The Virtual Element Method for Polygons with Curved Edges

Alessandro Russo

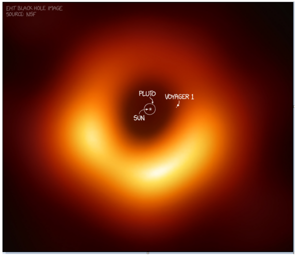
Department of Mathematics and Applications
University of Milano-Bicocca, Milan, Italy
and
IMATI-CNR, Pavia, Italy

POEMs 2019
CIRM, Marseille

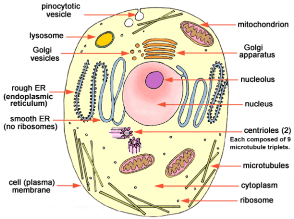
April 29 - May 3, 2019

The General Field Equations

Volcano Eruption



Black Holes



Cell Biology

$$i\hbar\frac{\partial}{\partial t}\boldsymbol{\Psi}(\boldsymbol{r},t)=\left[\frac{-\hbar^2}{2m}\nabla^2+\boldsymbol{V}(\boldsymbol{r},t)\right]\Psi(\boldsymbol{r},t), \quad [D_0+D_1 v]J\boldsymbol{C}^{-1}+\zeta(\nabla\cdot\boldsymbol{u})\boldsymbol{I}=\boldsymbol{D}(v,\boldsymbol{F},\boldsymbol{\Pi})$$

$$\boldsymbol{D}(v,\boldsymbol{F},\boldsymbol{\Pi})=[D_0+D_1v]J\boldsymbol{C}^{-1}+D_0/2Jf_0\oplus f_0+D_2F+\sum_{j=1}^{\infty}[\sigma_j(t,x)\cdot\nabla)u(t)+\nabla\tilde{p}_j(t,x)]\,dW_t^j\,\boldsymbol{F}^{-1}$$

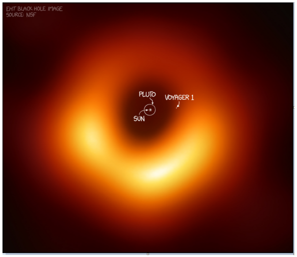
$$\boldsymbol{\mathcal{L}}(\varphi)=-\frac{(\partial\varphi)^2}{2}-\boldsymbol{m}^2\boldsymbol{M}^2\left[\left(1+\varphi^2/\boldsymbol{M}^2\right)^{1/2}-1\right], \quad \boldsymbol{\sigma}=\zeta(\nabla\cdot\boldsymbol{u})\boldsymbol{I}+\mu\left(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^T-\frac{2}{3}(\nabla\cdot\boldsymbol{u})\boldsymbol{I}\right)$$

$$d\boldsymbol{u}(t,x)=[\nu\Delta\boldsymbol{u}(t,x)-(\boldsymbol{u}(t,x)\cdot\nabla)\boldsymbol{u}(t,x)-\nabla p(t,x)]\,dt-[g(|\boldsymbol{u}(t,x)|^2)\,\boldsymbol{u}(t)-f(x,\boldsymbol{u}(t,x))]\,dt$$

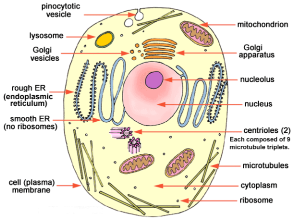
$$R_{\mu\nu}-\frac{1}{2}Rg_{\mu\nu}+\Lambda g_{\mu\nu}=\frac{8\pi G}{c^4}T_{\mu\nu}, \quad \rho\left(\frac{\partial\boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)=-\nabla p+\mu\nabla^2\boldsymbol{u}+\frac{1}{3}\mu\nabla(\nabla\cdot\boldsymbol{u})+\rho\boldsymbol{g}$$

The General Field Equations

Volcano Eruption



Black Holes

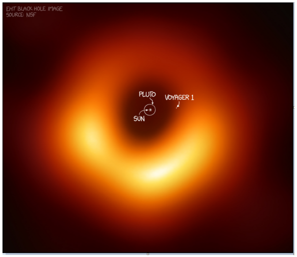


Cell Biology

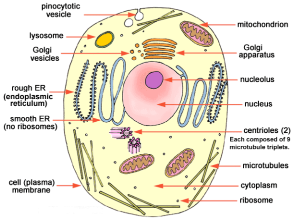
$$\begin{cases} -\Delta u = f & \text{in } \Omega \subset \mathbb{R}^2 \\ u = 0 & \text{on } \partial\Omega. \end{cases}$$

The General Field Equations

Volcano Eruption



Black Holes



Cell Biology

$$\begin{cases} -\Delta u = f & \text{in } \Omega \subset \mathbb{R}^2 \\ u = 0 & \text{on } \partial\Omega. \end{cases}$$

Joint work with L. Beirão da Veiga, F. Brezzi, D. Marini and G. Vacca

Poisson equation

Let Ω be a reasonable domain in $\mathbb{R}^2$, and $f \in L^2(\Omega)$. We define:

$$V := H_0^1(\Omega), \quad a(u,v) := \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x, \quad F(v) := \int_\Omega f \, v \, \mathrm{d}x$$

And we consider the following variational problem:

$$\begin{cases} \text{find } u \in V \text{ such that} \\ a(u,v) = F(v) \quad \text{for all } v \in V(\Omega). \end{cases} \tag{1}$$

This is the weak form of Poisson equation with homogeneous Dirichlet boundary conditions:

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ \quad\; u = 0 & \text{on } \partial\Omega. \end{cases}$$

We will consider conformal Galerkin approximations of (1), with $V_h \subset V$.

Key steps of the Virtual Element Method

- Decompose the domain into general polygons.
- Define local spaces using differential operators. Ensure that polynomials are inside. Choose carefully the degrees of freedom.
- Define projection operators onto polynomials which are computable from the degrees of freedom.
- Use the projection operators to discretize the problem, ensuring k-consistency ($\approx$ patch test). Add stability.

Key steps of the Virtual Element Method

- Decompose the domain into general polygons.
- Define local spaces using differential operators. Ensure that polynomials are inside. Choose carefully the degrees of freedom.
- Define projection operators onto polynomials which are computable from the degrees of freedom.
- Use the projection operators to discretize the problem, ensuring k-consistency ($\approx$ patch test). Add stability.

