The NURBS and GeoPDEs packages
Octave software for research on IGA

Rafael Vázquez

IMATI 'Enrico Magenes', Pavia
Consiglio Nazionale delle Ricerche

Motivation

In 2008/2009 in Pavia we started to work in isogeometric analysis within the GeoPDEs project. Software development was one of the objectives.

Starting point: different codes, different problems, different developers.

First goal: a uniform implementation of the different codes.

Second goal: it should be clear and easy to use, for didactic purposes, and for new researchers coming into the research group.

The result were two Octave packages: the **NURBS** package, for geometry construction and manipulation, and **GeoPDEs**, for isogeometric methods.

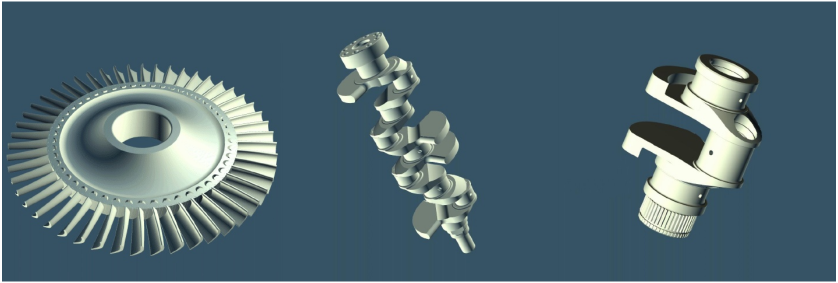
1. **The NURBS package**: B-splines and NURBS
    - B-splines and NURBS: mathematical definitions
    - Functions and examples
2. **The GeoPDEs package**: isogeometric analysis
    - Isogeometric analysis: definition
    - The development of GeoPDEs
    - Some examples

1. **The NURBS package**: B-splines and NURBS
    - B-splines and NURBS: mathematical definitions
    - Functions and examples

2. **The GeoPDEs package**: isogeometric analysis
    - Isogeometric analysis: definition
    - The development of GeoPDEs
    - Some examples

Non Uniform Rational B-Splines (NURBS)

NURBS (non-uniform rational B-splines) are probably the most commonly used CAD technology for engineering design.



NURBS are a generalization of **B-splines**.

B-splines: definition

Given an ordered knot vector $\xi_1 \leq \ldots \leq \xi_{n+p+1}$,

define the n B-splines of degree p by the recursion formula

$$N_{i,0}(\zeta) = \begin{cases} 1 & \text{if } \xi_i \leq \zeta \leq \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(\zeta) = \frac{\zeta - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\zeta) + \frac{\xi_{i+p+1} - \zeta}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\zeta)$$

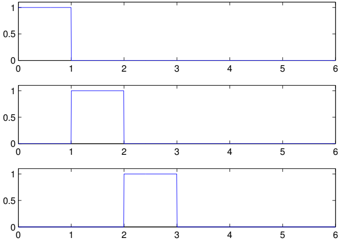
B-splines: definition

Given an ordered knot vector $\xi_1 \leq \ldots \leq \xi_{n+p+1}$,

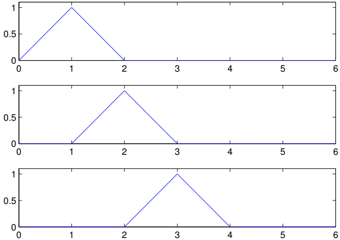
define the n B-splines of degree p by the recursion formula

$$N_{i,0}(\zeta) = \begin{cases} 1 & \text{if } \xi_i \leq \zeta \leq \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(\zeta) = \frac{\zeta - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\zeta) + \frac{\xi_{i+p+1} - \zeta}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\zeta)$$



