GETFEM++

Une Librairie Éléments finis générique en c++

Y. Renard*, J. Pommier*

*INSA de Toulouse

Motivations

- Outils de base pour faire des codes éléments finis (sauf mailleur).
- Gestion de maillage générique.
- Méthodes d’assemblage génériques.
- Outils divers pour méthodes avancées (méthodes mixte, éléments de joints, Xfem ...)
- interface MATLAB

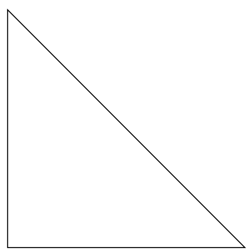
Le cœur de GETFEM

- Description des éléments géometriques
- Description des transformations géométriques
- Description d'un élément fini
- Calculs élémentaires

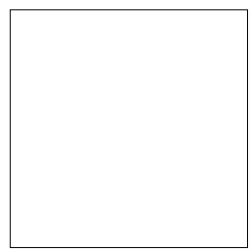
Description des éléments géométriques

Simplexes de références et leur produits :

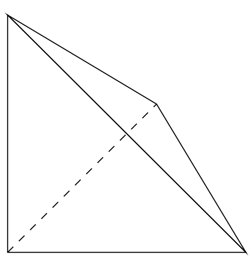
Segment



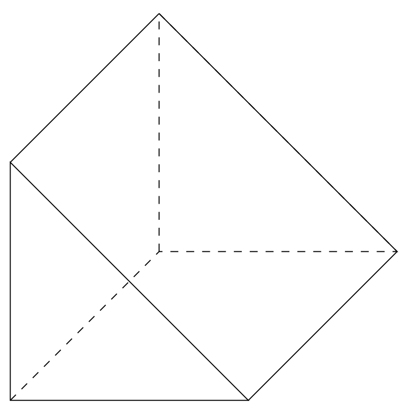
triangle



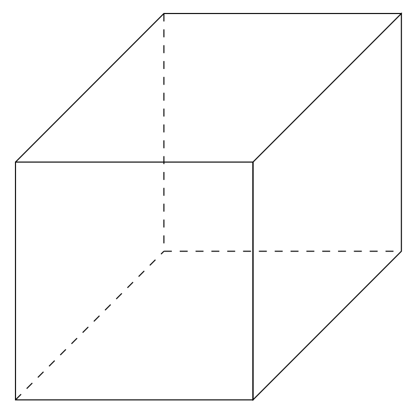
quadrilateron



tetrahedron



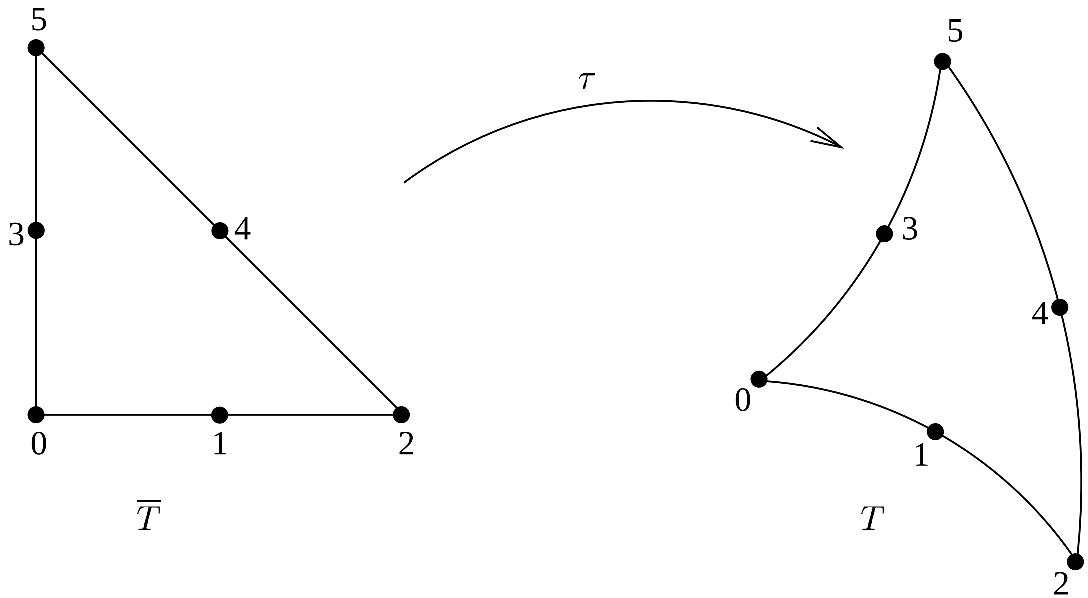
prism



hexahedron

...

