A polihedral 3–dimensional mesher.

An algorithm to approximate solutions to singular physical problems

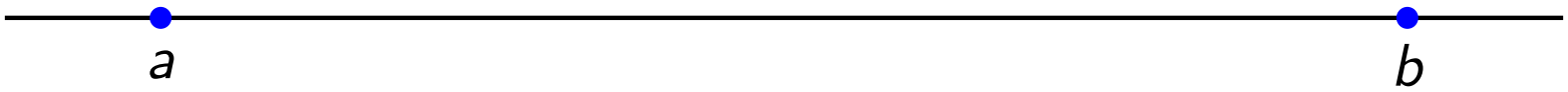
Alexis Jawtuschenko

Departamento de Matemática, FCEN
Universidad de Buenos Aires.
Departamento de Matemática y Ciencias,
Universidad de San Andrés.

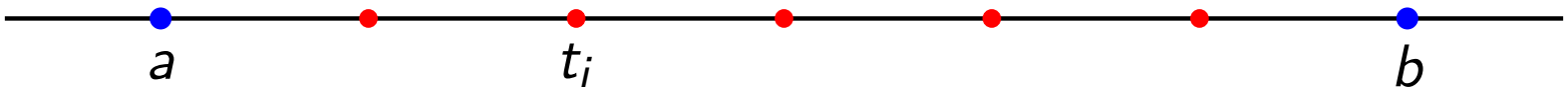
PyCon Brazil 2019

What is a mesher?