Degree 0



Degree 1

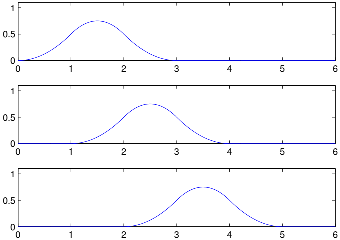
B-splines: definition

Given an ordered knot vector $\xi_1 \leq \ldots \leq \xi_{n+p+1}$,

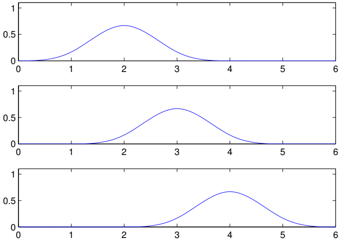
define the n B-splines of degree p by the recursion formula

$$N_{i,0}(\zeta) = \begin{cases} 1 & \text{if } \xi_i \leq \zeta \leq \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(\zeta) = \frac{\zeta - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\zeta) + \frac{\xi_{i+p+1} - \zeta}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\zeta)$$

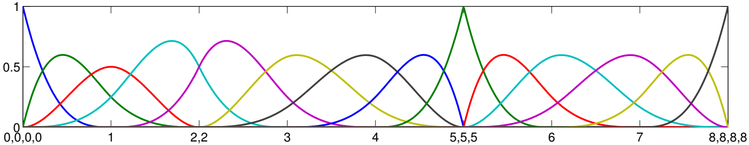


Degree 2



Degree 3

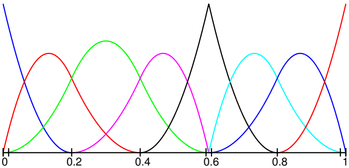
B-splines: the basis functions

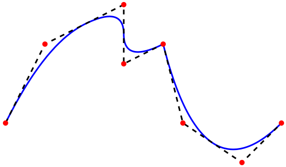


B-spline **basis functions** have the following properties:

- They are non-negative and form a partition of unity.
- Locally linearly independent on each knot span (ξ_i, ξ_{i+1})
- The function $N_{i,p}$ is supported in the interval $[\xi_i, \xi_{i+p+1}]$.
- Piecewise polynomials of degree p, and regularity at most $p-1$.
- The **regularity** at ξ_i is controlled by the **knot multiplicity**.

B-spline curves: definition





A B-spline curve in $\mathbb{R}^d$ is defined as a linear combination of B-splines:

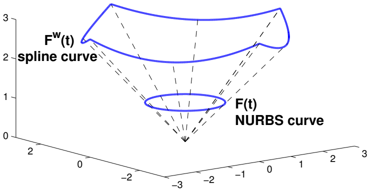
$$\mathbf{F}(\zeta) = \sum_{i=1}^{n} \mathbf{C}_i N_{i,p}(\zeta)$$

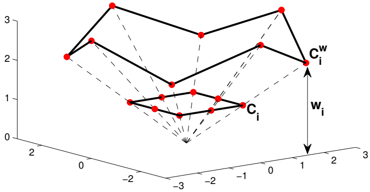
To define the parametrization $\mathbf{F}$ we only need:

- The basis functions $N_{i,p}$, given by the knot vector.
- The control points $\mathbf{C}_i \in \mathbb{R}^d$.

NURBS curves: definition

NURBS are rational B-splines, used to represent conic sections.





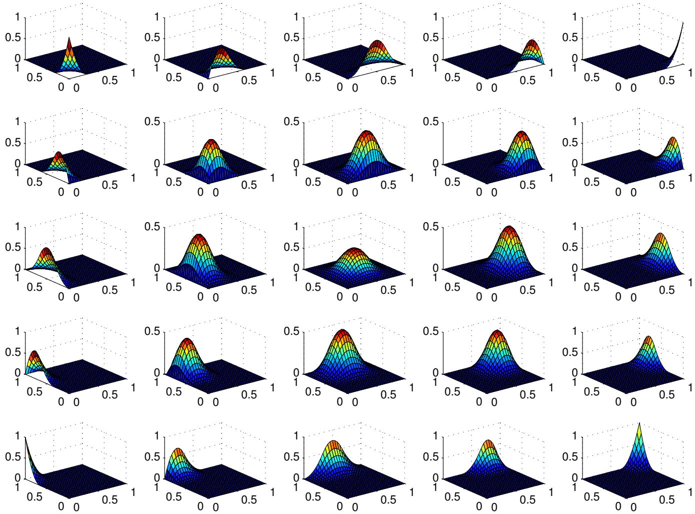
NURBS in $\mathbb{R}^d$ are projections of B-splines in $\mathbb{R}^{d+1}$.