Description des transformations géométriques



Définies comme l'ensemble :

- d'un élément de référence
- d'un certain nombre de nœuds géométriques
- de polynômes d'ordres arbitraires

$$\tau : \overline{T} \subset \mathbb{R}^P \longrightarrow T \subset \mathbb{R}^N$$

$$\tau(x) = \sum_{i=0}^{ng-1} \mathcal{N}_i(\overline{x}) g^i$$

Description d'un élément fini

Défini comme l'ensemble

- d'un élément de référence
- d'un ensemble de fonctions de base
- d'un ensemble de nœuds correspondants (+ description des ddl)

$$\bar{\varphi}^i : \bar{T} \subset I\!R^P \longrightarrow I\!R^Q$$

$$\tilde{\varphi}^i(x) = \bar{\varphi}^i(\bar{x}) = \bar{\varphi}^i(\tau^{-1}(x))$$

Transformation linéaire supplémentaire :

$$\varphi^i(x) = \sum_{i=0}^{nd-1} \tilde{M}_{ij}\tilde{\varphi}^j(x)$$

Éléments finis prédéfinis

`"FEM_PK(n,k)"`	Méthode P_K classique en dimension `n` et degré `k`.
`"FEM_QK(n,k)"`	Méthode Q_K classique en dimension `n` et degré `k`.
`"FEM_PRODUCT(a,b)"`	produit tensoriel de deux éléments finis polynomiaux `a` et `b`.
`"FEM_PK_DISCONTINUOUS(n,k)"`	Méthode P_K discontinue sur des simplexes.
`"FEM_PK_WITH_CUBIC_BUBBLE(n,k)"`	Élément P_K avec une bulle interne additionnelle ($k \leq n$)

Calculs élémentaires

Exemple de calcul élémentaire :

$$t(i_0, i_1, ..., i_7) = \int_T \varphi_{i_1}^{i_0} \, \partial_{i_4} \varphi_{i_3}^{i_2} \, \partial^2_{i_7/P, i_7} \text{ mod }_P \varphi_{i_6}^{i_5} dx,$$

```
getfem::pmat_elem_type pet = getfem::mat_elem_product(
   getfem::mat_elem_product(getfem::mat_elem_base(pfi1),
    getfem::mat_elem_grad(pfi2)),
    getfem::mat_elem_hess(pfi3));
```

où `pfi1`, `pfi2`, `pfi3` décrivent des méthodes éléments finis. La méthode d'intégration et la transformation géométrique sont spécifiées ensuite :

```
getfem::pmat_elem_computation pmec = mat_elem(pet,
   pintegration_method pi, pgeometric_trans pg)
```

Le calcul effectif :

```
pmec->gen_compute(base_tensor &t, const CONT &a)
pmec->gen_compute_on_face(base_tensor &t, const CONT &a, f)
```

Description d’un maillage

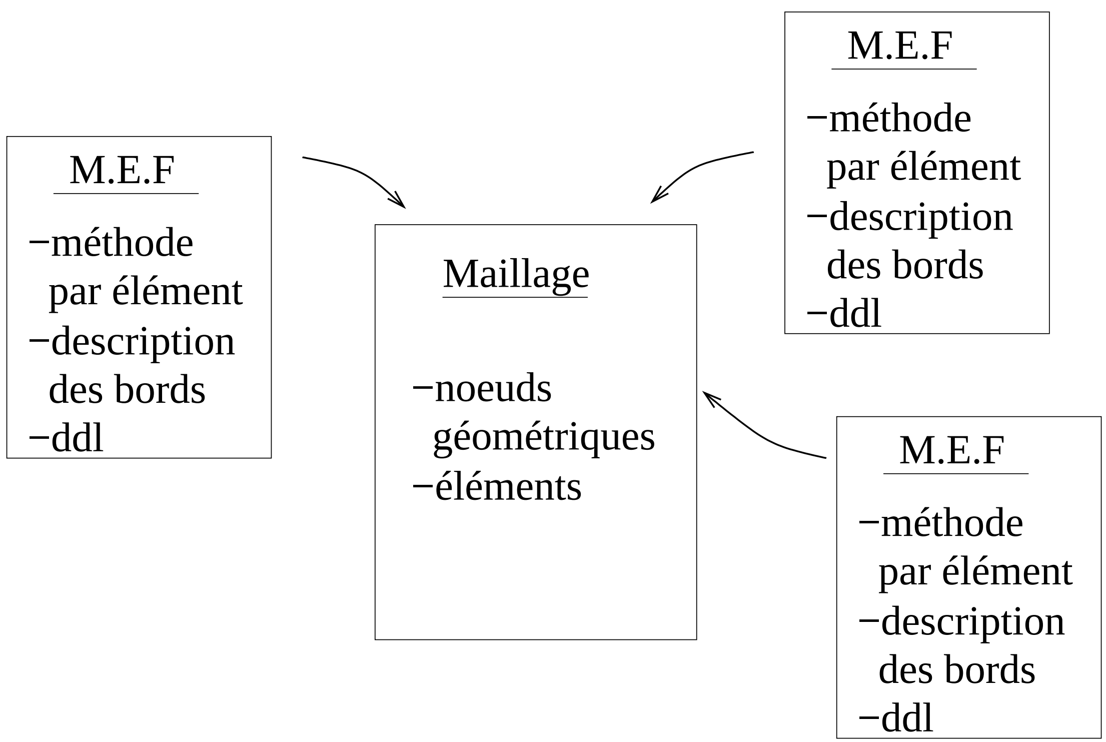
Une structure existe qui stocke des maillages quelconques (toutes dimensions, mixage possible) avec des transformations géométriques arbitraires. Elle permet de faire un certain nombre de choses basiques :

- ajout/suppression d’éléments.
- liste des éléments attachés à un nœud géométrique et réciproquement.
- recherche d’un élément, éléments voisins.
- faces d’un éléments, normales aux faces (en tenant compte de la transformation géométrique)
- recherche d’un élément sous un point quelconque (et inversion de la transformation géométrique).