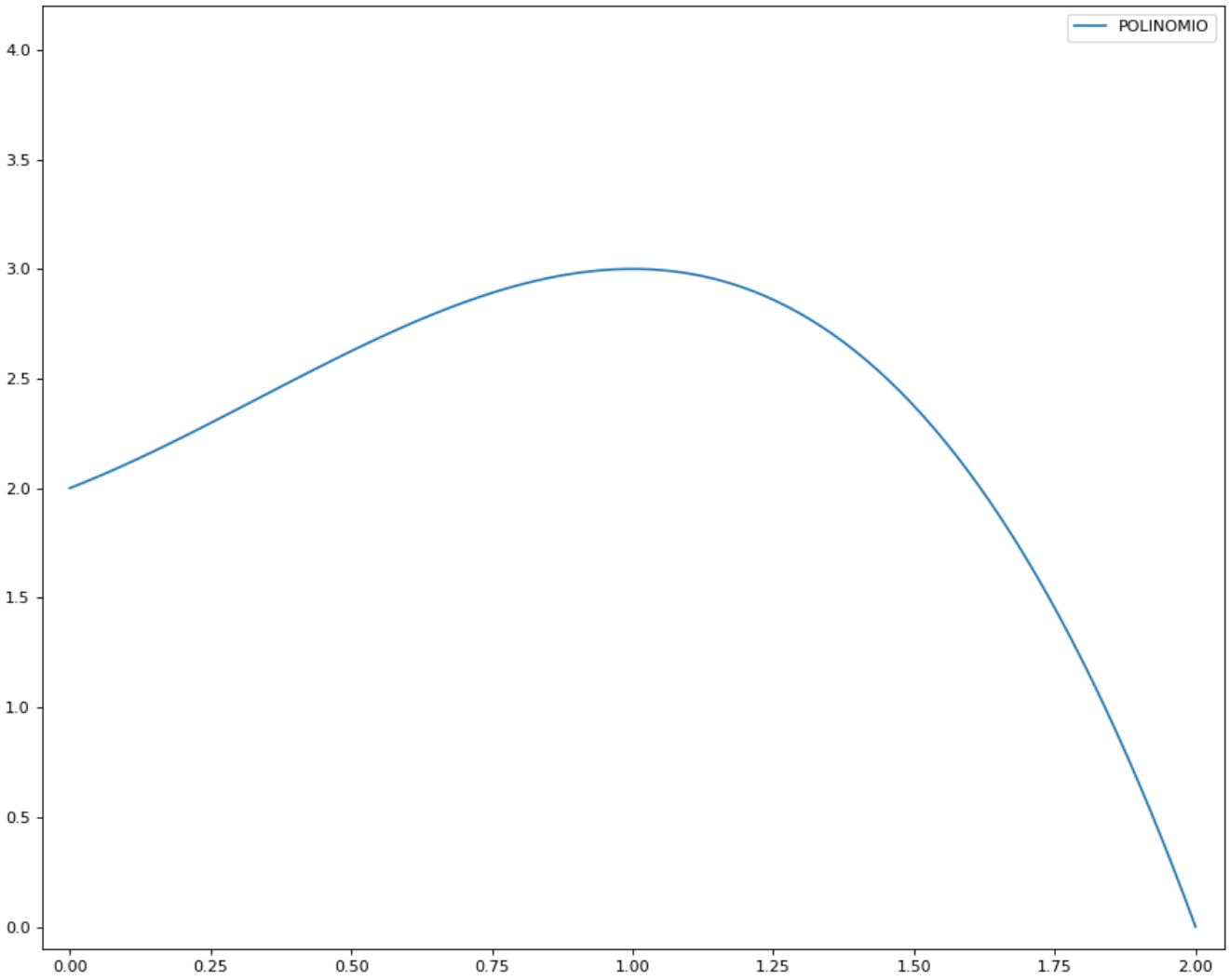
1D
intervals in the real line



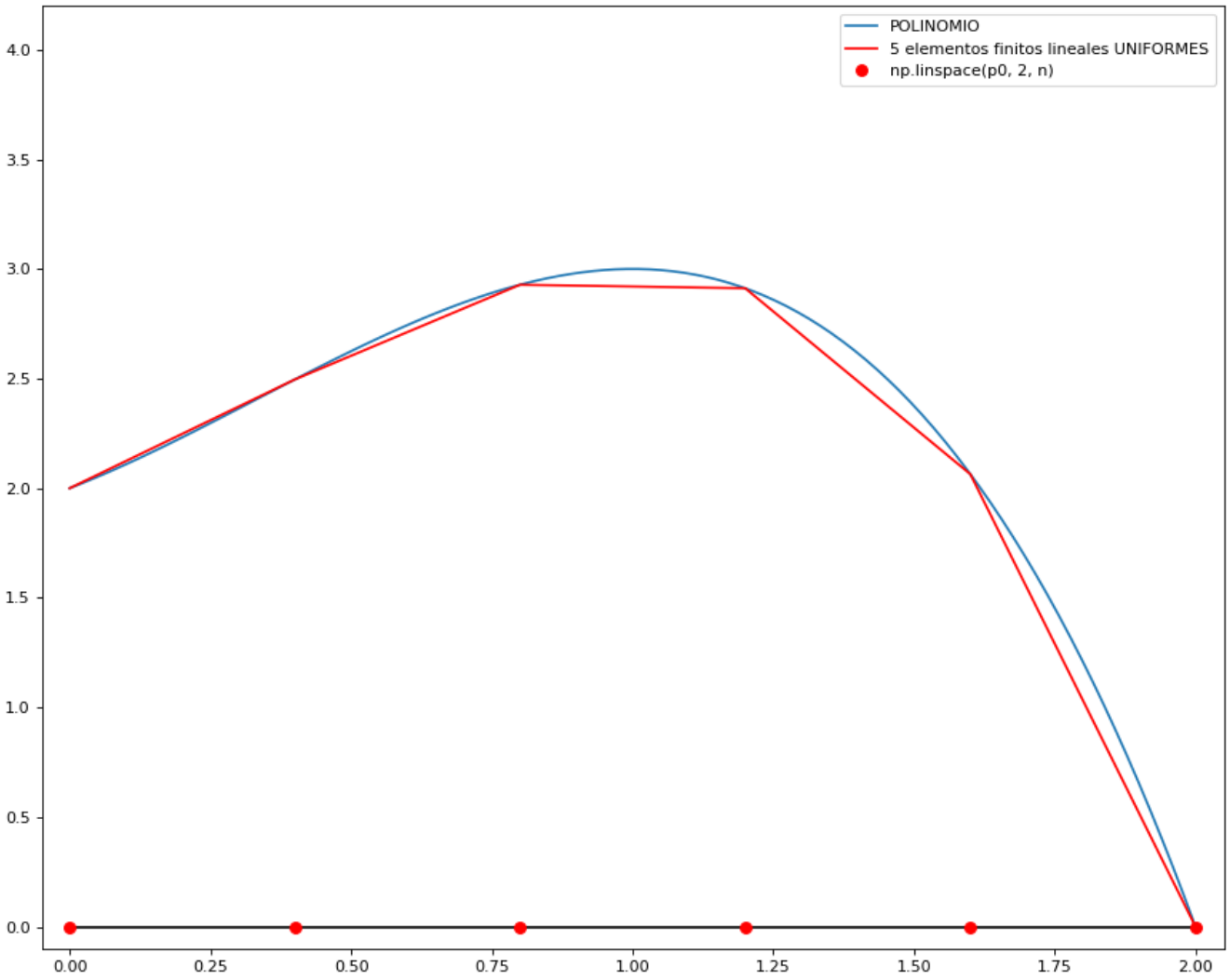
1D
intervals in the real line



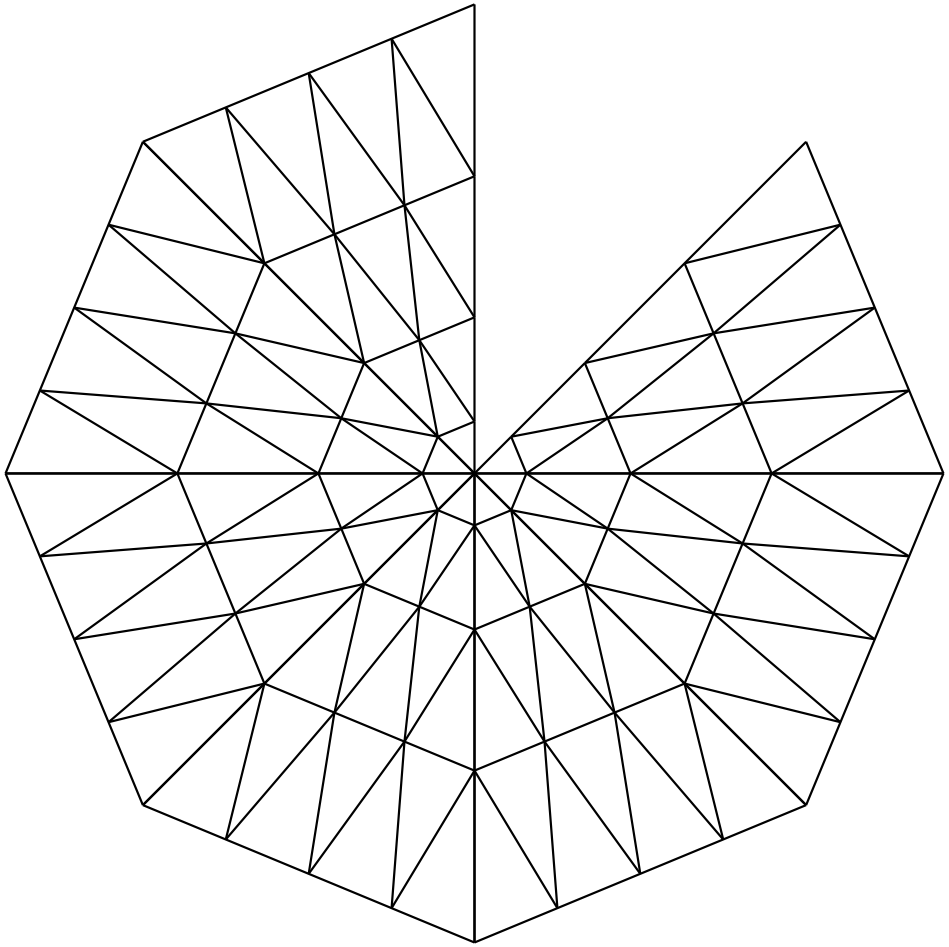
1D
intervals in the real line



1D
intervals in the real line

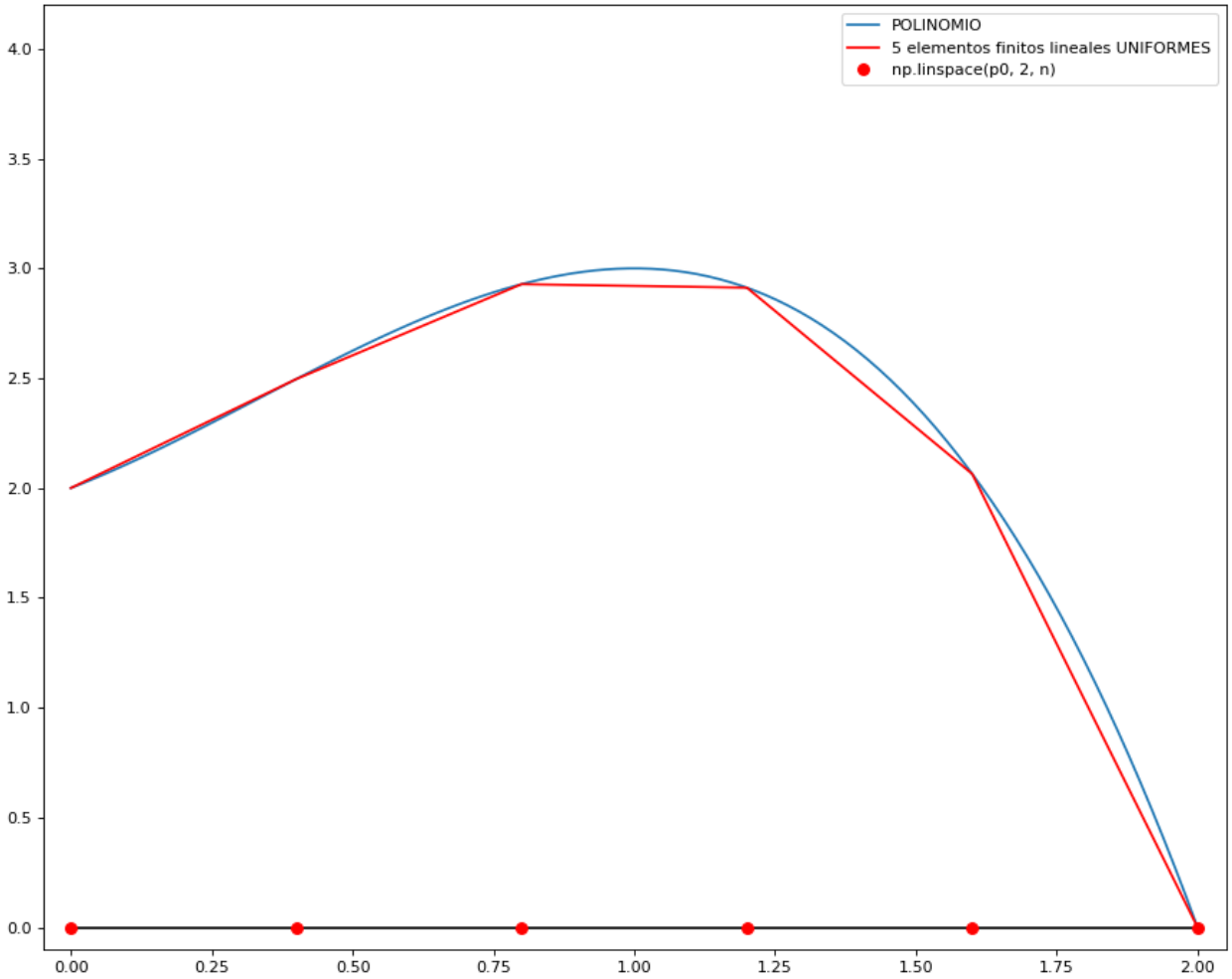


2D
polygons in the real plane

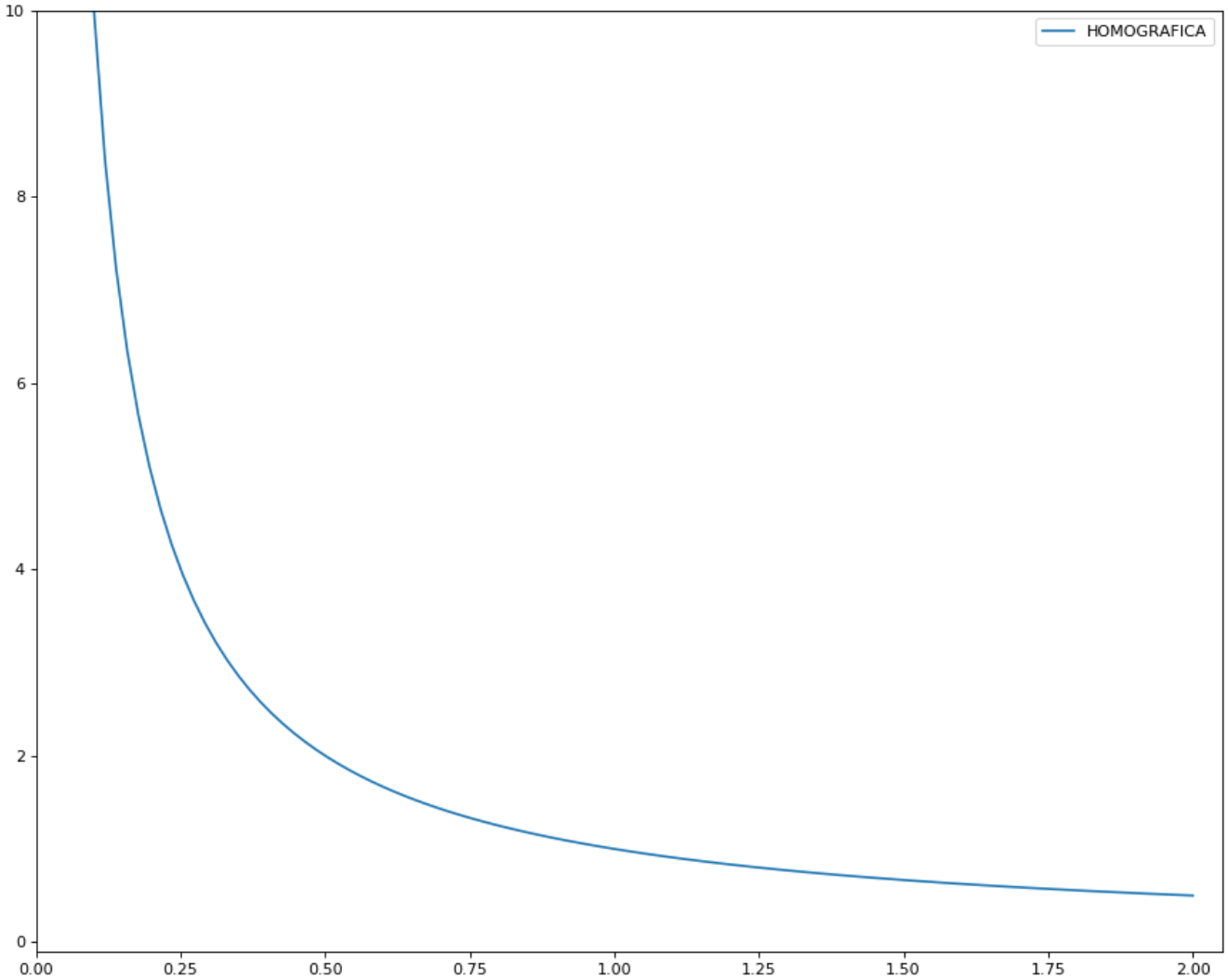


What do You mean with singular?

