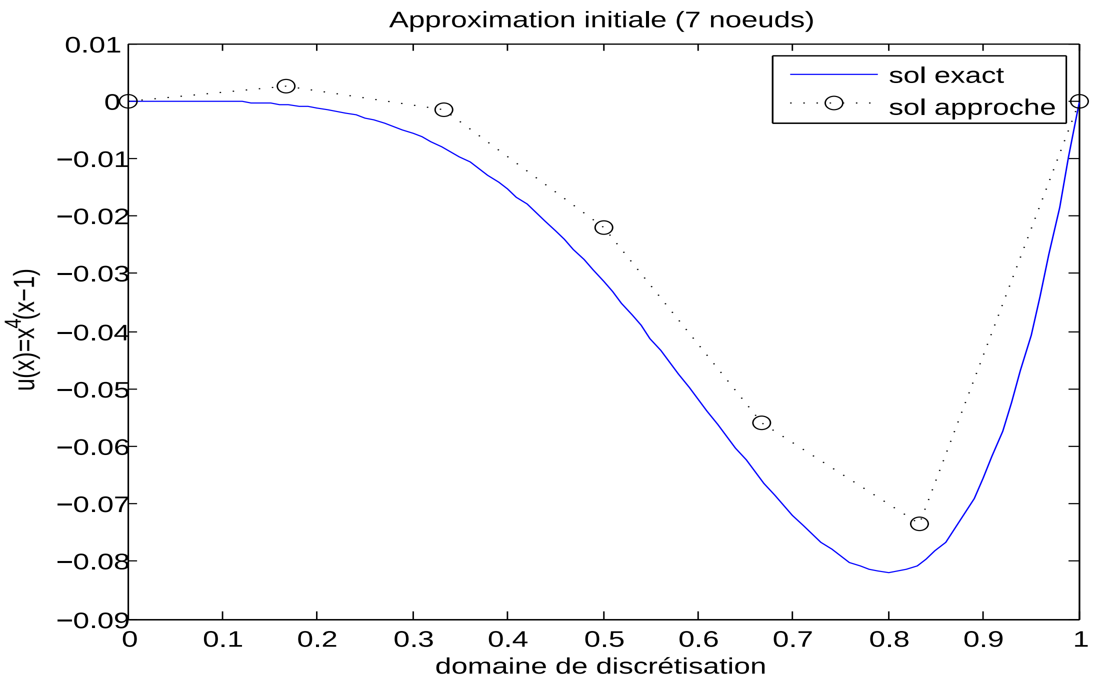
Step 9 Displacement (every k itérations)