Méthode éléments finis sur un maillage

A chaque maillage on peut associer un nombre indeterminé de structures définissant une méthode éléments finis sur ce maillage avec :

- spécification possible élément par élément de la M.E.F.
- spécification possible élément par élément de la méthode d'intégration
- repérage des faces pour les conditions aux bords.



$$\int_T a\nabla u.\nabla v dx$$

Méthodes d’assemblage génériques

Des méthodes d’assemblages génériques (i.e. qui fonctionnent quelque soit l’élément fini séléctionné) :

- Élasticité linéarisée
- Problème de Poisson
- Problème de Stokes
- Condition de Dirichlet
- Condition de Neumann et Fourier-Robin (voir Dirichlet-Neumann mélé)
- Termes sources
- Termes génériques

Interpolation

Il est possible d'interpoler toute fonction définie sur une élément fini donné sur une autre élément fini pourvu qu'il soit de Lagrange.

- maillages identiques ou différents,
- dimension quelconque, concordantes ou non,
- degré quelconque, transformation géométrique quelconque.

Intérêt :

- Interpoler une solution définie sur un élément quelconque sur un élément standard pour la visualisation.
- Faire des coupes d'une solution volumique.
- Comparer deux solutions en norme L^2ou H^1 définies sur deux maillages différents.

Interpolation d'éléments finis

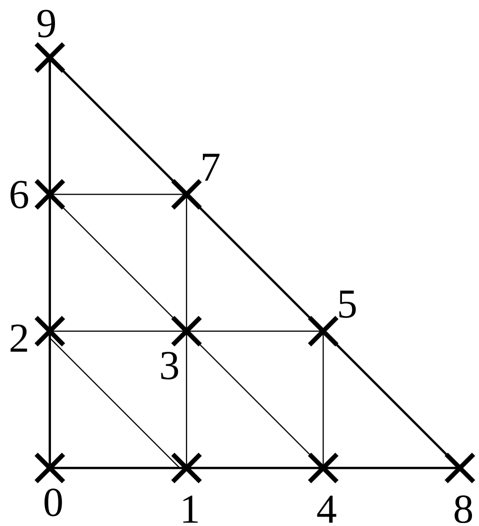
Il est possible d'interpoler un élément fini défini sur un maillage sur un autre maillage de même dimension. Il est alors utilisable dans toutes les fonctions habituelles : calcul élémentaires, procédures d'assemblage ...
Intérêt :

- On peut définir des méthodes mixtes où on maille différement les différentes inconnues.
- On peut faire les calcul de base pour les éléments de joints.
- On peut faire des calculs d'intégrales avec des fonctions définies globalement.

Il faut faire attention à mettre des méthodes d'intégration suffisement raffinées (plutôt que des méthodes d'ordre élevé).

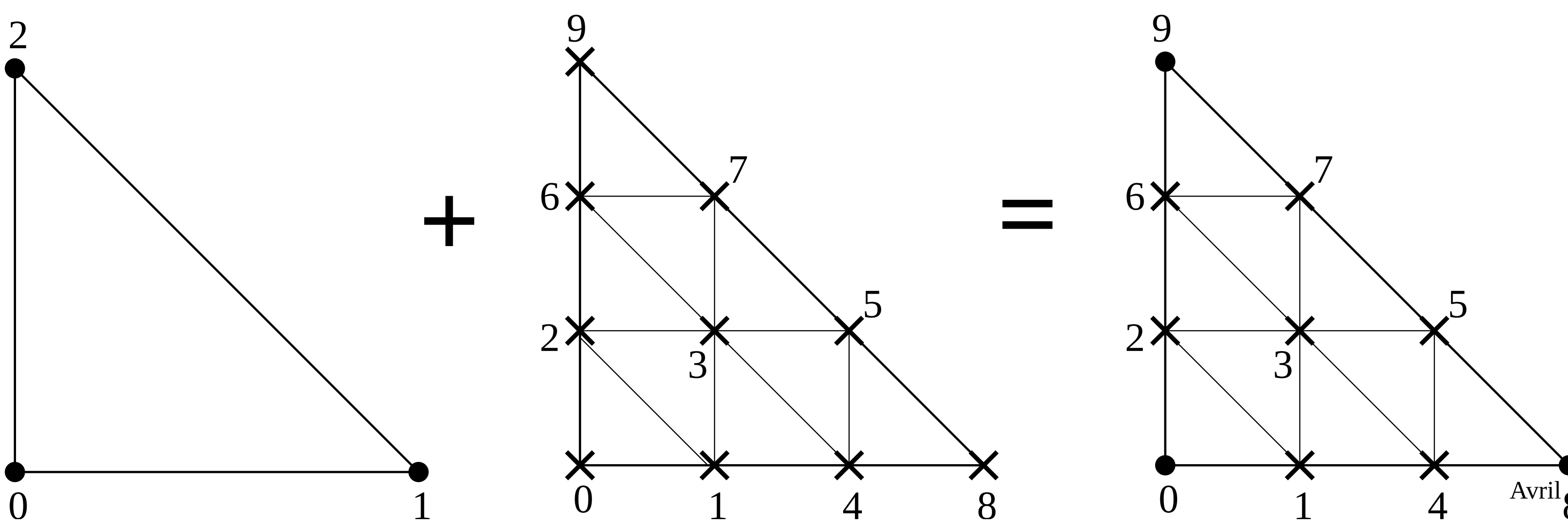
Éléments finis hierarchiques (et composites)