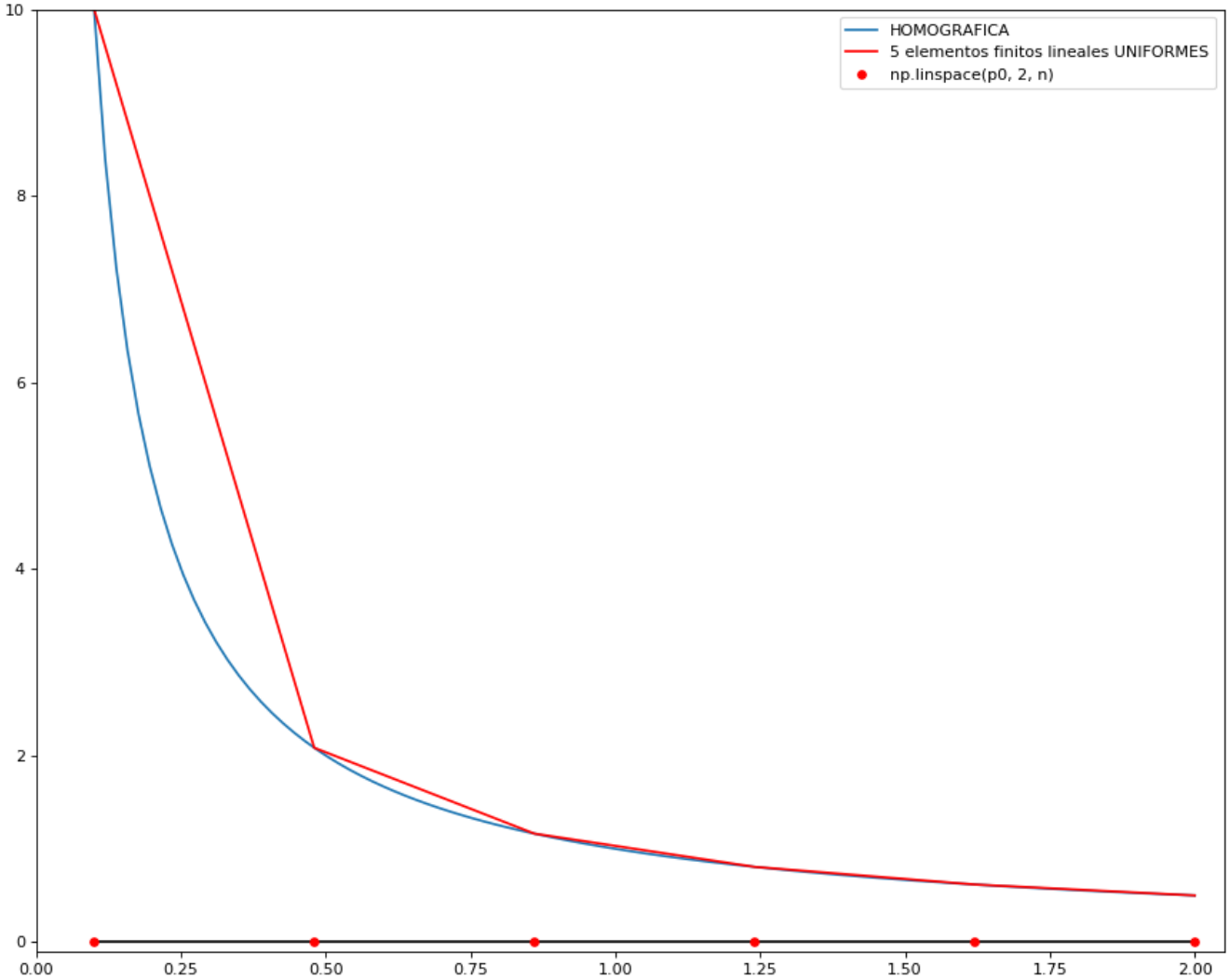
Returning to the regular function



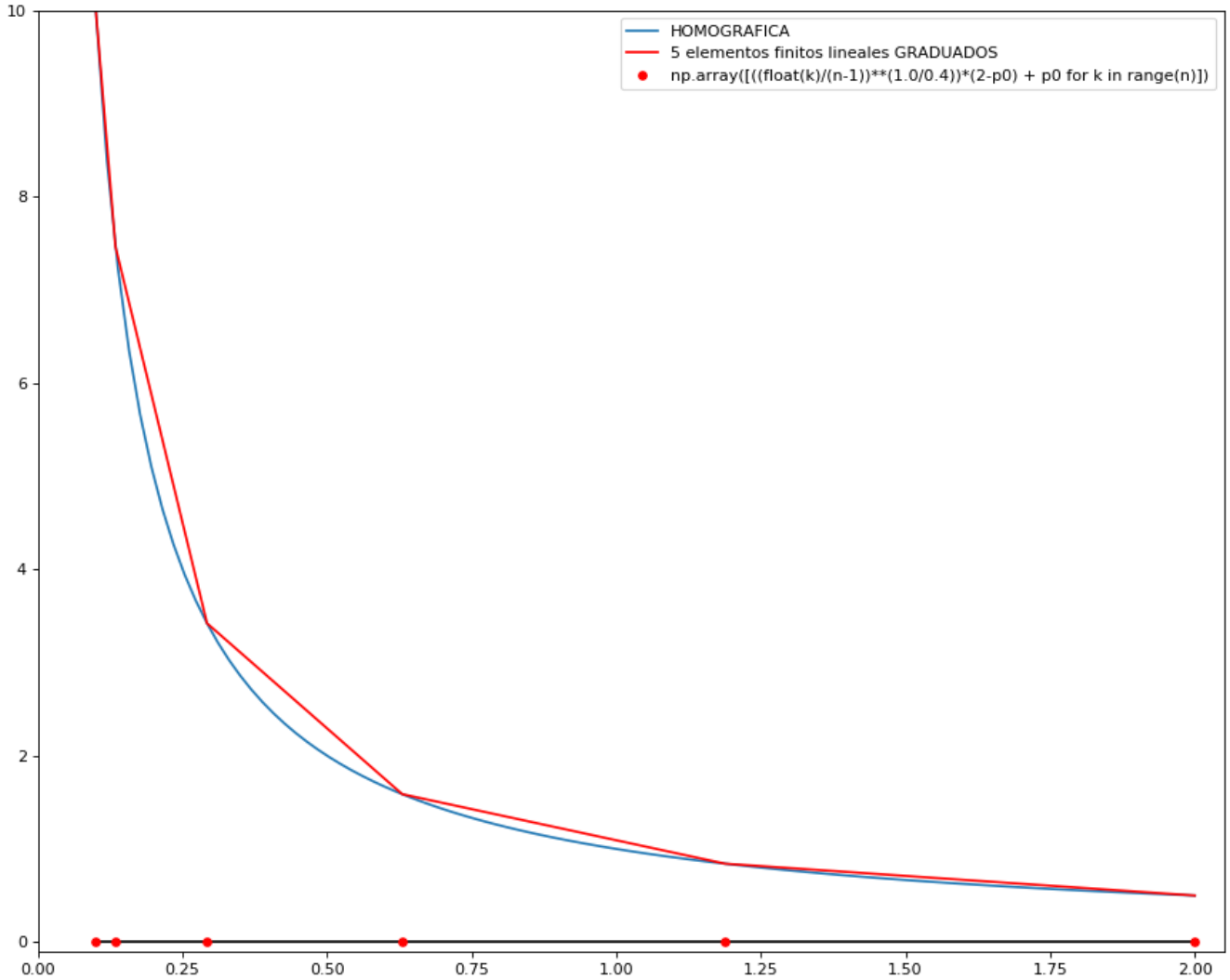
Approx. by pieces
Finite Elements



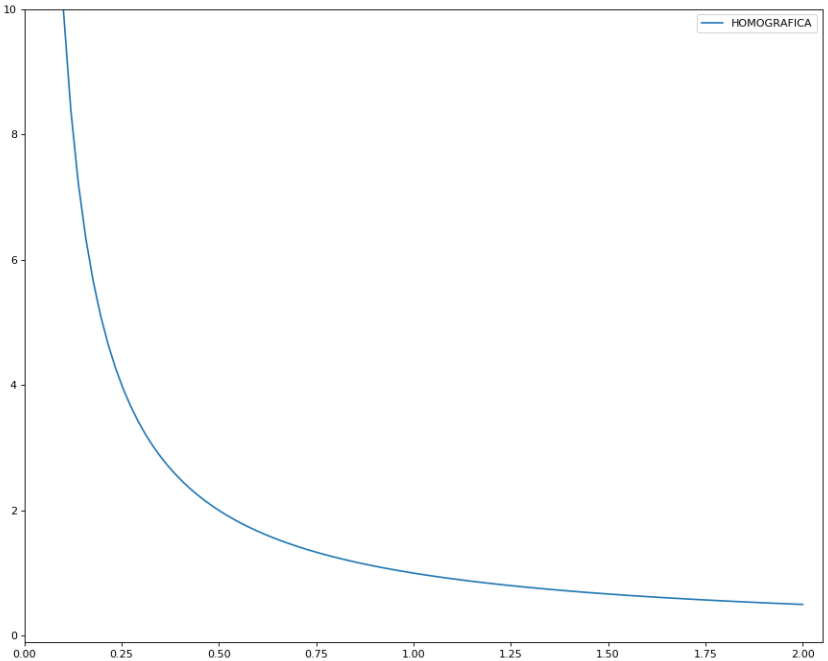
Approx. by pieces
Finite Elements

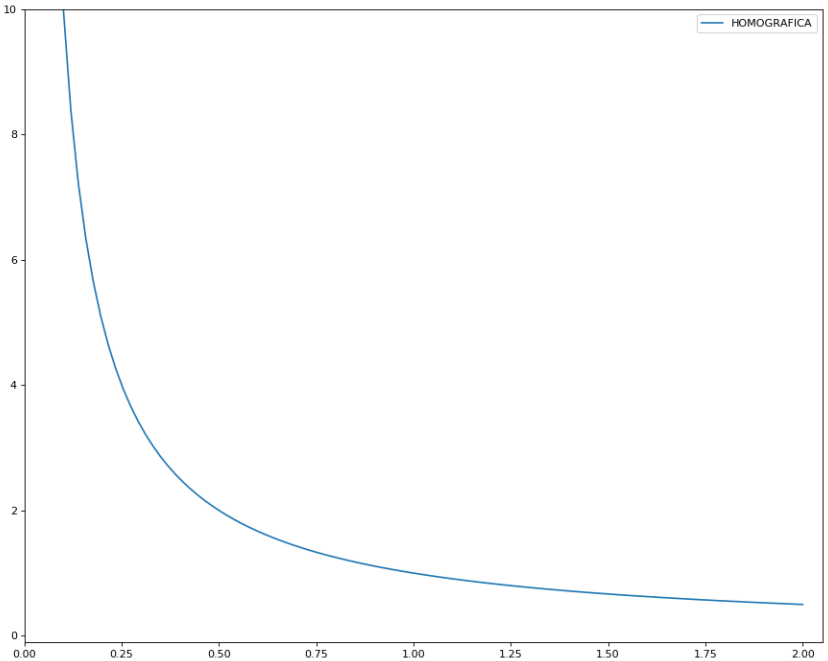


Approx. by pieces
Finite Elements



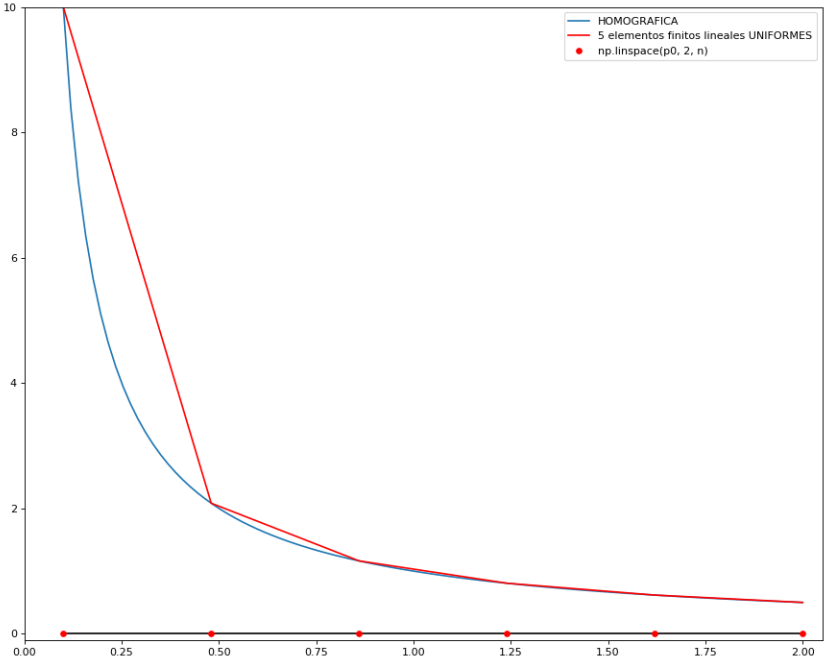
Approx. by pieces
Finite Elements uniform vs. graded *ad hoc*

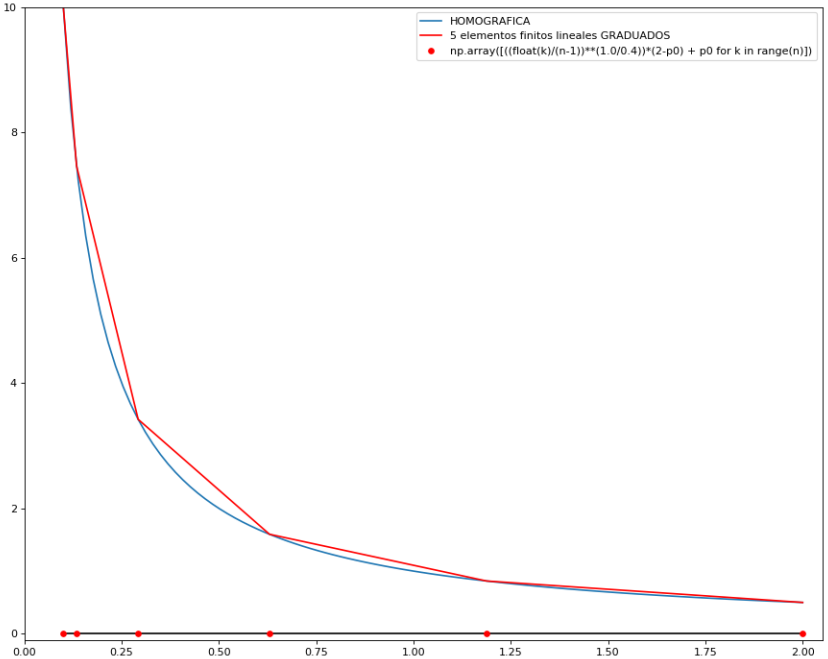




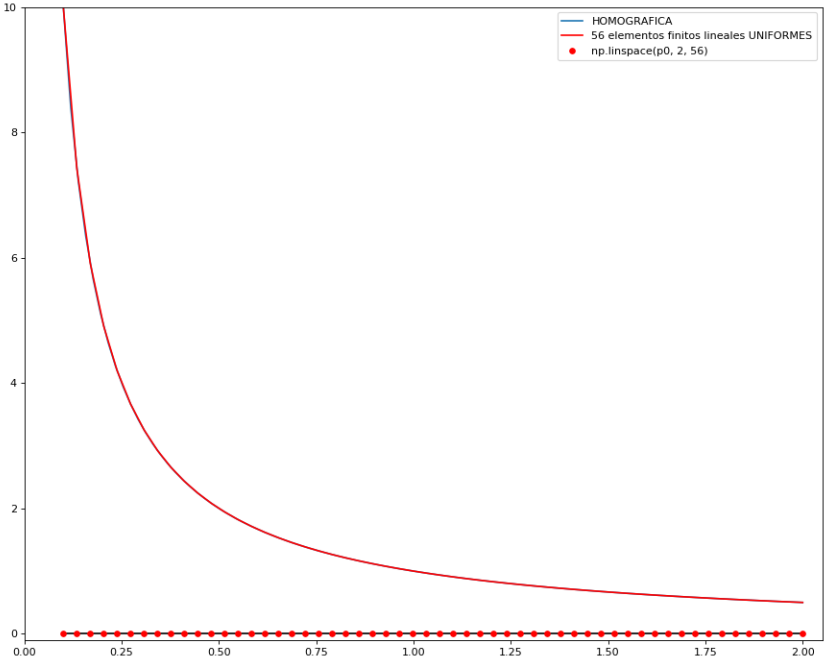
Approx. by pieces

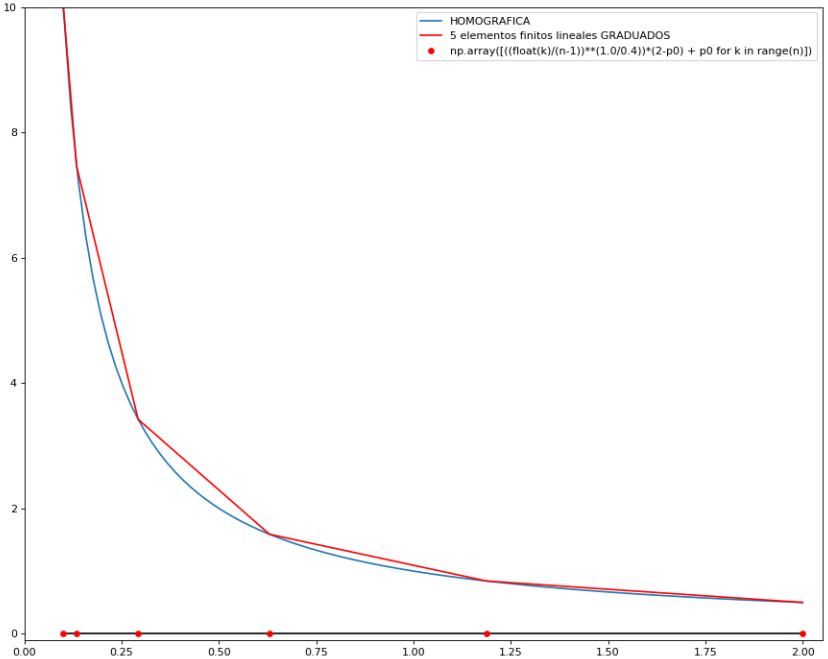
Finite Elements uniform vs. graded *ad hoc*





Approx. by pieces
Finite Elements uniform vs. graded *ad hoc*



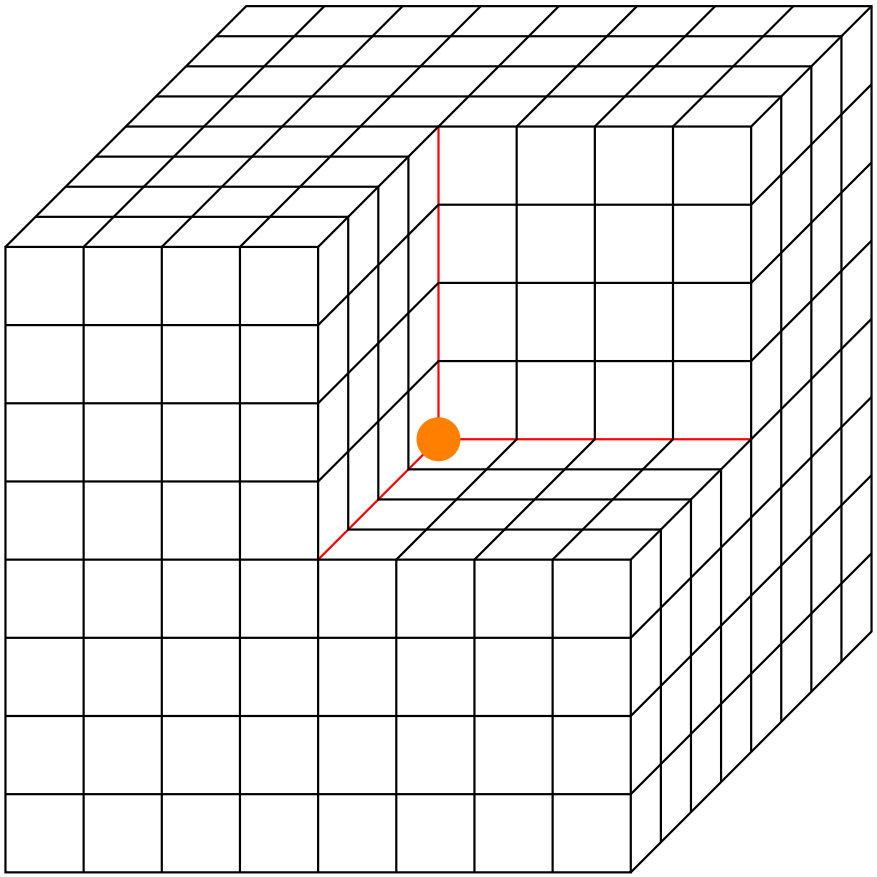


This Talk
3D polyhedra in the space

- 3D mesher (polyhedra in $\mathbb{R}^3$)