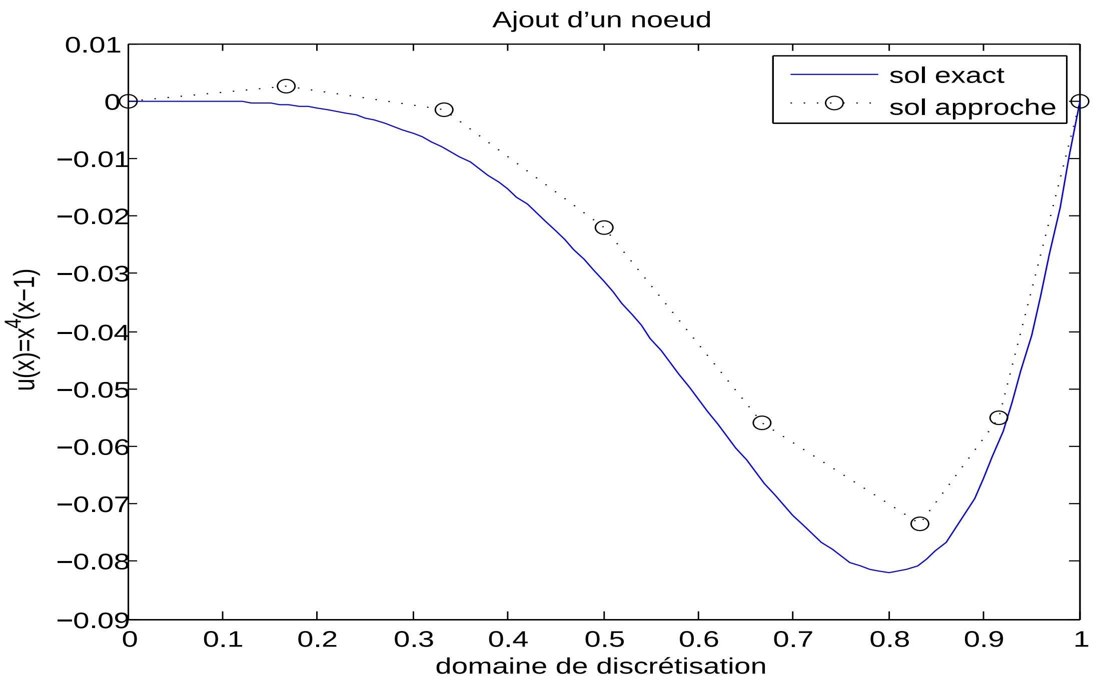
A Numerical Test

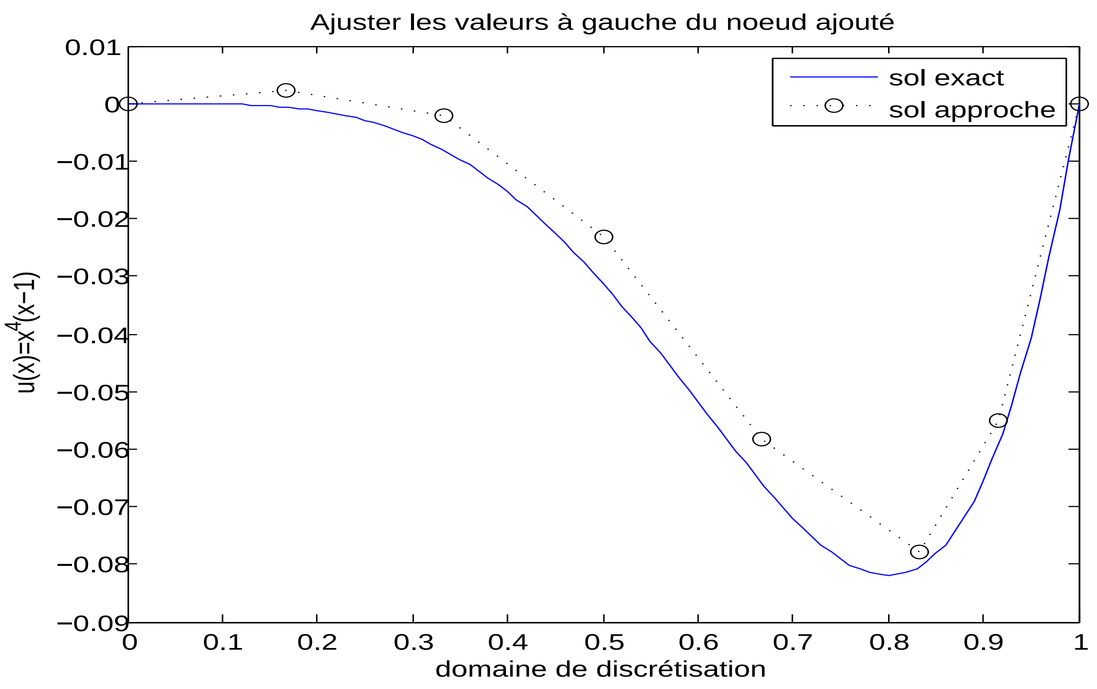
MATLAB

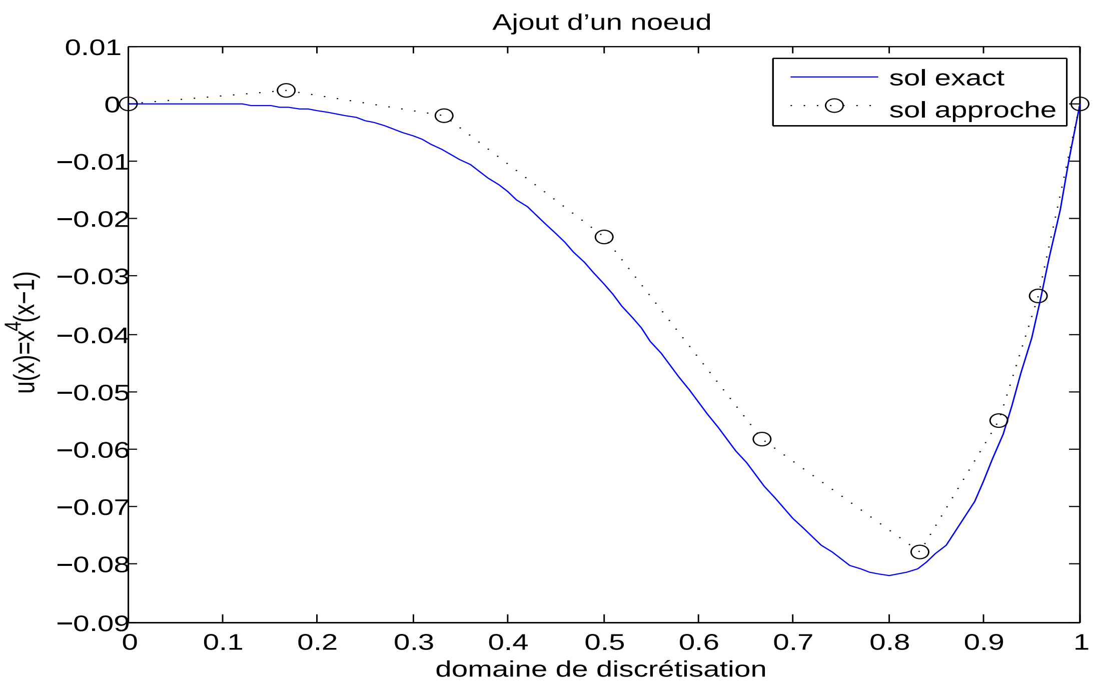
$\boldsymbol{\Omega = [0, 1]}$

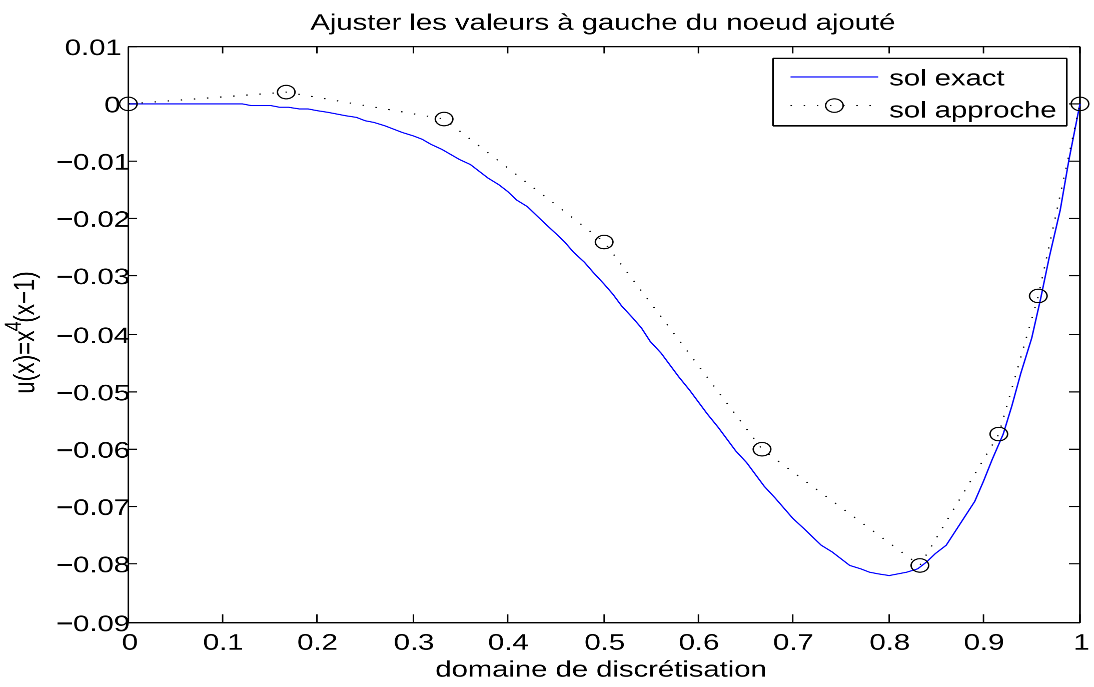
$\boldsymbol{u(x, y) = x^4(x - 1)}$


Approximation initiale (7 noeuds)
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

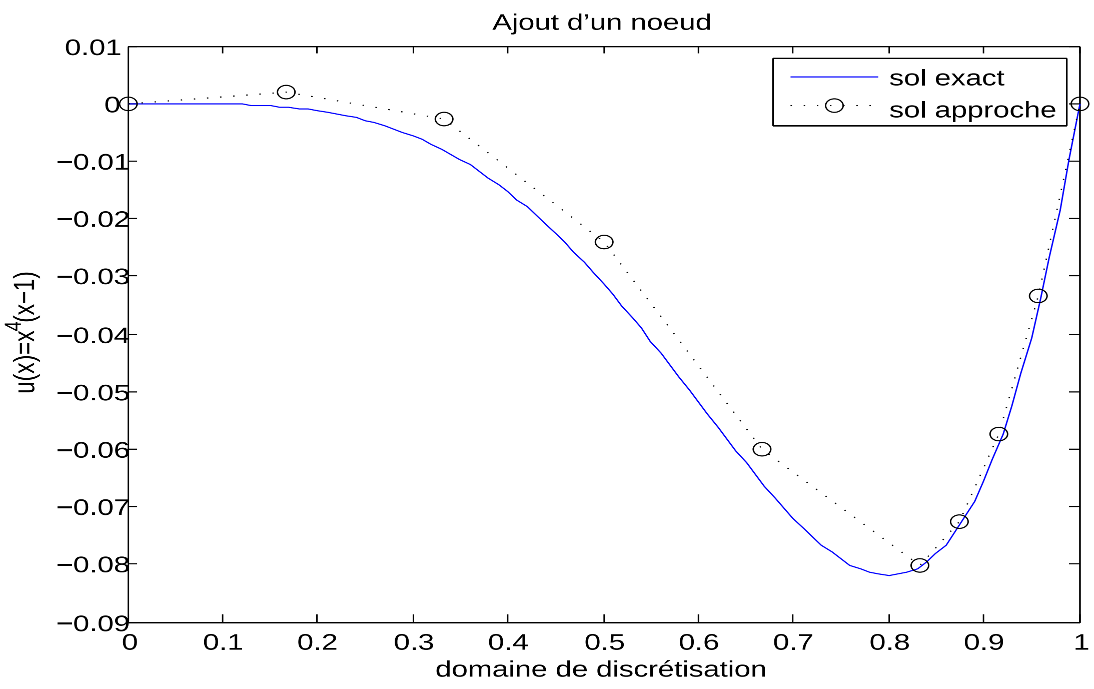

Ajout d'un noeud
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

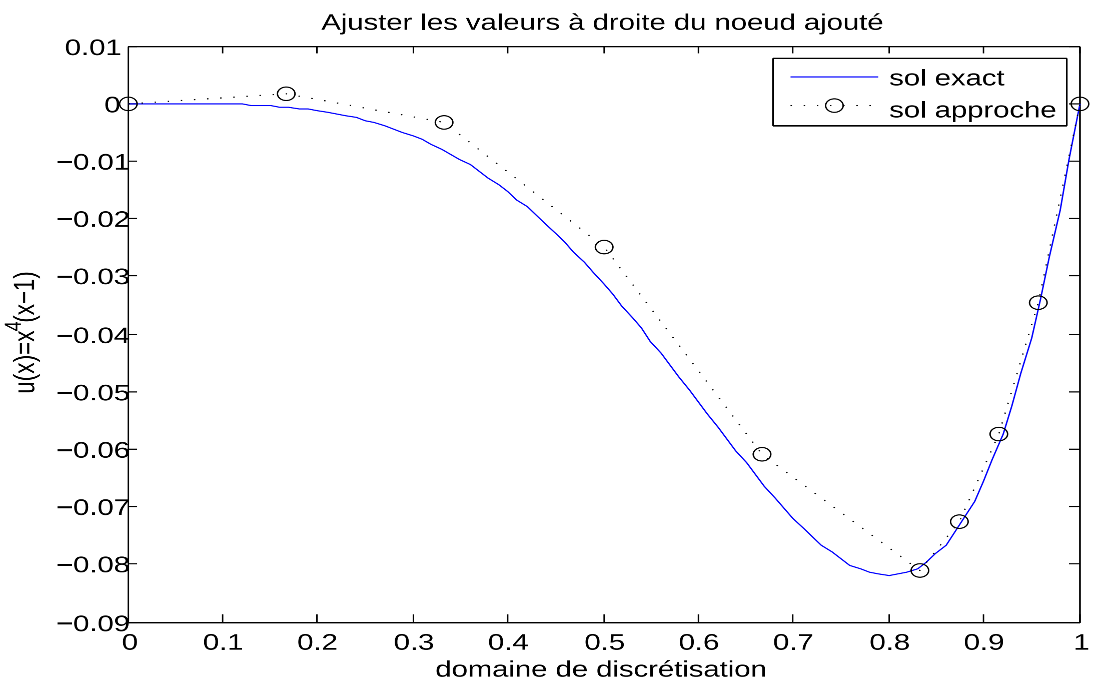

Ajuster les valeurs à gauche du noeud ajouté
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation


Ajout d'un noeud
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

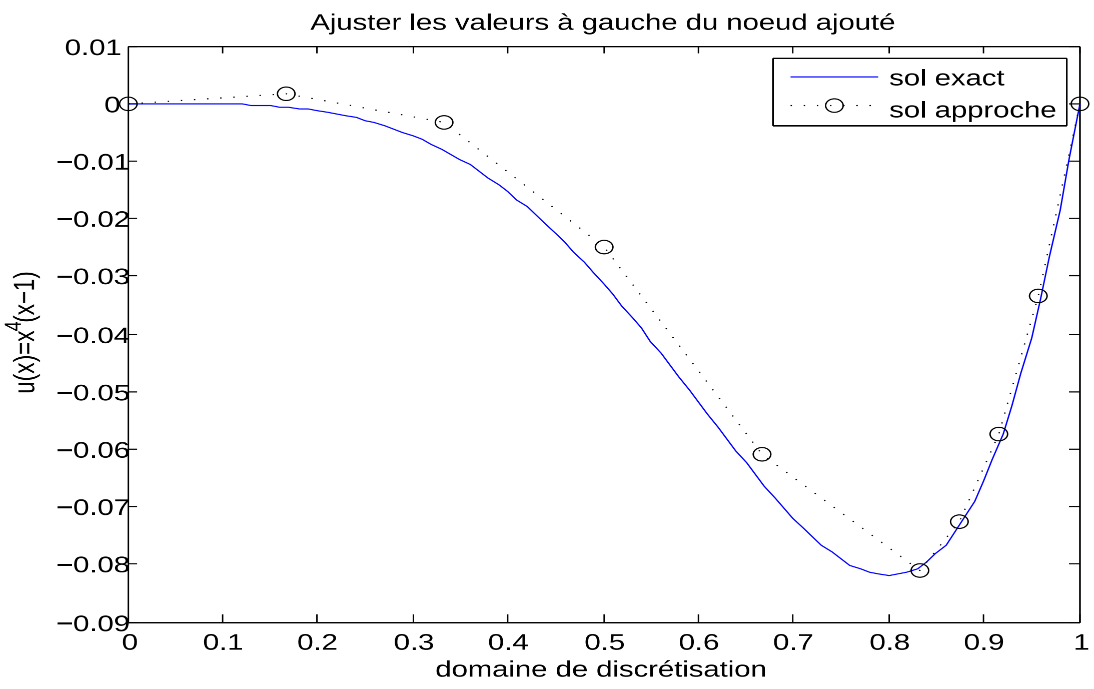


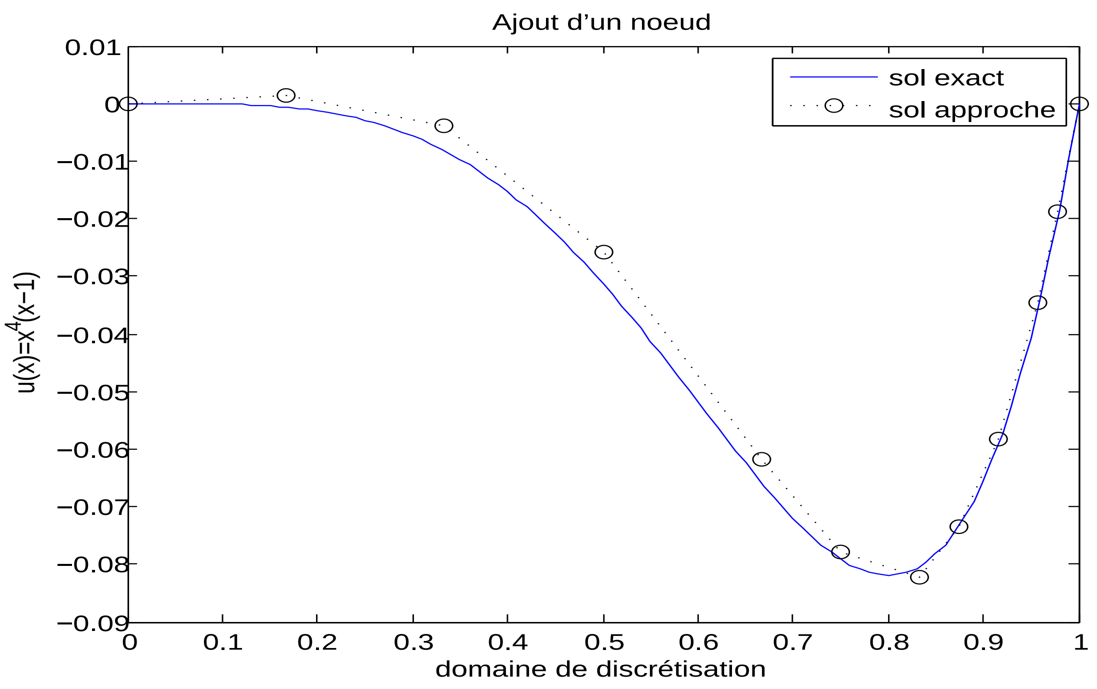
Ajuster les valeurs à gauche du noeud ajouté
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajout d'un noeud
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

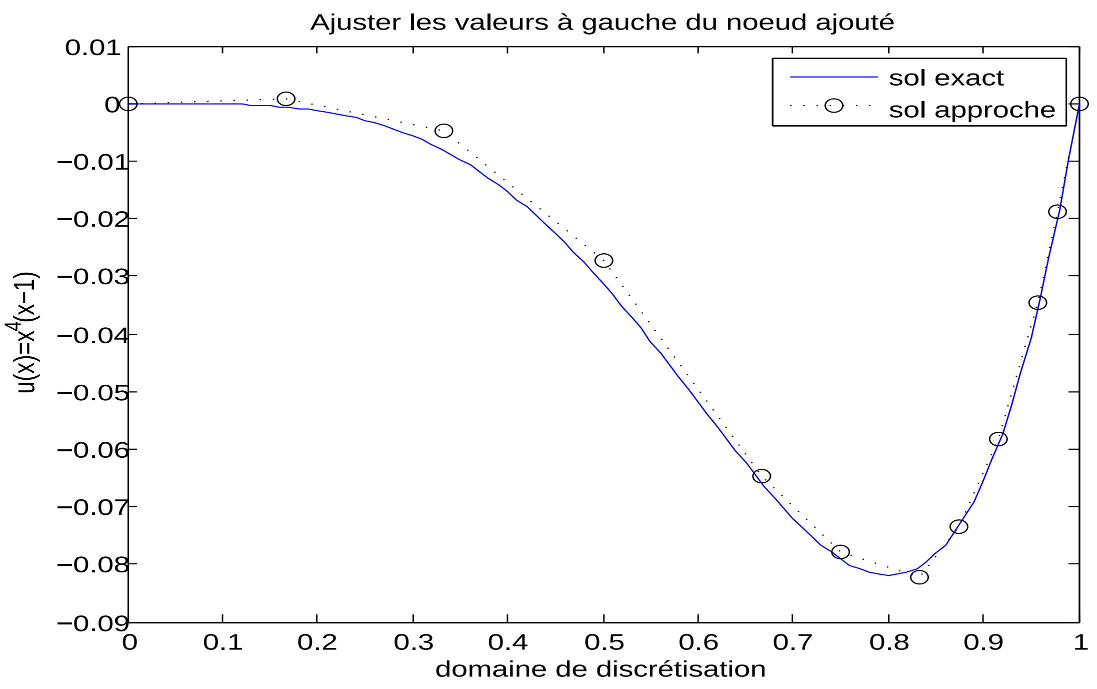


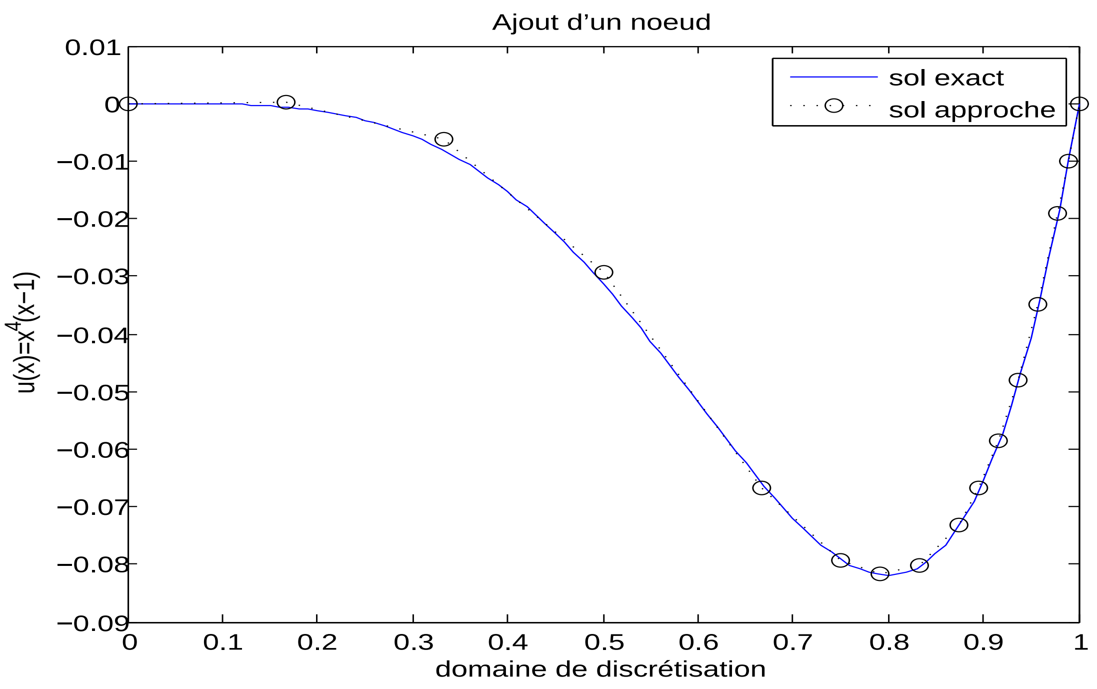
Ajuster les valeurs à droite du noeud ajouté
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajuster les valeurs à gauche du noeud ajouté
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