In practice, a weight w_i is associated to each B-spline function, to obtain the **NURBS basis functions** and the control points.

The NURBS curve is determined by: degree, knot vector, control points and weights.

Tensor product surfaces: B-splines

B-splines and NURBS surfaces are defined by **tensor product**.

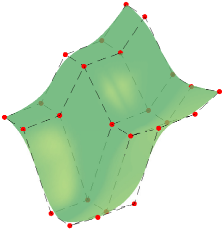


Tensor product surfaces: B-splines

B-splines and NURBS surfaces are defined by **tensor product**.

A control point $\mathbf{C_i} \in \mathbb{R}^d$ is associated to each basis function to define **F**:

$$\mathbf{F}(\zeta) = \sum_{\mathbf{i}} \mathbf{C_i} N_{\mathbf{i},\mathbf{p}}(\zeta)$$



The control points define the **control net**.

With a similar idea, one can define B-spline and NURBS volumes.

The NURBS package

The package is intended to work with NURBS geometries.

- Based on the **NURBS toolbox**, developed by M. Spink in 2000.
- From 2009, extended and maintained by Carlo de Falco and myself.
- Supports curves, surfaces and (simple) volumes.
- Geometry manipulation: rotation, extrusion, revolution...
- It also serves for basis function evaluation (B-splines and NURBS).
- Most of the algorithms come from **The NURBS book**.

The NURBS package

The package is intended to work with NURBS geometries.

- Based on the **NURBS toolbox**, developed by M. Spink in 2000.
- From 2009, extended and maintained by Carlo de Falco and myself.
- Supports curves, surfaces and (simple) volumes.
- Geometry manipulation: rotation, extrusion, revolution...
- It also serves for basis function evaluation (B-splines and NURBS).
- Most of the algorithms come from **The NURBS book**.

Some technical info:

- The package is part of the octave-forge project.
- All the functions are well documented, including several examples.
- Several tests included, but still some are missing (27 of 68 functions).
- The package contains 11 oct-files.

http://octave.sourceforge.net/nurbs/

NURBS curves: definition in the NURBS package

The construction and manipulation of NURBS geometries is based on a **structure** with the following fields:

- **number**: the number of control points.
- **coefs**: control points coordinates (for NURBS also the weights).
- **order**: the degree plus one.
- **knots**: the knot vector in each direction.

NURBS curves: definition in the NURBS package

The construction and manipulation of NURBS geometries is based on a **structure** with the following fields:

- **number**: the number of control points.
- **coefs**: control points coordinates (for NURBS also the weights).
- **order**: the degree plus one.
- **knots**: the knot vector in each direction.

NURBS geometries can be constructed "by hand" with the function **nrbmak**, giving the **knot vector** and the **control points**.

```
crv = nrbmak(coefs, knt);
```

The package contains several functions to define simple geometries: **nrbline**, **nrbrect**, **nrbcirc**, **nrbcylind**.

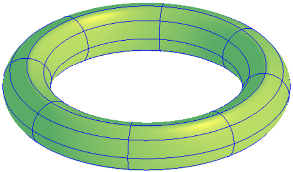
Examples of functions in the package

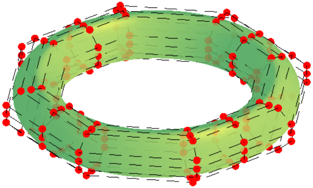
Functions to plot: **nrbkntplot**, **nrbctrlplot**

Revolution and extrusion: **nrbrevolve**, **nrbextrude**

Affine transformations: **nrbtform**

Extract the boundaries of a NURBS: **nrbextract**





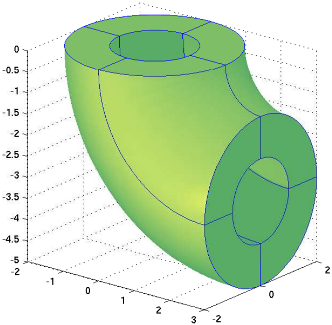
Examples of functions in the package

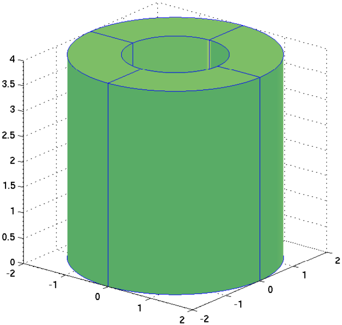
Functions to plot: **nrbkntplot**, **nrbctrlplot**