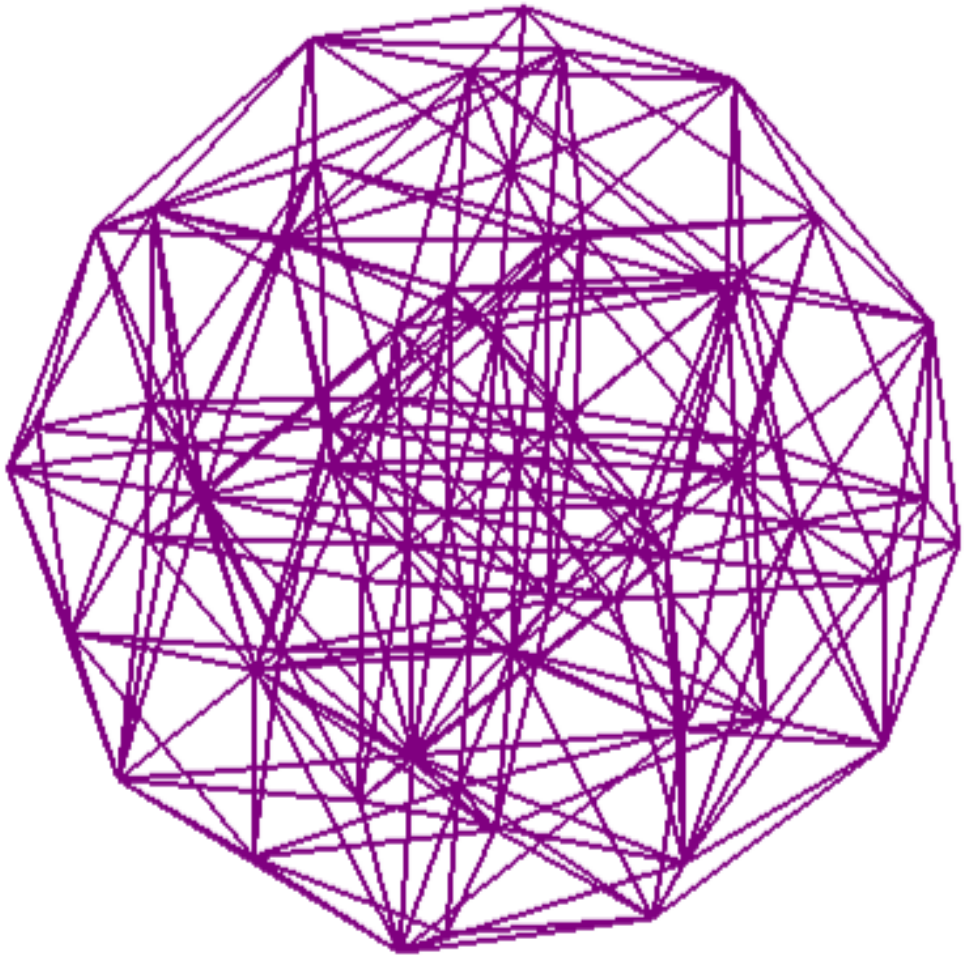
This Talk

3D polyhedra in the space



This Talk

3D polyhedra in the space



Anisotropy
A motivation

- Why did I do it? $\rightsquigarrow$

Anisotropy
A motivation

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$
- $\Omega \subseteq \mathbb{R}^3$ with concave edges or vertices

$$\begin{cases} \mathbf{u} = -\nabla p & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 1 + R(\mathbf{x})^{-3/2} \ln^{-1}(R(\mathbf{x})/4) & \text{in } \Omega \qquad (f \in L^2(\Omega)) \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$
- $\Omega \subseteq \mathbb{R}^3$ with concave edges or vertices

$$\begin{cases} \mathbf{u} = -\nabla p & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 1 + R(\mathbf{x})^{-3/2} \ln^{-1}(R(\mathbf{x})/4) & \text{in } \Omega \qquad (f \in L^2(\Omega)) \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

- $\mathbf{u}$ has singularities

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$
- $\Omega \subseteq \mathbb{R}^3$ with concave edges or vertices

$$\begin{cases} \mathbf{u} = -\nabla p & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 1 + R(\mathbf{x})^{-3/2} \ln^{-1}(R(\mathbf{x})/4) & \text{in } \Omega \qquad (f \in L^2(\Omega)) \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

- $\mathbf{u}$ has singularities
- that depend on the distance to concave edges or vertices.

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$
- $\Omega \subseteq \mathbb{R}^3$ with concave edges or vertices

$$\begin{cases} \mathbf{u} = -\nabla p & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 1 + R(\mathbf{x})^{-3/2} \ln^{-1}(R(\mathbf{x})/4) & \text{in } \Omega \qquad (f \in L^2(\Omega)) \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

- $\mathbf{u}$ has singularities
- that depend on the distance to concave edges or vertices.
- Numerical Methods with uniform meshes don't converge with optimal order

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$
- $\Omega \subseteq \mathbb{R}^3$ with concave edges or vertices

$$\begin{cases} \mathbf{u} = -\nabla p & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 1 + R(\mathbf{x})^{-3/2} \ln^{-1}(R(\mathbf{x})/4) & \text{in } \Omega \qquad (f \in L^2(\Omega)) \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

- $\mathbf{u}$ has singularities
- that depend on the distance to concave edges or vertices.
- Numerical Methods with uniform meshes don't converge with optimal order
- mesh has to be accordingly refined in order to recover optimal order w.r.t. number of dofs.

- Why did I do it? $\rightsquigarrow$
- Poisson equation in a polyhedron $\Omega \subseteq \mathbb{R}^3$
- $\Omega \subseteq \mathbb{R}^3$ with concave edges or vertices

$$\begin{cases} \mathbf{u} = -\nabla p & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 1 + R(\mathbf{x})^{-3/2} \ln^{-1}(R(\mathbf{x})/4) & \text{in } \Omega \qquad (f \in L^2(\Omega)) \\ p = 0 & \text{on } \partial\Omega. \end{cases}$$

- $\mathbf{u}$ has singularities
- that depend on the distance to concave edges or vertices.
- Numerical Methods with uniform meshes don't converge with optimal order
- mesh has to be accordingly refined in order to recover optimal order w.r.t. number of dofs.
- meshes have to be anisotropic.

And what is anisotropy?

- Anisotropic physical object:

And what is anisotropy?

- Anisotropic physical object:
  - has physical properties with different magnitudes in independent directions.

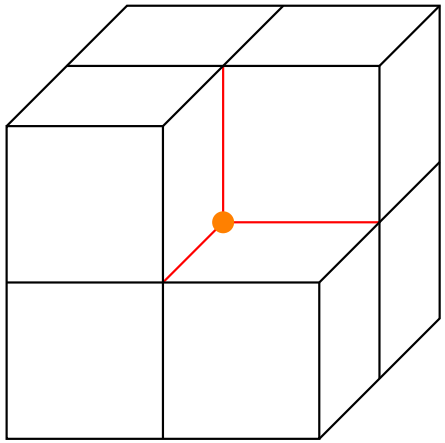
And what is anisotropy?

- Anisotropic physical object:
  - has physical properties with different magnitudes in independent directions.
- In numerical methods, anisotropic family of meshes:

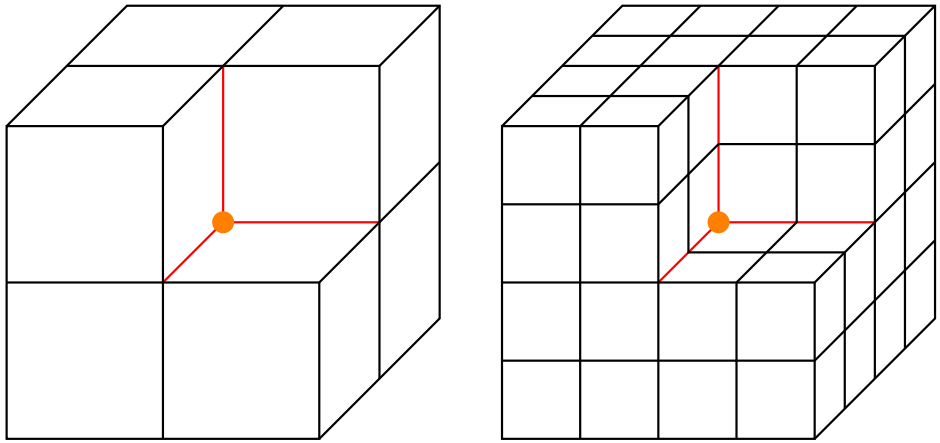
And what is anisotropy?

- Anisotropic physical object:
  - has physical properties with different magnitudes in independent directions.
- In numerical methods, anisotropic family of meshes:
  - contains sequences of elements with sizes decreasing more rapidly along one direction than in the others.