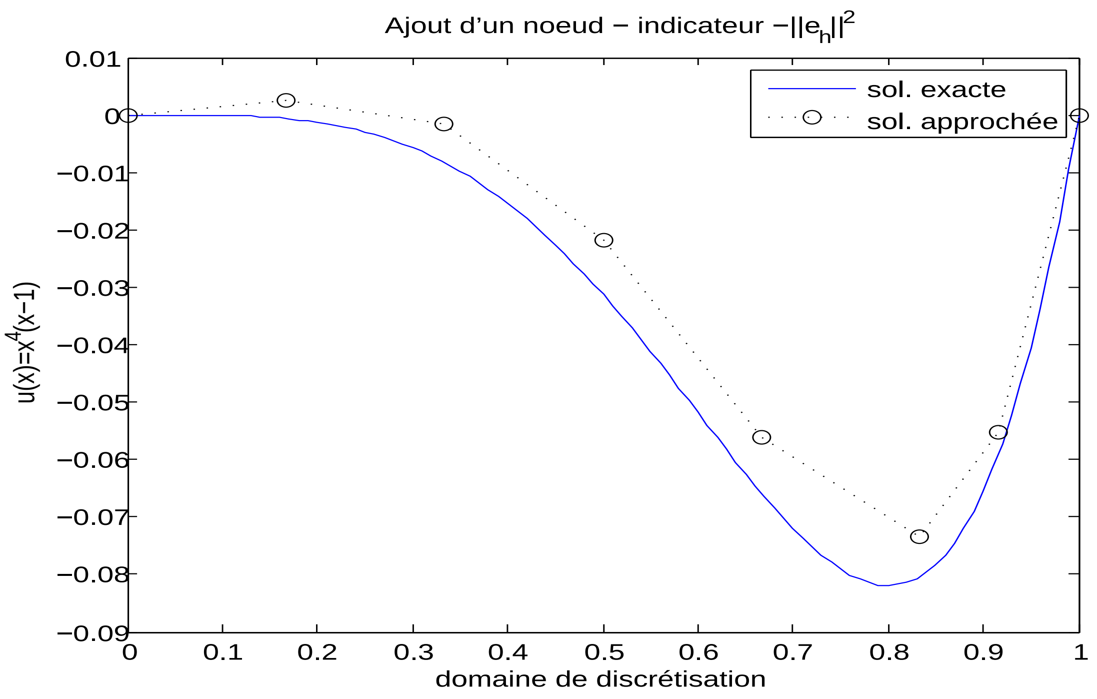


Ajout d'un noeud
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

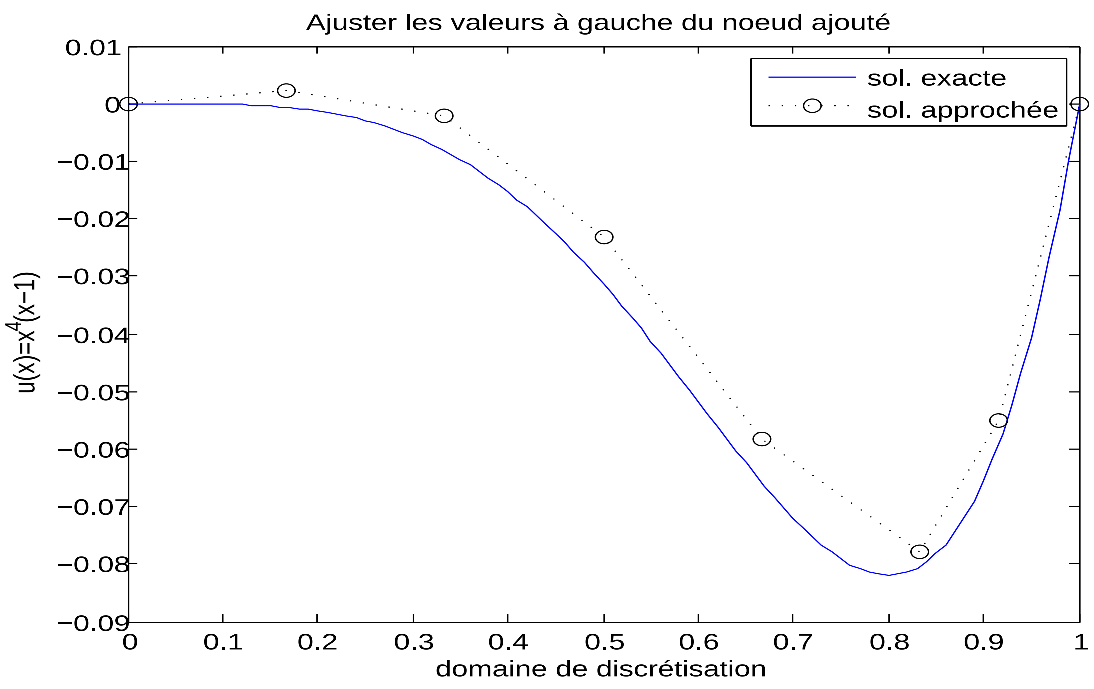

Ajuster les valeurs à gauche du noeud ajouté
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajout d'un noeud
sol exact
sol approche
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

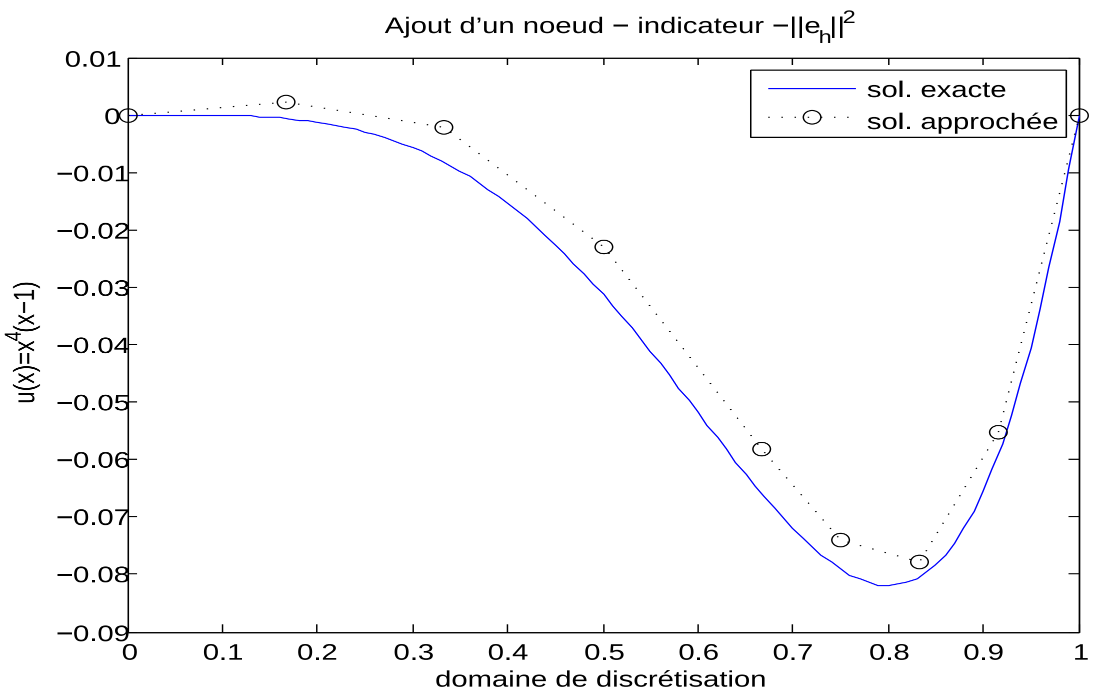
And with $\boldsymbol{g}(\boldsymbol{x_0}) = \boldsymbol{e_h}(\boldsymbol{x_0})\boldsymbol{q_h^0}(\boldsymbol{x_0})$ we obtain



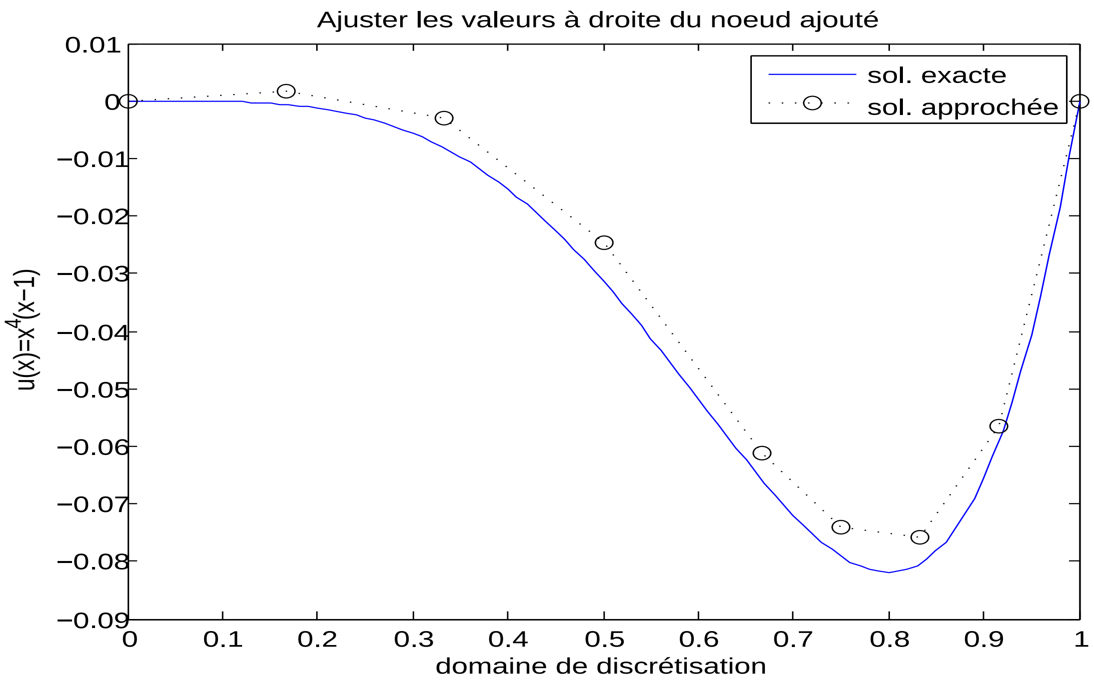
Ajout d'un noeud − indicateur $-||e_h||^2$
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

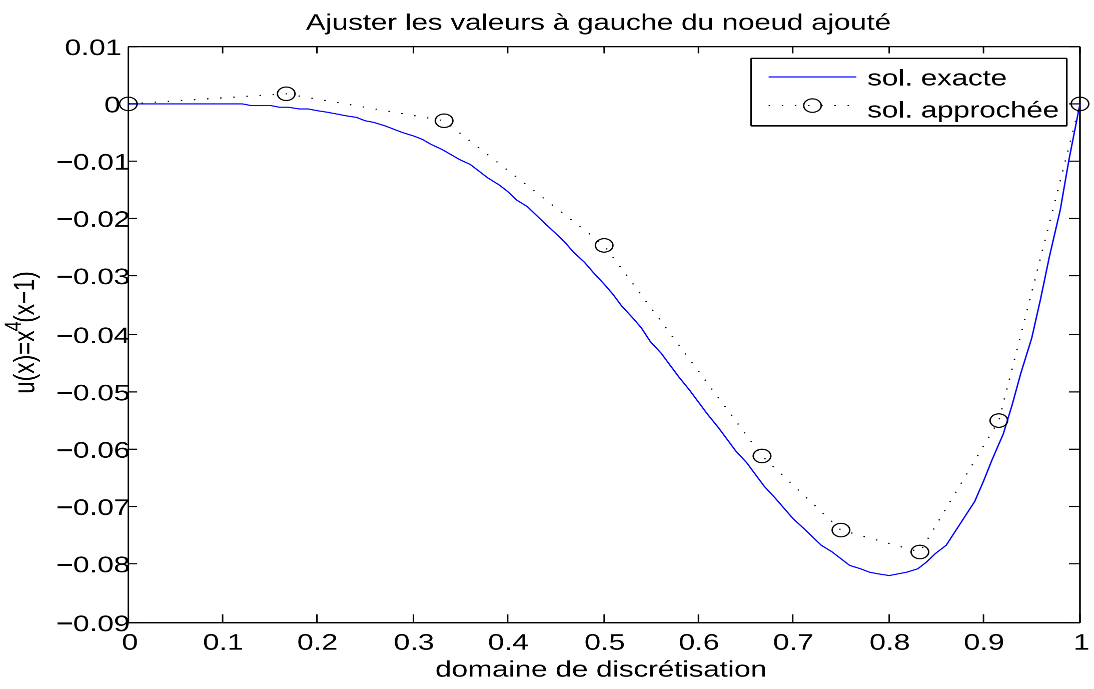


Ajuster les valeurs à gauche du noeud ajouté
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
$u(x)=x^4(x-1)$
domaine de discrétisation
sol. exacte
sol. approchée