Revolution and extrusion: **nrbrevolve**, **nrbextrude**

Affine transformations: **nrbtform**

Extract the boundaries of a NURBS: **nrbextract**





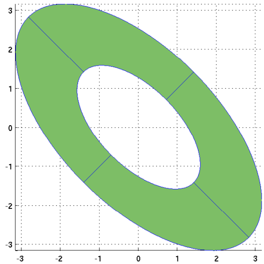
Examples of functions in the package

Functions to plot: **nrbkntplot**, **nrbctrlplot**

Revolution and extrusion: **nrbrevolve**, **nrbextrude**

Affine transformations: **nrbtform**

Extract the boundaries of a NURBS: **nrbextract**



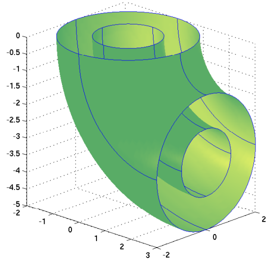
Examples of functions in the package

Functions to plot: **nrbkntplot**, **nrbctrlplot**

Revolution and extrusion: **nrbrevolve**, **nrbextrude**

Affine transformations: **nrbtform**

Extract the boundaries of a NURBS: **nrbextract**



Some doubts and work to do

- Add more tests.
- Compatibility with the **splines package**?
- The package is not intended to be a CAD software. But it would be so nice to move the control points in the figure...

Isogeometric Analysis: overview

Isogeometric Analysis (IGA) is a method for discretization of partial differential equations, similar to the finite element method (FEM).

The idea is to run the simulation directly on a NURBS geometry, approximating the solution also with NURBS (or splines) functions.

It is having a big impact in computational engineering, especially in computational mechanics.

Isogeometric Analysis: overview

Isogeometric Analysis (IGA) is a method for discretization of partial differential equations, similar to the finite element method (FEM).

The idea is to run the simulation directly on a NURBS geometry, approximating the solution also with NURBS (or splines) functions.

It is having a big impact in computational engineering, especially in computational mechanics.

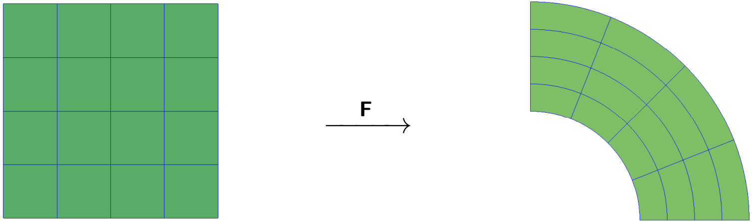
Compared to FEM, the method provides:

- Higher continuity of basis functions.
- Easier mesh generation and refinement.

The concept of IGA

Hughes, Cottrell, Bazilevs (2005)

Reference domain $\widehat{\Omega} = (0,1)^d$, and **physical domain** $\Omega = \mathbf{F}(\widehat{\Omega})$.

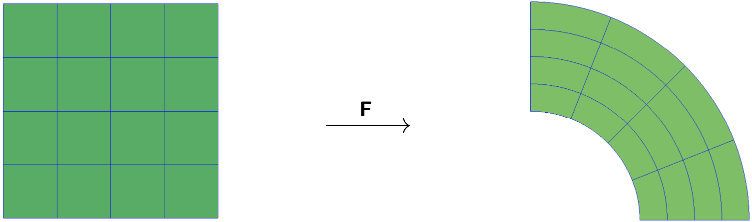


We want to solve numerically a certain problem in the NURBS domain Ω.

The concept of IGA

Hughes, Cottrell, Bazilevs (2005)

Reference domain $\widehat{\Omega} = (0,1)^d$, and **physical domain** $\Omega = \mathbf{F}(\widehat{\Omega})$.



