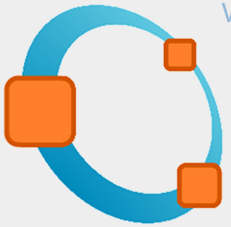
GNU Octave
A free high-level tool for Scientific Computing

Carlo de Falco, Jordi Gutiérrez Hermoso

May 21, 2013 - FEMTEC - Las Vegas



Outline

Outline

Outline

What is Octave?

Octave

"A free [a] numerical environment mostly compatible with MATLAB"

- What is compatibility? A point of much debate...
- If it works in MATLAB, it should work in Octave.
- If it breaks it is considered a bug.
- If it works in Octave, it can break in MATLAB.

[a] "free" = "libero" ≠ "gratis"

Lines of code

The stuff Octave is made of...

Lines of code

The stuff Octave is made of...

Core

- About 600,000 lines of C++
- About 100,000 lines of m-scripts
- About 50,000 lines of Fortran

Lines of code

The stuff Octave is made of...

Core

- About 600,000 lines of C++
- About 100,000 lines of m-scripts
- About 50,000 lines of Fortran

Octave-Forge

- About 200,000 lines of C++
- About 330,000 lines of m-scripts
- About 50,000 lines of Fortran

Features

Current features

- N-d arrays, linear algebra, sparse matrices
- Nonlinear equations
- Partial/Ordinary/Algebraic Differential Equations,
- Image processing, statistics, special functions
- Many more...

Features in development

- GUI
- JIT compiling
- classdef OOP

What does it look like

- Primarily a CLI interface

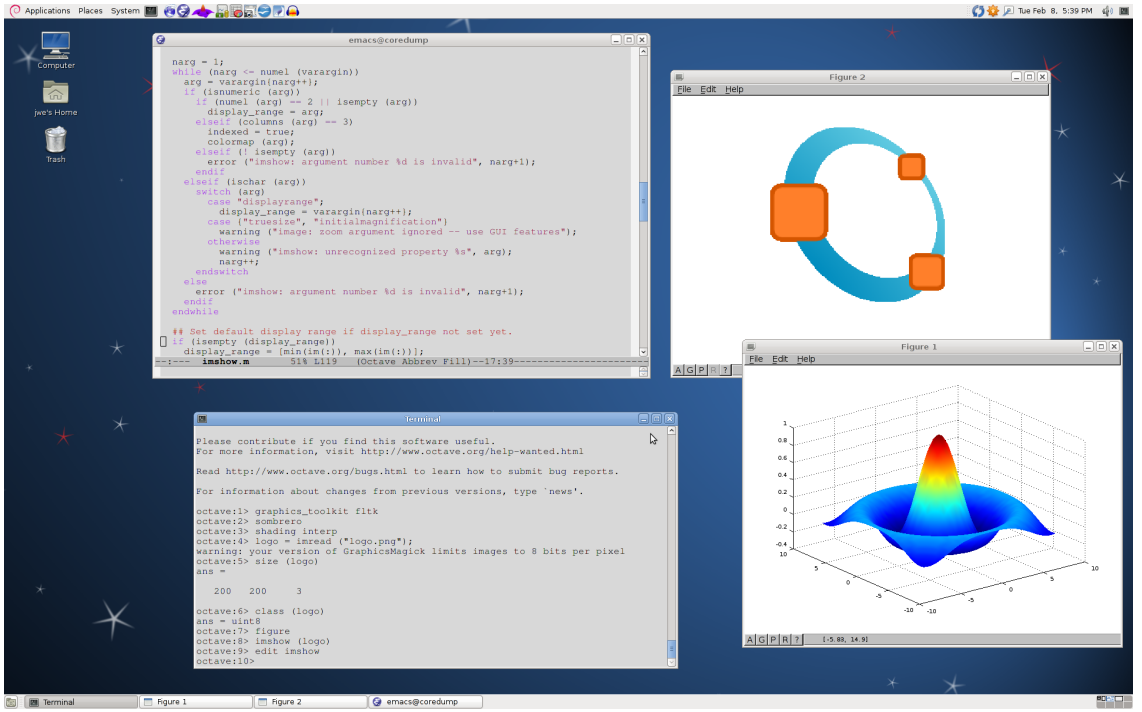


Figure : CLI screenshot

What does it look like

- Most requested feature: GUI!

What does it look like

- Most requested feature: GUI!

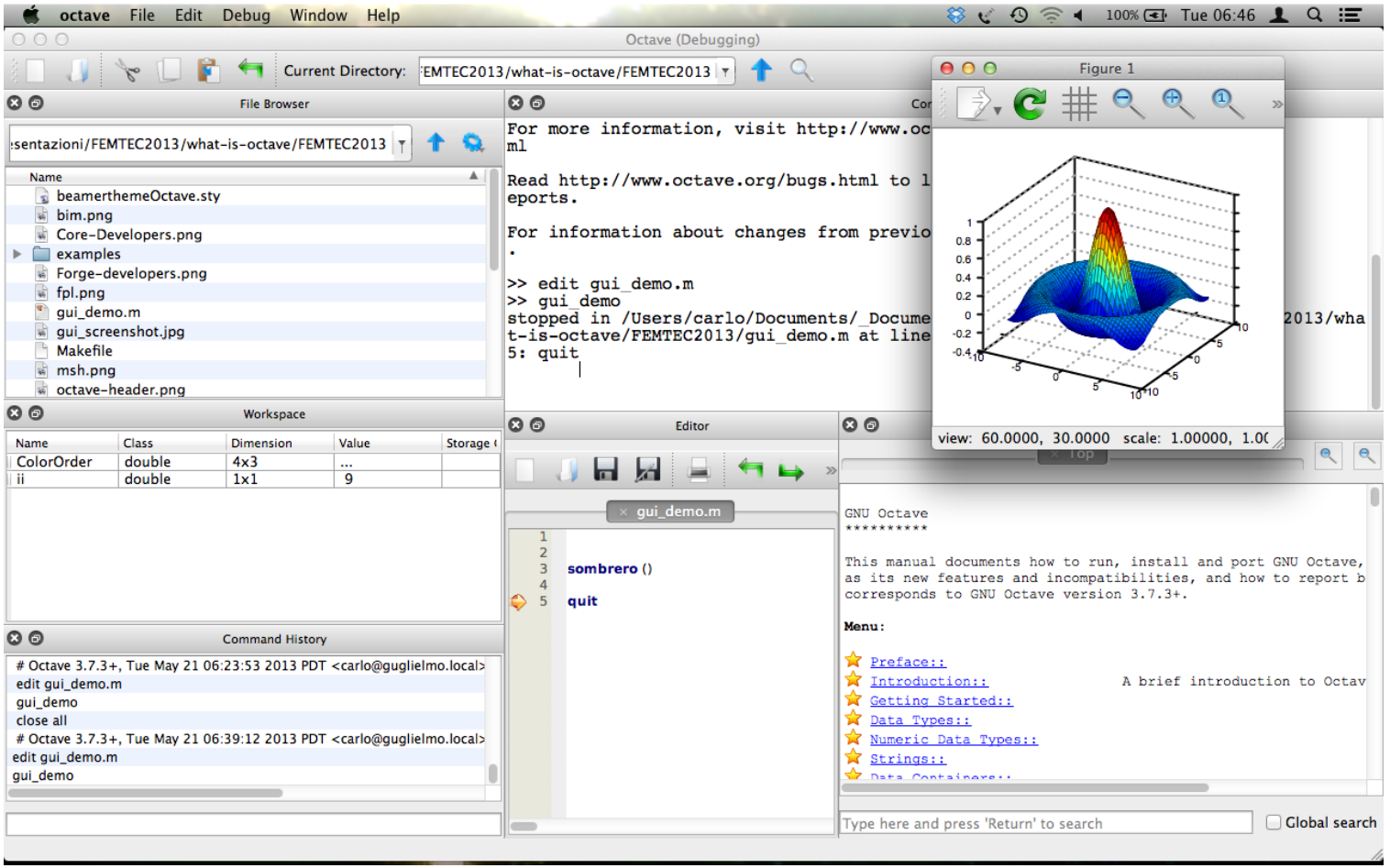


Figure : Qt based GUI Will ship with next release (4.0, expected 07/2013)

Outline

In the beginning...

- Companion software for chemical reactor textbook by James B. Rawlings and John G. Ekerdt
- John W. Eaton (hereafter, jwe) started coding in 1993

In the beginning...

- Companion software for chemical reactor textbook by James B. Rawlings and John G. Ekerdt
- John W. Eaton (hereafter, jwe) started coding in 1993

Rawlings said...

Why don't you call it "Octave"?

- Octave refers to Octave Levenspiel, nothing to do with music ...

jwe is a lone wolf...

jwe works almost completely alone for first four or five years.

jwe is a lone wolf...

jwe works almost completely alone for first four or five years.

In the very beginning...

- No mailing lists
- No widespread announcements
- No VCS (these were dark times)

Contributions slowly trickle in

Timeline

1989 Planning stages

1992 Development begins

1993 First public announcement

1994 Version 1.0

1996 Version 2.0

1998 Version 2.1 development

2004 Version 2.9 in preparation for 3.0 release

2007 Version 3.0 major upgrade

2010 Version 3.2.4, last before using hg

2011 Version 3.4.0

2012 Version 3.6.4

2012 Version 4.0

Contributions slowly trickle in

Milestones

1994 Most of the current basic functionality already in. (Much was written during its first two years!)

Contributions slowly trickle in

Milestones

1994 Most of the current basic functionality already in. (Much was written during its first two years!)

1995 Structs, MATLAB-style plot() command.

1998 Original sparse matrix implementation

2001 Octave-Forge's first commit

2006 MEX interface in core

2007 Implementation of handle graphics, full support for sparse matrices

2009 OpenGL plotting

2010 -forge option for pkg.m

2011 Profiler

2012 Nested functions

Contributions slowly trickle in

Milestones

1994 Most of the current basic functionality already in. (Much was written during its first two years!)

1995 Structs, MATLAB-style plot() command.

1998 Original sparse matrix implementation

2001 Octave-Forge's first commit

2006 MEX interface in core

2007 Implementation of handle graphics, full support for sparse matrices

2009 OpenGL plotting

2010 -forge option for pkg.m

2011 Profiler

2012 Nested functions

2013 GUI, Java, 64bit indexing

Outline

Web resources

Web pages

- Octave website
- Octave-Forge website
- Octave wiki

Users communication

- Octave users mailing list
- Octave-Forge mailing list
- #octave channel in Freenode
- Savannah bug tracker

Web resources

Web pages

- Octave website
- Octave-Forge website
- Octave wiki

Users communication

- Octave users mailing list
- Octave-Forge mailing list
- #octave channel in Freenode
- Savannah bug tracker

Developers collaboration

- Octave Mercurial repository
- Octave-Forge Subversion repository

Social structure

- Like all free projects, every user is a potential developer.
- 15 current Core developers (with write access to repo)

	Member
	Ben Abbott <bpabbott>
	Marco Caliari <caliari>
	Carlo de Falco <cdf>
	David Bateman <dbateman>
	Max Brister <fisheater>
	Michael Goffioul <goffioul>
	Jacob Dawid <jacobdawid>
	Jordi Gutiérrez Hermoso <jordigh>
	John W. Eaton <jwe>
	Kai Habel <kahacjde>
	Konstantinos Poulios <logari81>
	Mike Miller <mtmiller>
	Philip Nienhuis <philipnienhuis>
	Rik <rik5>
	Torsten <ttl>

Social structure

- Like all free projects, every user is a potential developer.
- 15 current Core developers (with write access to repo)
- 49 currently registered 'Forge developers (38 active)

Group	Users	Permissions
Admin	cdf (cdf) Thomas Treichl (treichl) Carnë Draug (carandraug) soren hauberg (hauberg) Paul Kienzle (pkienzle) David Bateman (adb014) Juan Pablo Carbajal (jpicarbajal) Add	 read admin create update
	All users in Admin group Francesco Potortë (fpoto) alex (abarth93) Rafael Vázquez (rafavzqz) Michele Martone (michelemartone) marco atzeri (matzeri) Alex (axkma) Andrius Sutas (eandrius) Etienne Grossmann (etienne) Thomas Weber (thomas-weber) andy buckle (blondandy) Paul Dreik (pauldreik) Marco Merlin (batman52) Pascal Dupuis (cdemills) Massimiliano Culpo (culpo) Ben Lewis (benjf5) Mike Miller (mtmiller) Dirk Schmidt (dreisamrd) jgpallero (jgpallero) slackydeb (slackydeb) Arno Onken (asnelt) Thomas Sailer (tsailer)	

Social structure

- Like all free projects, every user is a potential developer.
- 15 current Core developers (with write access to repo)
- 49 currently registered 'Forge developers (38 active)
- 305 total contributors over all time

Ben Abbott	Jean-Francois Cardoso	Paul Eggert	Jaroslav Hajek	Jarkko Kaleva
Andy Adler	Joao Cardoso	Stephen Eglen	Benjamin Hall	Mohamed Kamoun
Giles Anderson	Larrie Carr	Peter Ekberg	Kim Hansen	Lute Kamstra
Joel Andersson	David Castelow	Rolf Fabian	Søren Hauberg	Fotios Kasolis
Muthiah Annamalai	Vincent Cautaerts	Gunnar Farnebäck	Dave Hawthorne	Thomas Kasper
Marco Atzeri	Clinton Chee	Stephen Fegan	Daniel Heiserer	Joel Keay
Shai Ayal	Albert Chin-A-Young	R. Garcia Fernandez	Martin Helm	Mumit Khan
Roger Banks	Carsten Clark	Torsten Finke	Stefan Hepp	Paul Kienzle
Ben Barrowes	J. D. Cole	J.D.M. Frias	Martin Hepperle	Aaron A. King
Alexander Barth	Martin Costabel	Brad Froehle	Jordi Gutierrez Hermoso	Arno J. Klaassen
David Bateman	Michael Creel	Castor Fu	Yozo Hida	Alexander Klein
Heinz Bauschke	Jeff Cunningham	Eduardo Gallestey	Ryan Hinton	Geoffrey Knauth
Julien Bect	Martin Dalecki	Walter Gautschi	Roman Hodek	Heine Kolltveit
Roman Belov	Jorge Barros de Abreu	Klaus Gebhardt	A. Scottedward Hodel	Ken Kouno
Karl Berry	Carlo de Falco	Driss Ghaddab	Richard Allan Holcombe	Kacper Kowalik
David Billinghurst	Jacob Dawid	Nicolo Giorgetti	Tom Holroyd	Daniel Kraft
Don Bindner	Thomas D. Dean	Michael D. Godfrey	David Hoover	Aravindh Krishnamoorthy
Jakub Bogusz	Philippe Defert	Michael Goffioul	Kurt Hornik	Oyvind Kristiansen
Moritz Borgmann	Bill Denney	Glenn Golden	Christopher Hulbert	Piotr Krzyzanowski
Paul Boven	Fabian Deutsch	Tomislav Goles	Cyril Humbert	Volker Kuhlmann
Richard Bovey	Christos Dimitrakakis	Keith Goodman	John Hunt	Tetsuro Kurita
John Bradshaw	Pantxo Diribarne	Brian Gough	Teemu Ikonen	Miroslaw Kwasniak
Marcus Brinkmann	Vivek Dogra	Steffen Groot	Alan W. Irwin	Rafael Laboissiere
Max Brister	John Donoghue	Etienne Grossmann	Geoff Jacobsen	Kai Labusch
Remy Bruno	David M. Doolin	David Grundberg	Mats Jansson	Claude Lacoursiere
Ansgar Burchard	Carnë Draug	Kyle Guinn	Cai Jianming	Walter Landry
Marco Caliari	Pascal A. Dupuis	Peter Gustafson	Steven G. Johnson	Bill Lash
Daniel Calvelo	John W. Eaton	Kai Habel	Heikki Junes	Dirk Laurie
John C. Campbell	Dirk Eddelbuettel	Patrick Haecker	Matthias Jäschke	Maurice LeBrun
Juan Pablo Carbajal	Pieter Eendebak	W,P.Y. Hadisoeseno	Atsushi Kajita	Friedrich Leisch

Social structure

- Like all free projects, every user is a potential developer.
- 15 current Core developers (with write access to repo)
- 49 currently registered 'Forge developers (38 active)
- 305 total contributors over all time