Pas de limitations pour les fonctions de base élément fini. Deux types sont pré-définis : polynômes et polynômes par morceaux.



Un maillage d'un élément de référence avec une M.E.F., peut être déclaré comme étant un nouvel élément global.

A partir de deux éléments on peut définir un élément hierarchique

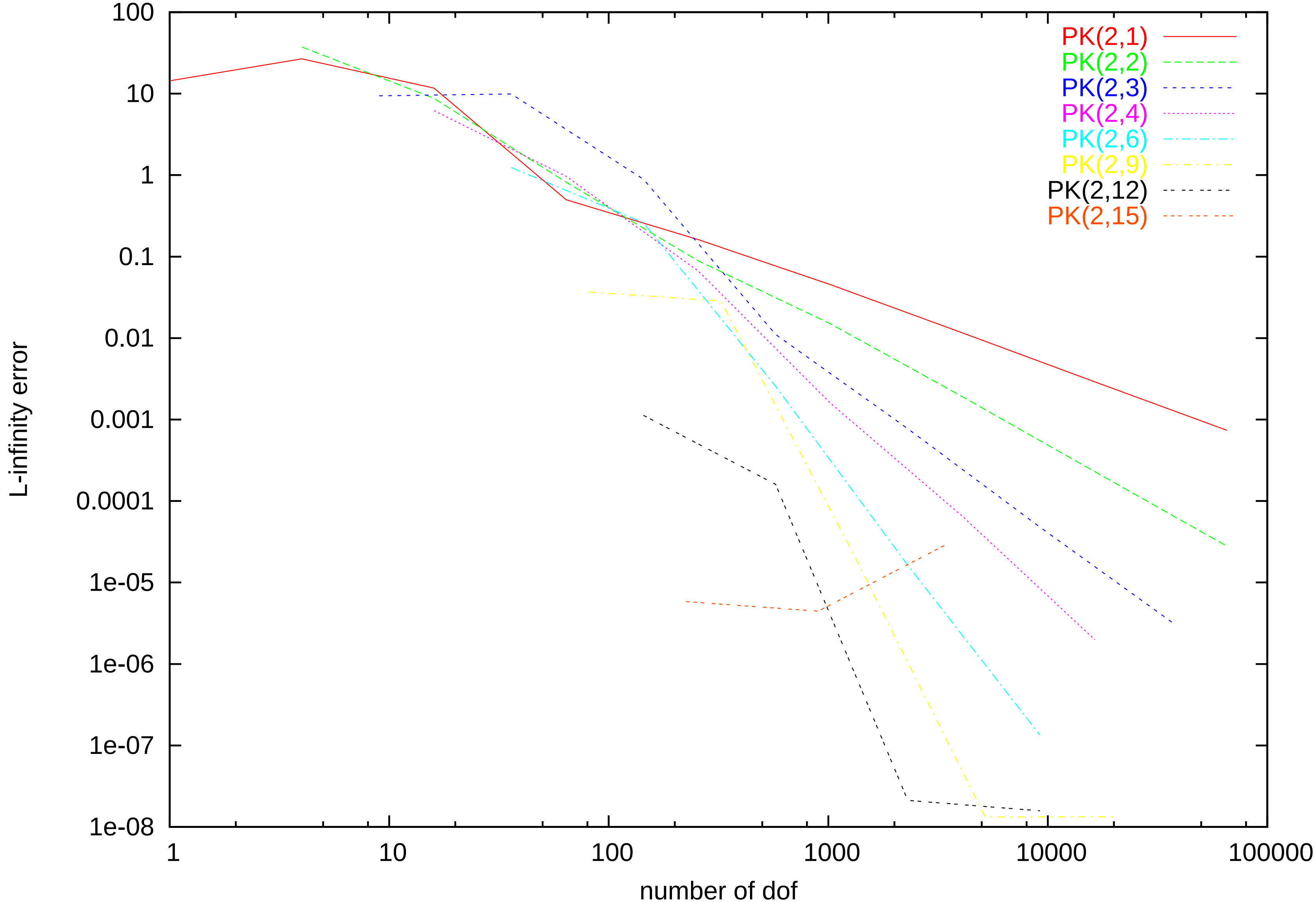


Degré élevé

Difficultés du degré élevé :