The local VEM space $V_k(P)$

Let P be a polygon with straight edges. Mimicking the classical Lagrange Finite Elements for triangles, we define a finite element space $V_k(P)$ on P such that:

- $V_k(P)$ contains the space $\mathbb{P}_k(P)$ of polynomials of degree less than or equal to k (plus - possibly - other non-polynomial functions);
- a function in $V_k(P)$ restricted to an edge e is in $\mathbb{P}_k(e)$, so that if two polygons P and P' have an edge in common, the two spaces $V_k(P)$ and $V_k(P')$ can be easily "glued" in $C^0(P \cup P')$ (compatibility with FEM);
- given functions $u_h, v_h \in V_k(P)$, I can "compute" the bilinear form $a_P(u_h, v_h)$ and the load term $F_P(v_h)$ (or at least reasonably good approximations).

The local VEM space $V_k(P)$

We start with a trivial remark:

$$p_k \in \mathbb{P}_k(P) \implies \Delta p_k \in \mathbb{P}_{k-2}(P).$$

Since we want that our space contains the polynomials of degree k, we define the local VEM space as follows:

$$V_k(P) := \{v_h \in C^0(\bar{P}) \text{ such that: } v_{h|e} \in \mathbb{P}_k(e),\ \Delta v_h \in \mathbb{P}_{k-2}(P)\}$$

If the polygon E has N_e edges (and $N_V = N_e$ vertices), it is clear that

$$\dim V_k(P) = N_V + (k-1)\,N_e + \dim \mathbb{P}_{k-2}(P) = k\,N_e + \frac{k(k-1)}{2}$$

Is the space $V_k(P)$ OK?

The local VEM space $V_k(P)$

Is the space $V_k(P)$ OK?

YES: it contains polynomials of degree up to k and is conformal, so (under some regularity assumptions on the polygonal mesh) order k approximation in H^1 is guaranteed...

BUT: the functions in $V_k(P)$ are not known explicitly; they are defined through the solution of a PDE in the element.

We need computable projectors!

Degrees of freedom in $V_k(P)$

Recall that

$$\dim V_k(P) = N_V + (k-1)\,N_e + \dim \mathbb{P}_{k-2}(P).$$

As degrees of freedom in $V_k(P)$, we choose:

- the value of v_h at the $\boxed{\text{vertices}}$ and at $\boxed{k-1 \text{ points on each edge;}}$
- the (scaled) moments $\boxed{\dfrac{1}{|P|}\displaystyle\int_P v_h m_{\boldsymbol{\alpha}}\,\mathrm{d}x}$ for $0 \le |\boldsymbol{\alpha}| \le k-2$

where $m_{\boldsymbol{\alpha}}$ is a scaled monomial of degree $|\boldsymbol{\alpha}|$.

We easily see that the degrees of freedom above are *unisolvent* in $V_k(P)$.

The Π_k^∇ projector onto $\mathbb{P}_k(P)$

We can define a projector $\Pi_k^\nabla : V_k(P) \to \mathbb{P}_k(P)$ which is computable.

The projector Π_k^∇ is orthogonal with respect to the energy scalar product in $H^1(P)$:

$$\int_P \nabla\left[\Pi_k^\nabla v_h\right] \cdot \nabla p_k \,\mathrm{d}x = \int_P \nabla v_h \cdot \nabla p_k \,\mathrm{d}x \quad \text{for all } p_k \in \mathbb{P}_k(P)$$

$$\int_P \Pi_k^\nabla v_h \,\mathrm{d}x = \int_P v_h \,\mathrm{d}x \quad \text{to fix the constant function}$$

By integration by parts we easily see that $\Pi_k^\nabla v_h$ is computable from the degrees of freedom of v_h.

The Π_k^∇ projector onto $\mathbb{P}_k(P)$

We basically only need to check that the right hand side

$$\int_P \nabla v_h \cdot \nabla p_k \,\mathrm{d}x$$

is computable from the degrees of freedom of v_h. Integrating by parts:

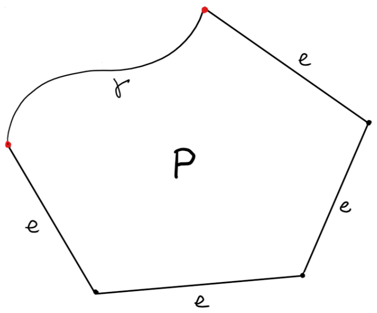
$$\int_P \nabla v_h \cdot \nabla p_k \,\mathrm{d}x = -\int_P v_h \,\Delta p_k \,\mathrm{d}x + \int_{\partial P} v_h \,(\nabla p_k \cdot \boldsymbol{n})\,\mathrm{d}s$$

and the result follows by observing that

- $\Delta p_k \in \mathbb{P}_{k-2}(P)$ which implies that $\displaystyle\int_P v_h \,\Delta p_k \,\mathrm{d}x$ is computable,
- v_h is a known polynomial of degree k on each edge,
- ∇p_k is a known polynomial of degree $k-1$ on each edge.

Polygons with curved edges

Assume that we have a polygon with a curved edge γ:



In the rest of the talk:

- e will denote a straight edge;
- γ a possibly curved edge.

The local VEM space $V_k(P)$

We make a crucial remark:

- In order to define the local VEM space $V_k(P)$, we only need to change the definition of the space on the curved edge γ.

The rest of the construction can be done in the same way as before.

A function $v_h \in V_k(P)$ is then defined by the following properties:

- for each straight edge e, $v_{h|e} \in \mathbb{P}_k(e)$
- for each curved edge γ, $v_{h|e} \in V_k(\gamma)$ (to be defined!)
- $v_{h|\partial P}$ continuous
- $\Delta v_h \in \mathbb{P}_{k-2}(P)$ as before.

Does $V_k(P)$ still contain polynomials of degree k?

Does $V_k(P)$ contain polynomials of degree k?

The answer is:
it depends on whether $V_k(\gamma)$ contains the restrictions to γ of polynomials in two variables.

We define

$$P_k(\gamma) = \{\text{restrictions to } \gamma \text{ of polynomials in two variables}\}$$

If $P_k(\gamma)$ is "almost contained" in $V_k(\gamma)$
then
$\mathbb{P}_k(P)$ is "almost contained" in $V_k(P)$.