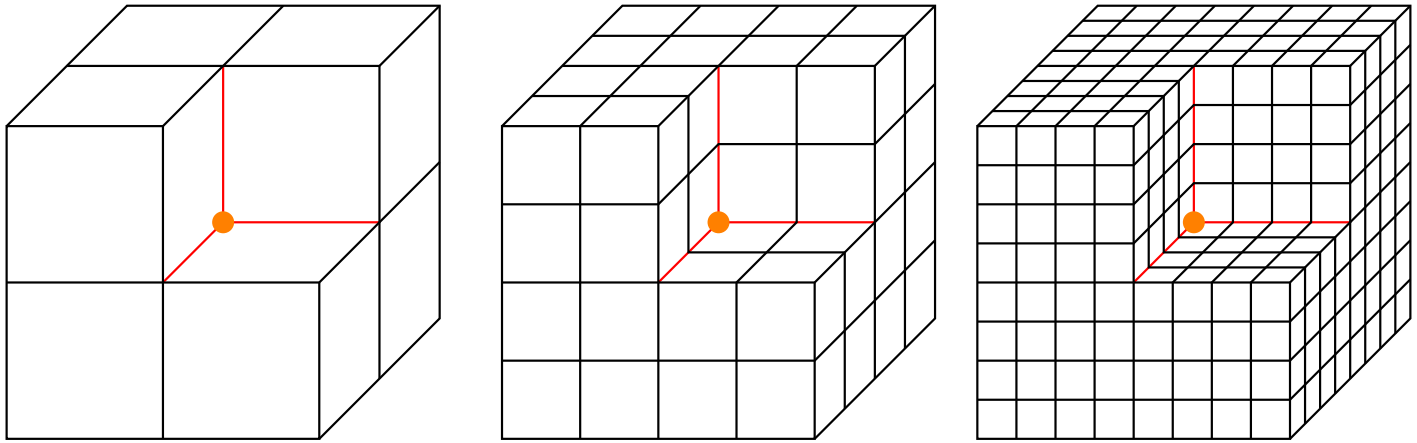
quasi–uniform vs. anisotropic



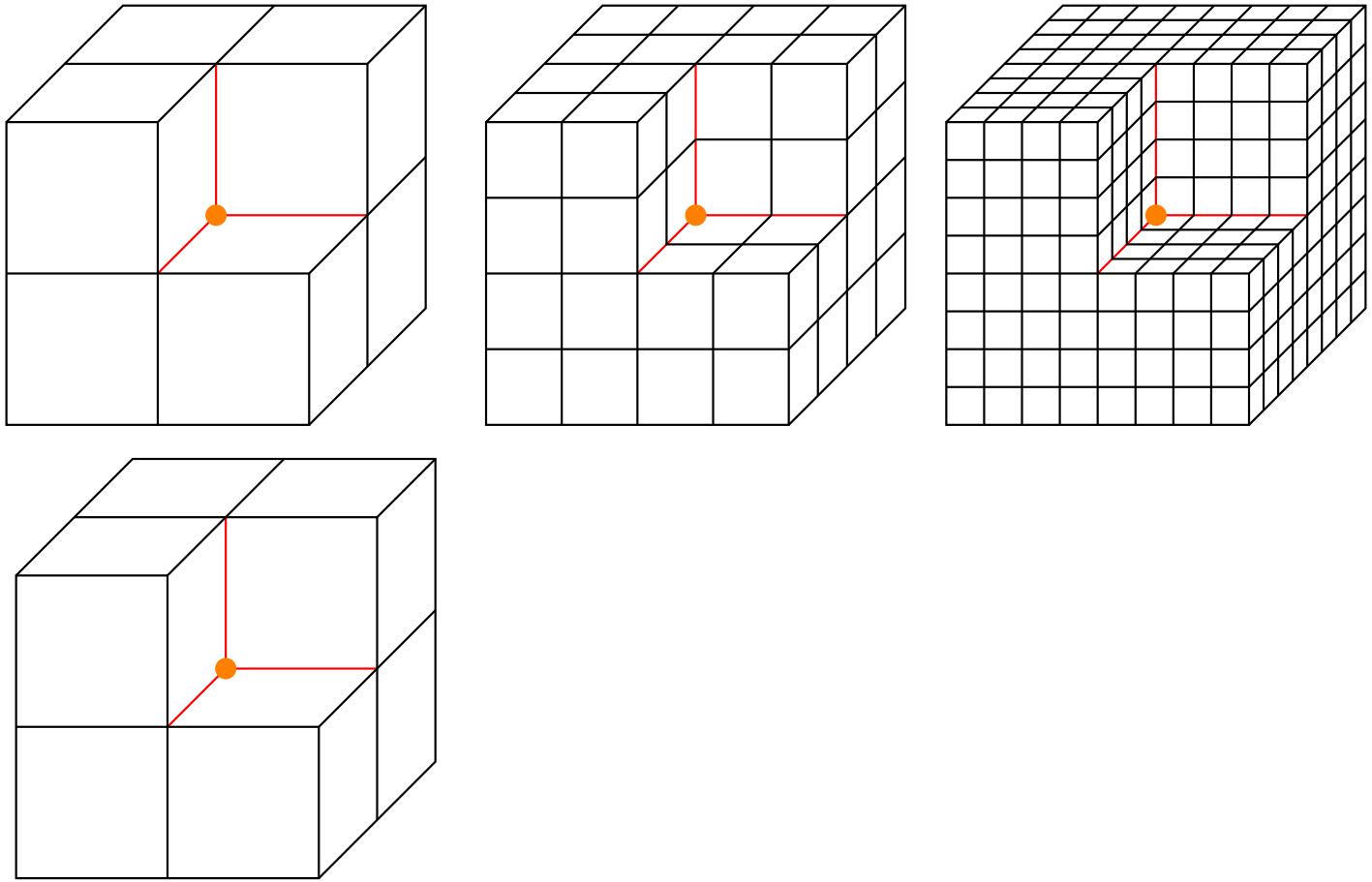
quasi–uniform vs. anisotropic



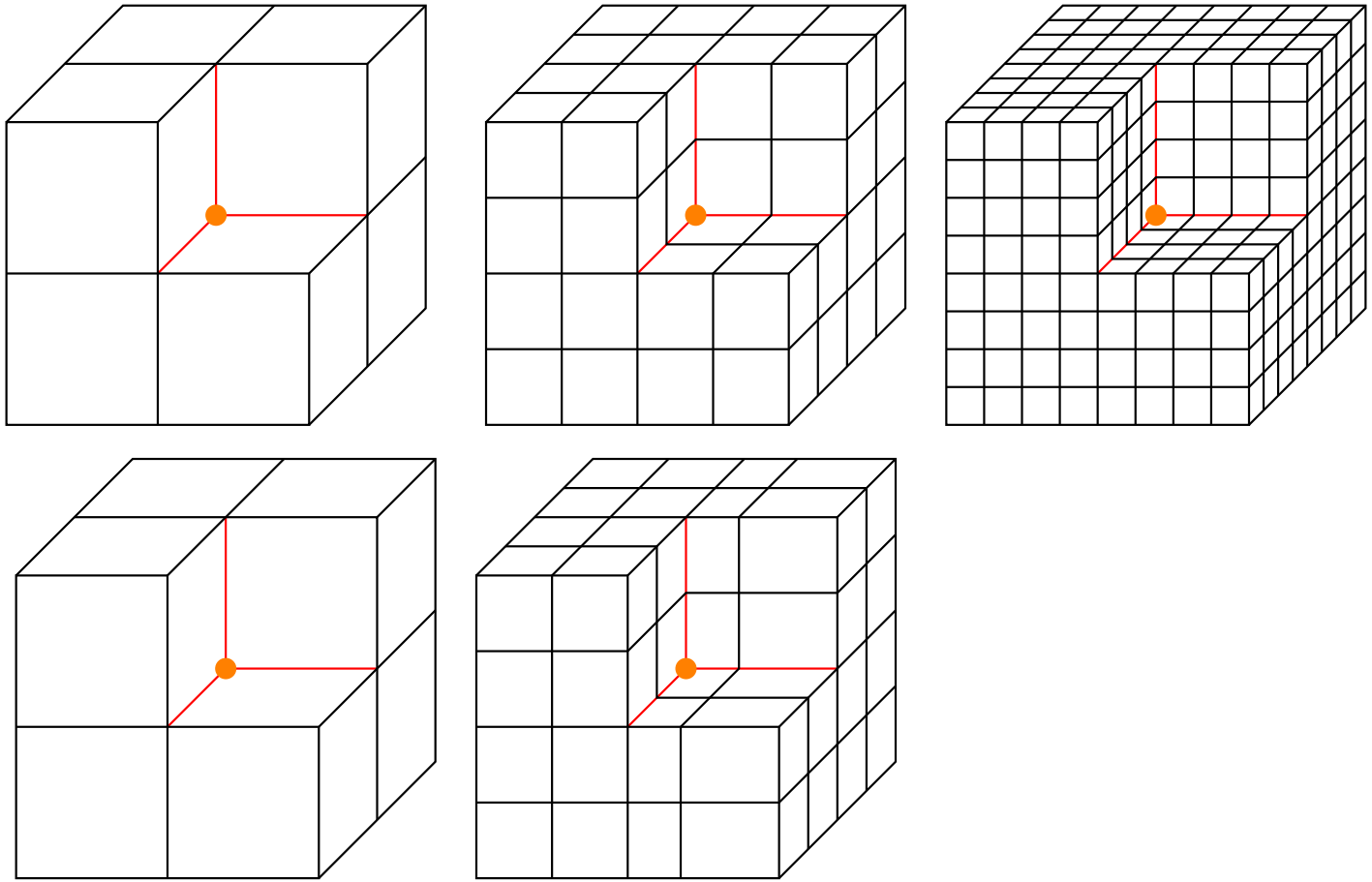
quasi–uniform vs. anisotropic



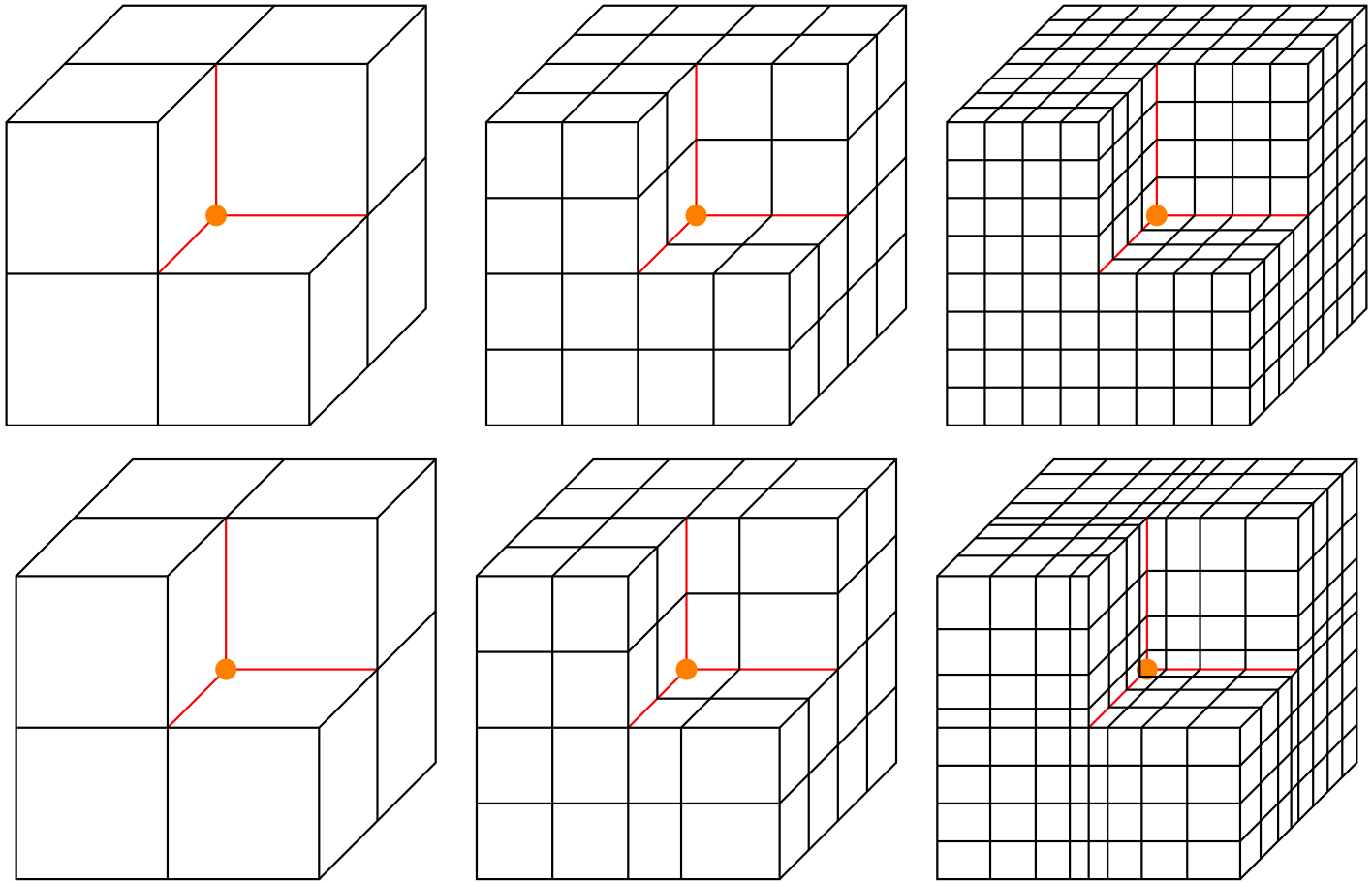
quasi–uniform vs. anisotropic



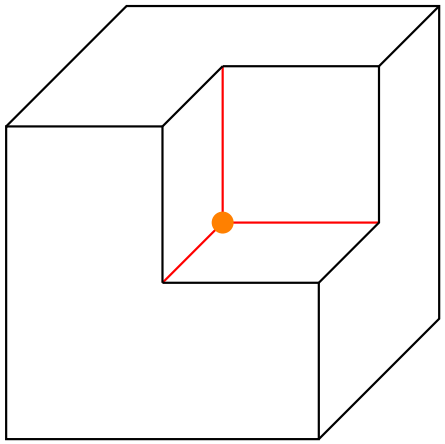
quasi–uniform vs. anisotropic



quasi–uniform vs. anisotropic

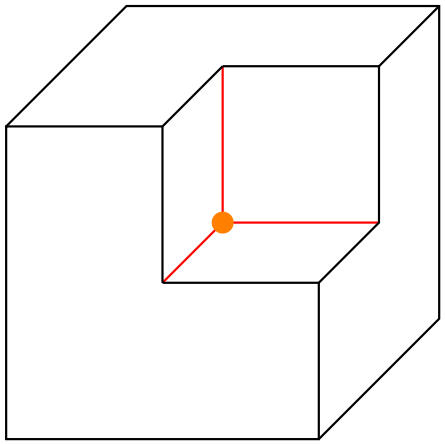


Two particular domains in my initial research

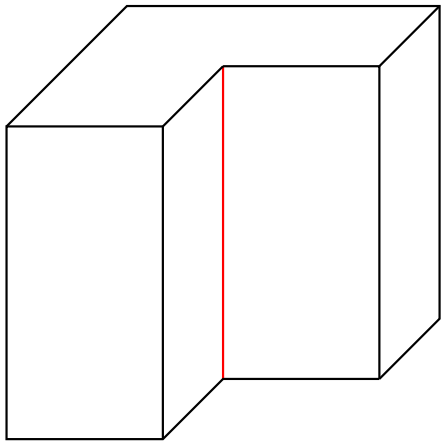


edges and vertices

Two particular domains in my initial research



edges and vertices



edges

Strategy of the proposed method

- $\bar{\Omega} = \cup_{\ell=1}^{M} \bar{\Lambda}_\ell$ (the macro–elements)

Strategy of the proposed method

- $\bar{\Omega} = \cup_{\ell=1}^{M} \bar{\Lambda}_\ell$ (the macro–elements)
  - prisms or tetrahedra

Strategy of the proposed method

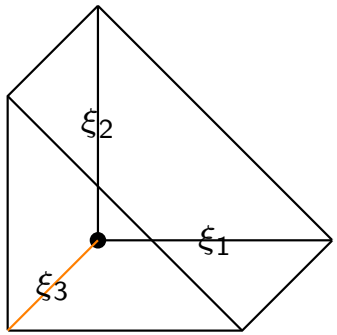
- $\bar{\Omega} = \cup_{\ell=1}^{M} \bar{\Lambda}_\ell$ (the macro–elements)
  - prisms or tetrahedra
  - just one concave edge or vertex each

Strategy of the proposed method

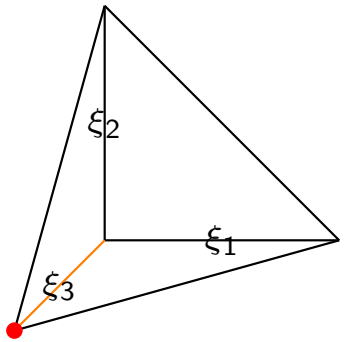
- $\bar{\Omega} = \cup_{\ell=1}^{M} \bar{\Lambda}_\ell$ (the macro–elements)
  - prisms or tetrahedra
  - just one concave edge or vertex each
  - that is, we isolate the singularities

Strategy of the proposed method

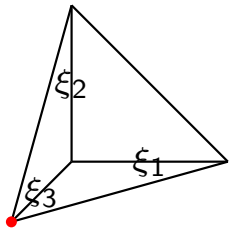
- $\bar{\Omega} = \cup_{\ell=1}^{M} \bar{\Lambda}_\ell$ (the macro–elements)
  - prisms or tetrahedra
  - just one concave edge or vertex each
  - that is, we isolate the singularities



(a) Prismatic Macroelement.



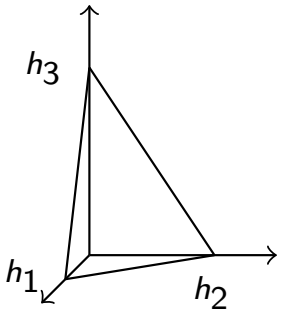
(b) Tetrahedral Macroelement.



(c) Tetrahedral Macroelement II.