Jyh-miin Lin	Kai P. Mueller	Tom Poage	Aleksej Saushev	John Swensen	Fook Fah Yap
Timo Lindfors	Hannes Müller	Orion Poplawski	Alois Schlögl	Daisuke Takago	Sean Young
Benjamin Lindner	Victor Munoz	Ondrej Popp	Michel D. Schmid	Ariel Tankus	Michael Zeising
Ross Lippert	Iain Murray	Jef Poskanzer	Julian Schnidder	Matthew Tenny	Federico Zenith
David Livings	Carmen Navarrete	Francesco Potorti	Nicol N. Schraudolph	Georg Thimm	Alex Zvoleff
Sebastien Loisel	Todd Neal	Konstantinos Poulios	Sebastian Schubert	Duncan Temple Lang	
E. de Castro Lopo	Philip Nienhuis	Jarno Rajahalme	Ludwig Schwardt	Kris Thielemans	
Massimo Lorenzin	Al Niessner	James B. Rawlings	Thomas L. Scofield	Olaf Till	
Emil Lucretiu	Rick Niles	Eric S. Raymond	Daniel J. Sebald	Christophe Tournery	
Hoxide Ma	Takuji Nishimura	Balint Reczey	Dmitri A. Sergatskov	Thomas Treichl	
James Macnicol	Kai Noda	Joshua Redstone	Vanya Sergeev	Karsten Trulsen	
Jens-Uwe Mager	Eric Norum	Lukas Reichlin	Baylis Shanks	Frederick Umminger	
Colin Macdonald	Krzesimir Nowak	Michael Reifenberger	Andriy Shinkarchuck	Utkarsh Upadhyay	
Rob Mahurin	Michael O'Brien	Anthony Richardson	Robert T. Short	Daniel Wagenaar	
Ricardo Marranita	Peter O'Gorman	Jason Riedy	Joseph P. Skudlarek	Stefan van der Walt	
Orestes Mas	Thorsten Ohl	E. Joshua Rigler	John Smith	Peter Van Wieren	
Axel MathÃ©i	Arno Onken	Petter Risholm	Julius Smith	James R. Van Zandt	
Makoto Matsumoto	V. Ortega-Clavero	Matthew W. Roberts	Shan G. Smith	Risto Vanhanen	
Tatsuro Matsuoka	Luis F. Ortiz	Andrew Ross	Peter L. Sondergaard	Gregory Vanuxem	
Laurent Mazet	Scott Pakin	Fabio Rossi	Joerg Specht	Ivana Varekova	
G. D. McBain	Gabriele Pannocchia	Mark van Rossum	Quentin H. Spencer	Thomas Walter	
Alexander Mamonov	Sylvain Pelissier	Joe Rothweiler	Christoph Spiel	Andreas Weber	
Christoph Mayer	Per Persson	Kevin Ruland	Richard Stallman	Olaf Weber	
J. Hoffimann Mendes	Primozz Peterlin	Kristian Rumberg	Russell Standish	Thomas Weber	
Ronald van der Meer	Jim Peterson	Ryan Rusaw	Brett Stewart	Rik Wehbring	
Thorsten Meyer	Danilo Piazzalunga	Olli Saarela	Doug Stewart	Bob Weigel	
Petr Mikulik	Nicholas Piper	Toni Saarela	Jonathan Stickel	Andreas Weingessel	
Mike Miller	Elias Pipping	Juhani Saastamoinen	Judd Storrs	Martin Weiser	
Stefan Monnier	Robert Platt	Radek Salac	Thomas Stuart	Michael Weitzel	
Antoine Moreau	Hans Ekkehard Plesser	Ben Sapp	Ivan Sutoris	David Wells	

Social structure

- Like all free projects, every user is a potential developer.
- 15 current Core developers (with write access to repo)
- 49 currently registered 'Forge developers (38 active)
- 305 total contributors over all time
- How many users? Thousands? Millions?

From user to developer

This is a FAQ

From user to developer

This is a FAQ

How can I contribute?

- Code (obviously)
- Money (pay-what-you-need)
- Documentation (especially examples)
- Wiki maintenance
- Help in the mailing list
- Bug reporting

From user to developer

This is a FAQ

How can I contribute?

- Code (obviously)
- Money (pay-what-you-need)
- Documentation (especially examples)
- Wiki maintenance
- Help in the mailing list
- Bug reporting

Student projects

Google Summer of Code

- GSoC 2011
  - Daniel Kraft, Profiler
- GSoC 2012
  - Jacob Dawid, Qt GUI; Max Brister, JIT; Ben Lewis, LSSA
- GSoC 2013
  - 8 Slots this year!
  - JIT, ILU/ICHOL, FEM, Agora

European Space Agency's Summer of Code in Space

- SOCIS 2012
  - Wendy Liu, Agora Octave; Andrius Sutas, Instrument-Control
- SOCIS 2013
  - Students apply!

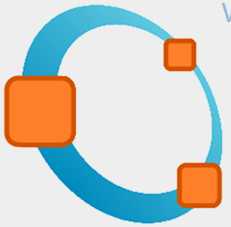
Student projects

Google Summer of Code

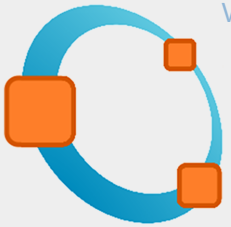
- GSoC 2011
  - Daniel Kraft, Profiler
- GSoC 2012
  - Jacob Dawid, Qt GUI; Max Brister, JIT; Ben Lewis, LSSA
- GSoC 2013
  - 8 Slots this year!
  - JIT, ILU/ICHOL, FEM, Agora

European Space Agency's Summer of Code in Space

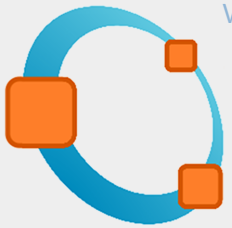
- SOCIS 2012
  - Wendy Liu, Agora Octave; Andrius Sutas, Instrument-Control
- SOCIS 2013
  - Students apply!



Outline



Outline

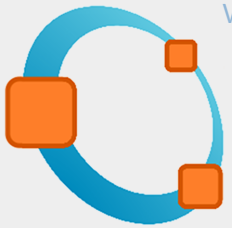


Octave-Forge

Octave-Forge

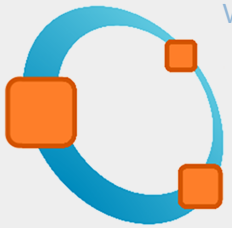
Octave Forge Is a place for concurrently developing and distributing extension packages for Octave.

- Each package has a *maintainer* responsible for updating and releasing new versions of the package
- Some packages are maintained by *The Community*
- Installation via an integrated *package manager*



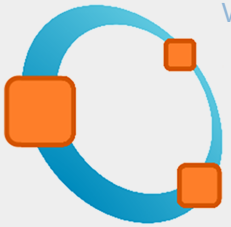
PKG

```
>> pkg install -forge miscellaneous
For information about changes from previous versions of the ↩
     miscellaneous package, run: news ("miscellaneous").
>> pkg list
Package Name           | Version | Installation directory
-----------------------+---------+----------------------
                 bim   |   1.1.1 | ~/octave/bim-1.1.1
                 fpl   |   1.3.3 | ~/octave/fpl-1.3.3
             general   |   1.3.1 | ~/octave/general-1.3.1
            geometry   |   1.6.0 | ~/octave/geometry-1.6.0
       miscellaneous   |   1.2.0 | ~/octave/miscellaneous-1.2.0
>> pkg load miscellaneous
>> pkg list
Package Name           | Version | Installation directory
-----------------------+---------+----------------------
                 bim   |   1.1.1 | ~/octave/bim-1.1.1
                 fpl   |   1.3.3 | ~/octave/fpl-1.3.3
             general   |   1.3.1 | ~/octave/general-1.3.1
            geometry   |   1.6.0 | ~/octave/geometry-1.6.0
       miscellaneous *|   1.2.0 | ~/octave/miscellaneous-1.2.0
```

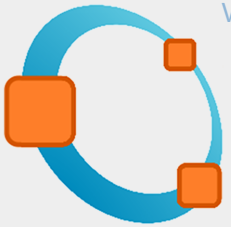


PKG

```
>> pkg describe bim -verbose
---
Package name:
bim
Version:
1.1.1
Short description:
Package for solving Diffusion Advection Reaction (DAR) Partial Differential Equations
Status:
Not loaded
---
Provides:
Matrix assembly
bim1a_advection_diffusion
bim1a_advection_upwind
bim2a_advection_diffusion
...
Pre-processing and Post-processing computations
bim2c_mesh_properties
...
>>
```

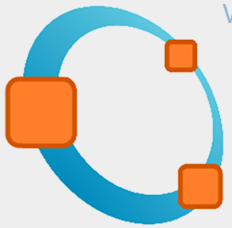


Outline



Broadcasting

- Since 3.6.0, Octave automatically broadcasts arrays when using elementwise binary operators.
- Corresponding array dimensions must either be equal or, one of them must be 1.
- In case all dimensions are equal, ordinary element-by-element arithmetic takes place.
- When one of the dimensions is 1, the array with that singleton dimension gets copied along that dimension until it matches the dimension of the other array.



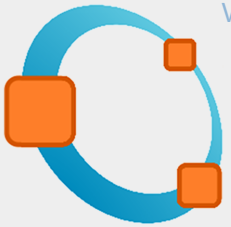
Broadcasting

```
x = [1 2 3; 4 5 6; 7 8 9];
y = [10 20 30];
x + y
    11   22   33
    14   25   36
    17   28   39
```

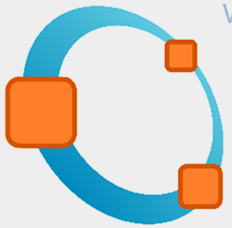
- Without broadcasting, x + y would be an error because dimensions do not agree.
- With broadcasting it is as if the following operation were performed

```
x = [1 2 3;4 5 6;7 8 9];
y = [10 20 30; 10 20 30; 10 20 30];
x + y
    11   22   33
    14   25   36
    17   28   39
```

Other notable differences with Matlab, listed in the wiki



Outline

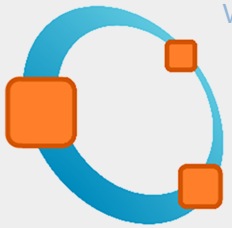


dld-functions

Implement an Octave interpreter function in C++

```
1  #include <octave/oct.h>
2
3  DEFUN_DLD(dld,args,nargout,"dld (array) \nreturn the elements of ↩
       the array in reverse order\n")
4  {
5    octave_value_list retval;
6    int nargin = args.length ();
7
8    if (nargin != 1)
9      print_usage ();
10   else
11     {
12       Array<double> a = args(0).array_value ();
13       if (! error_state)
14         {
15           Array<double> b (a);
```

source code of the example

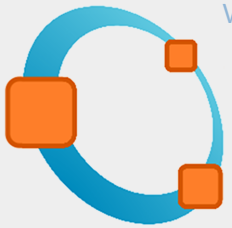


dld-functions

Implement an Octave interpreter function in C++

```
16              double* ap = a.fortran_vec ();
17              double* bp = b.fortran_vec ();
18              for (octave_idx_type i = a.numel () - 1, j = 0; i >= 0; ↩
                    i--, j++)
19                bp[i] = ap[j];
20              retval = octave_value (b);
21            }
22        }
23    return retval;
24  }
```

source code of the example

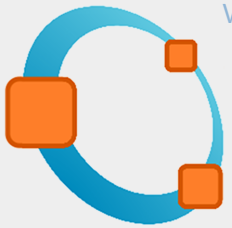


dld-functions

Implement an Octave interpreter function in C++

```
>> mkoctfile dld.cc
>> a = randn (5)
a =

   0.395421  -1.425232  -0.176544   1.055205   2.229371
  -0.241893   0.035004  -0.296543  -1.710613   0.444318
  -0.752467  -2.220469   2.380951   0.766246   1.196153
   1.404672   0.623112   1.182609   0.196125   0.609325
  -0.687019   0.646079   2.239012  -0.495169   1.488314

>> b = dld (a)
b =

   1.488314  -0.495169   2.239012   0.646079  -0.687019
   0.609325   0.196125   1.182609   0.623112   1.404672
   1.196153   0.766246   2.380951  -2.220469  -0.752467
   0.444318  -1.710613  -0.296543   0.035004  -0.241893
   2.229371   1.055205  -0.176544  -1.425232   0.395421

>>
```

source code of the example

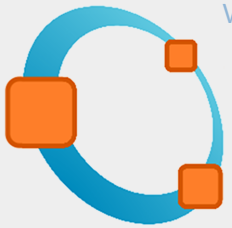


liboctave

Use Octave's Matrix/Array Classes in a C++ application

```
1  #include <iostream>
2  #include <octave/oct.h>
3
4  int main (void)
5  {
6
7    Matrix A (4, 4);
8    for (octave_idx_type i = 0; i < 4; i++)
9      for (octave_idx_type j = 0; j < 4; j++)
10       A(i,j) = 1.0 / (static_cast<double> (i) +
11                       static_cast<double> (j) + 1.0);
12
13   ColumnVector b (4, 1.0);
14   ColumnVector x = A.solve (b);
15
16   std::cout << "A = " << std::endl << A << std::endl
17             << "b = " << std::endl << b << std::endl
18             << "x = " << std::endl << x << std::endl;
19
20   return 0;
21 }
```

source code of the example

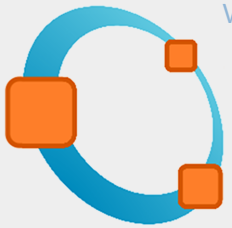


liboctave

Use Octave's Matrix/Array Classes in a C++ application

```
$ mkoctfile --link-stand-alone standalone.cc
$ ./a.out
A =
 1 0.5 0.333333 0.25
 0.5 0.333333 0.25 0.2
 0.333333 0.25 0.2 0.166667
 0.25 0.2 0.166667 0.142857
b =
1
1
1
1
x =
-4
60
-180
140
```

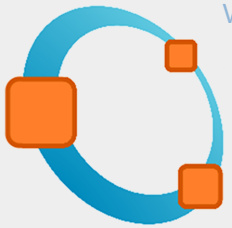
source code of the example



Embedding Octave

You can embed the Octave interpreter in your C++ application

```
1  #include <iostream>
2  #include <octave/oct.h>
3  #include <octave/octave.h>
4  #include <octave/parse.h>
5
6  int main (void)
7  {
8    string_vector octave_argv (2);
9    octave_argv(0) = "embedded";
10   octave_argv(1) = "-q";
11
12   octave_main (2, octave_argv.c_str_vec (), 1);
13
14
15   octave_value_list out = feval ("version", octave_value_list (), ↩
         1);
16   std::cout << out(0).string_value () << std::endl;
```

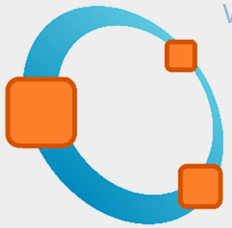


Embedding Octave

You can embed the Octave interpreter in your C++ application

```
18    Matrix A (4, 4);
19    for (octave_idx_type i = 0; i < 4; i++)
20      for (octave_idx_type j = 0; j < 4; j++)
21        A(i,j) = 1.0 / (static_cast<double> (i) +
22                        static_cast<double> (j) + 1);
23
24    ColumnVector b (4, 1.0);
25
26    out = feval ("mldivide", octave_value (A), octave_value (b), 1);
27
28    return 0;
29  }
```

source code of the example

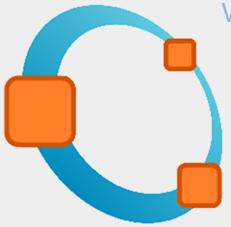


An advanced example

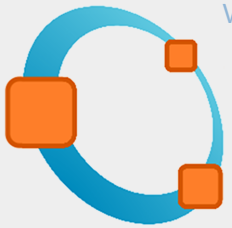
Add a new class to the Octave interpreter and work around Octave's pass-by-value semantics

source code of the example (.cc)

source code of the example (.h)



Outline

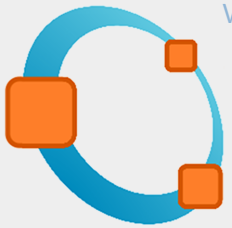


parcellfun and pararrayfun

Parcellfun is distributed in the package "general" it implements parallelization via `fork ()` and `pipe ()`

```
1  tic ();
2  nel = 100;
3  U0 = randn (200, 1);
4  us = zeros (101, 200);
5  for ii=1:numel (U0)
6
7    x = transpose (linspace (0, 1, nel+1));
8    A = bim1a_laplacian (x, 1, 1);
9    b = bim1a_rhs (x, 1, 1);
10
11   us(:,ii) = zeros (size (x));
12   us(1,ii) = U0(ii);
13
14   res = @(X) A(2:end-1, 2:end-1) * X - (b(2:end-1) - A(2:end-1, [1↩
         end]) * us([1 end], ii));
15   us(2:end-1,ii) = fsolve (res, us(2:end-1,ii));
16
17 endfor
18 toc ()
```

source code of the example

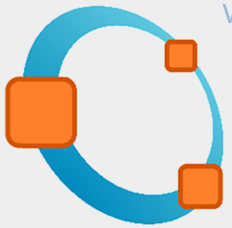


parcellfun and pararrayfun

Parcellfun is distributed in the package "general" it implements parallelization via `fork ()` and `pipe ()`

```
1  function u = poisson1d (u0)
2    nel = 100;
3    x = transpose (linspace (0, 1, nel+1));
4    A = bim1a_laplacian (x, 1, 1);
5    b = bim1a_rhs (x, 1, 1);
6
7    u = zeros (size (x));
8    u(1) = u0;
9
10   res = @(X) A(2:end-1, 2:end-1) * X - (b(2:end-1) - A(2:end-1, [1↩
         end]) * u([1 end]));
11   u(2:end-1) = fsolve (res, u(2:end-1));
12 endfunction
13 tic ();
14 U0 = num2cell (randn (1, 200));
15 up = parcellfun (2, @poisson1d, U0, "UniformOutput", true, "↩
       VerboseLevel", 2);
16 toc ()
```

source code of the example



openmpi_ext

The package `openmpi_ext` provides wrappers for the main MPI functions in `openmpi`

openmpi_ext

Package Version:	1.1.0
Last Release Date:	2012-8-29
Package Author:	Riccardo Corradini , Jaroslav Hajek, Carlo de Falco
Package Maintainer:	the Octave Community
License:	GPLv3+

Download Package
(older versions)

Function Reference

Description

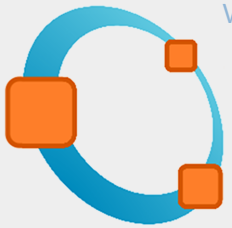
MPI functions for parallel computing using simple MPI Derived Datatypes.