Does $V_k(P)$ still contain polynomials of degree k?

Let $p_k(x,y)$ be a polynomial of degree k in two variables. We want to see to what extent "$p_k \in V_k(P)$".

Assume that $V_k(\gamma)$ contains "almost" the restriction of $\mathbb{P}_k(P)$ to γ, i.e. that there exists a small function ε which lives on γ such that

$$p_{k|\gamma} \notin V_k(\gamma) \quad \text{but} \quad \boxed{p_{k|\gamma} + \varepsilon \in V_k(\gamma)}$$

Consider the virtual function $w_h \in V_k(P)$ that solves:

- $w_h = p_k + \varepsilon$ on γ
- $w_h = p_k$ on the other (straight) edges e;
- $\Delta w_h = \Delta p_k \in \mathbb{P}_{k-2}(P)$ in P.

Does $V_k(P)$ still contain polynomials of degree k?

Then the difference $(w_h - p_k)$ solves the problem

- $(w_h - p_k) = \varepsilon$ on γ
- $(w_h - p_k) = 0$ on the other (straight) edges;
- $\Delta(w_h - p_k) = 0$ in P.

Hence

$$w_h - p_k = O(\varepsilon) \text{ on } \gamma \implies \boxed{w_h = p_k + O(\varepsilon) \text{ in } P} \quad \text{in the right norms.}$$

In other words:

- if the space $V_k(\gamma)$ on the curved edge "approximates well" the restrictions of polynomials,
- then the virtual space $V_k(P)$ "approximates well" the polynomials "automatically".

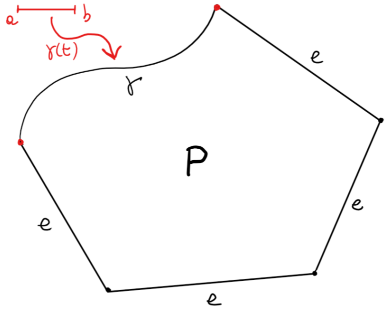
The “rough” and the “phony” way

So far we have explored two quite different ways for the definition of the space $V_k(\gamma)$: the unofficial names are “rough” and “phony” (“rozzi” and “farlocchi” in italian).

- The “rough” way. In this case, we assume that the curve γ is parametrized and we define $V_k(\gamma)$ as the functions which are polynomials of degree k in the parameter.
- The “phony” way. Here we bite the bullet and define $V_k(\gamma)$ directly as the restrictions to γ of all polynomials in two variables: $V_k(\gamma) = P_k(\gamma)$.

It is clear that if the edge γ happens to be straight, in both cases we are back to the space of polynomials in one variable (unless you are crazy and do not parametrize the edge with an affine map!).

The rough way



We then assume the the edge γ is given in parametric form:

$$t \mapsto \gamma(t) = (x(t), y(t)), \quad t \in [a, b].$$

The rough way

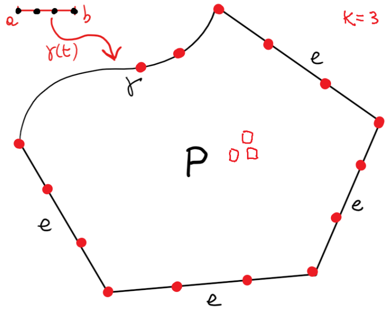
For each parametrized curved edge $t \mapsto \gamma(t)$, we define the edge space $V_k(\gamma)$ as

$$V_k(\gamma) = \{w : \gamma \to \mathbb{R} : \\ w(\gamma(t)) = v_h(x(t), y(t)) \text{ is a polynomial of degree } k\}$$

It is clear that if γ is straight, its parametrization is linear (affine) and we recover the standard VEM space.

The degrees of freedom are the same as before

- pointwise values at the vertices;
- pointwise values at $k-1$ "equispaced" points on the edge (equispaced with respect to $t \in [a,b]$ for a curved edge);
- moments against scaled monomials up to degree $k-2$ inside.



Hence the dimension of $V_k(P)$ is the same as before.

The Π_k^∇ projector onto $\mathbb{P}_k(P)$ can still be computed

We can still define the projector $\Pi_k^\nabla : V_k(P) \to \mathbb{P}_k(P)$.

As before, the projector Π_k^∇ is orthogonal with respect to the energy scalar product in $H^1(P)$:

$$\int_P \nabla\left[\Pi_k^\nabla v_h\right] \cdot \nabla p_k \,\mathrm{d}x = \int_P \nabla v_h \cdot \nabla p_k \,\mathrm{d}x \quad \text{for all } p_k \in \mathbb{P}_k(P)$$

$$\int_{\partial P} \Pi_k^\nabla v_h \,\mathrm{d}x = \int_{\partial P} v_h \,\mathrm{d}x \quad \text{to fix the constant function}$$

We show now that $\Pi_k^\nabla v_h$ is computable if we know the degrees of freedom of v_h.

The Π_k^∇ projector onto $\mathbb{P}_k(P)$ can still be computed

We only need to check that the right hand side

$$\int_P \nabla v_h \cdot \nabla p_k \,\mathrm{d}x$$

is computable from the degrees of freedom of v_h. Integrating by parts:

$$\int_P \nabla v_h \cdot \nabla p_k \,\mathrm{d}x = -\int_P v_h \,\Delta p_k \,\mathrm{d}x + \int_{\partial P} v_h \,(\nabla p_k \cdot \boldsymbol{n})\,\mathrm{d}s$$

and the result follows by observing that

- $\Delta p_k \in \mathbb{P}_{k-2}(P)$ which implies that $\displaystyle\int_P v_h \,\Delta p_k \,\mathrm{d}x$ is computable,
- v_h is a known polynomial of degree k on each straight edge e,
- v_h is a known function in $V_k(\gamma)$ on each curved edge γ,
- ∇p_k is a known polynomial of degree $k-1$ on each edge.

Issues

- The local VEM space $V_k(P)$ does not contain polynomials, since the corresponding edge space $V_k(\gamma)$ does not contain the restriction of polynomials in two variables. Hence the method will not pass the patch test.
- We need to integrate polynomials on curvilinear polygons and (known) functions on curves; of course in general we will not be able to compute the integrals exactly. Hence, the degree of precision of the quadrature formulas will come into play.