- Instabilité de l'intégration exacte
- Difficulté de calculer des méthodes d'intégrations d'ordre élevé.
- Perte de conditionnement de la matrice de rigidité

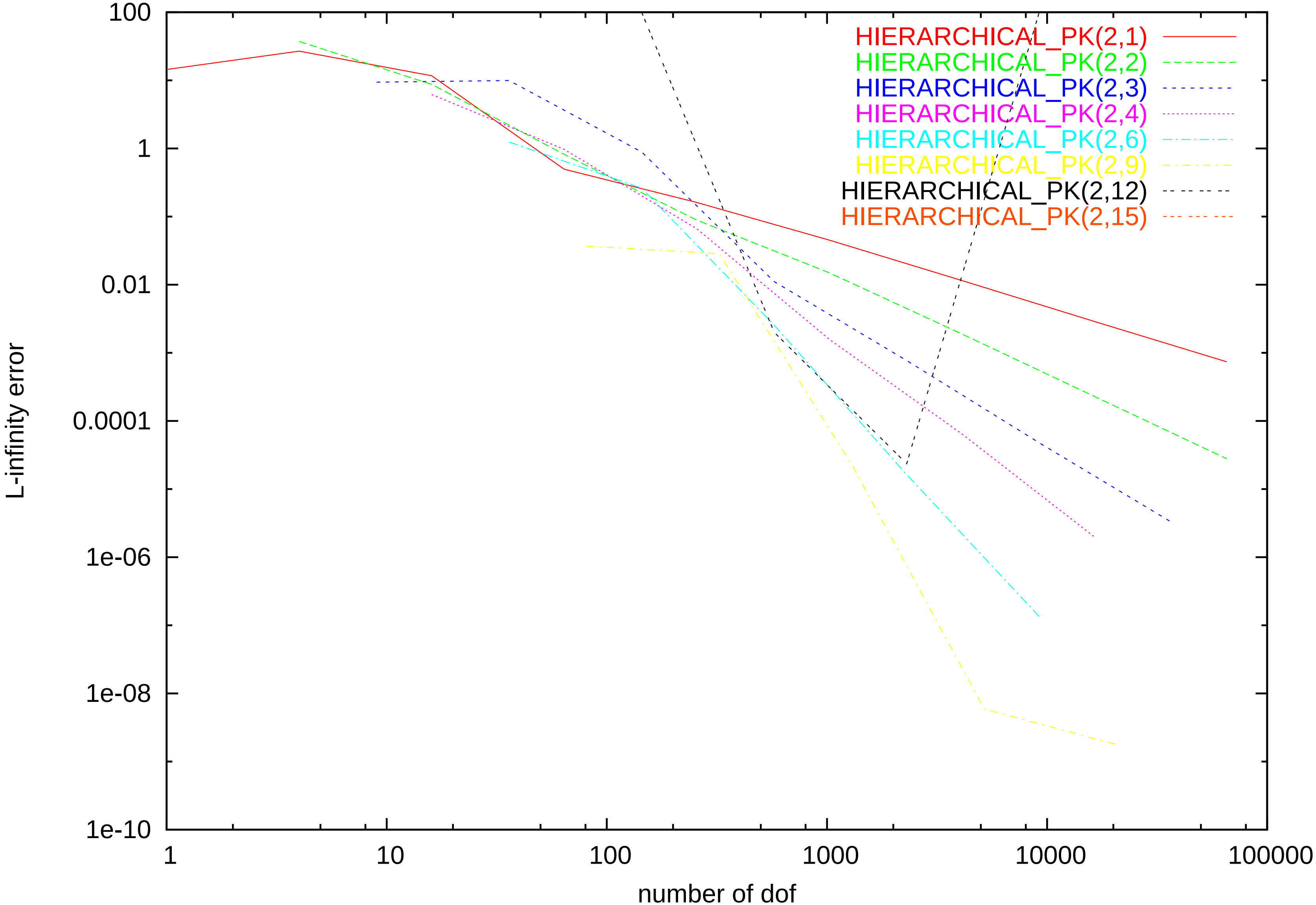
Degré élevé



Éléments P_K standards, problème de Poisson en dimension 2 sur un carré.

Solution exacte : $sin(10x - 10y)$.

Degré élevé



Éléments P_K hiérarchiques, problème de Poisson en dimension 2 sur un carré.

Solution exacte : $sin(10x-10y)$.

Module matriciel

- Au départ c'est une interface. Une classe de vecteurs ou de matrices pré-existante peut être utilisée après définition en particulier d'une classe "traits" définissant ses propriétés.
- Des classes de base fournies : vecteurs pleins et creux, matrices pleines, creuses et ligne de ciel.
- calculs de base : copie, addition, multiplications, accès aux sous-matrices.
- Quelques méthodes de résolution itératives : C.G., GMRES, BICGSTAB ...