We want to solve numerically a certain problem in the NURBS domain Ω.

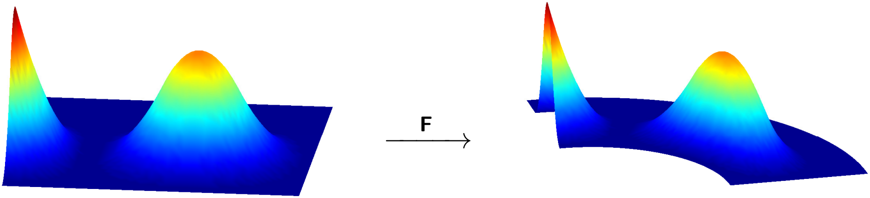
In $\widehat{\Omega}$ we take the space of NURBS functions, $\widehat{V}_h := \mathrm{span}\{R_{i,p}\}$.

Isoparametric paradigm: in Ω we define $V_h := \mathrm{span}\{v_i := R_{i,p} \circ \mathbf{F}^{-1}\}$.

The concept of IGA

Hughes, Cottrell, Bazilevs (2005)

Reference domain $\widehat{\Omega} = (0,1)^d$, and **physical domain** $\Omega = \mathbf{F}(\widehat{\Omega})$.



We want to solve numerically a certain problem in the NURBS domain Ω.

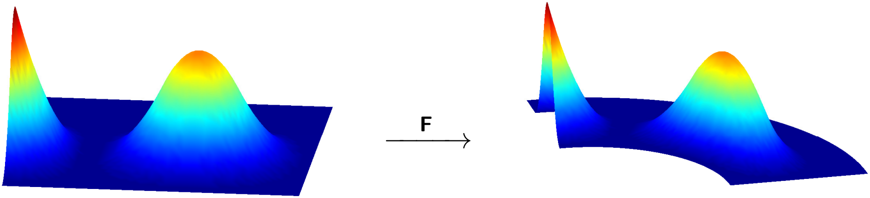
In $\widehat{\Omega}$ we take the space of NURBS functions, $\widehat{V}_h := \mathrm{span}\{R_{i,p}\}$.

Isoparametric paradigm: in Ω we define $V_h := \mathrm{span}\{v_i := R_{i,p} \circ \mathbf{F}^{-1}\}$.

The concept of IGA

Hughes, Cottrell, Bazilevs (2005)

Reference domain $\widehat{\Omega} = (0,1)^d$, and **physical domain** $\Omega = \mathbf{F}(\widehat{\Omega})$.



We want to solve numerically a certain problem in the NURBS domain Ω.

In $\widehat{\Omega}$ we take the space of NURBS functions, $\widehat{V}_h := \mathrm{span}\{R_{i,p}\}$.

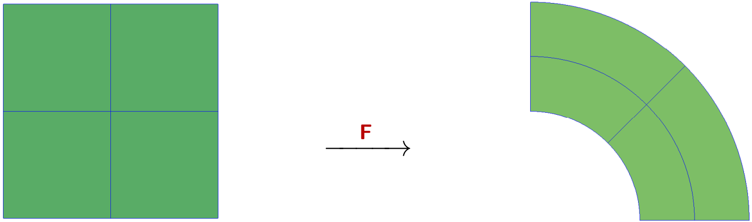
Isoparametric paradigm: in Ω we define $V_h := \mathrm{span}\{v_i := R_{i,p} \circ \mathbf{F}^{-1}\}$.

Once the space is defined, everything is similar to FEM: numerical integration, assembly of the global matrices, solution of the linear system...

Refinement in IGA

The coarsest mesh is given by the parametrization **F** of the geometry.

Coarsest mesh: geometry description

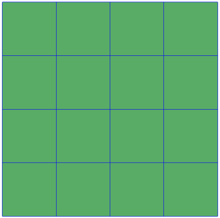


$\{R_i\}_{i=1}^{N_0}$ $\{v_i = R_i \circ \mathbf{F}^{-1}\}_{i=1}^{N_0}$

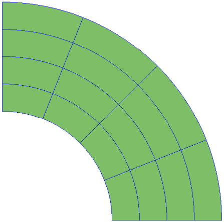
Refinement in IGA

The coarsest mesh is given by the parametrization **F** of the geometry.