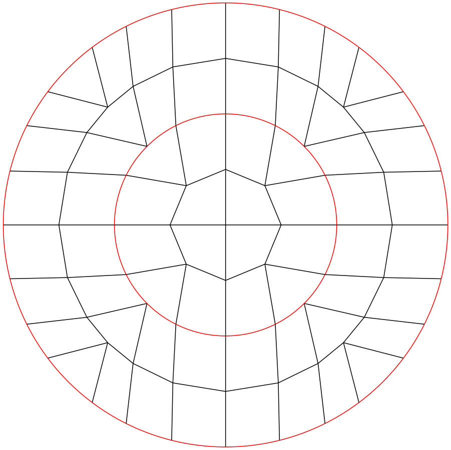
Does it work??????

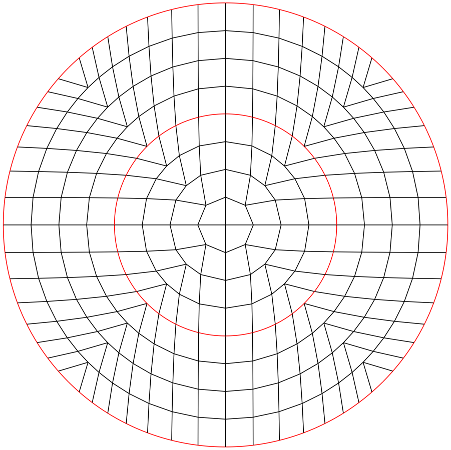
Roughly speaking, if a local space $V_k(P)$ contains all polynomials up to degree k, and we have a “regular” mesh sequence with h going to zero, a Gakerkin method converges with the right order.

We can miss the polynomials, but not too much… otherwise convergence is lost.

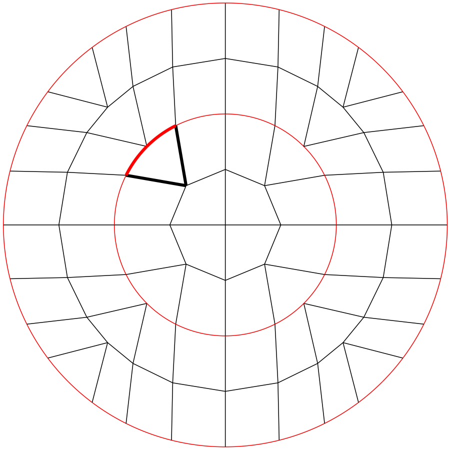
The easiest case is when the curved edges come from the boundary of the domain or from an internal, fixed interface, which is the case covered by the isoparametric classical finite elements.

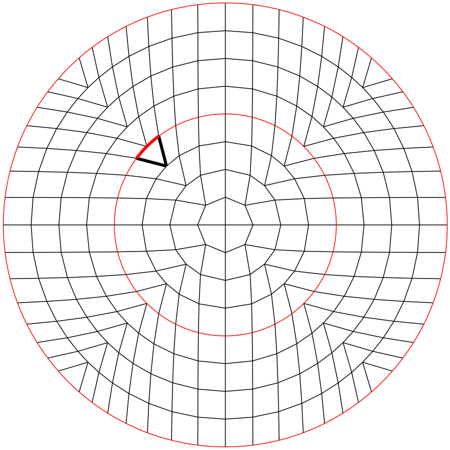
Does it work??????



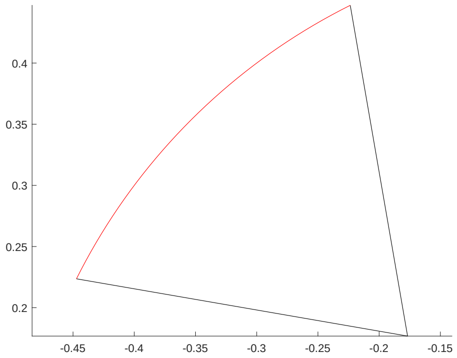


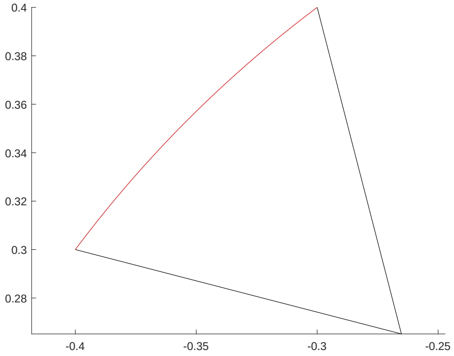
Does it work??????





Does it work??????





When the elements get smaller, they also get flatter.

Hence for the rough VEM, the gap between polynomials and $V_k(P)$ diminishes as $h \longrightarrow 0$ and the method works.

Main result

In the paper *The Virtual Element Method with Curved Edges*, ESAIM: M2AN 53 (2019), pp. 375–404 (also arXiv:1711.04306v2) together with L. Beirão da Veiga and G. Vacca we have proved that for the "fixed curved boundary" or the "fixed curved interface" problem, the Virtual Element Method described above converges with optimal rates, provided all integrals are computed without error.

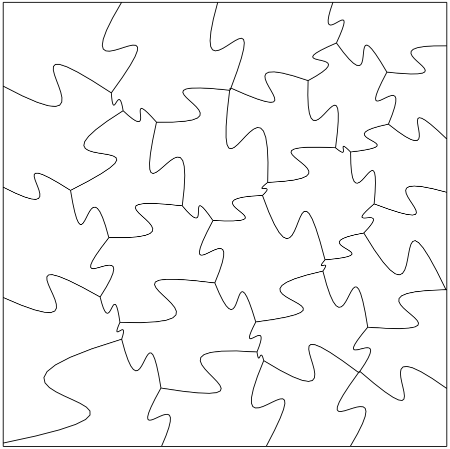
This is the extension to VEM of the results that hold for the classical isoparametric Finite Elements.