Details

Dependencies: Octave (>= 3.2.4)
Autoload: No

Package: openmpi_ext

source code of the example

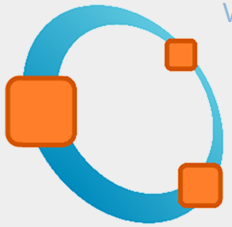


openmpi_ext

The package openmpi_ext provides wrappers for the main MPI functions in openmpi

```
1   T=clock;
2   MPI_ANY_SOURCE = -1;
3   MPI_Init ();
4   MPI_COMM_WORLD = MPI_Comm_Load ("NEWORLD");
5   rnk    = MPI_Comm_rank (MPI_COMM_WORLD);
6   siz    = MPI_Comm_size (MPI_COMM_WORLD);
7   SLV = logical(rnk);
8   MST = ~ SLV;
9   width=1/N; lsum=0;
10  i=rnk:siz:N-1;
11  x=(i+0.5)*width;
12  lsum=sum(4./(1+x.^2));
13  TAG=7;
14  if SLV
15    MPI_Send (lsum, 0, TAG, MPI_COMM_WORLD);
16  else
17    Sum =lsum;
18    for slv=1:siz-1
19      lsum = MPI_Recv (MPI_ANY_SOURCE, TAG, MPI_COMM_WORLD);
20      Sum += lsum;
21    endfor
22  endif
23  MPI_Finalize ();
```

source code of the example

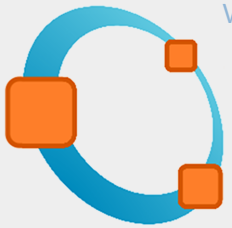


openmpi_ext

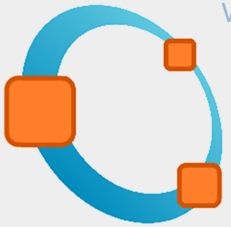
The package `openmpi_ext` provides wrappers for the main MPI functions in `openmpi`

```
1  mpirun --hostfile $HOSTFILE -np $NUMBER_OF_MPI_NODES octave --eval↩
       "pkg load openmpi_ext; Pi ()"
```

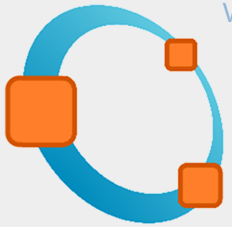
source code of the example



Outline



Outline



Some interesting packages

bim

Package Version:	1.1.1
Last Release Date:	2012-10-26
Package Author:	Carlo de Falco, Culpo Massimiliano
Package Maintainer:	Carlo de Falco
License:	GPLv2+

Download Package
(older versions)

Function Reference

Description

Package for solving Diffusion Advection Reaction (DAR) Partial Differential Equations

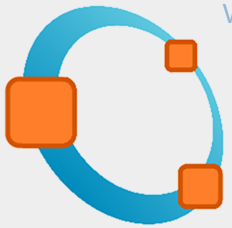
Details

Dependencies: Octave (>= 3.6.0) fpl (>= 0.0.0) msh (>= 0.0.0)
Autoload: No

Package: bim

usage

examples in the wiki



Some interesting packages

msh

Package Version:	1.0.6
Last Release Date:	2012-10-21
Package Author:	Carlo de Falco, Massimiliano Culpo
Package Maintainer:	Carlo de Falco
License:	GPLv2+

Download Package
(older versions)

Function Reference

Description

Create and manage triangular and tetrahedral meshes for Finite Element or Finite Volume PDE solvers.
Use a mesh data structure compatible with PDEtool. Rely on gmsh for unstructured mesh generation.

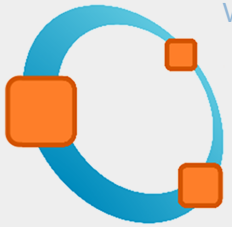
Details

Dependencies: Octave (>= 3.0.0) splines (>= 0.0.0)
Autoload: No

Package: msh

usage

examples in the wiki



Some interesting packages

fpl

Package Version:	1.3.3
Last Release Date:	2012-11-01
Package Author:	Carlo de Falco, Massimiliano Culpo and others
Package Maintainer:	Carlo de Falco, Massimiliano Culpo
License:	GPLv3+

Download Package
(older versions)

Function Reference

Description

Collection of routines to export data produced by Finite Elements or Finite Volume Simulations in formats used by some visualization programs.

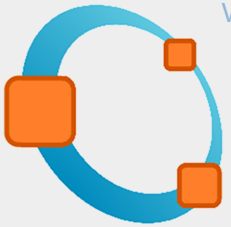
Details

Dependencies: Octave (>= 3.2.3)
Autoload: No

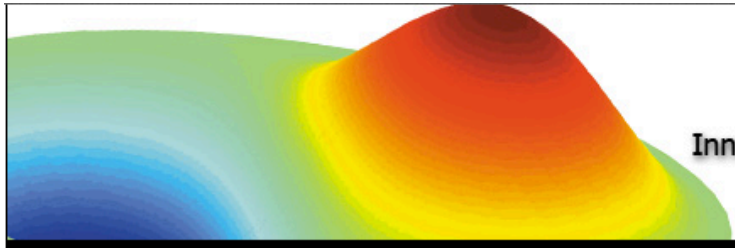
Package: fpl

usage

examples in the wiki



Outline



GeoPDEs

Innovative compatible discretization techniques for Partial Differential Equations

Software
Download
Documentation
FAQ
Contributions
How to contribute

NEW

A GeoPDEs package for T-splines has been released.

See the details or proceed to the download page.

GeoPDEs software

GeoPDEs is a suite of software tools for research on **Isogeometric Analysis** of PDEs. It provides a common and flexible framework for implementing and testing new isogeometric methods in different application areas. *GeoPDEs* is written in **Octave** and fully compatible with **Matlab**.

The suite consists of a set of interrelated **packages**. The main package, **geopdes_base**, defines the basic data-structures and methods, and should also serve as an entry point for understanding the implementation of an **Isogeometric Analysis** code.

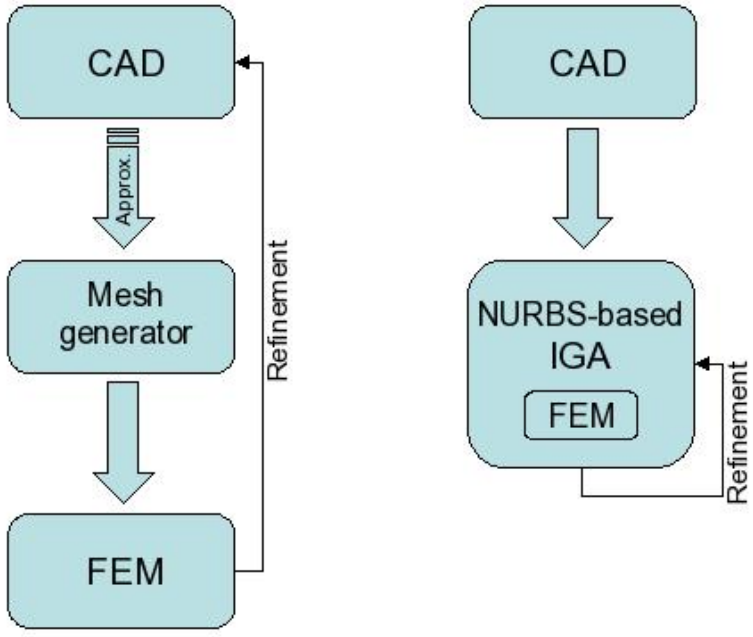
Other packages deal with applications in **linear elasticity**, **fluid mechanics** and **electromagnetism**. A package specifically meant to allow handling **multipatch** NURBS geometries is also available.

Download and installation

To download the packages, and to get installation instructions, please go to the **download page**.

New releases of *GeoPDEs* may appear from time to time, either for adding new features to the original code, or for fixing bugs. If you want to receive information about new releases, please subscribe to the **mailing list** of *GeoPDEs* users.

Iso-**g**eometric **A**nalysis

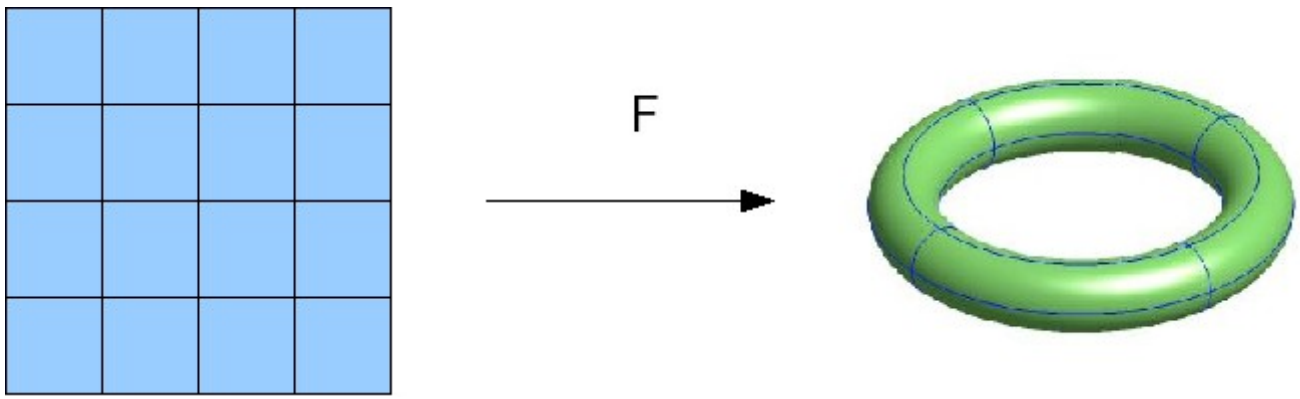


- Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(39-41):4135 – 4195
- Closer integration of CAD and CAE technologies
- Exact geometry representation
- Isoparametric philosophy

NURBS

Standard CAD software uses Non-Uniform Rational B-Splines (NURBS) for the description of geometries

$$\mathbf{F}(\xi) = \sum_{\mathbf{i}} \mathbf{C}_{\mathbf{i}} R_{\mathbf{i}}(\xi)$$



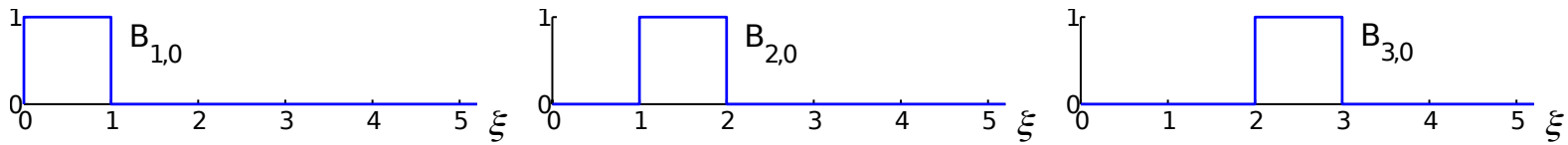
In Isogeometric Analysis (IgA) the same NURBS spaces used for geometry are chosen as approximation spaces for PDE unknowns

B-splines 1D

Given a *non-uniform knot vector* $\{\xi_1, ..., \xi_{n+p+1}\}$, in the parametric domain, B-spline basis functions are:

$$B_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi < \xi_{i+1} \\ 0 \text{ otherwise.} \end{cases}$$

$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi).$$

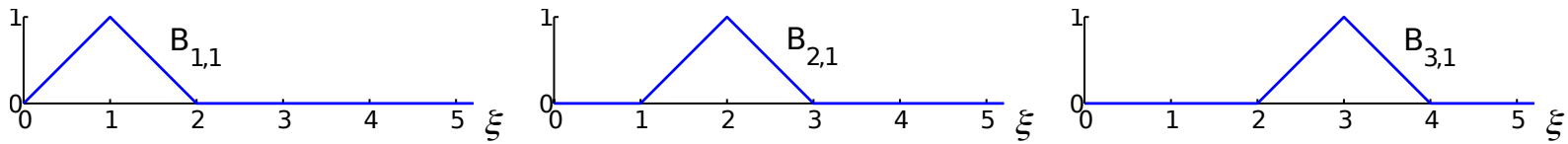


B-splines 1D

Given a *non-uniform knot vector* $\{\xi_1, ..., \xi_{n+p+1}\}$, in the parametric domain, B-spline basis functions are:

$$B_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi < \xi_{i+1} \\ 0 \text{ otherwise.} \end{cases}$$

$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi).$$

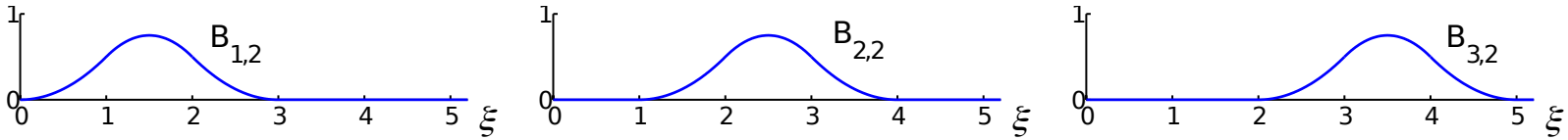


B-splines 1D

Given a *non-uniform knot vector* $\{\xi_1, ..., \xi_{n+p+1}\}$, in the parametric domain, B-spline basis functions are:

$$B_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi < \xi_{i+1} \\ 0 \text{ otherwise.} \end{cases}$$

$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi).$$

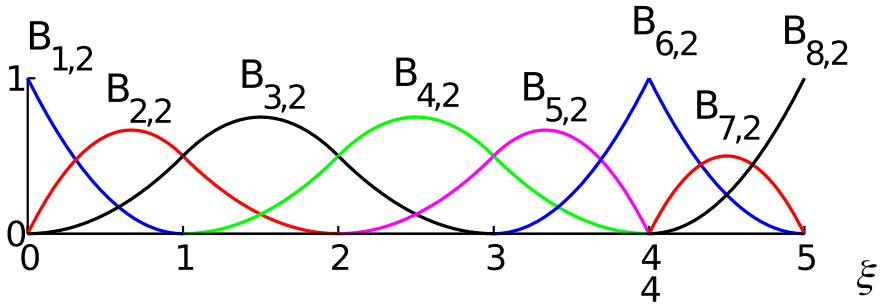


B-splines 1D

Given a *non-uniform knot vector* $\{\xi_1, ..., \xi_{n+p+1}\}$, in the parametric domain, B-spline basis functions are:

$$B_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi < \xi_{i+1} \\ 0 \text{ otherwise.} \end{cases}$$

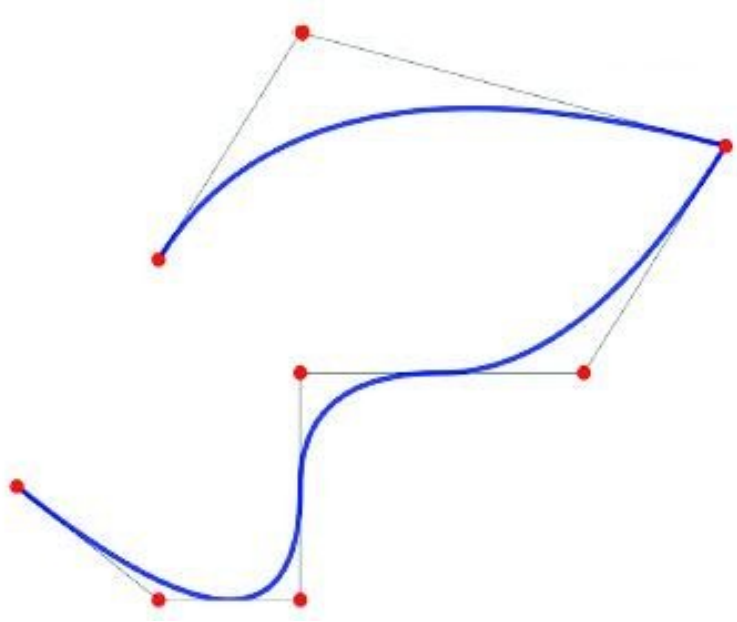
$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi).$$



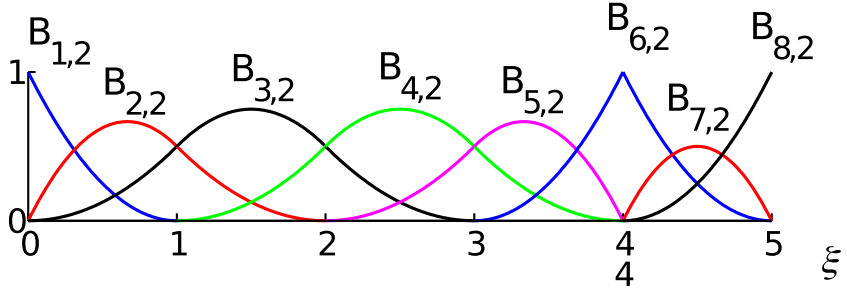
- The support of $B_{i,p}$ is (ξ_i, ξ_{i+p+1})
- The functions $B_{i,p}$ are non-negative and enjoy the partion–of–unity property
- $B_{i,p}$ is fully determined by the $p+2$ knots $\{\xi_j\}_{j=i}^{j=i+p+1}$ (local knot-vector)
- Repeating a knot reduces the smoothness of the basis at that knot