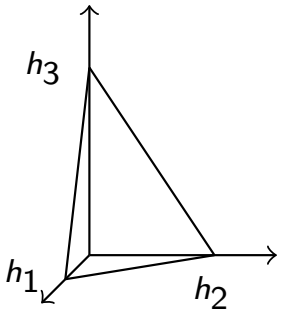
nested Sub–meshes

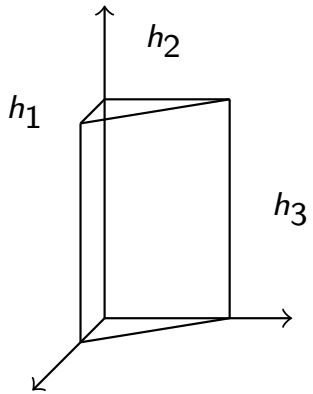
within each one of the three types of macro–elements:



nested Sub–meshes

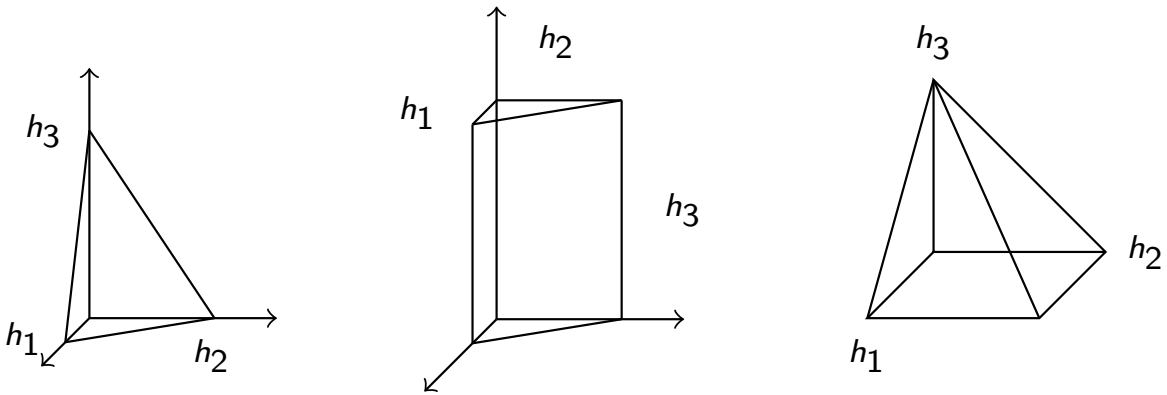
within each one of the three types of macro–elements:





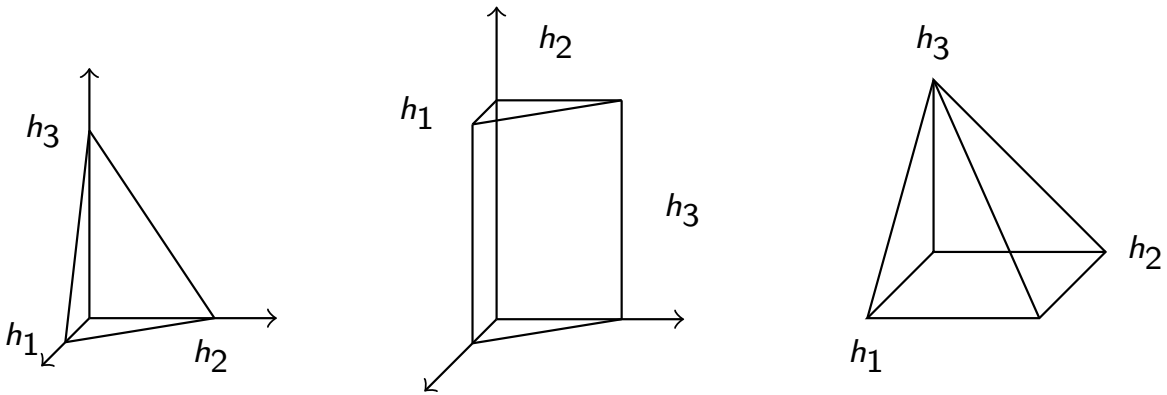
nested Sub–meshes

within each one of the three types of macro–elements:



nested Sub–meshes

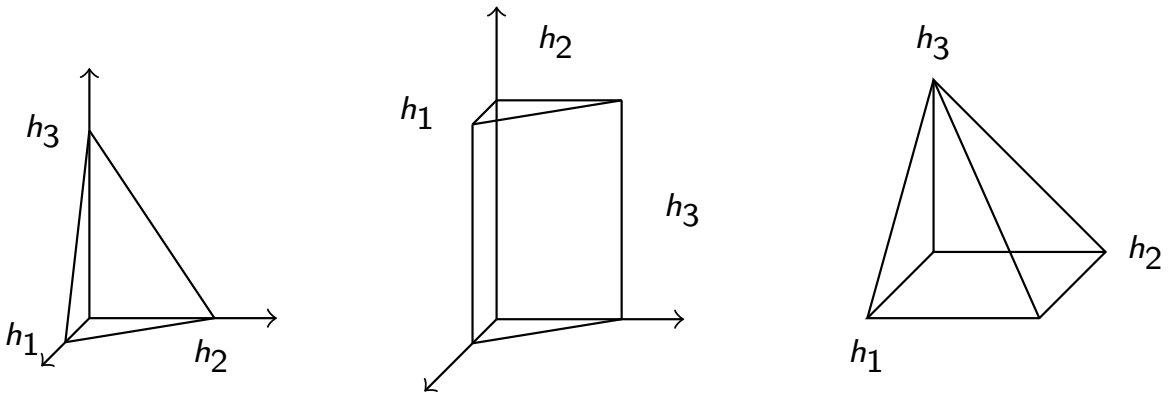
within each one of the three types of macro–elements:



tetrahedra, prisms y pyramids.

nested Sub–meshes

within each one of the three types of macro–elements:



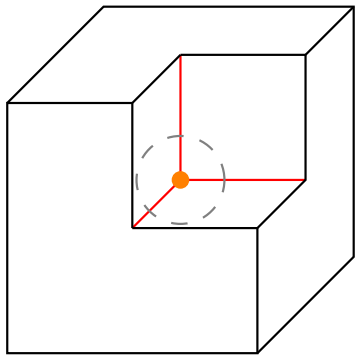
tetrahedra, prisms y pyramids.

The h_i are the sizes.

Subdomain only with tetrahedra

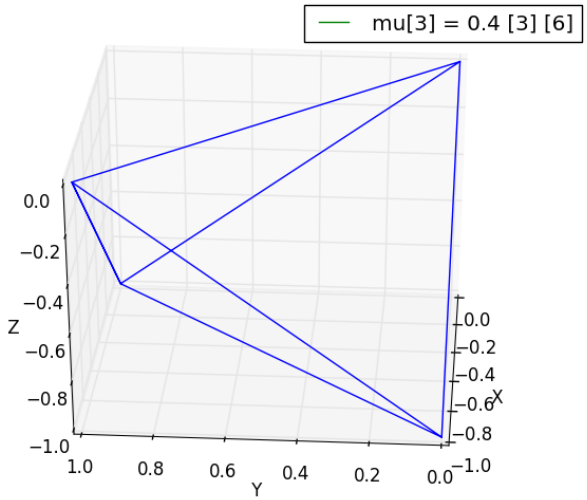
graded vs. uniform

Situation:

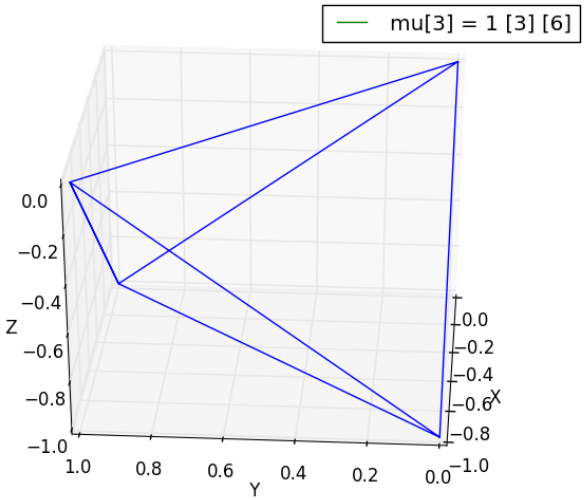


Subdomain only with tetrahedra

graded vs. uniform

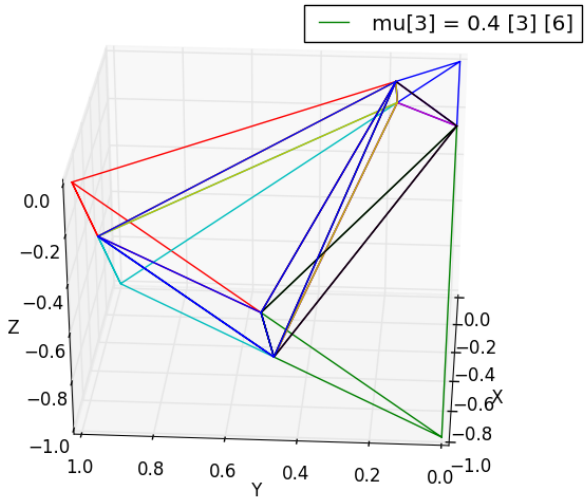


our method vs. standard method



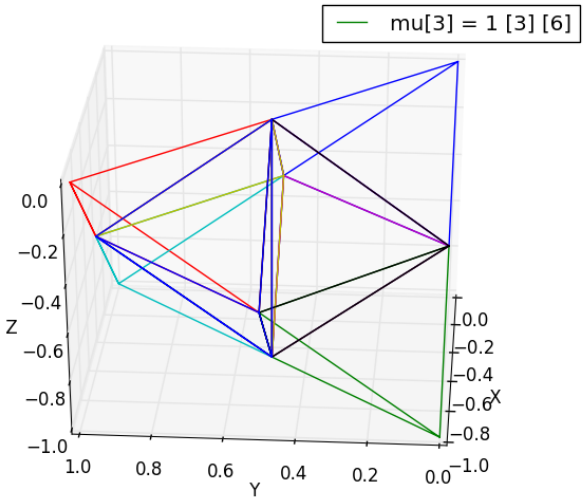
Subdomain only with tetrahedra

graded vs. uniform



our method

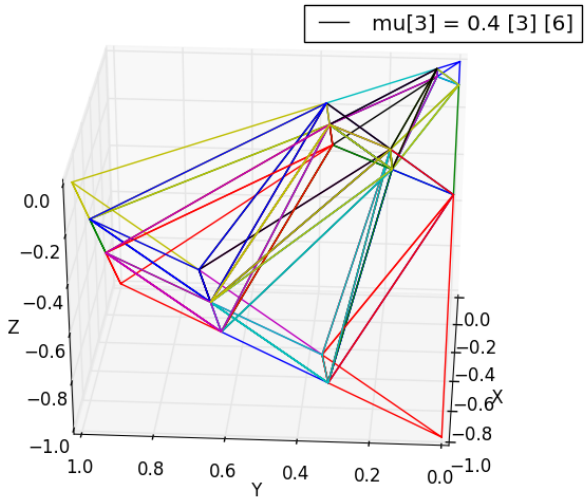
vs.



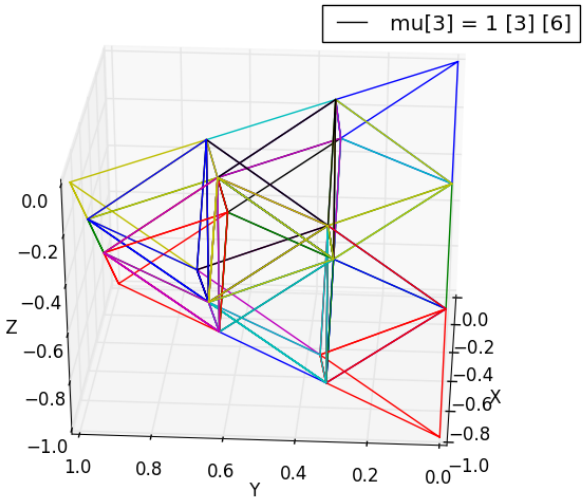
standard method

Subdomain only with tetrahedra

graded vs. uniform



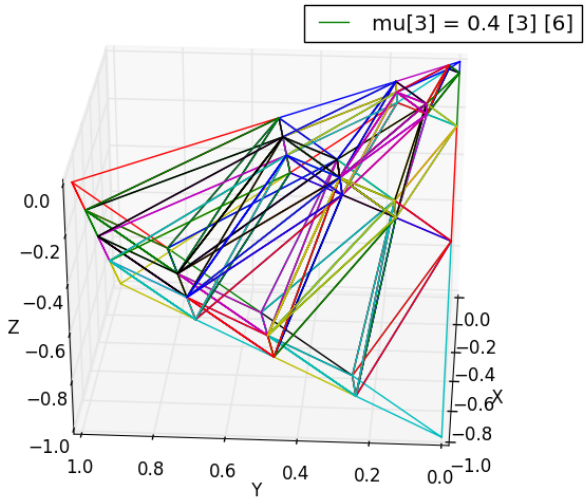
our method vs.



standard method

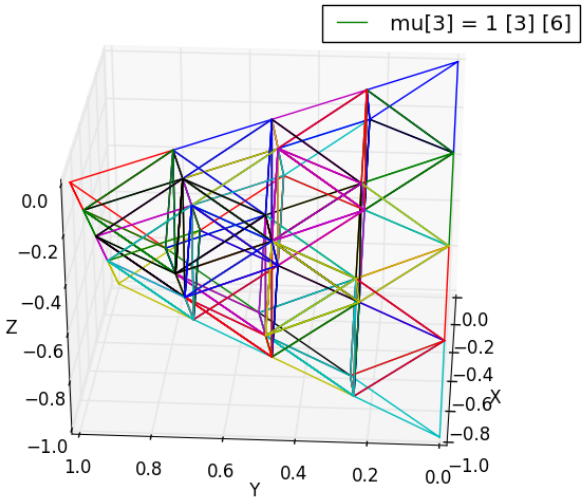
Subdomain only with tetrahedra

graded vs. uniform



our method

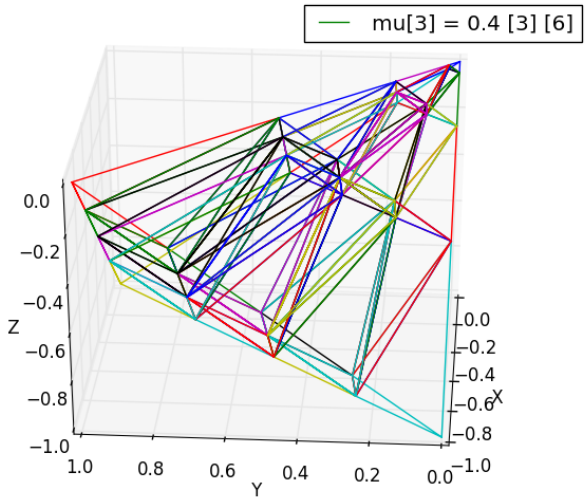
vs.