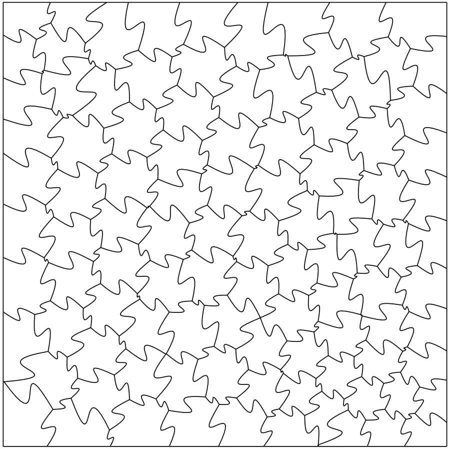
From the numerical experiments it is apparent that it is enough to compute all integrals with the same precision necessary for the straight edge case. In other words, the bottom line is:

take into account the curved geometry
but reuse the code for the straight edge case.

General curved edges - not flattening

Jigsaw mesh





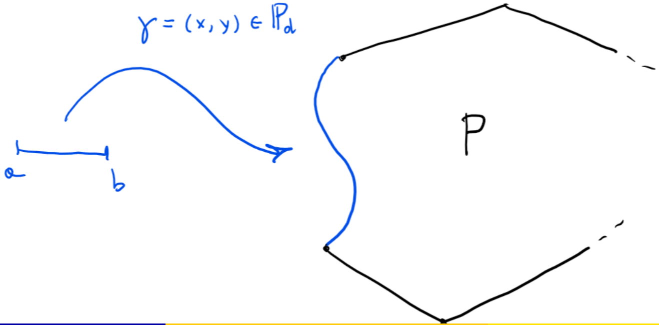
In this case we don't have convergence!

Polynomial edges

Idea: we can raise the degree of the VEM on the edges independently of the accuracy k. The VEM local space is still well defined.

Assume that the curve describing the curved edge γ is a polynomial of degree d:

$$\gamma(t) = (x(t), y(t)), \quad x, y \in \mathbb{P}_d(\mathbb{R})$$



The "enhanced rough" way

Now if $p_k(x, y)$ is a polynomial of degree k in two variables, the function

$$p_k \circ \gamma : t \mapsto p_k(x(t), y(t))$$

is a polynomial of degree kd in t.

Hence, in order to include all polynomials of degree k in the local VEM space $V_k(P)$, we change the definition of $V_k(\gamma)$ as follows:

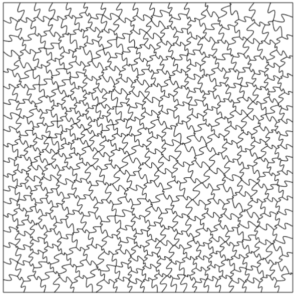
- for each curved edge $\gamma(t)$ which is a polynomial of degree d,

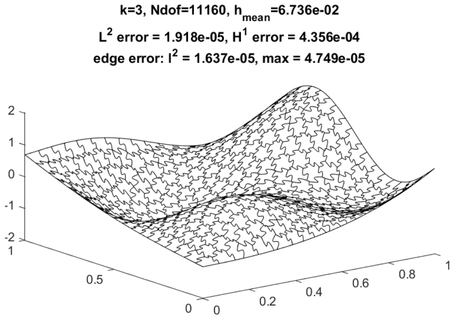
  $$v_h \in V_k(\gamma) \quad \text{means} \quad v_h \circ \gamma \in \mathbb{P}_{kd}(\mathbb{R})$$

  In other words, the function $\boxed{t \mapsto v_h(x(t), y(t))}$ must a polynomial of degree $\boxed{kd}$ in the paramatrization parameter t.
- for each straight edge e, $v_{h|e} \in \mathbb{P}_k(e)$ (as before)
- $\Delta v_h \in \mathbb{P}_{k-2}(P)$ (as before)

The “enhanced rough” way

Jigsaw mesh, enhanced rough VEM

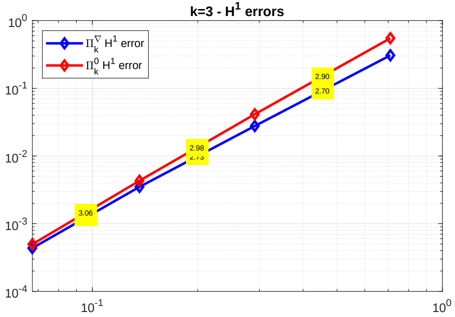


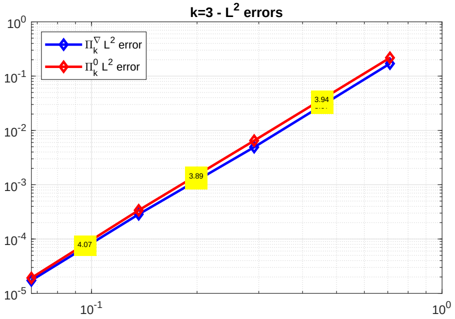

k=3, Ndof=11160, h_mean=6.736e-02
L^2 error = 1.918e-05, H^1 error = 4.356e-04
edge error: l^2 = 1.637e-05, max = 4.749e-05
2
1
0
-1
-2
1
0.5
0
0
0.2
0.4
0.6
0.8
1

$\gamma =$ cubic polynomial, degree $3 \times 3 = 9$ on each edge

The “enhanced rough” way

Jigsaw mesh, enhanced rough VEM





The phony way

If we want to keep polynomials inside our local VEM spaces, we can simply define the edge space as the restriction to the edge of polynomials of degree k in two variables, i.e.

$$V_k(\gamma) = P_k(\gamma)$$

If we proceed in this way, as we observed before we ensure that polynomials are included in $V_k(P)$, whatever the shape of γ.

This seems very natural, maybe a little bit expensive, but we don't need any parametrization - at least for the definition.

The real problem is the dimension of the space $P_k(\gamma)$!

Dimension of $P_k(\gamma)$

If γ is a "generic" curve (i.e. not the zeros of some polynomial) then

$$\dim \mathbb{P}_k(\gamma) = \dim \mathbb{P}_k(\mathbb{R}^2) = \frac{(k+1)(k+2)}{2}$$

However, if γ is straight then

$$\dim \mathbb{P}_k(\gamma) = \dim \mathbb{P}_k(\mathbb{R}^1) = k+1$$

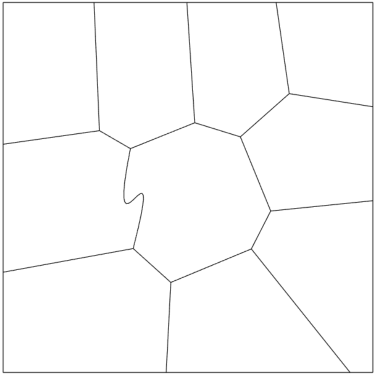
and all intermediate case are possible!