B-spline curve

The image of $\mathbf{F}(\xi) = \sum_{\mathbf{i}} \mathbf{C_i} B_{\mathbf{i}}(\xi)$ is the B-spline curve:

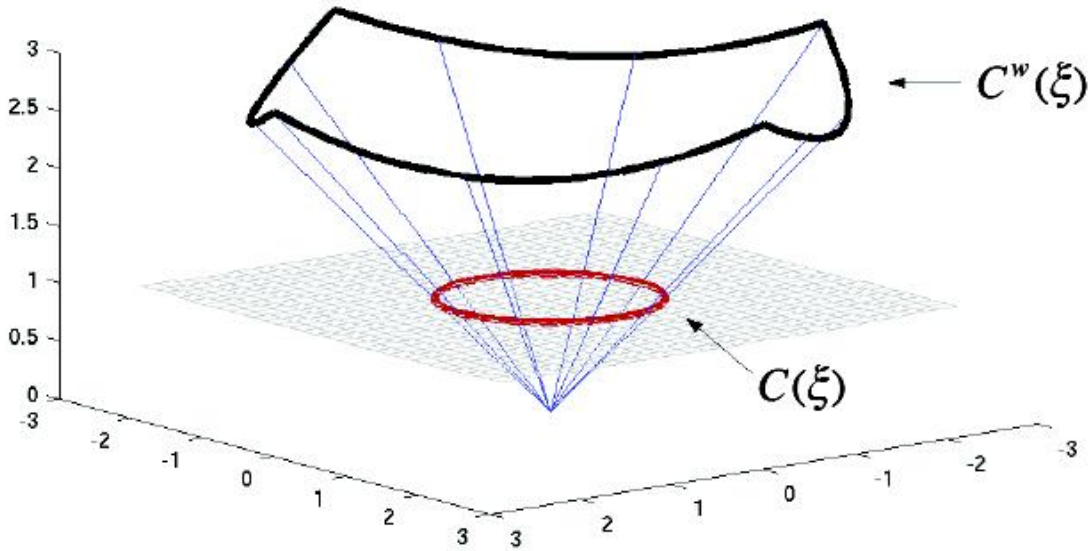


where $\mathbf{C_i} = \bullet$ and $B_{\mathbf{i}} =$



NURBS curve

A NURBS curve in $\mathbb{R}^2$ is the projection of a B-spline in $\mathbb{R}^3$



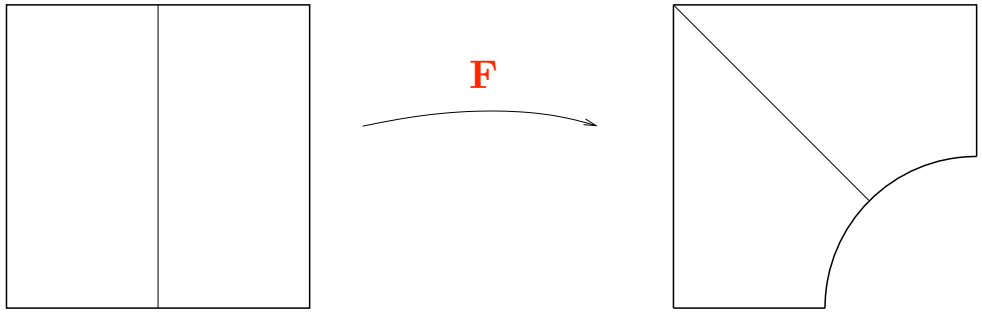
$$C(\xi) = \frac{[C_x^w(\xi), C_y^w(\xi)]}{C_z^w(\xi)} = \sum_{i=1}^{n} \mathbf{c}_i \frac{w_i B_{i,p}(\xi)}{\sum_{\hat{i}=1}^{n} w_{\hat{i}} B_{\hat{i},p}(\xi)} = \sum_{i=1}^{n} \mathbf{c}_i R_{i,p}(\xi).$$

NURBS surfaces and volumes are defined as tensor product of 1d spaces

Isogeometric and Isoparametric discretization

parametric domain

physical domain



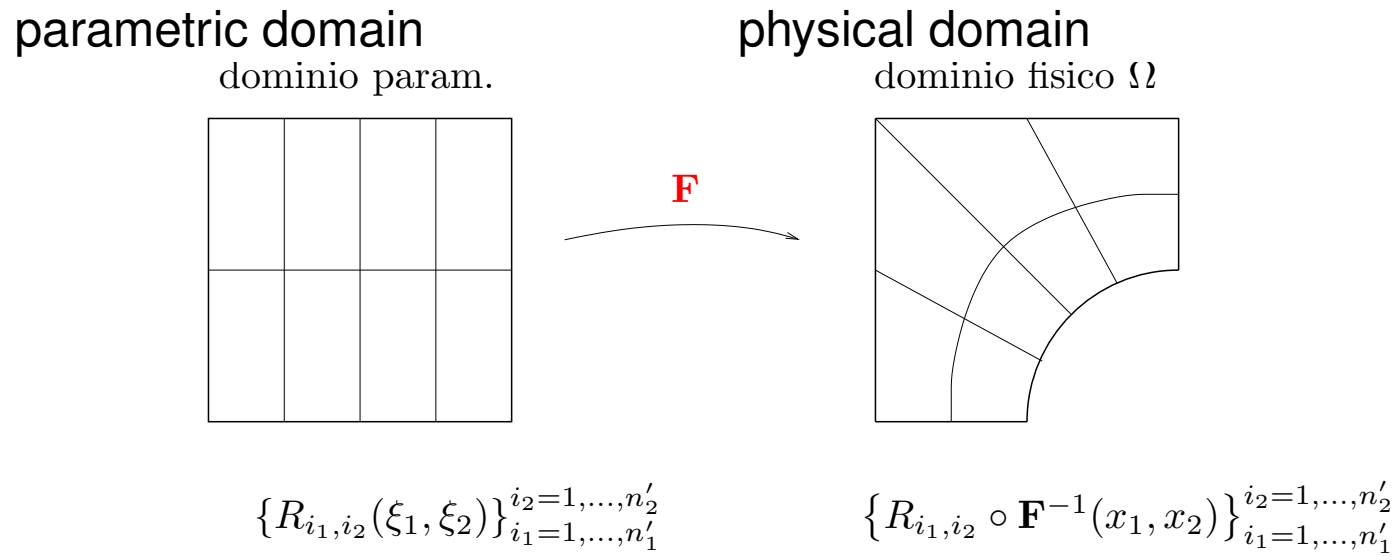
$\{R_{i_1,i_2}(\xi_1,\xi_2)\}_{i_1=1,\dots,n_1}^{i_2=1,\dots,n_2}$ $\quad$ $\{R_{i_1,i_2}\circ\mathbf{F}^{-1}(x_1,x_2)\}_{i_1=1,\dots,n_1}^{i_2=1,\dots,n_2}$

- The geometry is defined via $\mathbf{F}(\xi_1,\xi_2)=\sum_{i_1,i_2}\mathbf{C}_{i_1,i_2}G_{i_1,i_2}(\xi_1,\xi_2)$
- The discrete space V_h on Ω is the span of *push-forward* of basis functions on the parametric domain:

$$V_h=\operatorname{span}\left\{(x_1,x_2)\mapsto R_{i_1,i_2}\circ\mathbf{F}^{-1}(x_1,x_2)\right\}_{i_1=1,\dots,n_1}^{i_2=1,\dots,n_2}$$

- The discretization is isoparametric if $\mathbf{F}\in\operatorname{span}\{R_{i_1,i_2}\}$

Isogeometric and Isoparametric discretization



- The geometry is defined via $\mathbf{F}(\xi_1,\xi_2)=\sum_{i_1,i_2}\mathbf{C}_{i_1,i_2}G_{i_1,i_2}(\xi_1,\xi_2)$
- The discrete space V_h on Ω is the span of *push-forward* of basis functions on the parametric domain:

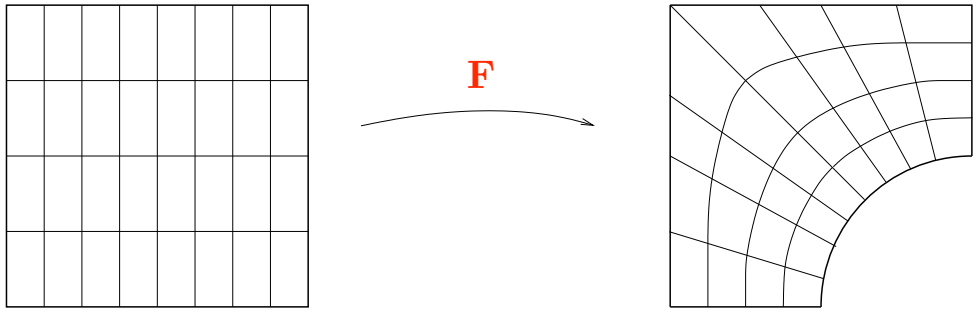
$$V_h=\operatorname{span}\left\{(x_1,x_2)\mapsto R_{i_1,i_2}\circ\mathbf{F}^{-1}(x_1,x_2)\right\}_{i_1=1,\dots,n_1''}^{i_2=1,\dots,n_2''}$$

- The discretization is isoparametric if $\mathbf{F}\in\operatorname{span}\{R_{i_1,i_2}\}$

Isogeometric and Isoparametric discretization

parametric domain

physical domain



$\{R_{i_1,i_2}(\xi_1,\xi_2)\}_{i_1=1,\dots,n_1''}^{i_2=1,\dots,n_2''}$ $\quad$ $\{R_{i_1,i_2}\circ\mathbf{F}^{-1}(x_1,x_2)\}_{i_1=1,\dots,n_1''}^{i_2=1,\dots,n_2''}$

- The geometry is defined via $\mathbf{F}(\xi_1,\xi_2)=\sum_{i_1,i_2}\mathbf{C}_{i_1,i_2}G_{i_1,i_2}(\xi_1,\xi_2)$
- The discrete space V_h on Ω is the span of *push-forward* of basis functions on the parametric domain:

$$V_h=\operatorname{span}\left\{(x_1,x_2)\mapsto R_{i_1,i_2}\circ\mathbf{F}^{-1}(x_1,x_2)\right\}_{i_1=1,\dots,n_1}^{i_2=1,\dots,n_2}$$

- The discretization is isoparametric if $\mathbf{F}\in\operatorname{span}\{R_{i_1,i_2}\}$

Some Notation: B-spline spaces in 1d

- Given positive integers p, n, s.t. $n \geq p+1$

$$\Xi := \{0 = \xi_1, \xi_2, \ldots, \xi_{n+p+1} = 1\}, \quad \xi_1 \leq \xi_2 \leq \cdots \leq \xi_{n+p+1}$$

- we only work with *open* knot vectors: the first $p+1$ knots in Ξ are equal to 0, and the last $p+1$ are equal to 1
- we assume all internal knots have multiplicity r, $1 \leq r \leq p+1$:

$$\Xi = \{\underbrace{\zeta_1, \ldots, \zeta_1}_{p+1 \text{ times}}, \underbrace{\zeta_2, \ldots, \zeta_2}_{r \text{ times}}, \ldots, \underbrace{\zeta_m, \ldots, \zeta_m}_{p+1 \text{ times}}\}.$$

- $\mathcal{Z} = \{0 = \zeta_1, \zeta_2, \ldots, \zeta_m = 1\}$ = knots without repetitions, $m = \frac{n-p-1}{r} + 2$
- we denote by B_i the p-degree B-spline basis functions, $i = 1, \ldots, n$
- B_i are piecewise polynomials of degree p on the subdivision $\{\zeta_1, \ldots, \zeta_m\}$ with $\alpha := p - r$ continuous derivatives at ζ_i, $-1 \leq \alpha \leq p-1$
- $r = p+1$, gives $\alpha = -1$, discontinuity at each ζ_i
- $S_\alpha^p := \text{span}\{B_i\}_{i=1}^n, \quad \left\{\frac{d}{dx} v : v \in S_{\alpha+1}^{p+1}\right\} = S_\alpha^p, \quad \#S_{\alpha+1}^{p+1} = \#S_\alpha^p + 1$

B-splines in 2d

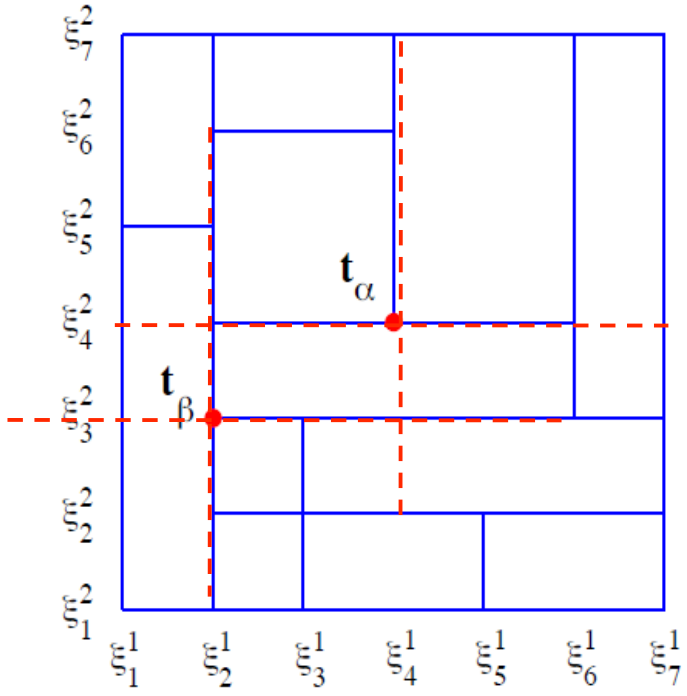
- $\widehat{\Omega} = (0,1)^2 \subset \mathbb{R}^2$ (*parametric domain*)
- p_d, r_d, n_d, $\alpha_d = p_d - r_d$, $d = 1, 2$,
- $\Xi_d = \{\xi_{1,d}, \xi_{2,d}, \ldots, \xi_{n_d+p_d+1,d}\}$, $\mathcal{Z}_d = \{\zeta_{1,d}, \ldots, \zeta_{m_d,d}\}$
- $\mathcal{Q}_h$ *mesh* of the parametric domain:
  $\mathcal{Q}_h = \{Q = \otimes_{d=1,2}(\zeta_{i_d,d}, \zeta_{i_d+1,d}),\ 1 \le i_d \le m_d - 1\}$
- $h = \max\{\mathrm{diam}(Q),\ Q \in \mathcal{Q}_h\}$
- $B_{i,d}$, with $i = 1, \ldots, n_d$ = p_d-degree univariate B-splines basis functions associated with Ξ_d, $d = 1, 2$
- tensor-product B-spline basis functions:

$$B_{ij} := B_{i,1} \otimes B_{j,2}, \quad i = 1, \ldots, n_1,\ j = 1, \ldots, n_2$$

- $S^{p_1,p_2}_{\alpha_1,\alpha_2} \equiv S^{p_1,p_2}_{\alpha_1,\alpha_2}(\mathcal{Q}_h) := S^{p_1}_{\alpha_1} \otimes S^{p_2}_{\alpha_2} = \mathrm{span}\{B_{ij}\}^{n_1,n_2}_{i=1,j=1}$
- $\mathcal{C}^\infty_\alpha$ space of univariate piece-wise smooth functions
- $\mathcal{C}^\infty_{\alpha_1,\alpha_2} = \mathcal{C}^\infty_{\alpha_1,\alpha_2}(\mathcal{Q}_h) = \mathcal{C}^\infty_{\alpha_1} \otimes \mathcal{C}^\infty_{\alpha_2}$

Breaking the tensor product structure: T-Splines

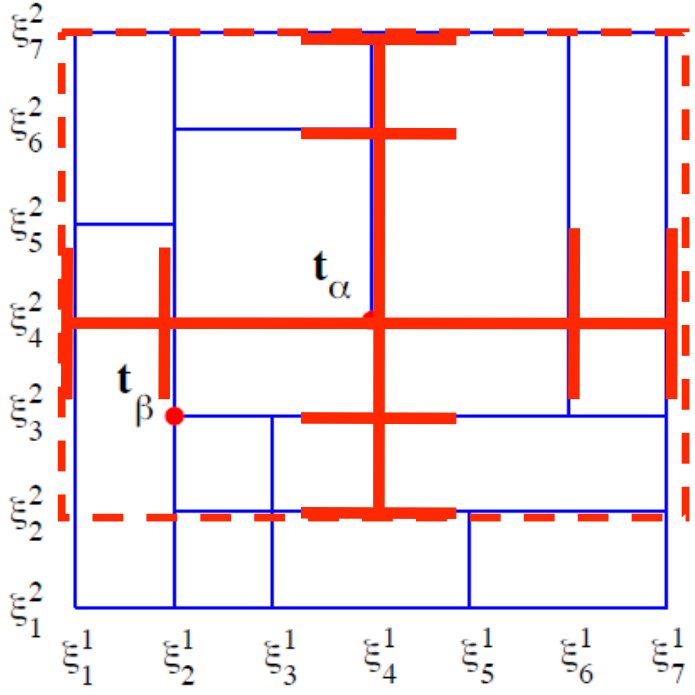
Since recently (2004) the CAD community started considering T-Splines, which are associated with a T-mesh:



Anchors

Breaking the tensor product structure: T-Splines

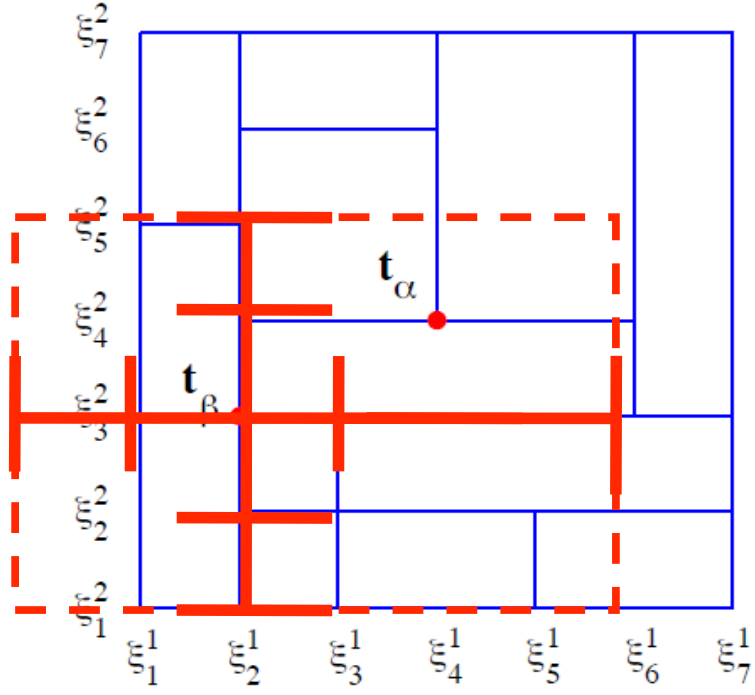
Since recently (2004) the CAD community started considering T-Splines, which are associated with a T-mesh:



Local knot-vectors

Breaking the tensor product structure: T-Splines

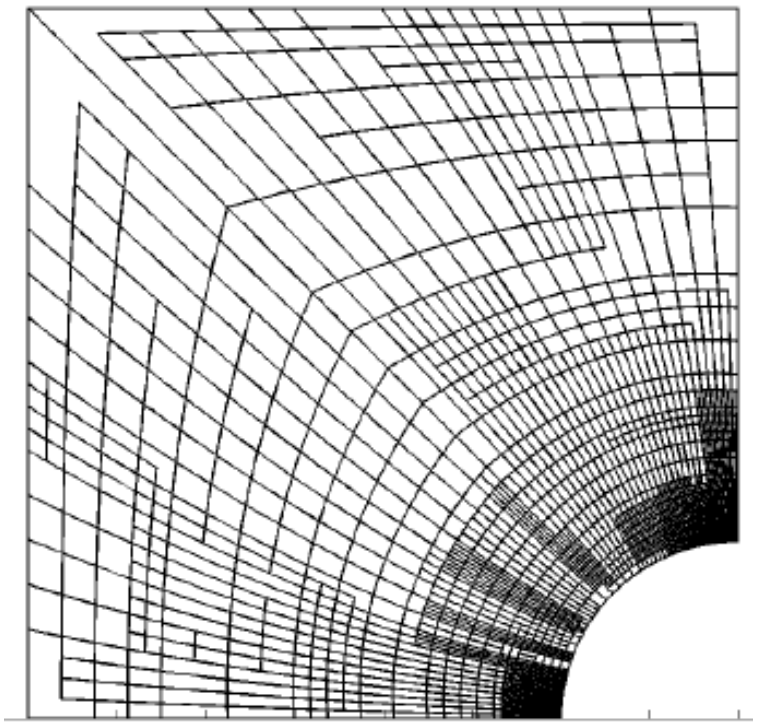
Since recently (2004) the CAD community started considering T-Splines, which are associated with a T-mesh:



Local knot-vectors

Breaking the tensor product structure: T-Splines

Since recently (2004) the CAD community started considering T-Splines, which are associated with a T-mesh:



T-NURBS

IGA in an abstract framework

- Problem to solve at the continuous level.

Abstract framework

$$a(u, v) = (f, v), \quad \forall v \in V.$$

Simple example: Poisson equation

$$\int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v = \int_\Omega f\, v, \ \forall v \in H_0^1(\Omega).$$

IGA in an abstract framework

- Problem to solve at the continuous level.
- Parametric domain and parameterization of the physical domain.

Abstract framework

$$\mathbf{F} : \widehat{\Omega} \longrightarrow \Omega \subset \mathbb{R}^d, \text{ and } \mathbf{F} \text{ is known and computable.}$$

Simple example: Poisson equation

$$\widehat{\Omega} = (0,1)^d, \text{ and } \mathbf{F} \text{ is a NURBS.}$$

IGA in an abstract framework

- Problem to solve at the continuous level.
- Parametric domain and parameterization of the physical domain.
- Discrete problem and spaces in the parametric and physical domain.

Abstract framework

$$a(u_h, v_h) = (f, v_h), \quad \forall v_h \in V_h.$$

$$V_h = \{v_h : \iota(v_h) = \widehat{v}_h \in \widehat{V}_h\}, \text{ and } \widehat{V}_h = \operatorname{span}\{\widehat{v}_j\}_{j=1}^{N_h}$$

discrete and computable spaces.

Simple example: Poisson equation

$$\int_\Omega \mathbf{grad}\, u_h \cdot \mathbf{grad}\, v_h = \int_\Omega f\, v_h, \ \forall v_h \in V_h.$$

$$V_h = \{v_h : v_h \circ \mathbf{F} = \widehat{v}_h \in \widehat{V}_h\},$$

with $\widehat{V}_h = \operatorname{span}\{N_j\}_{i=1}^{N_h}$ a space of NURBS.

IGA in an abstract framework

- Problem to solve at the continuous level.
- Parametric domain and parameterization of the physical domain.
- Discrete problem and spaces in the parametric and physical domain.
- Solve a linear system to find the discrete solution.

Abstract framework

Trial function $u_h = \sum_{i=1}^{N_h} \alpha_i v_i$, and test again every v_j, to get

$\sum_{i=1}^{N_h} \alpha_i a(v_i, v_j) = (f, v_j)$, $j = 1, \ldots, N_h$, or $\sum_{i=1}^{N_h} A_{ji}\alpha_i = b_j$.

Simple example: Poisson equation

Trial function $u_h = \sum_{i=1}^{N_h} \alpha_i R_i$, and test functions R_j:

$$\sum_{i=1}^{N_h} \alpha_i \int_\Omega \mathbf{grad}\, R_i \cdot \mathbf{grad}\, R_j = \int_\Omega f\, R_j, \quad j = 1, \ldots, N_h.$$

IGA in an abstract framework

To numerically compute the integrals, we define a partition $\widehat{\Omega} = \cup_{k=1}^{N_e} \widehat{K}_k$,

and on each "element" $\widehat{K}_k$ a quadrature rule: $\{(\widehat{\mathbf{x}}_{\ell,k}, w_{\ell,k})\}_{\ell=1}^{n_k}$.

$\int_\Omega \phi(\mathbf{x}) = \sum_{k=1}^{N_e} \int_{\widehat{K}_k} \phi(\mathbf{F}(\widehat{\mathbf{x}}))|\det(D\mathbf{F}(\widehat{\mathbf{x}}))|$

IGA in an abstract framework

To numerically compute the integrals, we define a partition $\widehat{\Omega} = \cup_{k=1}^{N_e} \widehat{K}_k$,

and on each "element" $\widehat{K}_k$ a quadrature rule: $\{(\widehat{\mathbf{x}}_{\ell,k}, w_{\ell,k})\}_{\ell=1}^{n_k}$.

$\int_\Omega \phi(\mathbf{x}) \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \phi(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$

IGA in an abstract framework

To numerically compute the integrals, we define a partition $\widehat{\Omega} = \cup_{k=1}^{N_e} \widehat{K}_k$,

and on each "element" $\widehat{K}_k$ a quadrature rule: $\{(\widehat{\mathbf{x}}_{\ell,k}, w_{\ell,k})\}_{\ell=1}^{n_k}$.

$\int_\Omega \phi(\mathbf{x}) \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \phi(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$

Using the quadrature rule, the stiffness matrix is computed as

$$A_{ij} \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \mathbf{grad}\, v_j(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) \cdot \mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$$

And recall that $\mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) = D\mathbf{F}^{-T} \widehat{\mathbf{grad}}\, \widehat{v}_i(\widehat{\mathbf{x}}_{\ell,k})$.

IGA in an abstract framework

To numerically compute the integrals, we define a partition $\widehat{\Omega} = \cup_{k=1}^{N_e} \widehat{K}_k$,

and on each "element" $\widehat{K}_k$ a quadrature rule: $\{(\widehat{\mathbf{x}}_{\ell,k}, w_{\ell,k})\}_{\ell=1}^{n_k}$.

$\int_\Omega \phi(\mathbf{x}) \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \phi(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) |\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$

Using the quadrature rule, the stiffness matrix is computed as

$$A_{ij} \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \mathbf{grad}\, v_j(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) \cdot \mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) |\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$$

And recall that $\mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) = D\mathbf{F}^{-T} \widehat{\mathbf{grad}}\, \widehat{v}_i(\widehat{\mathbf{x}}_{\ell,k})$.

Summarizing, what we need is

- A partition of $\widehat{\Omega}$ and a quadrature rule (nodes and weights).
- The evaluation of $\mathbf{F}$ and the Jacobian $D\mathbf{F}$ at the quadrature points.
- The value of the shape functions at the "mapped" quadrature points.
- A routine for matrix assembly.

IGA in an abstract framework

The implementation of GeoPDEs

- The geometry structure
- The mesh structure
- The space structure

Application in some simple examples

- Poisson problem
- Linear elasticity problem
- Multipatch domains

The main structures of GeoPDEs

GeoPDEs has been implemented following the abstract framework.

The code is based on three main structures:

- **Geometry**: the parameterization **F** and its derivatives.
- **Mesh**: the partition of the domain and the quadrature rule.
- **Space**: the shape functions of the discrete space V_h.

The main structures of GeoPDEs

GeoPDEs has been implemented following the abstract framework.

The code is based on three main structures:

- **Geometry**: the parameterization **F** and its derivatives.
- **Mesh**: the partition of the domain and the quadrature rule.
- **Space**: the shape functions of the discrete space V_h.

In version 1.x everything is precomputed and store.
In version 2.x mesh and space are computed by columns exploiting tensor product structure.
The structures are usable for a wide range of problems and applications and are easily extended.

The parameterization: geometry structure

Computation of the parameterization **F** and its derivatives.

- **map**: function handle to compute **F** at given points in $\widehat{\Omega}$.
- **map_der**: function handle to compute $D\mathbf{F}$, the derivatives of **F**.

The fields contain the handles to evaluate **F**, NOT the values of **F**.

For NURBS and B-splines, we make use of the **NURBS toolbox**.

The parameterization: geometry structure

Computation of the parameterization $\mathbf{F}$ and its derivatives.

- **map**: function handle to compute $\mathbf{F}$ at given points in $\widehat{\Omega}$.
- **map_der**: function handle to compute $D\mathbf{F}$, the derivatives of $\mathbf{F}$.

The fields contain the handles to evaluate $\mathbf{F}$, NOT the values of $\mathbf{F}$.

For NURBS and B-splines, we make use of the **NURBS toolbox**.

The computation of the geometry is separated from the shape functions.

- Necessary for non-isoparametric discretizations.
- Geometry evaluations can be made in the coarsest given geometry.

The quadrature rule: mesh structure

Contains information on the partition of the domain, $\Omega = \cup_{k=1}^{N_e} K_k$, and the quadrature rule $\{(\mathbf{x}_{\ell,k}, w_{\ell,k})\}_{\ell=1}^{n_k}$.

Recall the expression for the entries of the stiffness matrix
$A_{ij} \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \mathbf{grad}\, v_j(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) \cdot \mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) |\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$

- **nel**: N_e, number of elements of the partition.
- **nqn**: n_k, number of quadrature points per element.
- **quad_nodes**: $\widehat{\mathbf{x}}_{\ell,k}$, quadrature nodes in $\widehat{\Omega}$.
- **quad_weights**: $w_{\ell,k}$, quadrature weights.

The quadrature rule: mesh structure

Contains information on the partition of the domain, $\Omega = \cup_{k=1}^{N_e} K_k$, and the quadrature rule $\{(\mathbf{x}_{\ell,k}, w_{\ell,k})\}_{\ell=1}^{n_k}$.

Recall the expression for the entries of the stiffness matrix
$A_{ij} \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \mathbf{grad}\, v_j(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) \cdot \mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) |\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$

- **nel**: N_e, number of elements of the partition.
- **nqn**: n_k, number of quadrature points per element.
- **quad_nodes**: $\widehat{\mathbf{x}}_{\ell,k}$, quadrature nodes in $\widehat{\Omega}$.
- **quad_weights**: $w_{\ell,k}$, quadrature weights.

- **geo_map**: $\mathbf{x}_{\ell,k} = \mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})$, quadrature nodes in Ω.
- **geo_map_jac**: $D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})$, Jacobian matrix evaluated at **quad_nodes**.
- **jacdet**: $|\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$, absolute value of the Jacobian.

The discrete space: space structure

Information on the basis functions of the space

$$V_h = \mathrm{span}\{v_i\}_{i=1}^{N_h}.$$

- **ndof**: N_h, total number of degrees of freedom.
- **nsh**: N_s, number of non-vanishing functions in each "element".
- **connectivity**: indices associated to the non-vanishing basis functions.
- **shape_functions**: $v_i(\mathbf{x}_{\ell,k})$, shape functions evaluated at the quadrature points.
- **shape_function_gradients**: **grad** $v_i(\mathbf{x}_{\ell,k})$, gradients of the shape functions evaluated at the quadrature points.

Other fields may be necessary (curl, divergence, Laplacian...)

A simple example on how to use GeoPDEs

```
geometry = geo_load('ring_refined.mat');
knots = geometry.nurbs.knots;
[qn, qw] = msh_set_quad_nodes(knots, msh_gauss_nodes(ngauss));
msh = msh_2d_tensor_product (knots, qn, qw);
msh = msh_push_forward_2d (msh, geometry);
space = sp_nurbs_2d_phys (geometry.nurbs, msh);
[x, y] = deal (msh.geo_map(1,:,:), msh.geo_map(2,:,:));
mat = op_gradu_gradv (space, msh);
rhs = op_f_v (space, msh, rhs_fun(x, y));
```

- Create the **geometry** structure, from a NURBS toolbox file.
- Create the **mesh** structure in the parametric domain.
- Map the **mesh** structure to the physical domain, using **geometry**.
- Construct the **space** structure (the knots are stored in **geometry**).
- Build the matrix and right-hand side.

Matrices and vector construction

Recall the expression for the entries of the stiffness matrix

$$A_{ij} \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \mathbf{grad}\, v_j(\mathbf{F}(\mathbf{x}_{\ell,k})) \cdot \mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))| \det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$$

- Everything is precomputed in the previous structures.
- To construct the matrices, it is enough to correctly gather the information.
- The computation of the matrices is clearer.
- The main drawback: large memory consumption.

Matrices and vector construction

Recall the expression for the entries of the stiffness matrix

$$A_{ij} \simeq \sum_{k=1}^{N_e} \sum_{\ell=1}^{n_k} w_{\ell,k} \mathbf{grad}\, v_j(\mathbf{F}(\mathbf{x}_{\ell,k})) \cdot \mathbf{grad}\, v_i(\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k})) |\det(D\mathbf{F}(\widehat{\mathbf{x}}_{\ell,k}))|$$

```
function mat = op_gradu_gradv (space, msh)
for iel = 1:msh.nel
 mat_loc = zeros (space.nsh(iel), space.nsh(iel));
 for idof = 1:space.nsh(iel)
  ishp = space.shape_function_gradients(:,idof,iel);
  for jdof = 1:space.nsh(iel)
   jshp = space.shape_function_gradients(:,jdof,iel);
   for inode = 1:msh.nqn
    mat_loc(idof,jdof) +=  ishp(inode) .* jshp(inode) * ...
       msh.jacdet(inode,iel) * msh.quad_weights(inode,iel);
   end %inode
  end %jdof
 end %idof
 mat(space.connect(:,iel), space.connect(:,iel)) += mat_loc;
end %iel
```

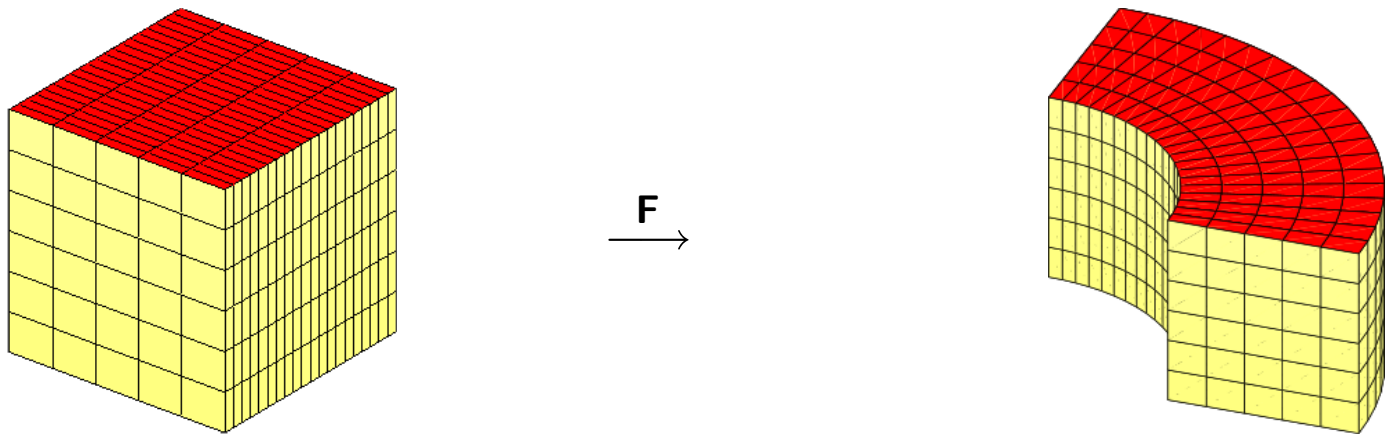
Boundary conditions: the **boundary** substructures

The structures **mesh** and **space** are enriched with a field **boundary**.