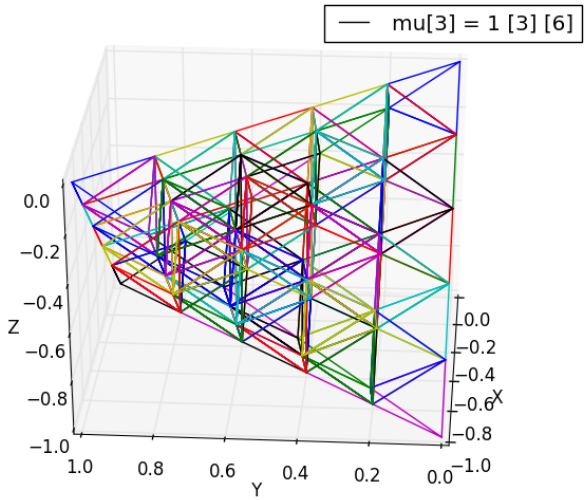
standard method

Subdomain only with tetrahedra

graded vs. uniform

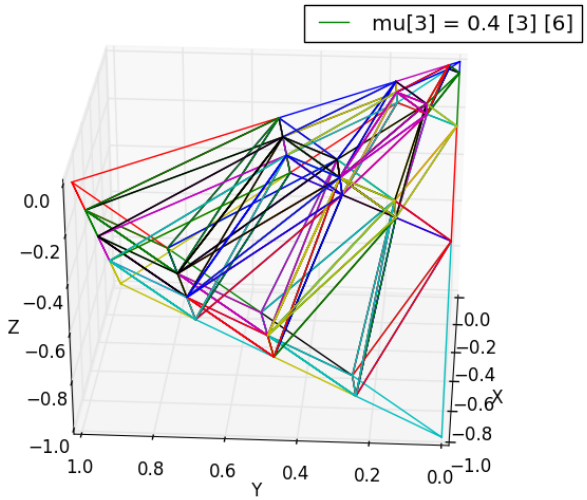


our method vs. standard method

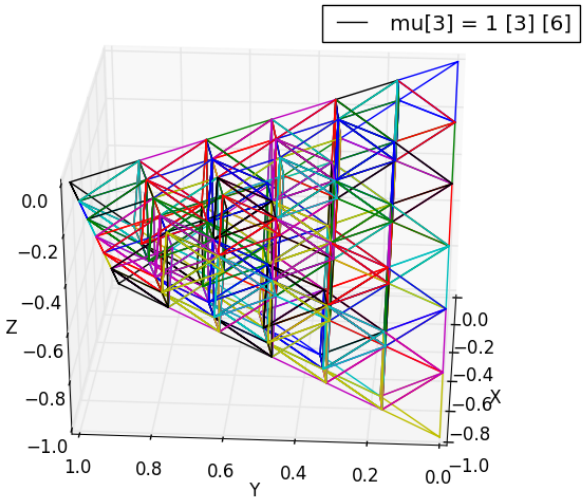


Subdomain only with tetrahedra

graded vs. uniform

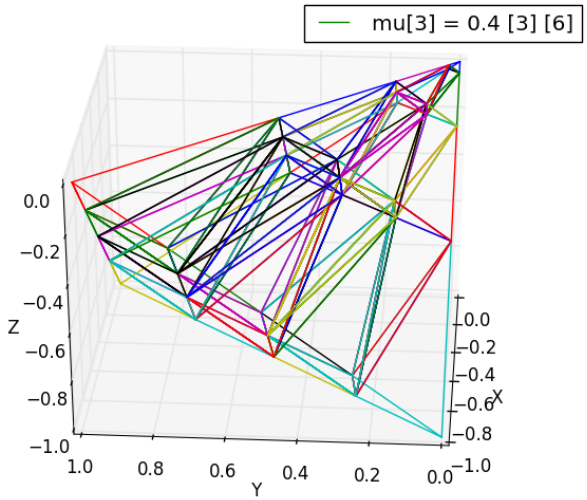


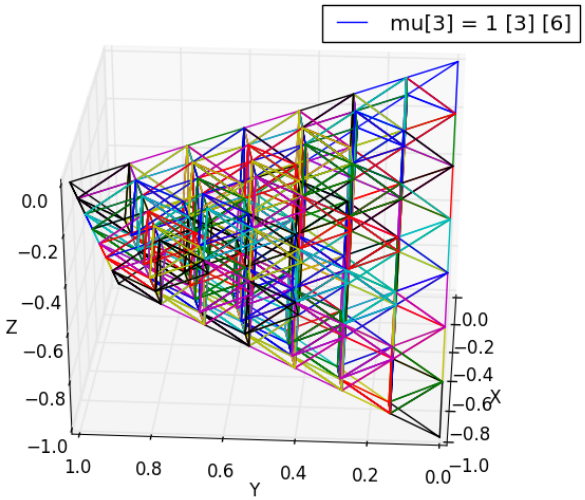
our method vs. standard method



Subdomain only with tetrahedra

graded vs. uniform

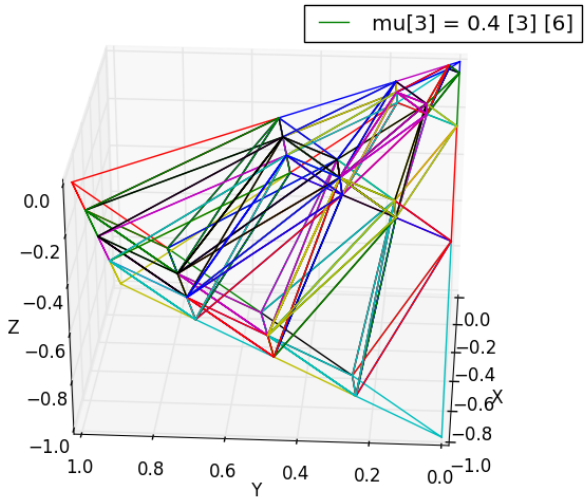


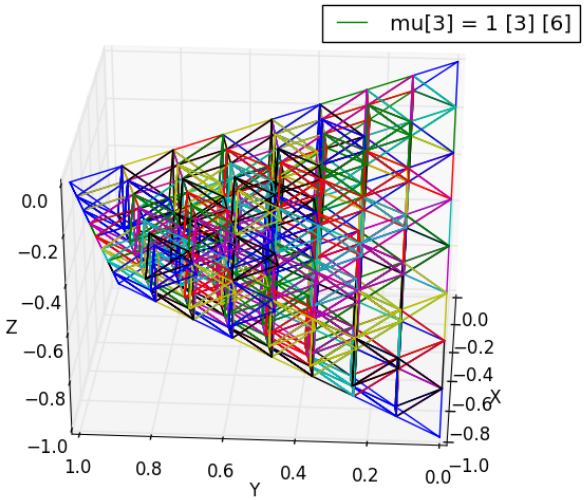


our method vs. standard method

Subdomain only with tetrahedra

graded vs. uniform

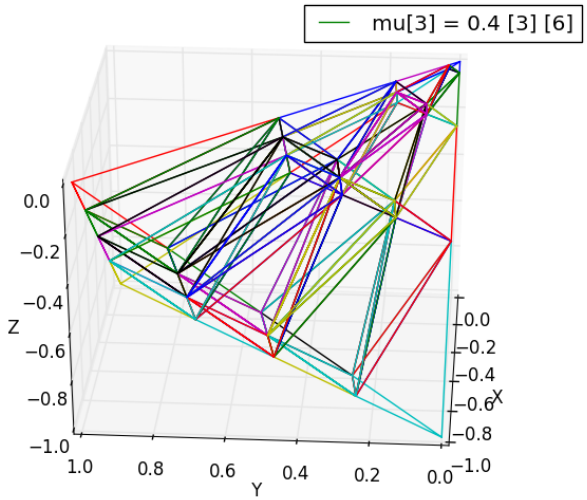


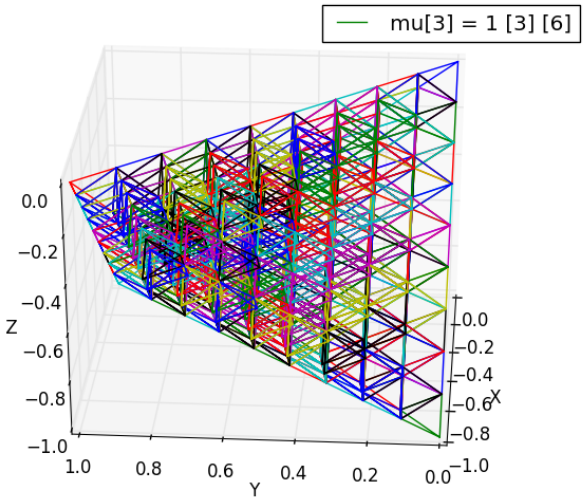


our method vs. standard method

Subdomain only with tetrahedra

graded vs. uniform

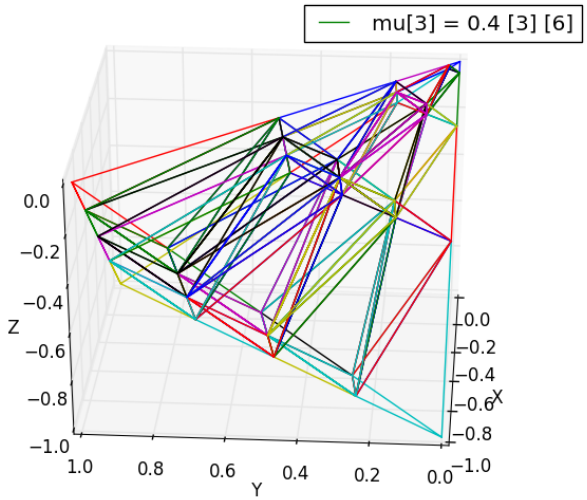




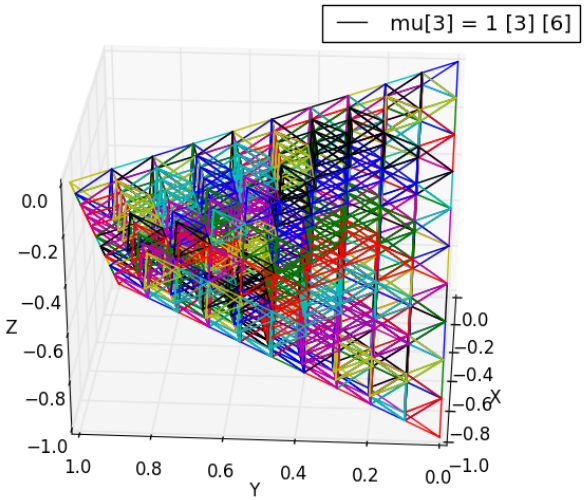
our method vs. standard method

Subdomain only with tetrahedra

graded vs. uniform

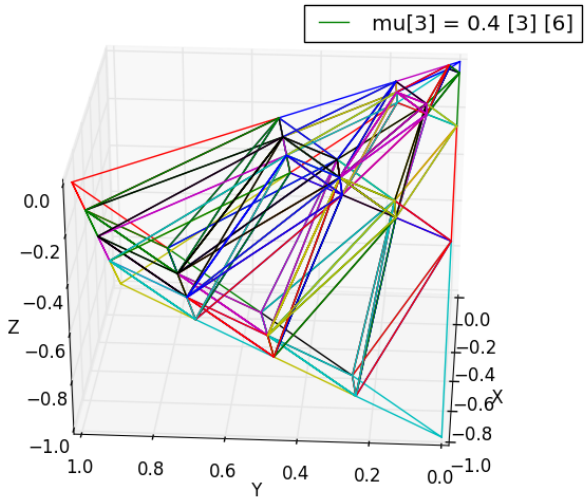


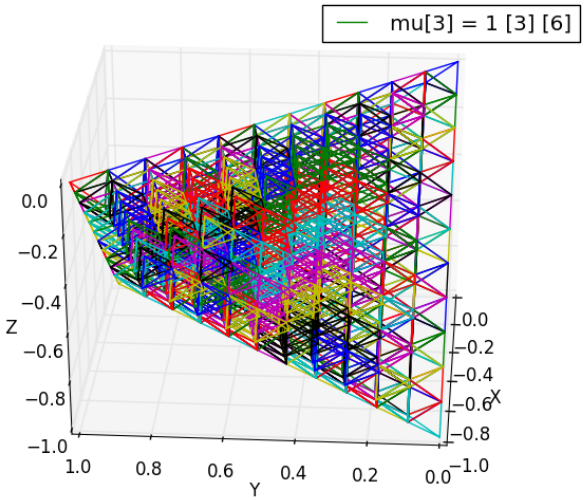
our method vs. standard method



Subdomain only with tetrahedra

graded vs. uniform

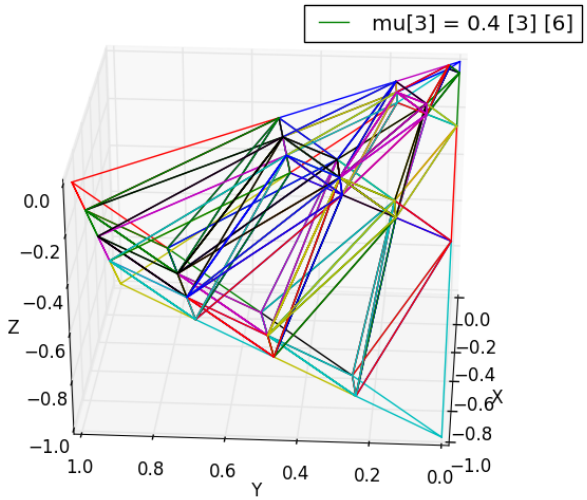




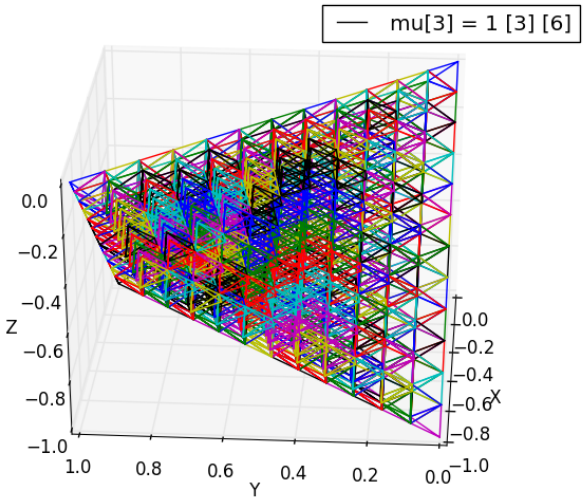
our method vs. standard method

Subdomain only with tetrahedra

graded vs. uniform