The situation is even worse, since the dimension of $\mathbb{P}_k(\gamma)$ is unstable with respect to small changes in γ, and I think it would be really hard to decide in the code case by case the "numerical" dimension of $\mathbb{P}_k(\gamma)$.

After long discussions, our point of view is the following: we don't want to check the dimension of $\mathbb{P}_k(\gamma)$. We want a method that works in ALL cases, even when γ is a segment. In other words: we don't want to care.

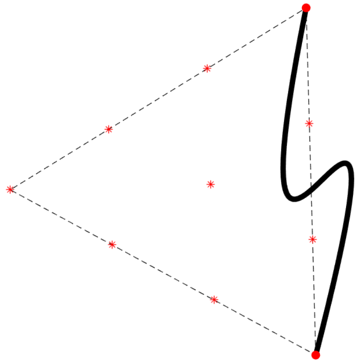
Degrees of freedom for $P_k(\gamma)$

Since $P_k(\gamma)$ are the restrictions to γ of polynomials in two variables up to degree k, we describe $P_k(\gamma)$ with the polynomials in two variables up to degree k. Hence we start from a mesh with a curved edge γ:



Degrees of freedom for $P_k(\gamma)$

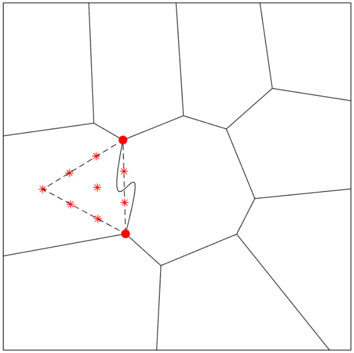
For the curved edge γ we construct the Lagrange nodes for polynomials in two variables up to degree k:



We identify a function in $V_k(\gamma)$ by its pointwise values in the Lagrange nodes: the “phony” dofs.

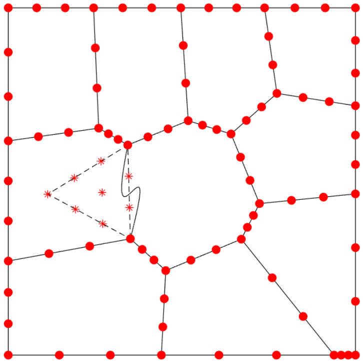
Degrees of freedom for $P_k(\gamma)$

The phony dofs at the vertices are special because they are also “true” dofs, connecting the two worlds:



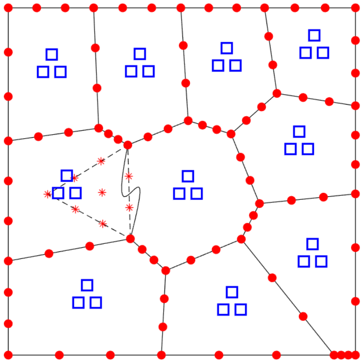
Degrees of freedom for $P_k(\gamma)$

The phony dofs at the vertices are special because they are also “true” dofs, connecting the two worlds:



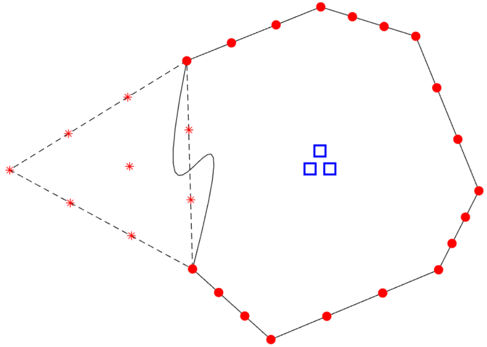
Degrees of freedom for $P_k(\gamma)$

The phony dofs at the vertices are special because they are also “true” dofs, connecting the two worlds:



Degrees of freedom for $V_k(P)$

We go back now to our polygon P and the local VEM space $V_k(P)$:



There are 8 phony dofs $\{*\}$ and 22 true dofs $\{\bullet,\square\}$. If the edge γ becomes straight, then 6 of the phony dofs become useless. All intermediate cases are possible.

Degrees of freedom for $V_k(P)$

Hence $\dim V_k(P)$ is between 24 and 30 depending on γ.

Since we don't want to decide what kind of curve γ is, we keep 30 parameters to describe $V_k(P)$.

Note that we do not assume that they are degrees of freedom.

Given an array $\boldsymbol{V_h}$ in $\mathbb{R}^{30}$, we define the associated function $v_h \in V_k(P)$ in the following way:

- Take the 10 values of $\boldsymbol{V_h}$ corresponding to the 8 phony dofs and the 2 true dofs on the vertices of γ;
- build the corresponding interpolation polynomial p_k in two variables (recall that in the example $k = 3$);
- define v_h on γ as the restriction of p_k on γ.
- Use the true edge dofs of $\boldsymbol{V_h}$ to determine v_h on the other edges.
- Use the true internal dofs of $\boldsymbol{V_h}$ to determine v_h inside.

Degrees of freedom for $V_k(P)$

The reverse path would be:

- Take a function $v_h \in V_k(P)$;
- restrict v_h on the edge γ;
- from the restriction of v_h to γ identify a polynomial p_k in two variables;
- the array $\boldsymbol{V_h} \in \mathbb{R}^{30}$ is given by
  - the true dofs of v_h for straight edges and in the interior of P;
  - the value of p_k at the phony dofs

As we have already seen, this path is unstable.

$$\boldsymbol{V_h} \Longrightarrow v_h \text{ is OK;}$$

$$v_h \Longrightarrow \boldsymbol{V_h} \text{ is forbidden.}$$

Degrees of freedom for $V_k(P)$

The reverse path would be:

- Take a function $v_h \in V_k(P)$;
- restrict v_h on the edge γ;
- from the restriction of v_h to γ identify a polynomial p_k in two variables;
- the array $\boldsymbol{V_h} \in \mathbb{R}^{30}$ is given by
  - the true dofs of v_h for straight edges and in the interior of P;
  - the value of p_k at the phony dofs

As we have already seen, this path is unstable.

$$\boldsymbol{V_h} \Longrightarrow v_h \text{ is OK;}$$

$$v_h \Longrightarrow \boldsymbol{V_h} \text{ is forbidden.}$$

Degrees of freedom for $\mathbb{P}_k(P)$

We have seen that we can go from the parameters in $\mathbb{R}^{30}$ to a function but not viceversa.