Module matriciel

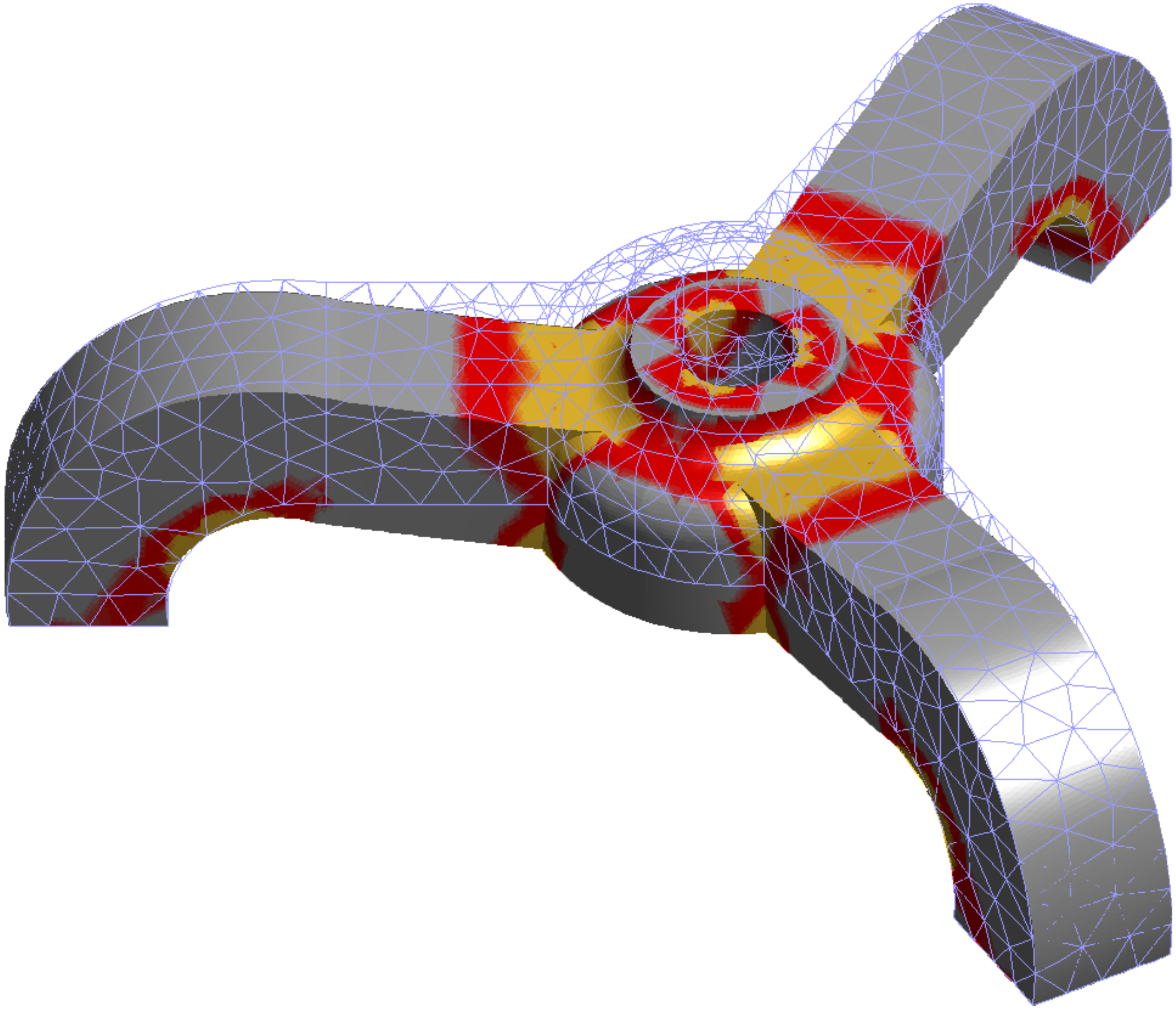
Exemple de l'interfaçage de `std::vector<T>`

```
template <class T, class alloc> struct linalg_traits<std::vector<T,alloc> > {
  typedef std::vector<T, alloc> this_type;
  typedef linalg_false is_reference;
  typedef abstract_vector linalg_type;
  typedef T value_type;
  typedef T& reference;
  typedef typename this_type::iterator iterator;
  typedef typename this_type::const_iterator const_iterator;
  typedef abstract_plain storage_type;
  typedef plain_access<iterator,const_iterator> access_type;
  typedef plain_clear<iterator> clear_type;
  static size_type size(const this_type &v) { return v.size(); }
  static iterator begin(this_type &v) { return v.begin(); }
  static const_iterator begin(const this_type &v) { return v.begin(); }
  static iterator end(this_type &v) { return v.end(); }
  static const_iterator end(const this_type &v) { return v.end(); }
  static const void* origin(const this_type &v) { return &v; }
  static void do_clear(this_type &v) { clear_type()(origin(v), begin(v), end(v));
}
```