First refinement step



$\xrightarrow{\mathbf{F}}$



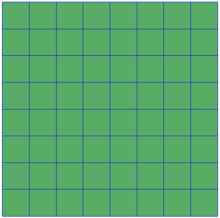
$\{R_i\}_{i=1}^{N_1}$ $\{v_i = R_i \circ \mathbf{F}^{-1}\}_{i=1}^{N_1}$

A better approximation is obtained by refining the space, either taking a finer mesh (*h*-refinement) or raising the degree (*p*-refinement).

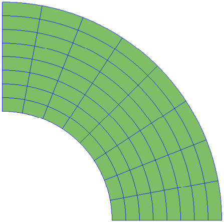
Refinement in IGA

The coarsest mesh is given by the parametrization **F** of the geometry.

Second refinement step



$\xrightarrow{\mathbf{F}}$



$\{R_i\}_{i=1}^{N_2}$ $\{v_i = R_i \circ \mathbf{F}^{-1}\}_{i=1}^{N_2}$

A better approximation is obtained by refining the space, either taking a finer mesh (*h*-refinement) or raising the degree (*p*-refinement).

The geometry Ω and the parametrization **F** remain **fixed** after refinement.

Since the mesh is structured, refinement is very easy.

GeoPDEs: development

GeoPDEs was originally developed in Pavia, by Carlo de Falco, Alessandro Reali, and myself.

- 2009/2010: join forces to obtain a single/uniform code.
- 2010: first public release of GeoPDEs.
- 2011: presentation at the first IGA conference.
- 2012: version 2.0, efficient use of tensor-product features.
- 2015: version 2.1 (to be released), dimension independent.
- 2016 (expected): adaptivity with hierarchical splines.

GeoPDEs: development

GeoPDEs was originally developed in Pavia, by Carlo de Falco, Alessandro Reali, and myself.

- 2009/2010: join forces to obtain a single/uniform code.
- 2010: first public release of GeoPDEs.
- 2011: presentation at the first IGA conference.
- 2012: version 2.0, efficient use of tensor-product features.
- 2015: version 2.1 (to be released), dimension independent.
- 2016 (expected): adaptivity with hierarchical splines.

Other developers and contributors

Andrea Bressan, Elena Bulgarello, Durkbin Cho, Jacopo Corno, Adriano Côrtes, Luca Dedè, Sara Frizziero, Eduardo M. Garau, Timo Lähivaara, Marco Pingaro, Anna Tagliabue.

GeoPDEs: description of the software

GeoPDEs consists of a set of interrelated **packages** for different problems:

- **base**: the main package, with examples for Laplace problem.
- **elasticity**: a simple package for linear elasticity problems.
- **fluid**: Stokes' equations, with generalization of face finite elements.
- **maxwell**: Maxwell equations, generalization of edge finite elements.
- **multipatch**: extension to multi-patch defined geometries.

GeoPDEs: description of the software

GeoPDEs consists of a set of interrelated **packages** for different problems:

- **base**: the main package, with examples for Laplace problem.
- **elasticity**: a simple package for linear elasticity problems.
- **fluid**: Stokes' equations, with generalization of face finite elements.
- **maxwell**: Maxwell equations, generalization of edge finite elements.
- **multipatch**: extension to multi-patch defined geometries.

The **main features** (structures, classes and functions) are defined in **geopdes_base**.

The other packages are based on the structures defined in **base**, with the same nomenclature in each package.

All the packages are built as in octave-forge.

The main structures of GeoPDEs

GeoPDEs has been implemented following an abstract framework.

The code is based on **three** main **structures/classes**:

- **Geometry**: the parametrization **F** and its derivatives.
- **Mesh**: the partition of the domain for **numerical integration**.
- **Space**: the **basis functions** of the approximation space.

The main structures of GeoPDEs

GeoPDEs has been implemented following an abstract framework.

The code is based on **three** main **structures/classes**:

- **Geometry**: the parametrization $\mathbf{F}$ and its derivatives.
- **Mesh**: the partition of the domain for **numerical integration**.
- **Space**: the **basis functions** of the approximation space.