These are mesh and space **substructures** of dimension $N - 1$.

The boundary structures have some particularities:

- **jacdet** contains the norm of the differential of the boundary parameterization.
- The **space** structure uses a local numbering for each boundary.
- A new field, **dofs**, is added to recover the global numbering.



Boundary conditions: the **boundary** substructures

For **Neumann** conditions, $\frac{\partial u}{\partial n} = g$ on $\partial\Omega$, we must compute $\int_{\partial\Omega} g v_j$.

- The integral is computed in the same manner as for bulk forces.
- It is assembled into the global r.h.s. using the field **dofs**.

```
x = msh.boundary.geo_map(1,:,:);
y = msh.boundary.geo_map(2,:,:);
rhs_bnd = op_f_v(space.boundary,msh.boundary,g(x,y));
rhs(space.boundary.dofs) += rhs_bnd;
```

For **Dirichlet** conditions we must assign the d.o.f. in **boundary.dofs**.

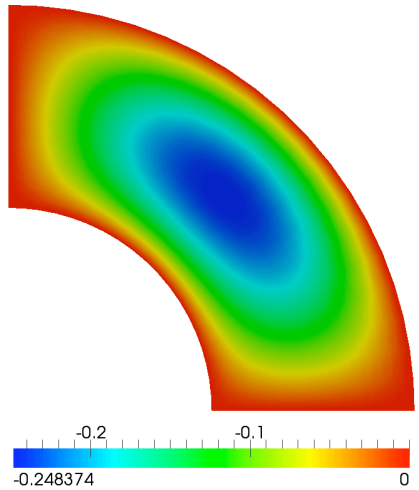
- The needed information should already be in the **boundary** structures.
- As an example we have included the least squares best fit.

A simple example on how to use GeoPDEs (again)

```
geometry = geo_load('ring_refined.mat');
knots = geometry.nurbs.knots;
[qn, qw] = msh_set_quad_nodes(knots,msh_gauss_nodes(ngauss));
msh = msh_2d_tensor_product (knots, qn, qw);
msh = msh_push_forward_2d (msh, geometry);
space = sp_nurbs_2d_phys (geometry.nurbs, msh);
[x, y] = deal (msh.geo_map(1,:,:), msh.geo_map(2,:,:));
mat = op_gradu_gradv (space, msh);
rhs = op_f_v (space, msh, rhs_fun(x, y));
drchlt_dofs = unique ([space.boundary(:).dofs]);
int_dofs = setdiff (1:space.ndof, drchlt_dofs);
u(drchlt_dofs) = 0;
u(int_dofs) = mat(int_dofs, int_dofs) \ rhs(int_dofs);
sp_to_vtk_2d (u, space, geometry, [20 20], filename, 'u');
err = sp_l2_error (space, msh, u, exact_solution(x, y));
```

A simple example on how to use GeoPDEs (again)

And the final result is something like this.



The package contains **several** simple **examples**:

h-,*p*-,*k*-refinement, 2D and 3D, B-splines and NURBS...

The structures can be **easily extended** to solve more **complex problems**.

Linear elasticity problems in GeoPDEs

Let us see how to solve the following **linear elasticity** problem:

Find $\mathbf{u} \in V = (H^1_{0,\Gamma_D}(\Omega))^d$ *such that*

$$\int_\Omega (2\mu\,\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) + \lambda\,\mathrm{div}(\mathbf{u})\,\mathrm{div}(\mathbf{v})) = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_{\Gamma_N} \mathbf{g}\cdot\mathbf{v} \quad \forall \mathbf{v} \in V,$$

- The **geometry** and **mesh** are described as in the previous example.
- The basis functions in the **space** structure are now vector-valued.
- A specific **operator** for this problem must be defined.
- Imposing the boundary conditions is similar to the previous problem.

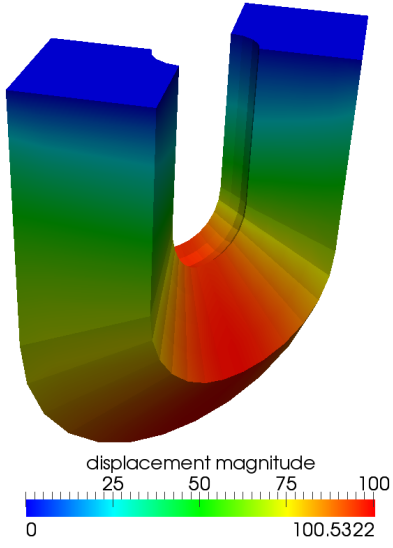
Definition of the vectorial space

We first define one **space** structure for **each component**.

From these we define the vector-valued **space** structure for our problem.

```
spx = spy = spz = sp_nurbs_3d_phys (geometry, msh);
space = sp_scalar_to_vector_3d (spx, spy, spz, msh);
```

This command computes the new vector-valued **space** and the numbering.

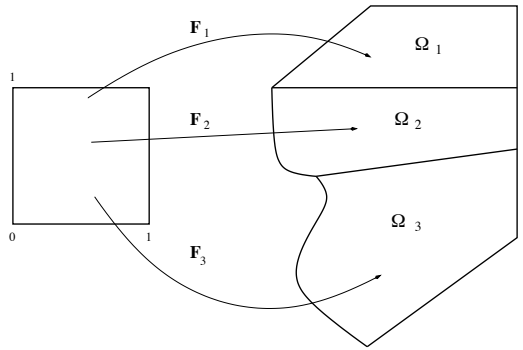


Thanks to T.J.R. Hughes' group for the horseshoe

Problems defined in multipatch domains

Multipatch domains are defined as $\overline{\Omega} = \cup_{j=1}^{N} \overline{\Omega}_j$,

with $\mathbf{F}_j : \widehat{\Omega} \longrightarrow \Omega_j \subset \mathbb{R}^d$ known and computable parameterizations.



One **geometry**, **mesh** and **space** structures for **each patch**.

Two main strategies to deal with multipatch geometries:

- Boundary "meshes" match exactly: C^0 continuity. (Released)
- Domain decomposition methods. (Future work)

In every case the coupling should be done using the **boundary structures**.

Part II

Examples of Applications

Testing Discretization Schemes

- Incompressible fluids
- Extension: Higher-order operators

More complex applications

- A Non-linear Problem: Schrödinger-Poisson
- A "Multiphysics" Problem: Twisted Wave-Guide

Conclusions

Stokes' Problem

- **Mixed formulation of the Stokes' problem**

$$\begin{aligned} &\textit{Find } \mathbf{u} \in V \textit{ and } p \in Q \textit{ s.t.} \\ &\int_\Omega \mu \boldsymbol{\nabla} \mathbf{u} : \boldsymbol{\nabla} \mathbf{v} - \int_\Omega p \ \mathrm{div}\ \mathbf{v} = \int_\Omega \mathbf{f} \cdot \mathbf{v} \qquad \forall \mathbf{v} \in V \\ &\int_\Omega q \ \mathrm{div}\ \mathbf{u} = 0 \qquad \forall q \in Q. \end{aligned}$$

- **Discrete inf-sup condition** for the discrete spaces V_h and Q_h

$$\inf_{q \in Q_h, q \neq 0} \sup_{\mathbf{v} \in V_h, \mathbf{v} \neq 0} \frac{b(\mathbf{v}, q)}{\|q\|_{L^2} \|\mathbf{v}\|_{H^1}} \geq c_{is} > 0, \text{ independent of } h$$

- **Sufficient condition for divergence-free velocities**

$$Q_h = \{\mathrm{div}\mathbf{v} : \mathbf{v} \in V_h\}$$

Some compatible spaces

- Some possible choices of B-Spline spaces
- On the parametric domain
  - Let, for simplicity, the pressure space be

$$\widehat{Q}_h \equiv \widehat{Q}_h(p, \alpha) = S^{p,p}_{\alpha,\alpha}$$

  - Consider as velocity space

$$\widehat{V}_h^{\mathrm{TH}} = S^{p+1,p+1}_{\alpha,\alpha} \times S^{p+1,p+1}_{\alpha,\alpha}$$

- On the physical domain
  - The pressure space is mapped as

$$Q_h = \{q : q \circ \mathbf{F} \in \widehat{Q}_h\}$$

  - The velocity space is mapped as

$$V_h^{\mathrm{TH}} = \{\mathbf{v} : \mathbf{v} \circ \mathbf{F} \in \widehat{V}_h^{\mathrm{TH}}\}, \quad Q_h = \{q : q \circ \mathbf{F} \in \widehat{Q}_h\}$$

  - [Bazilevs, da Veiga, Cottrell, Hughes, Sangalli 2006][Bressan 2009, 2010]

Some compatible spaces

- Some possible choices of B-Spline spaces
- On the parametric domain
  - Let, for simplicity, the pressure space be

$$\widehat{Q}_h \equiv \widehat{Q}_h(p, \alpha) = S^{p,p}_{\alpha,\alpha}$$

  - Consider as velocity space

$$\widehat{V}_h^{\mathrm{NDL}} = S^{p+1,p+1}_{\alpha+1,\alpha} \times S^{p+1,p+1}_{\alpha,\alpha+1}$$

- On the physical domain
  - The pressure space is mapped as

$$Q_h = \{q : q \circ \mathbf{F} \in \widehat{Q}_h\}$$

  - The velocity space is mapped as

$$V_h^{\mathrm{NDL}} = \left\{ \frac{D\mathbf{F}}{\det(D\mathbf{F})} \mathbf{v} : \mathbf{v} \circ \mathbf{F} \in \widehat{V}_h^{\mathrm{NDL}} \right\}$$

  - [Buffa, de Falco, Sangalli 2010]

Some compatible spaces

- Some possible choices of B-Spline spaces
- On the parametric domain
  - Let, for simplicity, the pressure space be

$$\widehat{Q}_h \equiv \widehat{Q}_h(p, \alpha) = S^{p,p}_{\alpha,\alpha}$$

  - Consider as velocity space

$$\widehat{V}_h^{\mathrm{RT}} = S^{p+1,p}_{\alpha+1,\alpha} \times S^{p,p+1}_{\alpha,\alpha+1}$$

- On the physical domain
  - The pressure space is mapped as

$$Q_h = \{q : q \circ \mathbf{F} \in \widehat{Q}_h\}$$

  - The velocity space is mapped as

$$V_h^{\mathrm{RT}} = \left\{ \frac{D\mathbf{F}}{\det(D\mathbf{F})} \mathbf{v} : \mathbf{v} \circ \mathbf{F} \in \widehat{V}_h^{\mathrm{RT}} \right\}$$

  - [Buffa, de Falco, Sangalli 2010]

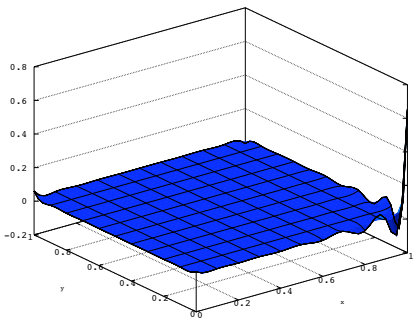
Divergence free solutions with the RT space

- Without boundary conditions

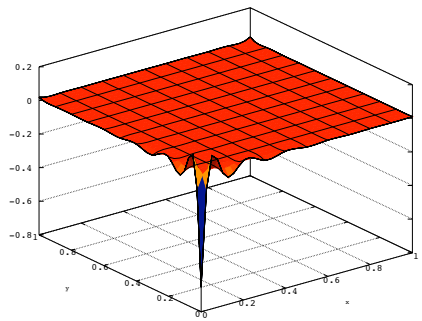
$$\widehat{V}_h^{\mathrm{RT}} := S_{\alpha+1,\alpha}^{p+1,p} \times S_{\alpha,\alpha+1}^{p,p+1} \xrightarrow{\mathrm{div}} S_{\alpha,\alpha}^{p,p} =: \widehat{Q}_h$$

- With no-slip boundary conditions

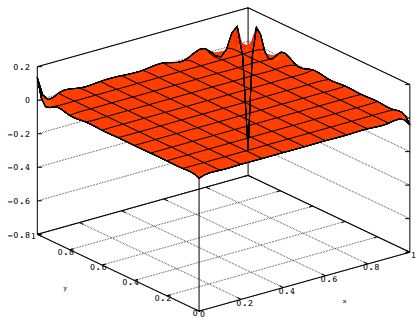
$$V_{h,0}^{RT} := (H_0^1)^2 \cap S_{\alpha+1,\alpha}^{p+1,p} \times S_{\alpha,\alpha+1}^{p,p+1} \xrightarrow{\mathrm{div}} \widetilde{Q}_{h,0} \subset S_{\alpha,\alpha}^{p,p}$$



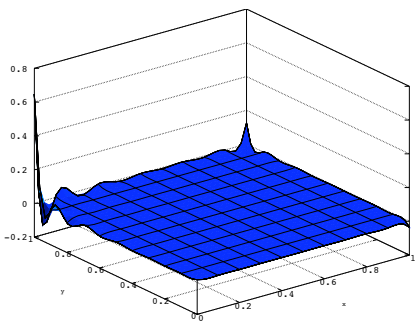
(a)



(b)



(c)



(d)

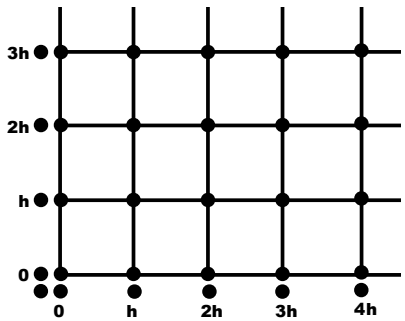
Divergence free solutions with the RT space

- Without boundary conditions

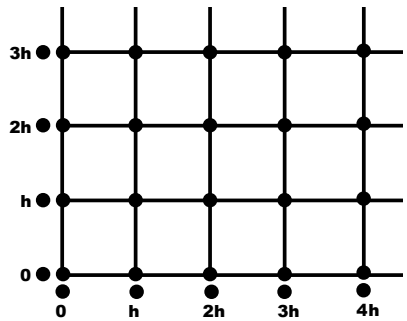
$$\widehat{V}_h^{\mathrm{RT}} := S_{\alpha+1,\alpha}^{p+1,p} \times S_{\alpha,\alpha+1}^{p,p+1} \xrightarrow{\mathrm{div}} S_{\alpha,\alpha}^{p,p} =: \widehat{Q}_h$$

- With no-slip boundary conditions

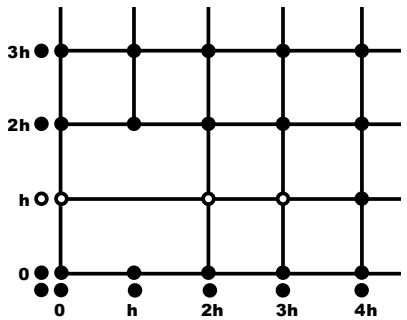
$$V_{h,0}^{RT} := (H_0^1)^2 \cap S_{\alpha+1,\alpha}^{p+1,p} \times S_{\alpha,\alpha+1}^{p,p+1} \xrightarrow{\mathrm{div}} \widetilde{Q}_{h,0} \subset S_{\alpha,\alpha}^{p,p} \approx \widehat{Q}_{h,0}$$



(e)



(f)



(g)

Implementation

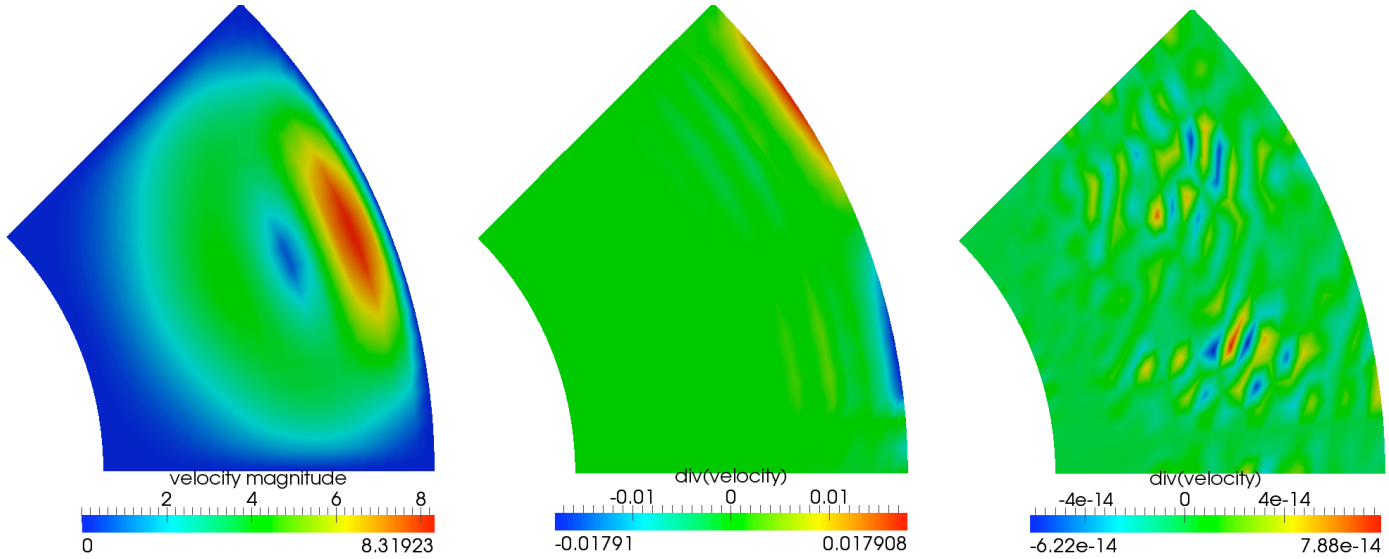
- GeoPDEs provides functions to build the function space structures corresponding to the TH, NDL and RT space pairs

```
[spv, spp]     = sp_bspline_th_2d_phys  (knotsp, degree,
    msh);
[spv, spp]     = sp_bspline_ndl_2d_phys (knotsp, degree,
    msh);
[spv, spp, P] = sp_bspline_rt_2d_phys  (knotsp, degree,
    msh);
```