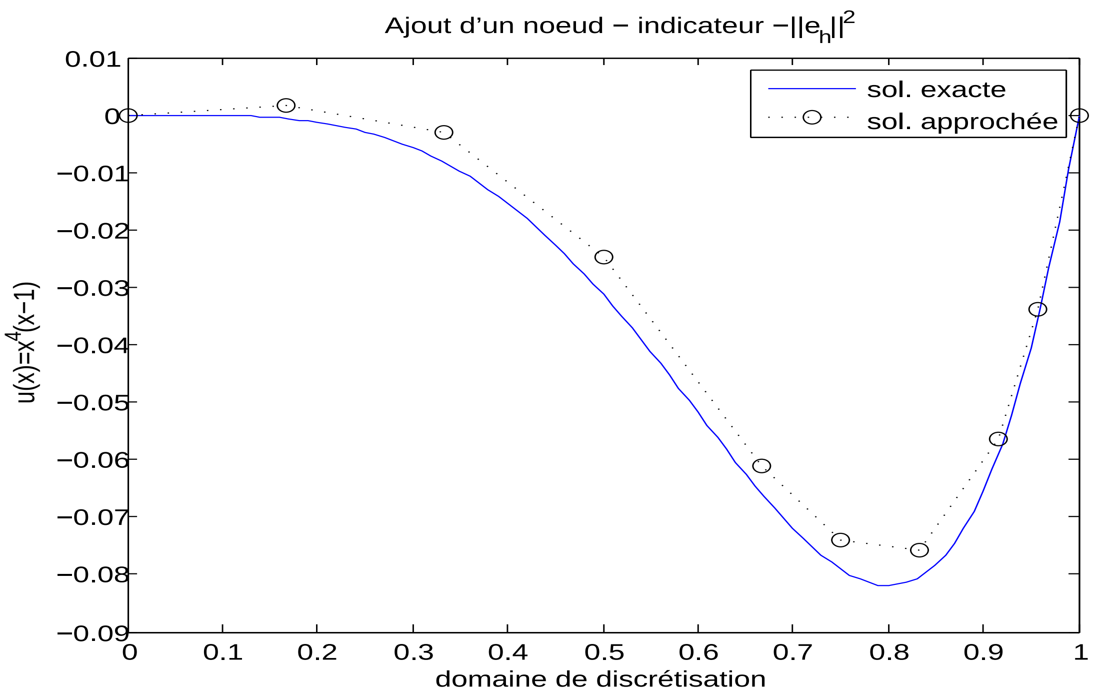


Ajout d'un noeud − indicateur −$||e_h||^2$
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
domaine de discrétisation
$u(x)=x^4(x-1)$
sol. exacte
sol. approchée

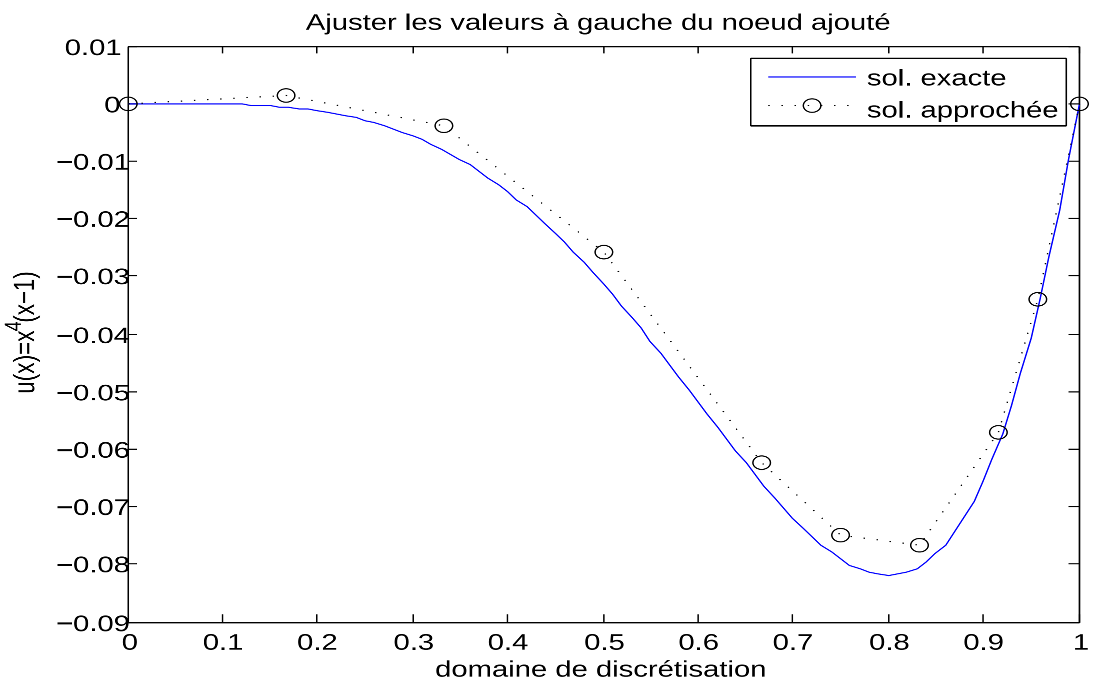

Ajuster les valeurs à droite du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

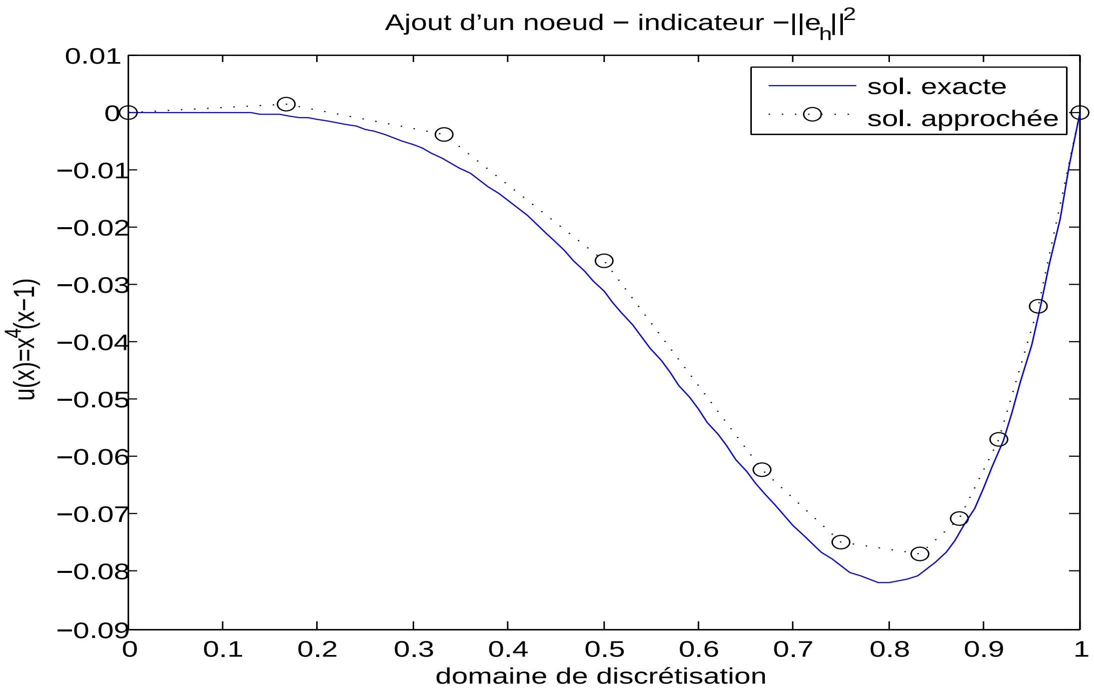


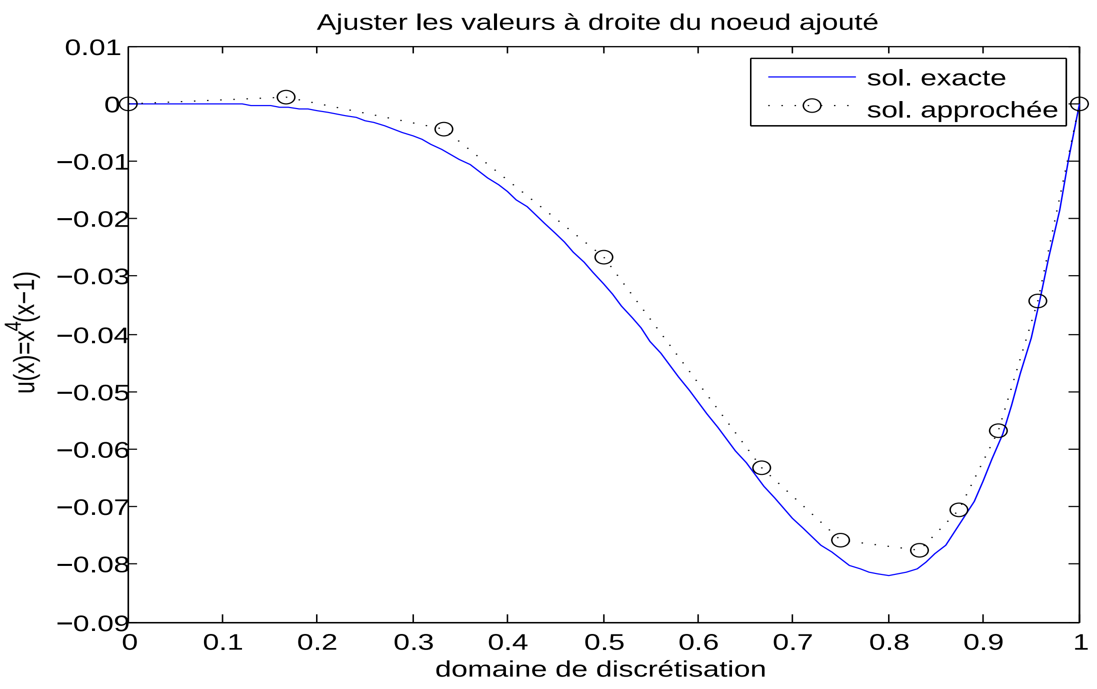
Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