There is a notable exception: we can reverse the path when the function is a polynomial p_k.

In fact, regardless of the shape of γ, we can always define an array $\boldsymbol{P_k}$ associated with a polynomial p_k by simply computing the value of p_k at the phony dofs, simply observing that polynomials are globally defined.

$$\boldsymbol{P_k} \Longleftrightarrow p_k \text{ is OK for polynomials}$$

$$\boldsymbol{P_k} \Longrightarrow p_k \Longrightarrow \boldsymbol{P_k}$$

The phony method

The simplest VEM for the Laplace operator defines the local approximate bilinear form on P in the following way:

$$a_h^P(u_h, v_h) = \int_P \nabla\Pi_k^\nabla u_h \cdot \nabla\Pi_k^\nabla v_h \,\mathrm{d}x + \sum_\ell \mathsf{dof}_\ell(u_h - \Pi_k^\nabla u_h)\,\mathsf{dof}_\ell(v_h - \Pi_k^\nabla v_h)$$

where the index ℓ in the summation runs across all dofs.

This is the so-called “dofi-dofi” stabilization.

The phony method

Since our unknown will be the array $\boldsymbol{U_h}$, we need to find a related approximate bilinear form defined on arrays:

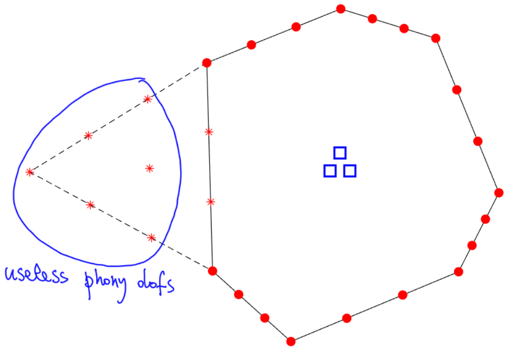
$$a_h^P(\boldsymbol{U_h}, \boldsymbol{V_h}) =$$
$$\int_P \nabla \Pi_k^\nabla \boldsymbol{U_h} \cdot \nabla \Pi_k^\nabla \boldsymbol{V_h} \, \mathrm{d}x + \sum_\ell \mathsf{dof}_\ell(\boldsymbol{U_h} - \Pi_k^\nabla \boldsymbol{U_h}) \, \mathsf{dof}_\ell(\boldsymbol{V_h} - \Pi_k^\nabla \boldsymbol{V_h})$$

Now:

- $\Pi_k^\nabla \boldsymbol{V_h}$ is defined through the projection of the associated function v_h:
$$\boldsymbol{V_h} \Longrightarrow v_h \Longrightarrow \Pi_k^\nabla v_h$$
- $\mathsf{dof}_\ell(\boldsymbol{V_h})$ is simply the ℓ-th component of the array $\boldsymbol{V_h}$;
- $\Pi_k^\nabla \boldsymbol{V_h} = \Pi_k^\nabla v_h$ is a polynomial of degree k, and we have seen that we can associate to it an array of dofs in a consistent way.

Local stiffness matrix in the singular case

When the curve γ “degenerates”, the consistency part becomes singular but the stability takes care of it:



Local stiffness matrix in the singular case

We see now what happens to the local stiffness matrix in this case.

- Suppose that γ is straight and take the array $\boldsymbol{\Phi_i}$ which is 1 only on the useless phony dof i and 0 on the other dofs (true and phony).
- The corresponding basis function φ_i is identically zero, and the same obviously happens for $\Pi_k^\nabla \varphi_i$.
- Hence the consistency term is all zero:

$$\int_P \nabla \Pi_k^\nabla \boldsymbol{\Phi_j} \cdot \nabla \Pi_k^\nabla \boldsymbol{\Phi_i} \, \mathrm{d}x = 0 \text{ for all } j.$$

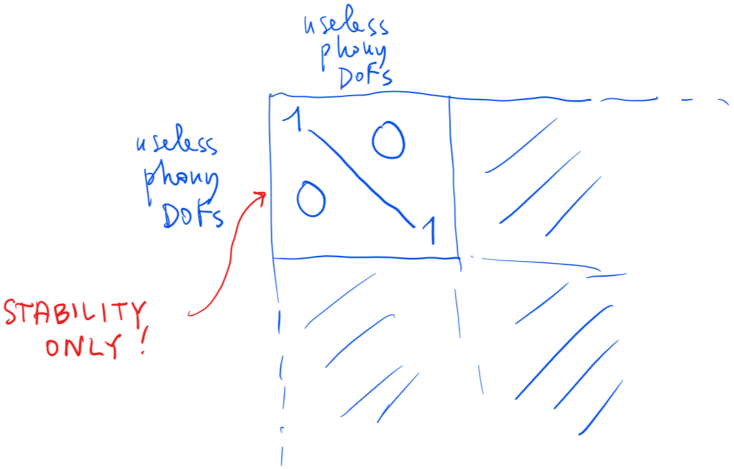
- The stability term reduces to

$$\sum_\ell \mathrm{dof}_\ell(\boldsymbol{\Phi_j} - \Pi_k^\nabla \boldsymbol{\Phi_j}) \, \mathrm{dof}_\ell(\boldsymbol{\Phi_i} - \Pi_k^\nabla \boldsymbol{\Phi_i}) = \sum_\ell \mathrm{dof}_\ell(\boldsymbol{\Phi_j} - \Pi_k^\nabla \boldsymbol{\Phi_j}) \, \delta_{\ell i} = \delta_{ji} - \mathrm{dof}_i(\Pi_k^\nabla \boldsymbol{\Phi_j})$$

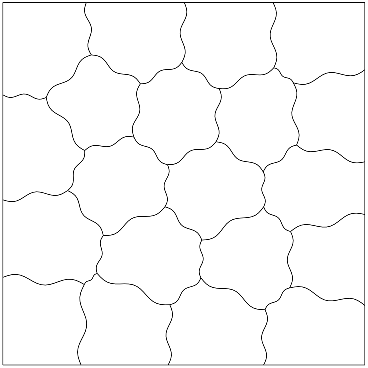
and if j is also a useless phony dof, then we get the identity matrix.