Interface MATLAB

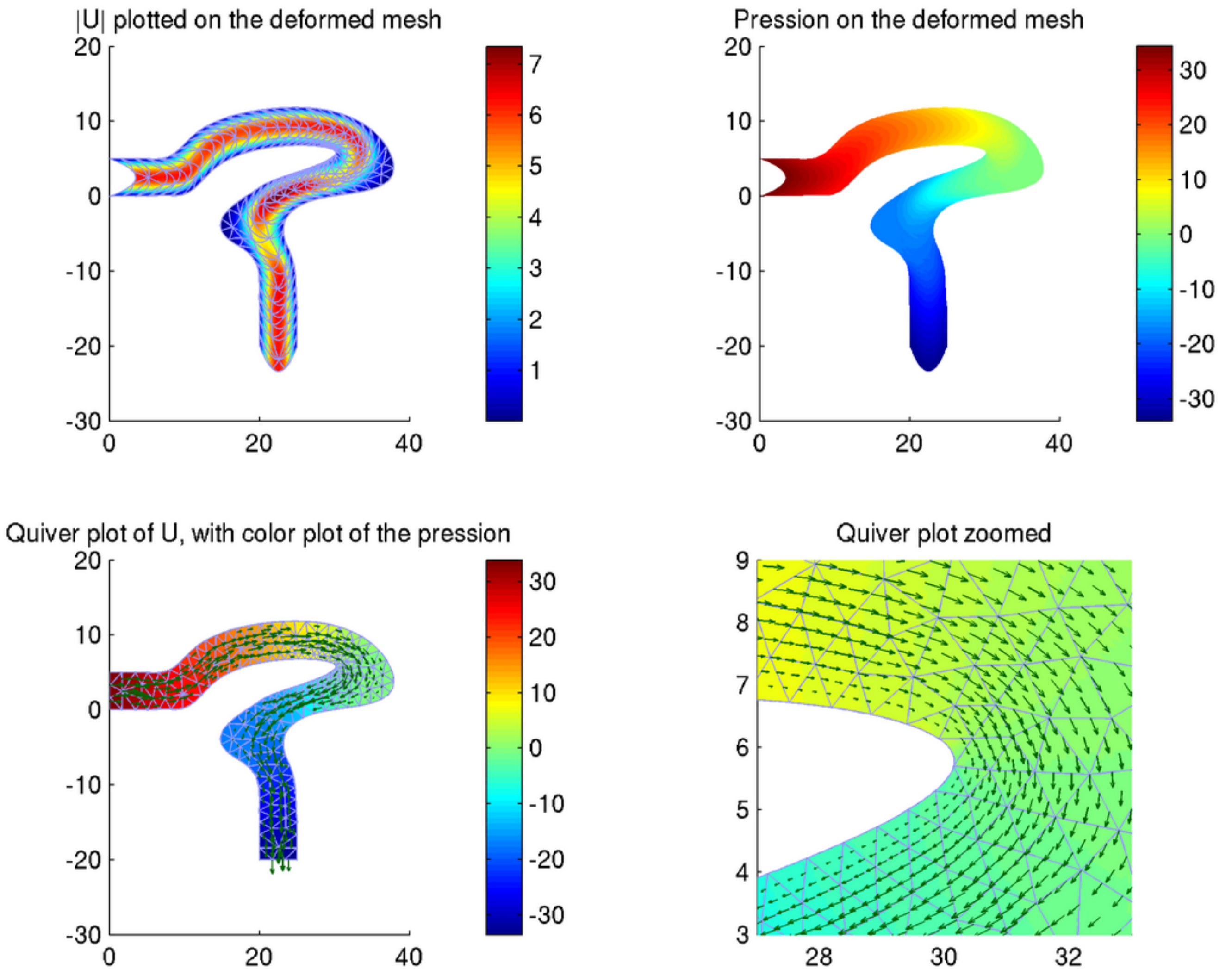
- interfaçage de (presque) toutes les fonctionnalités de GETFEM, sans se préocuper des structures C++ sous-jacentes.
- documentation extensive.
- permet de récuperer un problème complet de la pdetool de Matlab et d'y mettre des éléments finis arbitraires.
- bon support pour des T.P. ou des projets.

Exemple 1 : Le tripode



Maillage issu de GiD. Calculs en P_2 isoparametrique. Contrainte de Von Mises.