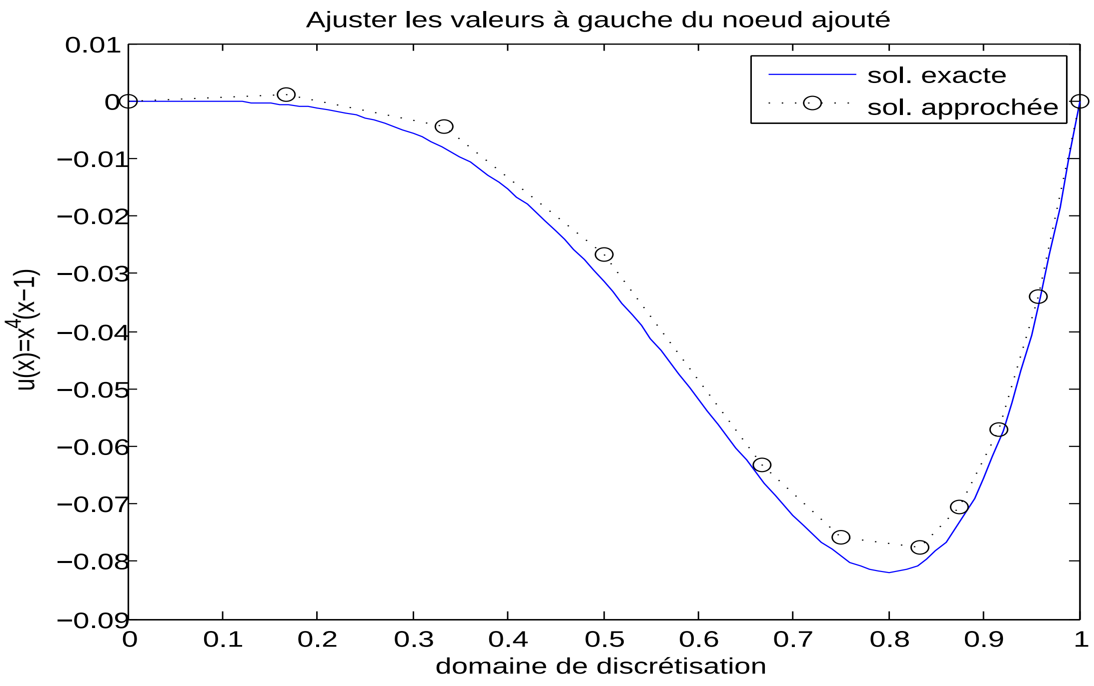


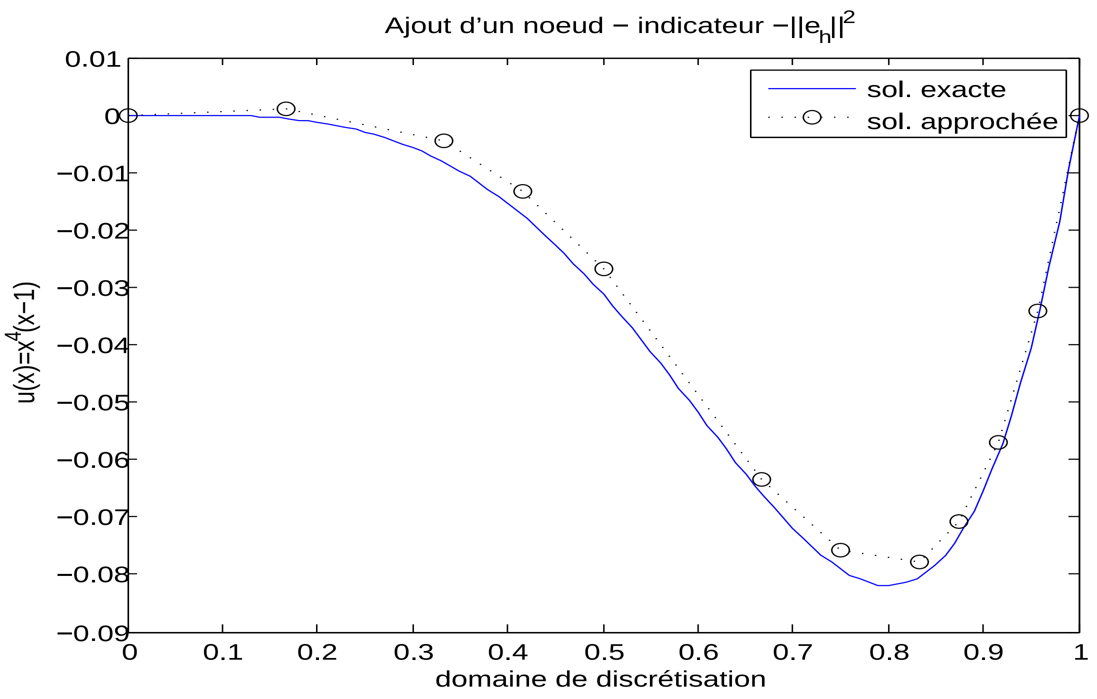
Ajout d'un noeud − indicateur $-||e_h||^2$
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

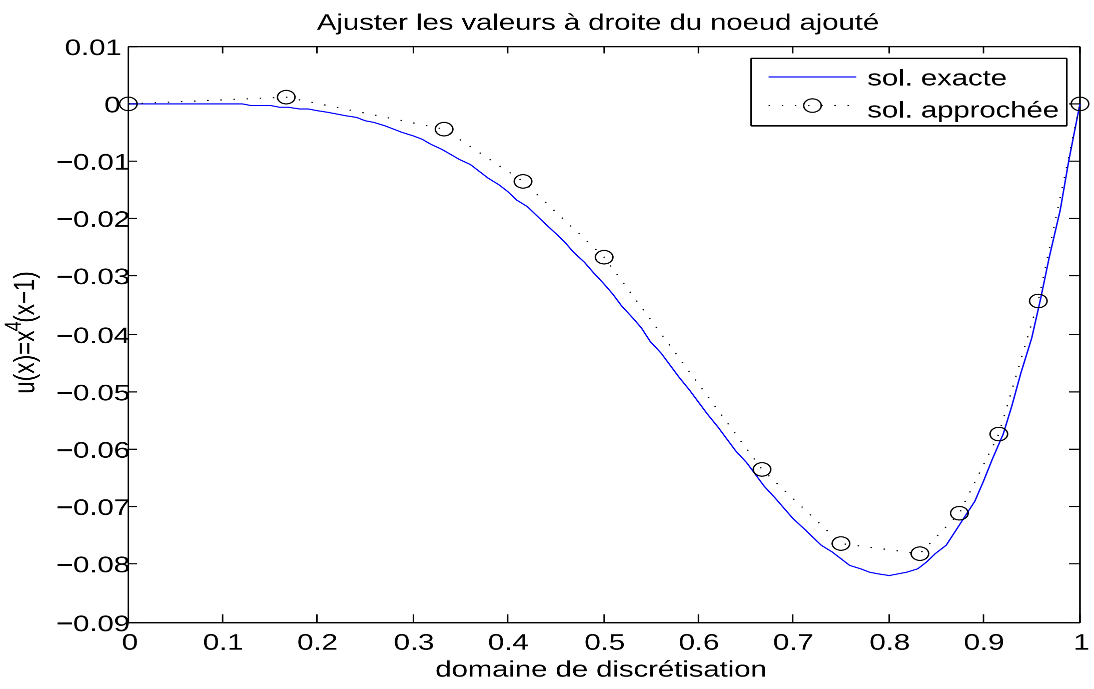

Ajout d'un noeud − indicateur −$||e_h||^2$
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

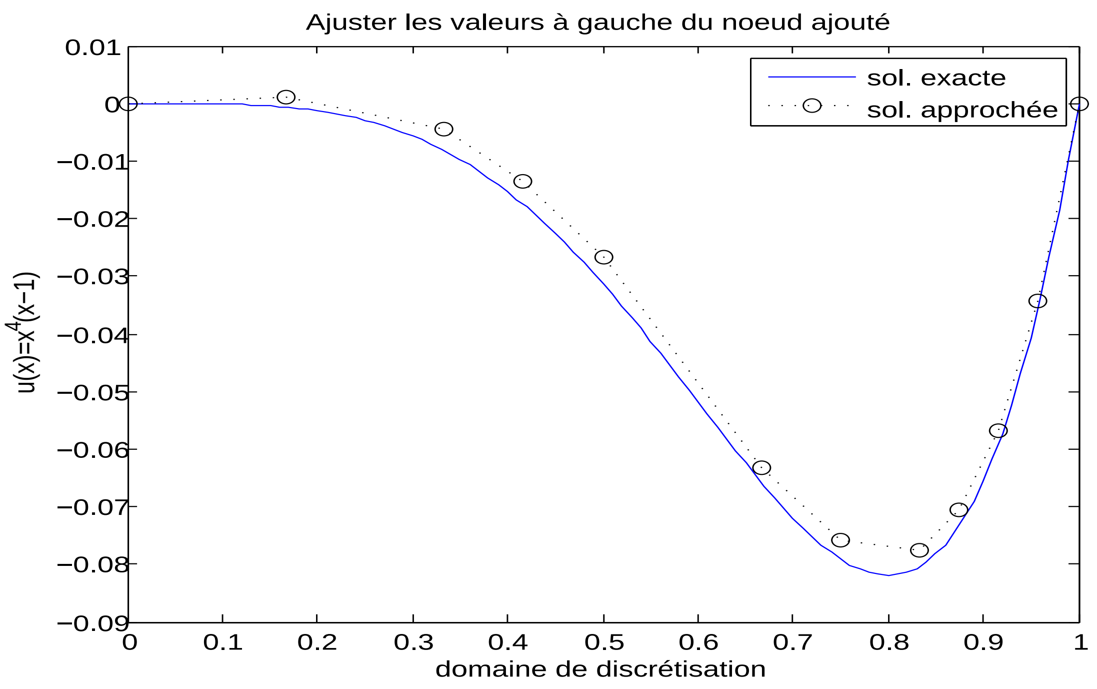

Ajuster les valeurs à droite du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