Local stiffness matrix in the singular case

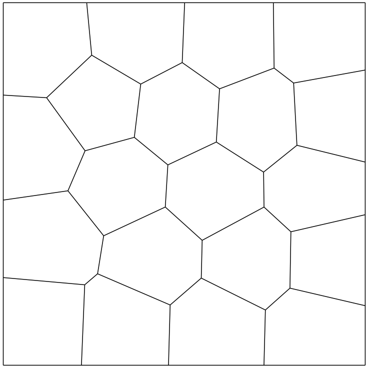
Local VEM stiffness matrix



Numerical experiments



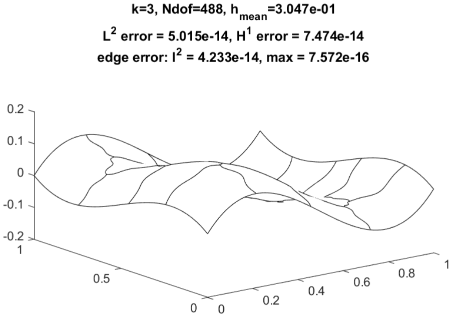
$\gamma = \sin$



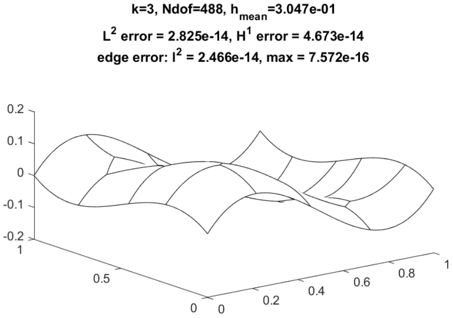
γ straight

The curved edges degenerate to straight edges.

Numerical experiments - patch test $k = 3$



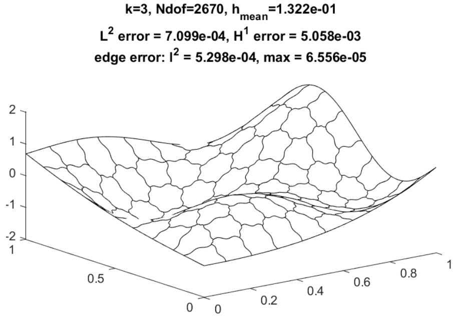
$\gamma = \sin$



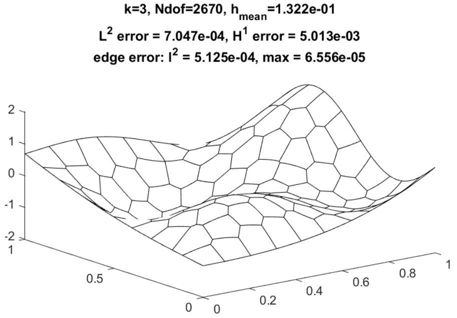
γ straight

The solution doesn't change when the curved edges degenerate to straight edges.

Numerical experiments - Poisson equation $k = 3$



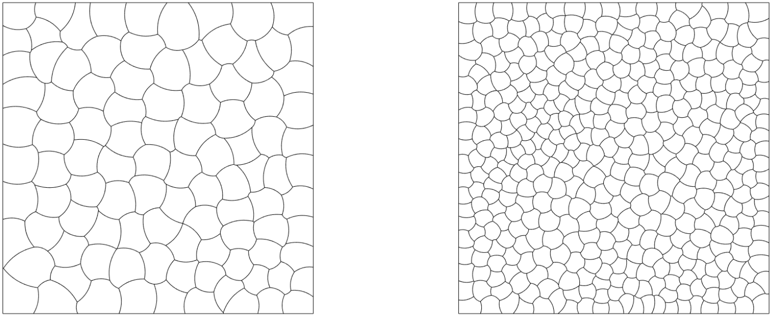
$\gamma = \sin$



γ straight

The solution doesn't change when the curved edges degenerate to straight edges.

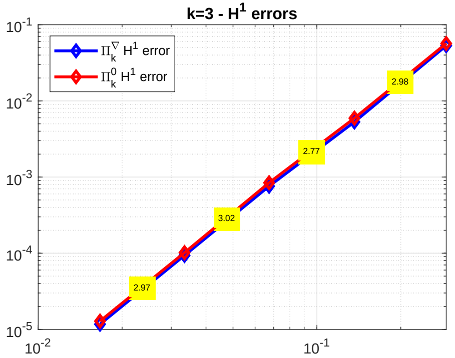
Numerical experiments - convergence

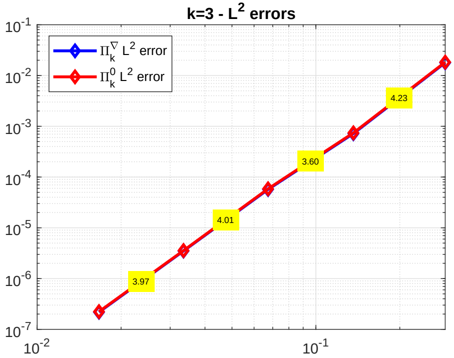


$\gamma =$ quadratic

Here the curved edges γ are quadratic polynomials and $k = 3$. Hence the dimension of $V_k(\gamma)$ is not equal to the dimension of polynomials in two variables, and there are phony dofs that are useless. The methods works in any case, without need to check!

Numerical experiments - convergence $k = 3$





$\gamma =$ quadratic

In all the experiments I have overkilled the integration over polygons with curved edges. The minimum degree of precision of the quadrature formulas to be used in order to preserve optimal convergence is unknown.

Conclusion and perspectives

- There is a lot of work to do!
- The classical VEM machinery continue to work (enhancing for L^2 projectors, variable coefficients, ...).
- The methods presented could be of interest also for triangular and quadrilateral FEM meshes with curved edges.
- Integration over curved polygons and polyhedra must be investigated.
- Extension to three dimensions?

Conclusion and perspectives

- There is a lot of work to do!
- The classical VEM machinery continue to work (enhancing for L^2 projectors, variable coefficients,. . .).
- The methods presented could be of interest also for triangular and quadrilateral FEM meshes with curved edges.
- Integration over curved polygons and polyhedra must be investigated.
- Extension to three dimensions?

Thanks for your attention!