Exemple 2 : Problème de Stockes

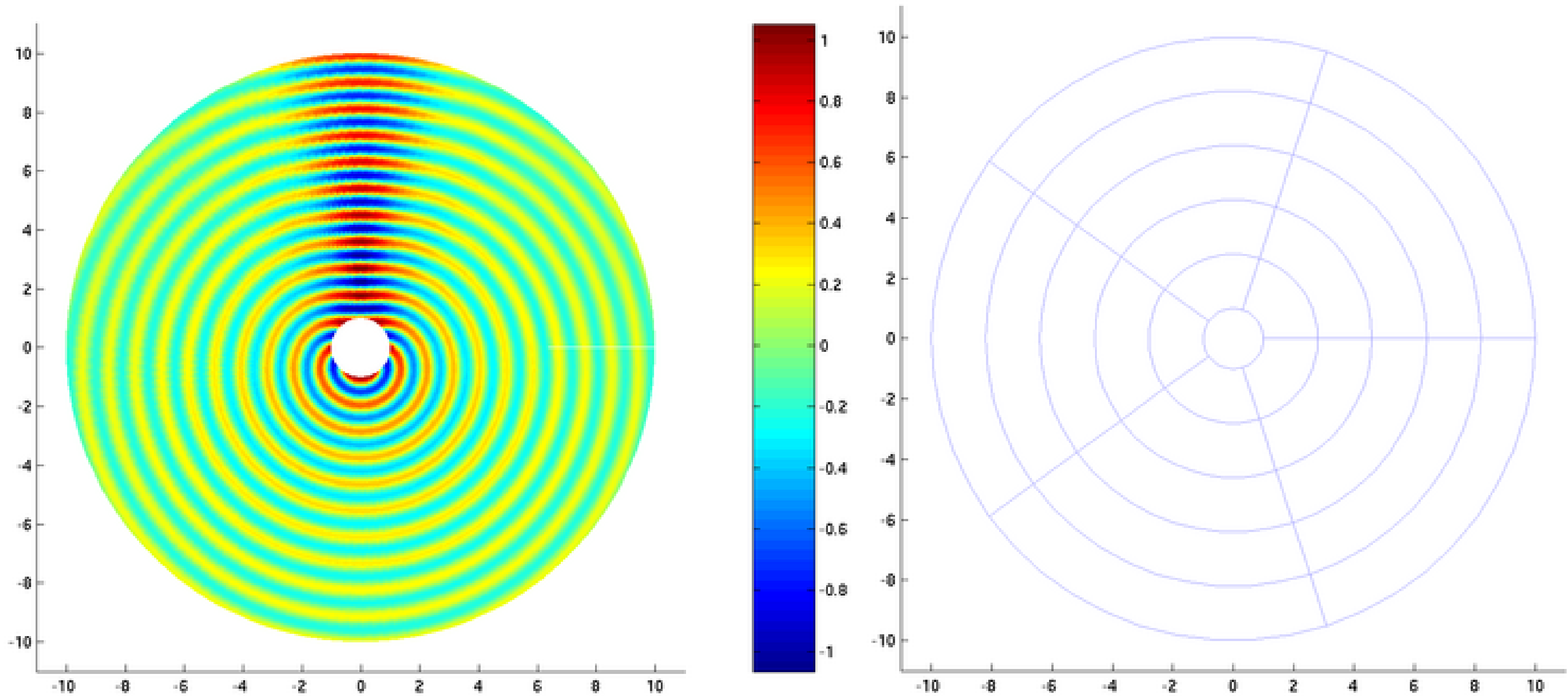


Élément mixte P_2+/P_1 avec transformation géométrique d'ordre 2.

code Matlab du problème de Stockes

```
pde.type = ’stokes’; pde.viscos=1.0; pde.F = 0,0;
pde.bound1.R  = ’-y.*(y-5)’,0; pde.bound2.R  = 0,’+(x-20).*(x-25)’;
pde.bound1.type = ’Dirichlet’; pde.bound2.type = ’Dirichlet’;
m=gf_mesh(’import’,’GiD’,’tube_2D_spline.GiD.msh’);
pde.mf_u=gf_mesh_fem(m,2); mfulag=gf_mesh_fem(m,2);
pde.mf_p=gf_mesh_fem(m,1); pde.mf_d=gf_mesh_fem(m,1);
gf_mesh_fem_set(pde.mf_u,’fem’,gf_fem(’FEM_PK_WITH_CUBIC_BUBBLE(2,2)’),
                gf_integ(’IM_TRIANGLE(5)’));
gf_mesh_fem_set(pde.mf_d,’fem’,gf_fem(’FEM_PK(2,2)’), gf_integ(’IM_TRIANGLE(5)’));
gf_mesh_fem_set(pde.mf_p,’fem’,gf_fem(’FEM_PK_DISCONTINUOUS(2,1)’),
                gf_integ(’IM_TRIANGLE(5)’));
gf_mesh_fem_set(mfulag,’fem’,gf_fem(’FEM_PK(2,3)’), gf_integ(’IM_TRIANGLE(5)’));
all_faces = gf_mesh_get(m, ’outer faces’, gf_mesh_get(m, ’cvid’));
P=gf_mesh_get(m,’pts’); INpid=find(abs(P(1,:)) < 1e-4); OUTpid=find(abs(P(2,:)+20) <
INfaces=gf_mesh_get(m, ’faces from pid’, INpid);
OUTfaces=gf_mesh_get(m, ’faces from pid’, OUTpid);
for mf=[pde.mf_u pde.mf_p pde.mf_d],
  gf_mesh_fem_set(mf, ’boundary’, 1, INfaces);
  gf_mesh_fem_set(mf, ’boundary’, 2, OUTfaces);
end;
[U,P]=gf_solve(pde); Ul=gf_compute(pde.mf_u,U,’interpolate on’,mfulag);
gf_plot(mfulag,Ul,’norm’,’on’,’deformation’,Ul,’deformation_scale’,’10%’, ’deformed_
```

Exemple 3 : Helmholtz



Élément P$_{10}$ sur transformation géométrique d'ordre 6.

Résultat : deux longueurs d'onde par élément.

Distribution

GETFEM++ est distribué sous licence L.G.P.L.
Il est disponible à l'adresse :

`http://www.gmm.insa-tlse.fr/getfem`

avec les documentations et exemples commentés en MATLAB.