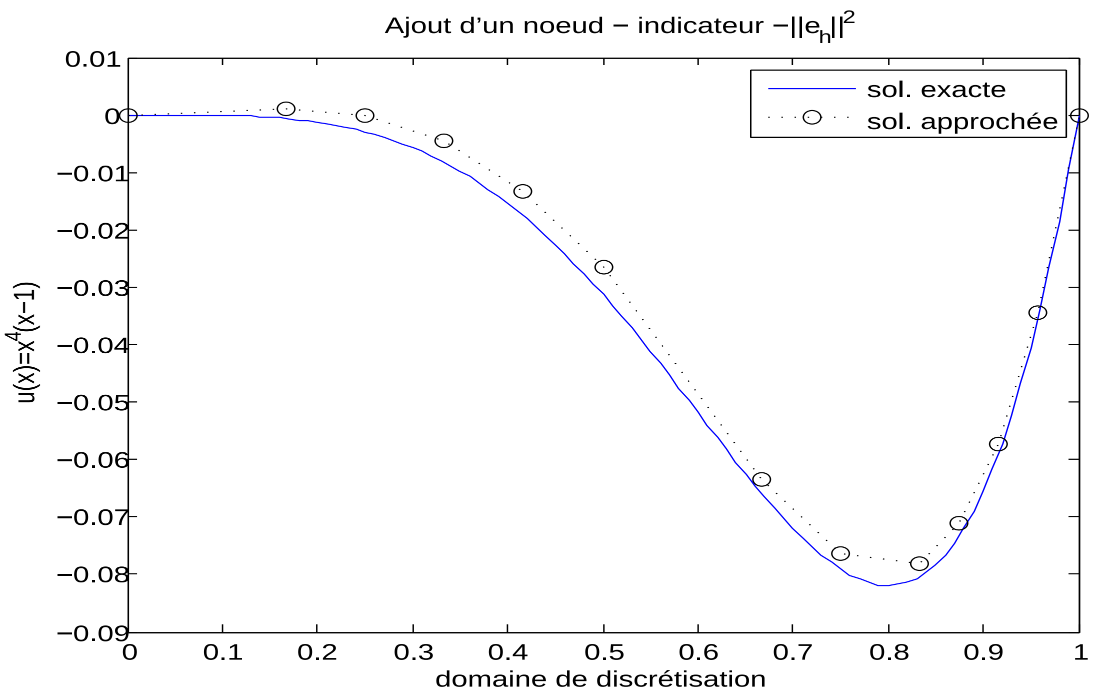


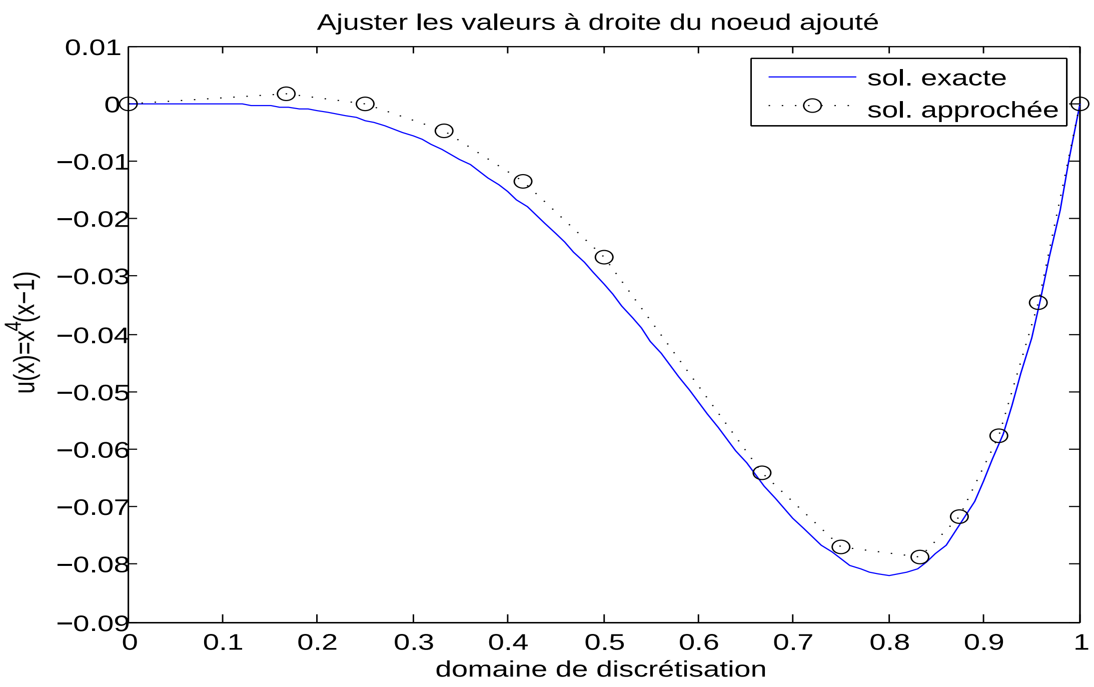
Ajout d'un noeud − indicateur $-\|e_h\|^2$
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajuster les valeurs à droite du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

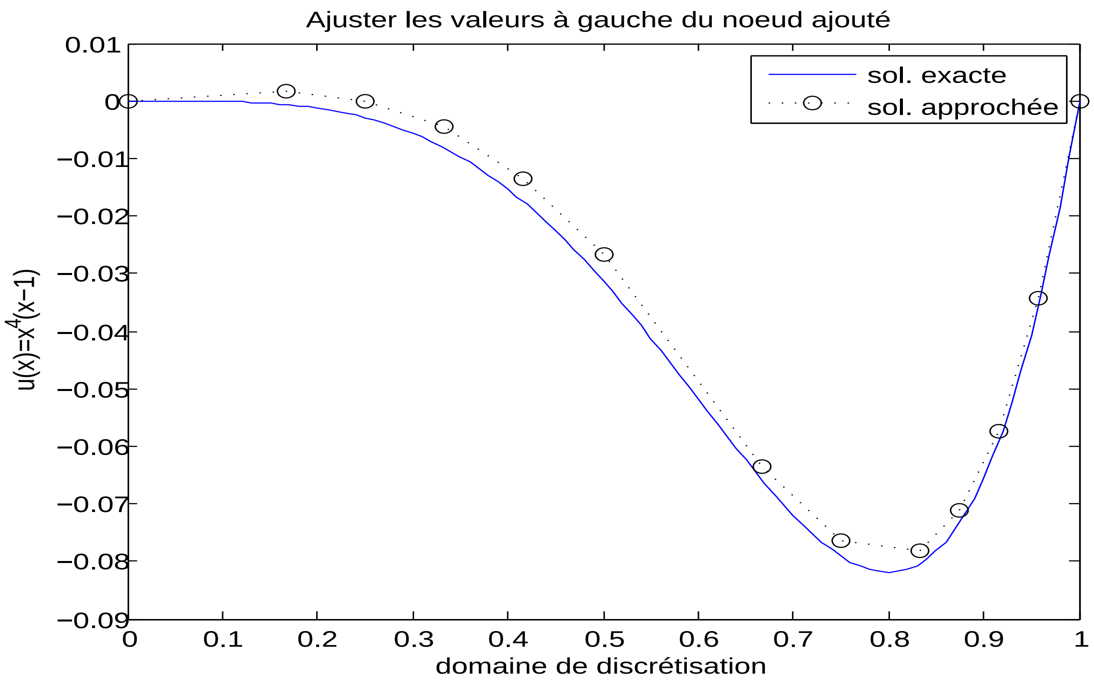


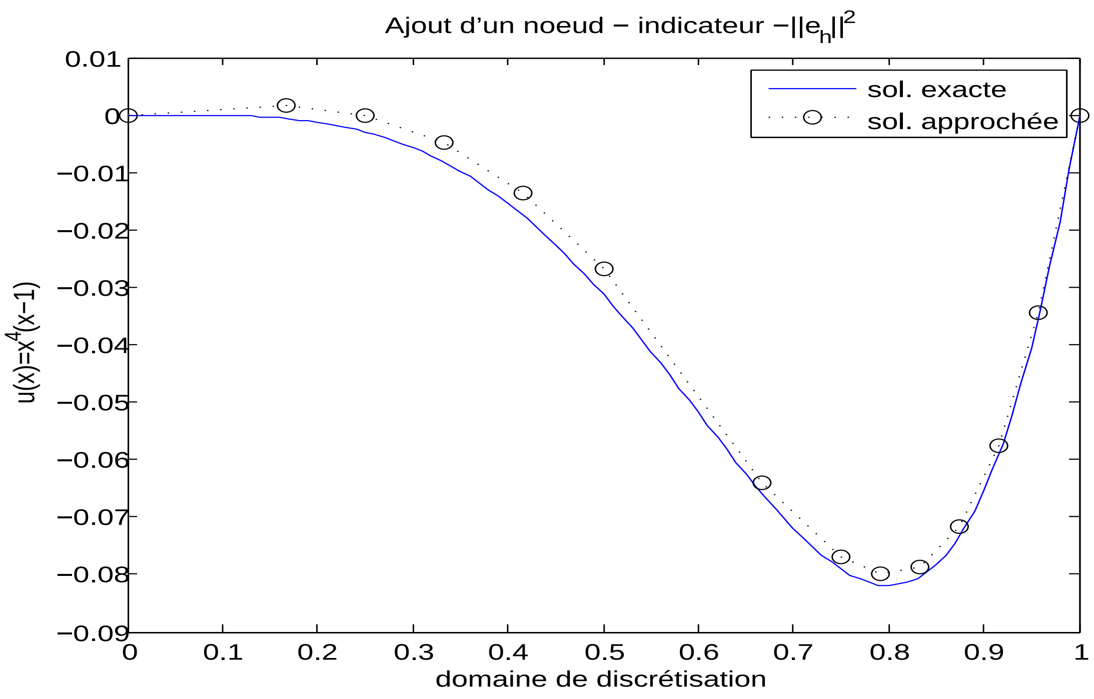
Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