- knotsp and degree refer to the pressure space
- under the hood:
  - the pressure space structure **spp** is computed as in Part I
  - the knot vectors and dofs for each component of the velocity are automatically computed from those of the pressure and the **spv** structure is computed
  - for NDL and RT spaces the Piola-mapping is also applied.
  - the additional output P of the RT space constructor is a matrix that operates the change of basis from Q_h to $\widetilde{Q}_h$
- assembling the matrices

```
A = op_gradu_gradv (spv, spv, msh, mu (x, y));
B = P' * op_div_v_q (spv, spp, msh);
```

Numerical results



(a) (b) (c)

(a) Velocity magnitude for RT

(b) divergence for TH

(c) divergence for RT

Stream-function formulation of Stokes' problem

Find $\psi \in \Phi$ *s.t.*

$$\int_\Omega \mu \nabla \mathbf{curl}\, \psi \,:\, \nabla \mathbf{curl}\, \phi = \int_\Omega \mathbf{f} \cdot \mathbf{curl}\, \phi \qquad \forall \phi \in \Phi$$

$$\mathbf{u} = \mathbf{curl}\, \psi$$

- $\mathbf{curl} := \partial/\partial y, -\partial/\partial x$
- $\Phi = \left\{ \phi \in H^2(\Omega) /, \mathbf{curl}\, \phi = 0 \text{ on } \partial\Omega \right\}$
- $\mathbf{K} = \{ \mathbf{u} \in V_0, \mathrm{div}\ \mathbf{u} = 0 \text{ in } \Omega \}$
- $\psi \in \Phi \Rightarrow \mathbf{u} \in \mathbf{K}$
- **curl** provides a topological and algebraic isomorphism between Φ and $\mathbf{K}$
- this formulation is the basis for the discretization method proposed in [Auricchio, da Veiga, Buffa, Lovadina, Reali, Sangalli 2007]
- the method takes advantage of the smoothness of the B-Spline basis functions

Implementation

- original approach:
  - higher order derivatives of the basis functions

```
sp = sp_bspline_2d_phys (knots, degree, msh,
    'hessian', true);
```

  - discretization of the fourth order differential operator

```
A = op_gradcurlu_gradcurlv_2d (sp, sp, msh, mu (x,
    y));
b = op_f_curlv_2d (sp, msh, f (x, y));
```

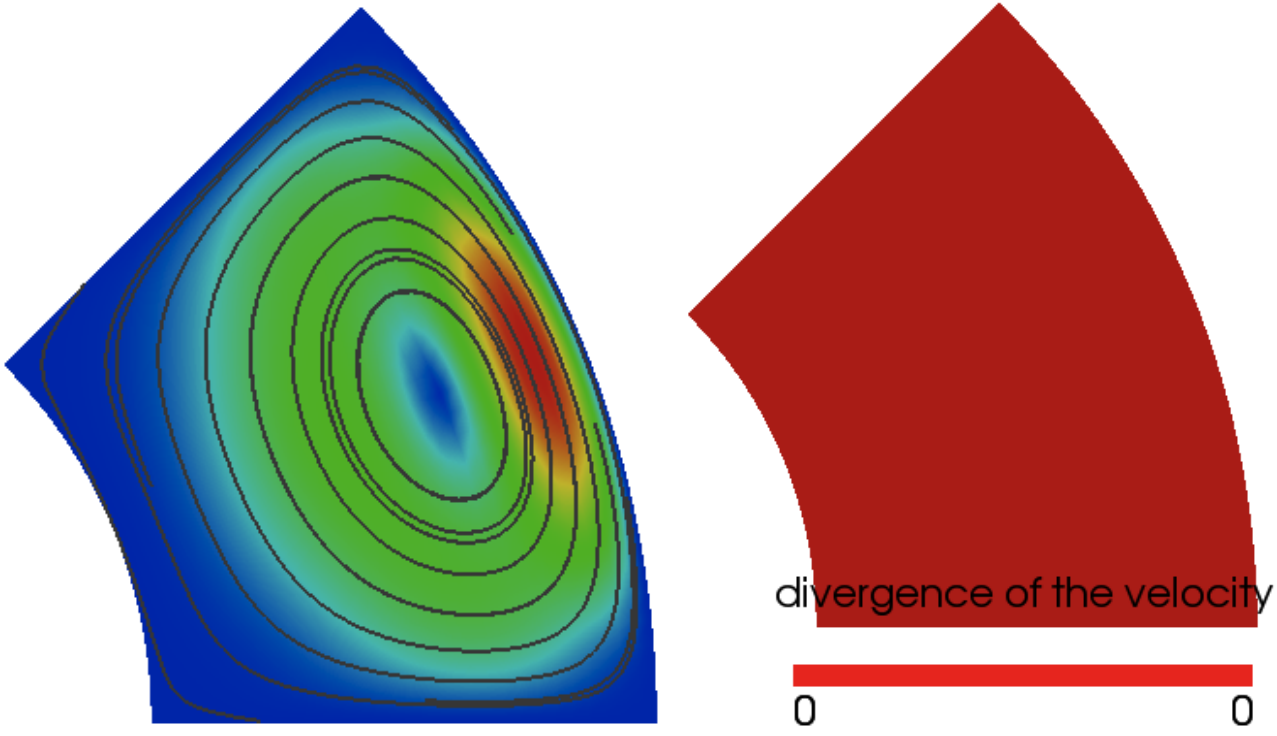
- alternative approach:
  - Use as basis functions the curls of B-Spline basis functions

```
sp = sp_bspline_curl_2d_phys (knotsp, degree, msh);
```

  - use the "usual" functions to assemble the matrices

```
A = op_gradu_gradv (sp, sp, msh, mu (x, y));
b = op_f_v (sp, msh, f (x, y));
```

Numerical results



(a) (b)

(a) velocity

(b) divergence of the velocity

- the result is divergence-free by construction

A highly non-linear problem

$$-\mathrm{div}\left(\varepsilon\ \mathbf{grad}\,\varphi\right) + qn_i\gamma\exp\left[\frac{\varphi}{KT/q}\right] = qD$$

$$\left[-\frac{\hbar^2}{2m^*}\triangle - q\varphi\right]\psi_i = E_i\psi_i$$

$$\gamma = \sum_i 2|\psi_i|^2\exp\left[-\frac{E_i/q+\varphi}{KT}\right]$$

- nested iteration solution algorithm

(1) Solve the non-linear Poisson equation for φ

  (1.1) Newton iteration

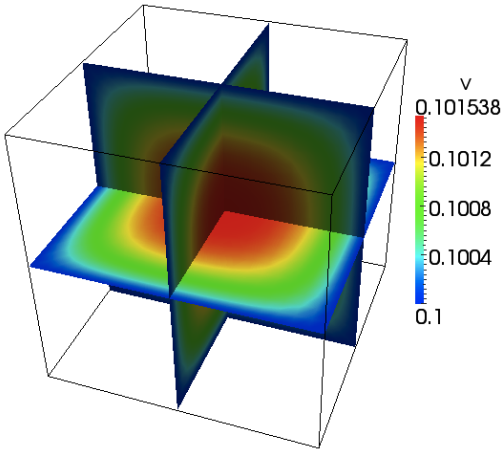
(2) Solve the Schrödinger eigenproblem for ψ_i and E_i

(3) Update the quantum correction γ

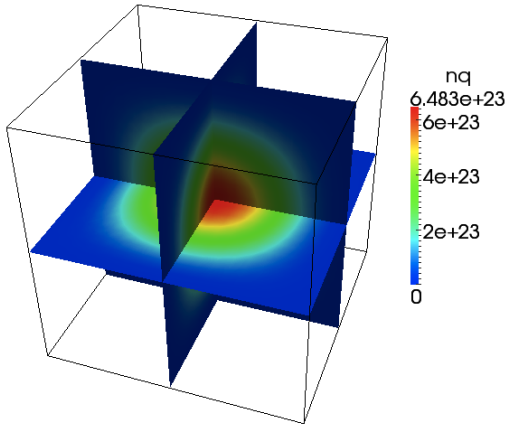
(4) Check convergence and restart

- Nested iteration requires recomputing matrices MANY times

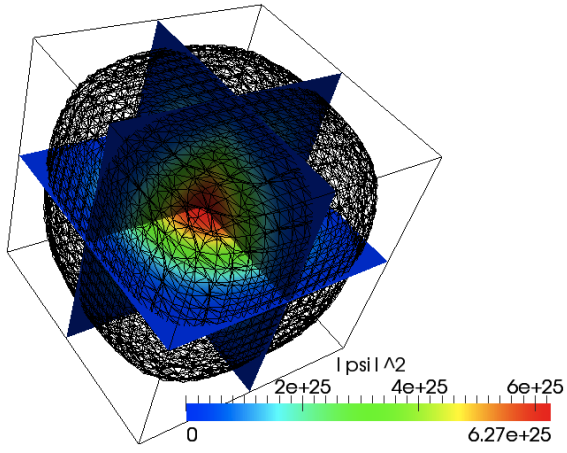
Implementation and Numerical results



(a)



(b)



(c)

- Cadmium Sulfide nanoparticle

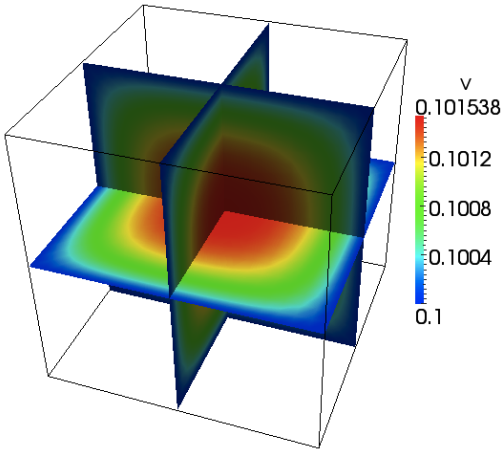
(a) Electric potential

(b) Quantum-corrected charge density

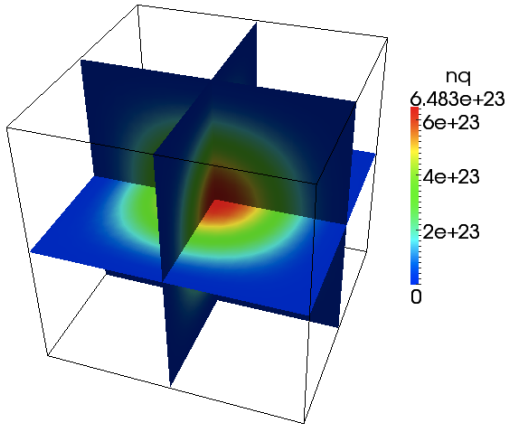
(c) Wave functions

- Using the precomputed space structure of `GeoPDEs` reduces computation time by orders of magnitude
- 3D results from [Ciucci, de Falco et. al. 2011]

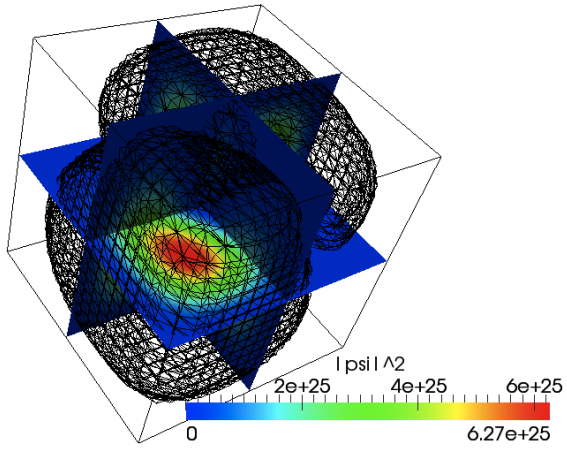
Implementation and Numerical results



(a)



(b)



(c)

- Cadmium Sulfide nanoparticle

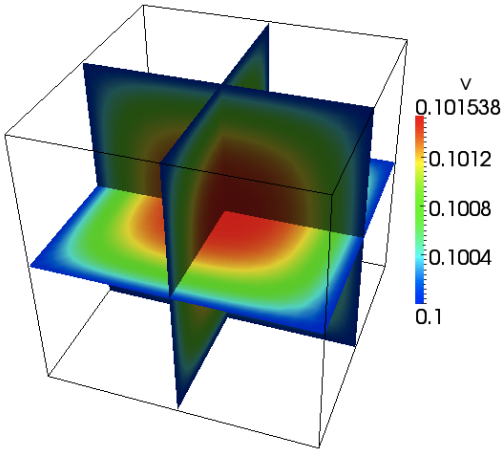
(a) Electric potential

(b) Quantum-corrected charge density

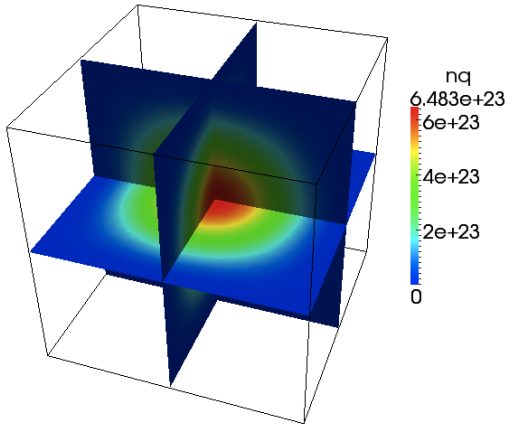
(c) Wave functions

- Using the precomputed space structure of `GeoPDEs` reduces computation time by orders of magnitude
- 3D results from [Ciucci, de Falco et. al. 2011]

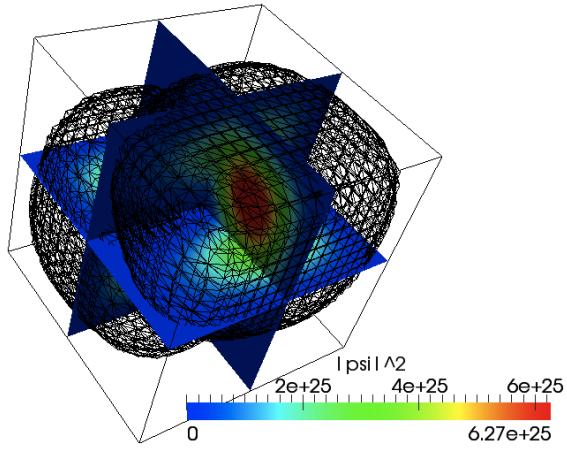
Implementation and Numerical results



(a)



(b)



(c)

- Cadmium Sulfide nanoparticle

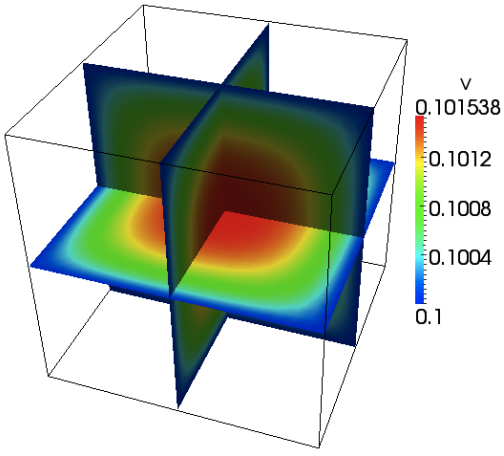
(a) Electric potential

(b) Quantum-corrected charge density

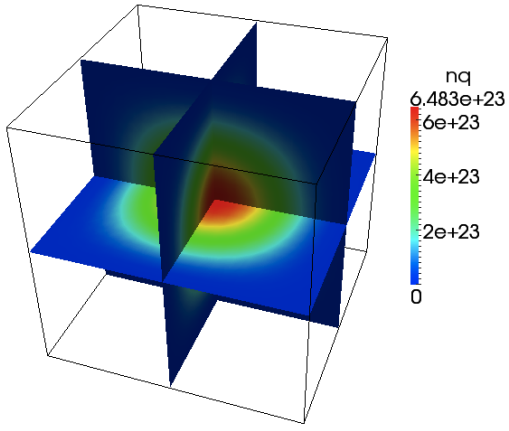
(c) Wave functions

- Using the precomputed space structure of `GeoPDEs` reduces computation time by orders of magnitude
- 3D results from [Ciucci, de Falco et. al. 2011]