The **geometry** can use the NURBS package, but is not limited to it.

In version 2.0, **mesh** and **space** became classes, to avoid precomputing.

The structures/classes are used in **different applications** without changes.

Other important features

The core of GeoPDEs are the **operator** functions, a family of functions to assemble the matrices and vectors of the method.

This is the most time-consuming part. For efficiency, they are implemented in **oct-files**.

Several examples are already present: Laplacian, bilaplacian, convection terms, SUPG stabilization, Stokes, linear elasticity, Maxwell equations ...

Other important features

The core of GeoPDEs are the **operator** functions, a family of functions to assemble the matrices and vectors of the method.

This is the most time-consuming part. For efficiency, they are implemented in **oct-files**.

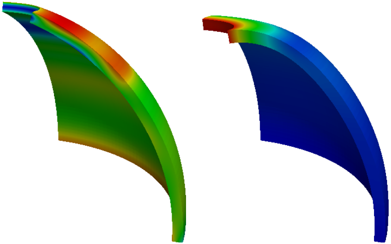
Several examples are already present: Laplacian, bilaplacian, convection terms, SUPG stabilization, Stokes, linear elasticity, Maxwell equations ...

GeoPDEs also includes several functions for postprocessing.

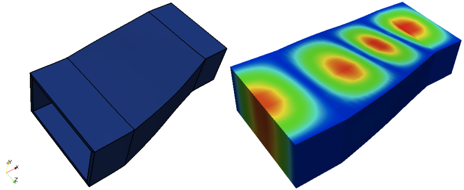
- Export to Paraview.
- Evaluate the solution at given points.
- Compute the error in academic problems with known solution.

Some examples

Linear elasticity.

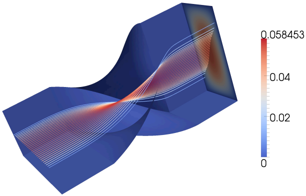


Maxwell problem in a deformed waveguide.



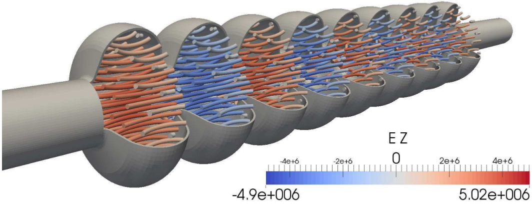
Stokes flow in a twisted pipe.

Bressan, Sangalli (2012)



The TM_{010} mode for a TESLA cavity.

Courtesy of Jacopo Corno



Current and future work

I am currently working on

- A dimension-independent implementation (it will be released soon).
  - Reduction of number of classes, functions and lines of code.
  - Simplifies the imposition of boundary conditions.
- Hierarchical splines with adaptivity (joint work with E. Garau).
  - Still in a preliminary stage.
  - Works in geopdes_base. Needs to be extended to other packages.
- Some theorems that (almost) nobody will understand.

Current and future work

I am currently working on

- A dimension-independent implementation (it will be released soon).
  - Reduction of number of classes, functions and lines of code.
  - Simplifies the imposition of boundary conditions.
- Hierarchical splines with adaptivity (joint work with E. Garau).
  - Still in a preliminary stage.
  - Works in geopdes_base. Needs to be extended to other packages.
- Some theorems that (almost) nobody will understand.

Actually, I should also work on

- Adding a test suite. Only 3 functions have a test.
- Change the repository (subversion in sourceforge).
- Rewrite the classes using classdef.
- A new web page.

Current and future work

I am currently working on

- A dimension-independent implementation (it will be released soon).
  - Reduction of number of classes, functions and lines of code.
  - Simplifies the imposition of boundary conditions.
- Hierarchical splines with adaptivity (joint work with E. Garau).
  - Still in a preliminary stage.
  - Works in geopdes_base. Needs to be extended to other packages.
- Some theorems that (almost) nobody will understand.

Actually, I should also work on

- Adding a test suite. Only 3 functions have a test.
- Change the repository (subversion in sourceforge).
- Rewrite the classes using classdef.
- A new web page.

http://geopdes.sourceforge.net

Thanks for your attention!