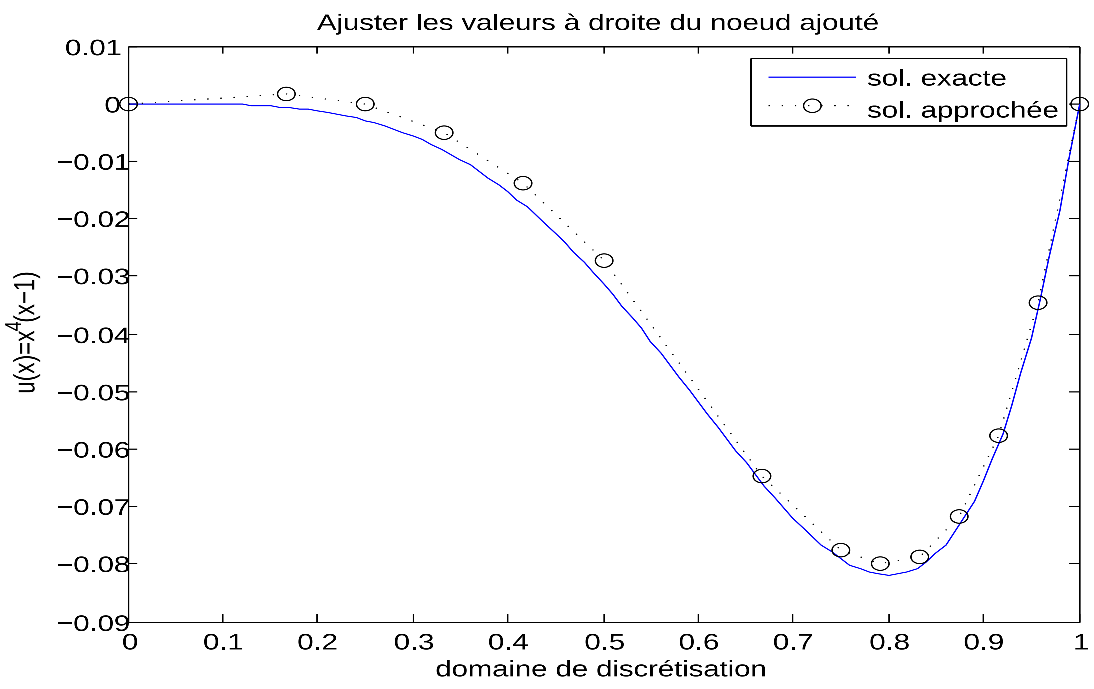

Ajout d'un noeud − indicateur $-||e_h||^2$
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

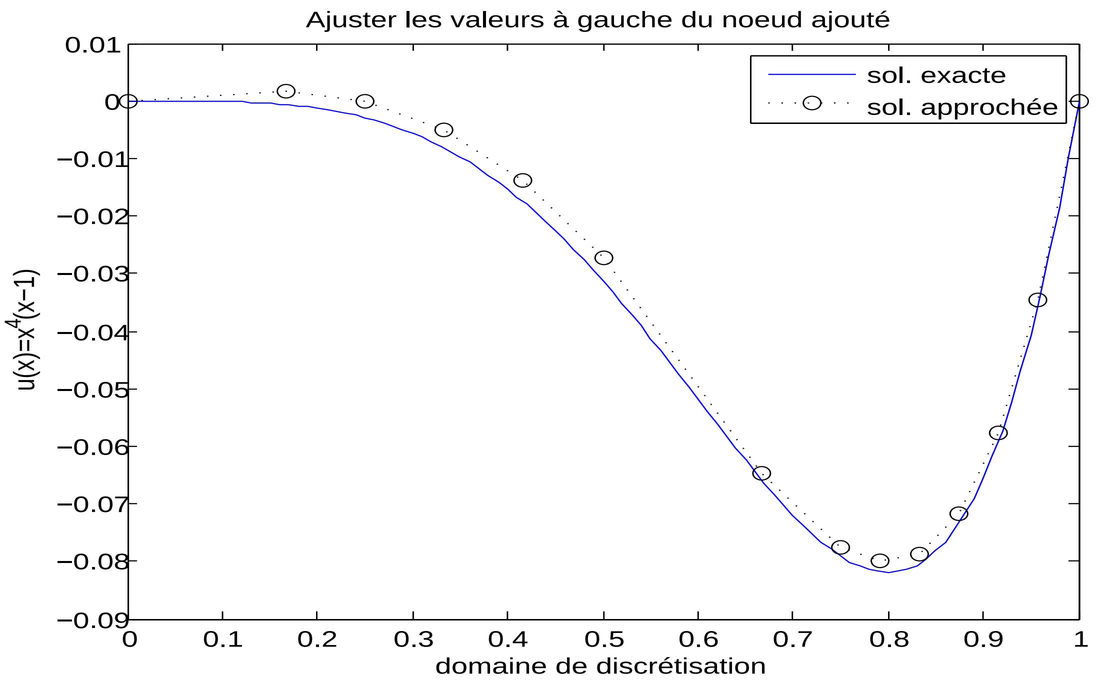


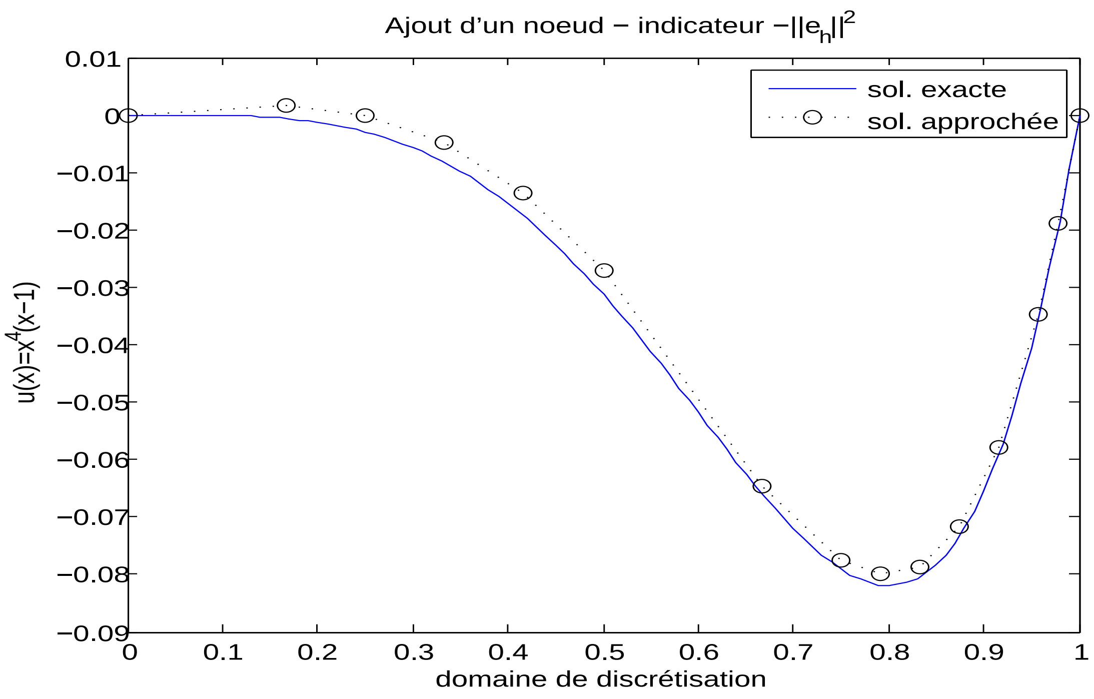
Ajuster les valeurs à droite du noeud ajouté
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

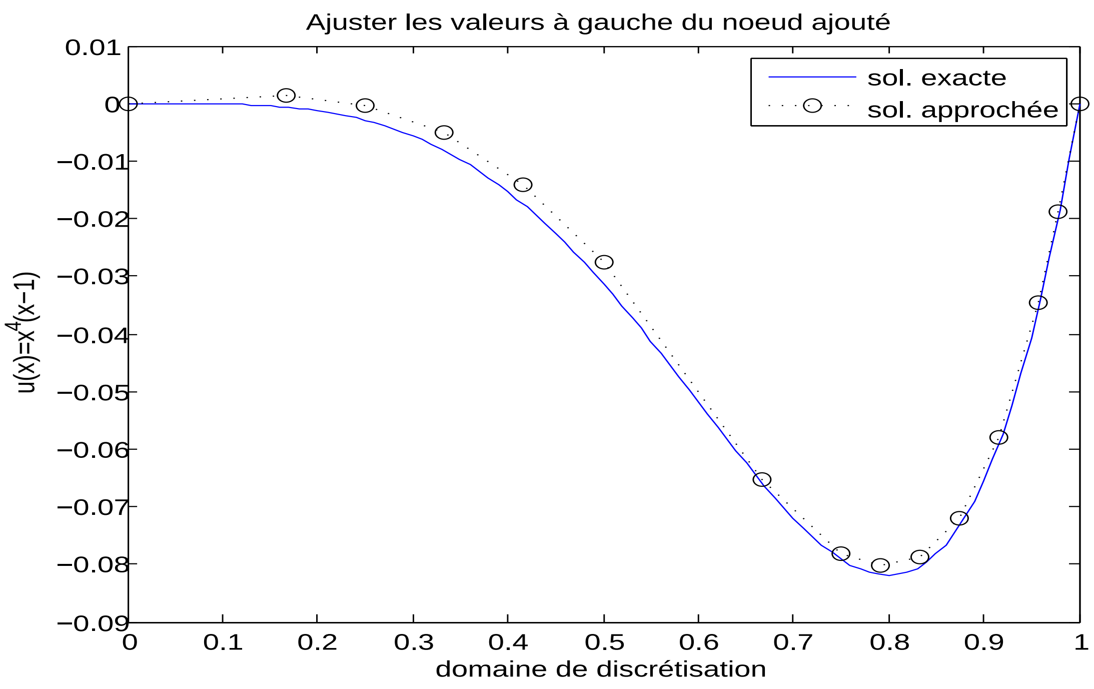

Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajout d'un noeud − indicateur $-||e_h||^2$
sol. exacte
sol. approchée
$u(x)=x^4(x-1)$
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajuster les valeurs à droite du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ajout d'un noeud − indicateur −$||e_h||^2$
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
domaine de discrétisation
$u(x)=x^4(x-1)$
sol. exacte
sol. approchée



Ajuster les valeurs à gauche du noeud ajouté
sol. exacte
sol. approchée
u(x)=x^4(x−1)
domaine de discrétisation
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
−0.08
−0.09
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Conclusion and perspective

- Displacement

- minimize the number of nodes

We can say that the node displacement algorithm works quite well. It remains to find the criterion for swapping edge to complete this algorithm.

- up date: solving the linear system only one times

- Hidden error

it remains to determine the exact topological gradient to determine the right places to add or delete node