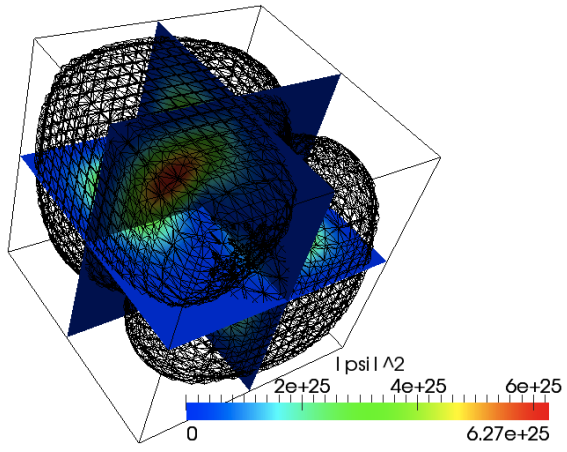
Implementation and Numerical results



(a)



(b)



(c)

- Cadmium Sulfide nanoparticle

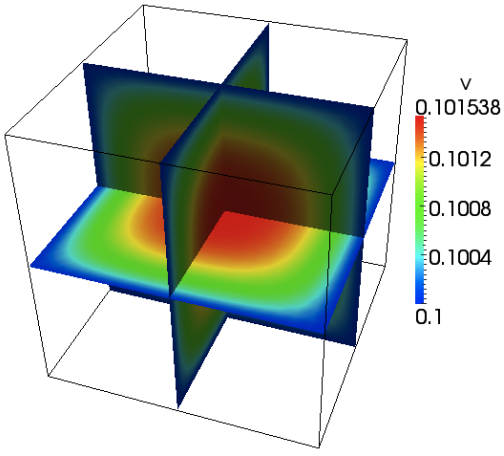
(a) Electric potential

(b) Quantum-corrected charge density

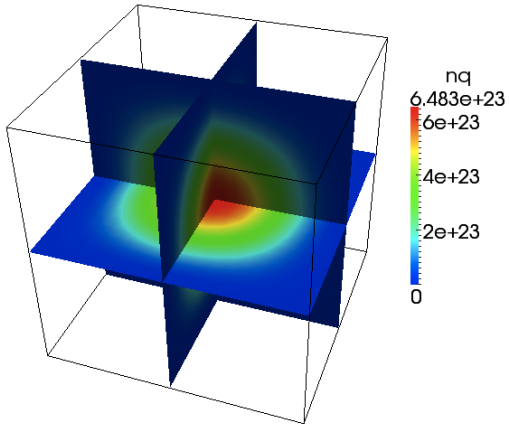
(c) Wave functions

- Using the precomputed space structure of `GeoPDEs` reduces computation time by orders of magnitude
- 3D results from [Ciucci, de Falco et. al. 2011]

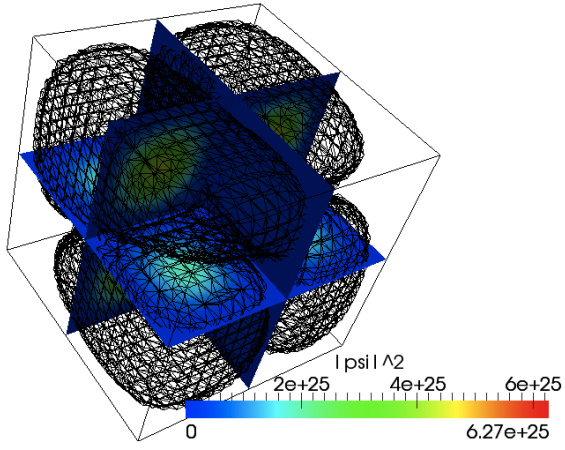
Implementation and Numerical results



(a)



(b)



(c)

- Cadmium Sulfide nanoparticle

(a) Electric potential

(b) Quantum-corrected charge density

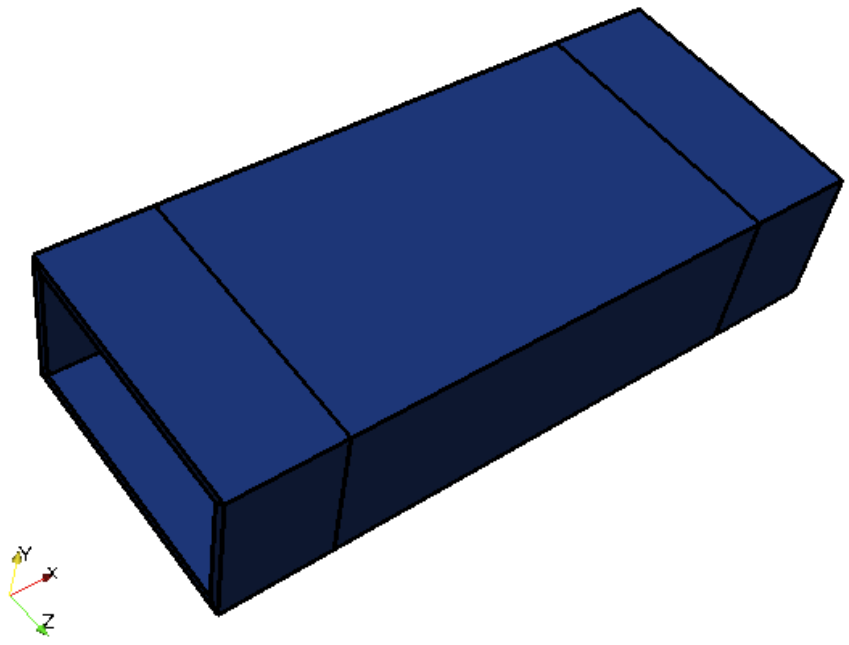
(c) Wave functions

- Using the precomputed space structure of `GeoPDEs` reduces computation time by orders of magnitude
- 3D results from [Ciucci, de Falco et. al. 2011]

Physical problem description

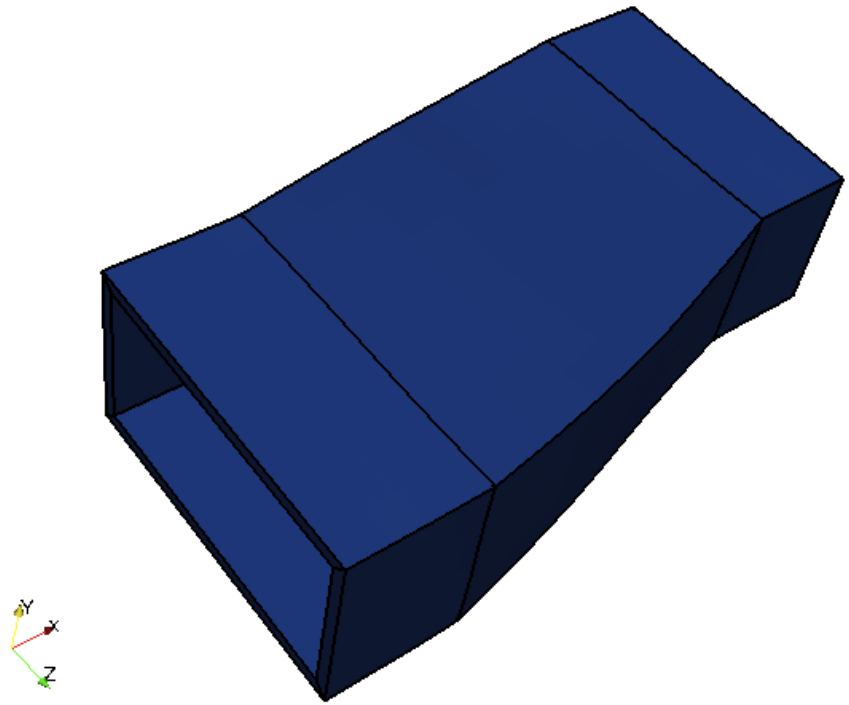
- Consider an infinitely long hollow metallic WR-2300 waveguide
- TE_{10} mode at $\simeq 0.35\,GHz$
- A mechanical deformation of the waveguide produces a reflected wave
- Compute relative amplitudes of the reflected and transmitted waves

Step 1: Shell deformation



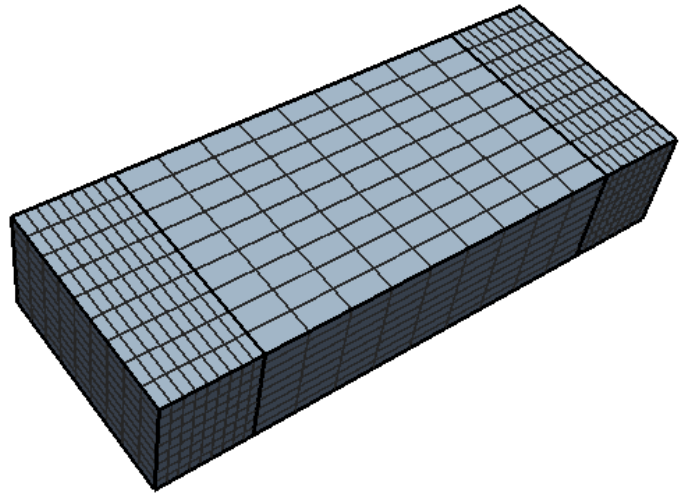
- multi-patch B-Spline model of the straight waveguide shell + interior
- extract patches that represent the exterior shell
- displaced geometry by solving linear elasticity problem

Step 1: Shell deformation



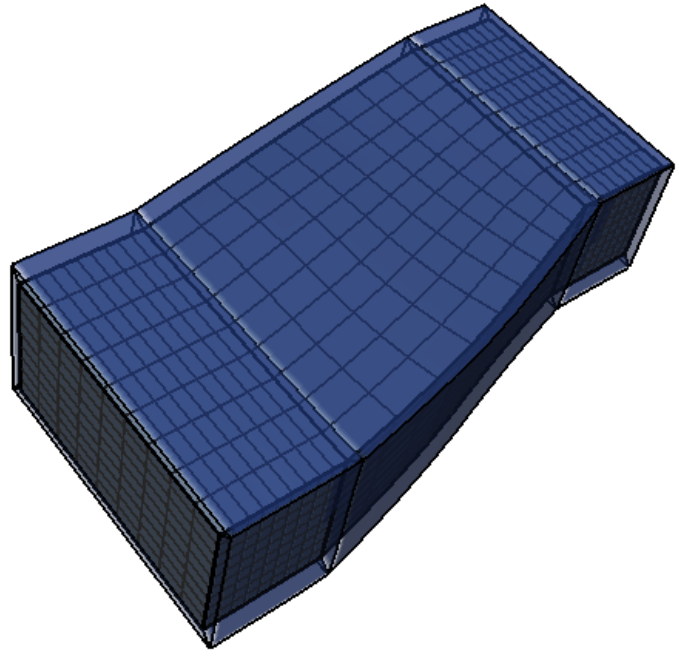
- multi-patch B-Spline model of the straight waveguide shell + interior
- extract patches that represent the exterior shell
- displaced geometry by solving linear elasticity problem

Step 2: Parametrization of the interior



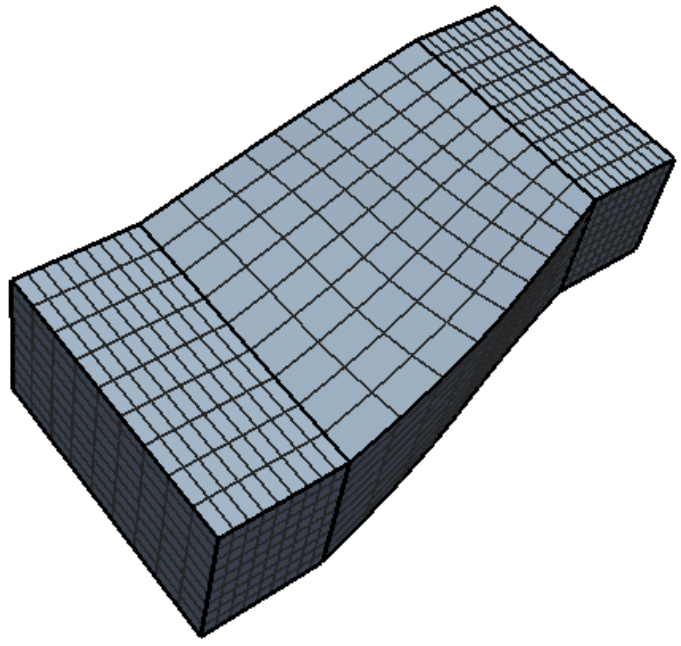
- we need a parametrization of the interior of the deformed waveguide
- extract the patches that represent the interior

Step 2: Parametrization of the interior



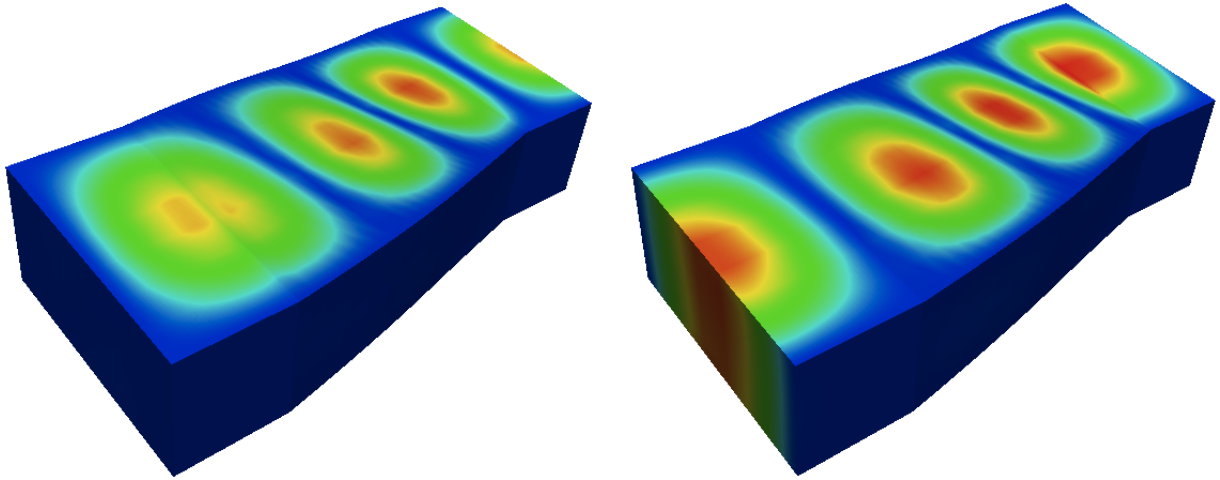
- we need a parametrization of the interior of the deformed waveguide
- extract the patches that represent the interior
- "copy" the boundary dof value from the shell

Step 2: Parametrization of the interior



- we need a parametrization of the interior of the deformed waveguide
- extract the patches that represent the interior
- "copy" the boundary dof value from the shell

Step 3: Computing Electric field in the waveguide



(d) real part (e) imaginary part

$$\mathbf{curl}\left(\frac{1}{\mu}\mathbf{curl}\,E\right) - \omega^2 \varepsilon E = 0$$

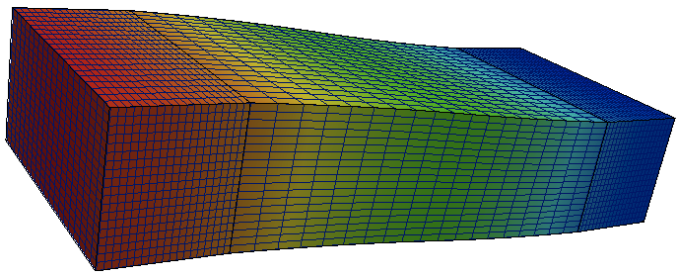
$E \times n = 0$ on Γ_D

forward + reverse(reflected) traveling waves at Γ_{in}

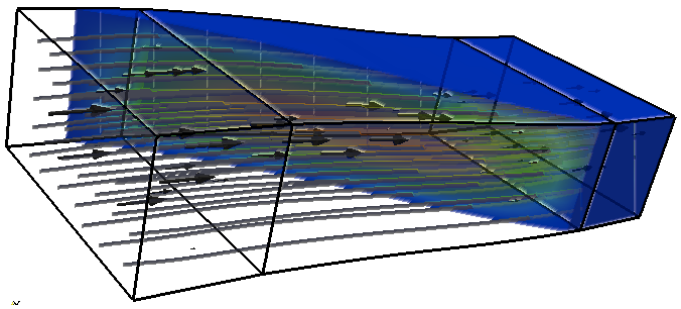
forward wave only at Γ_{out}

B-Spline discretization [Buffa, Sangalli, Vazquez 2010]

Step 3: Computing fluid velocity and pressure in the "pipe"



(f) pressure



(g) velocity

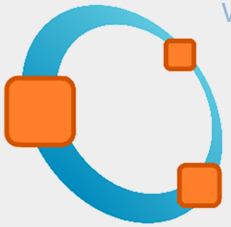
Testing Discretization Schemes

- Incompressible fluids
- Extension: Higher-order operators

More complex applications

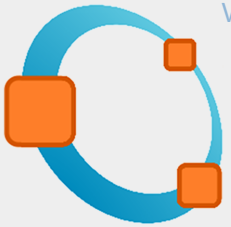
- A Non-linear Problem: Schrödinger-Poisson
- A "Multiphysics" Problem: Twisted Wave-Guide

Conclusions



OctConf 2013

- MOX - Politecnico di Milano 24-26 June 2013
- Register now!



How to get Octave

- Source on www.octave.org
- Binaries on octave.sourceforge.net
- Through package managers
- In NCLab
- I'll be available to help with installation.