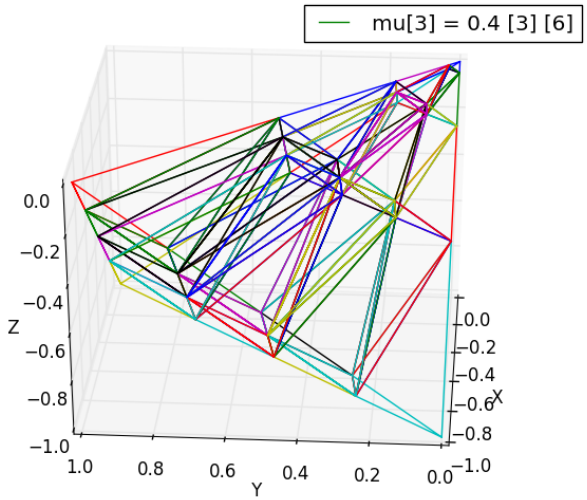


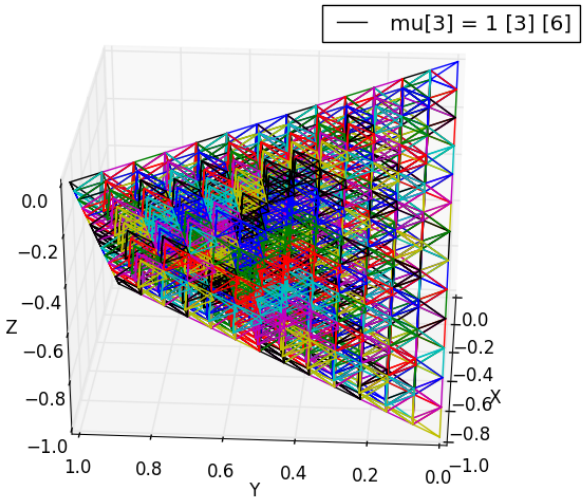
our method vs. standard method



Subdomain only with tetrahedra

graded vs. uniform

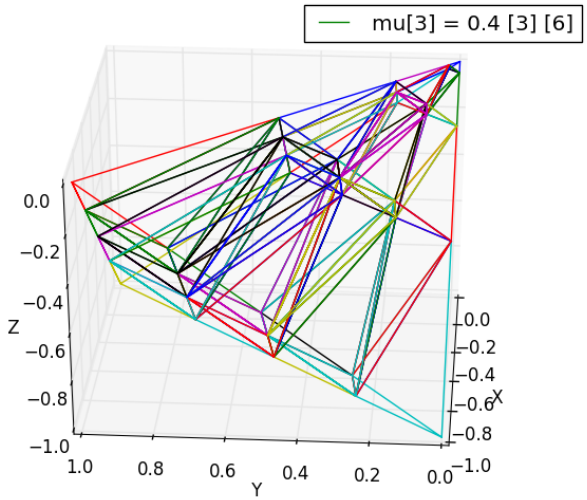




our method vs. standard method

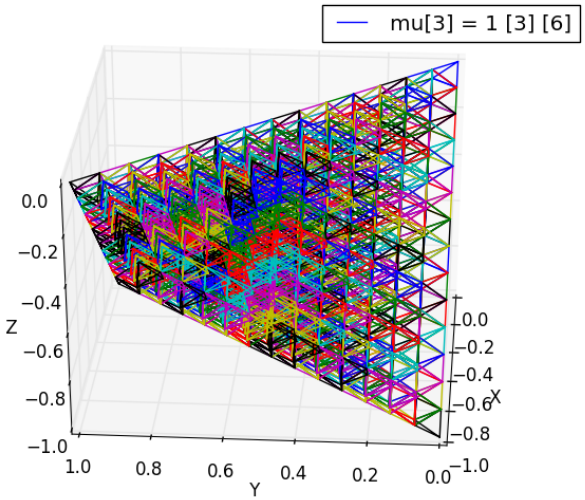
Subdomain only with tetrahedra

graded vs. uniform



our method

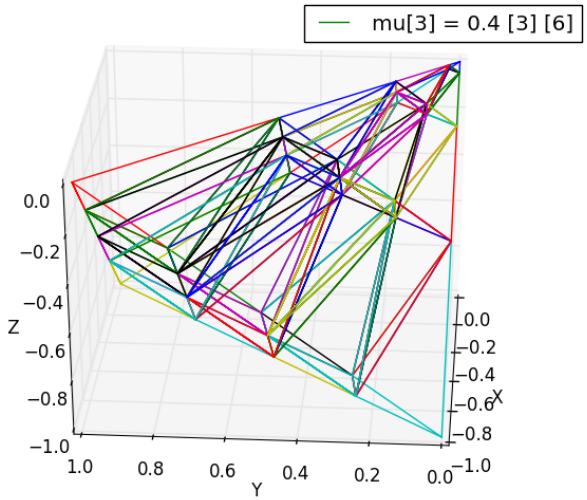
vs.

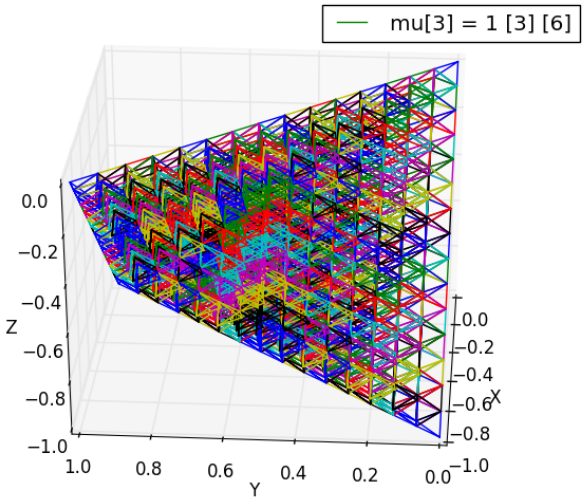


standard method

Subdomain only with tetrahedra

graded vs. uniform

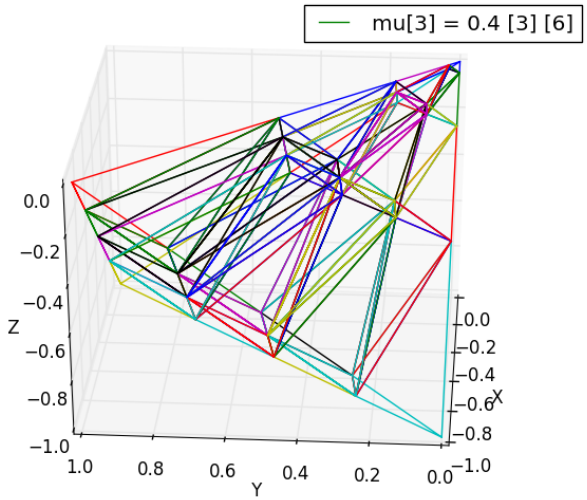




our method vs. standard method

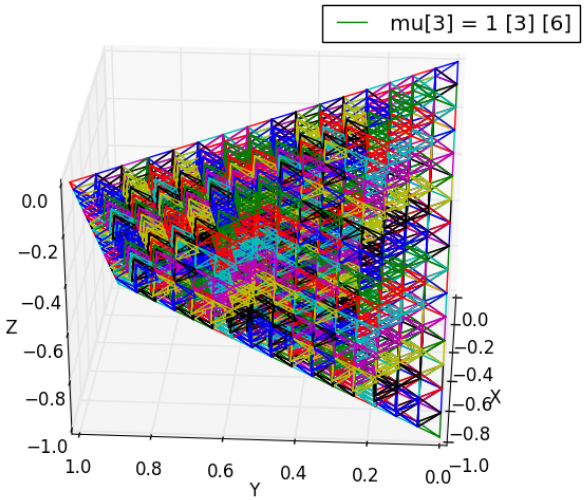
Subdomain only with tetrahedra

graded vs. uniform



our method

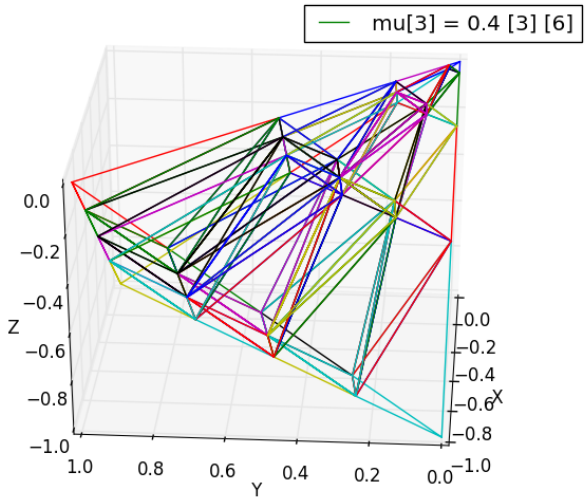
vs.



standard method

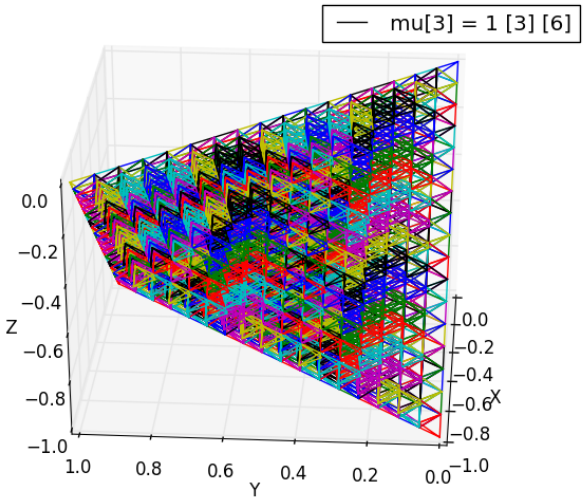
Subdomain only with tetrahedra

graded vs. uniform



our method

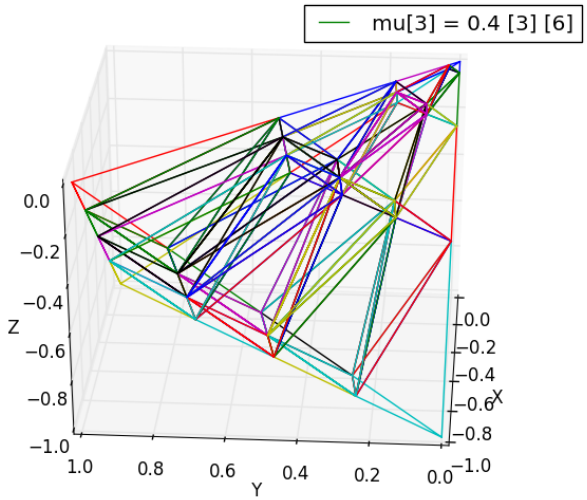
vs.



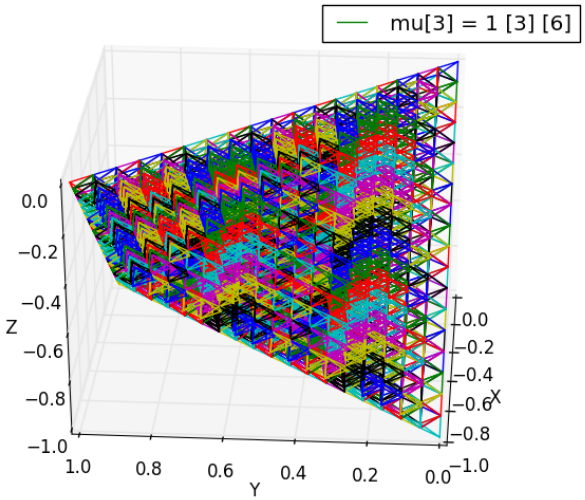
standard method

Subdomain only with tetrahedra

graded vs. uniform

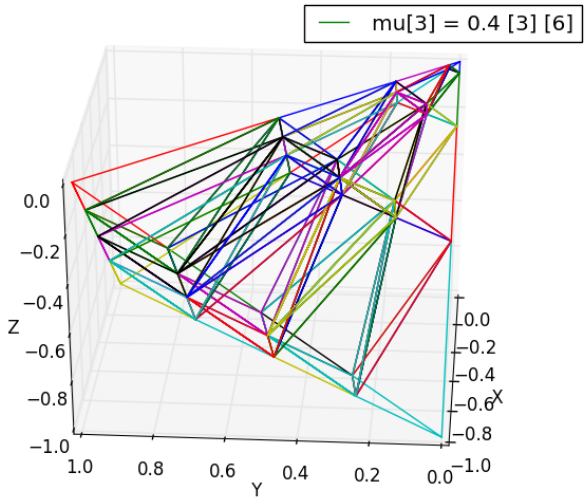


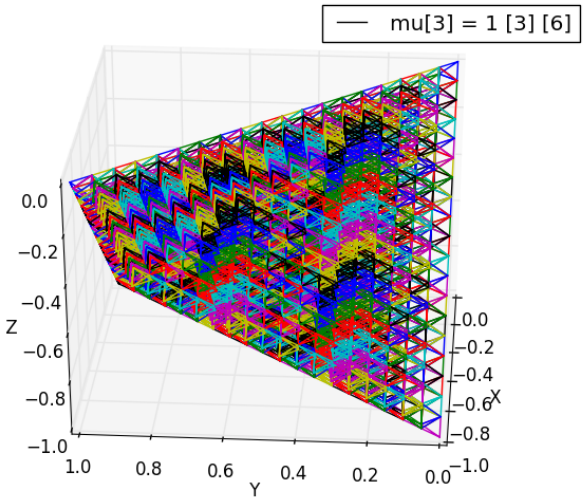
our method vs. standard method



Subdomain only with tetrahedra

graded vs. uniform



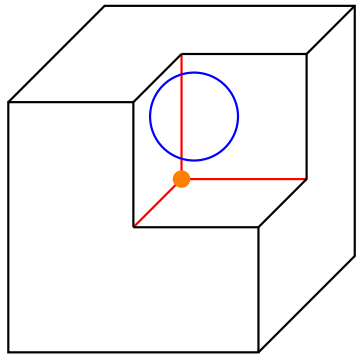


our method vs. standard method

Subdomain only with prisms

graded vs. uniform

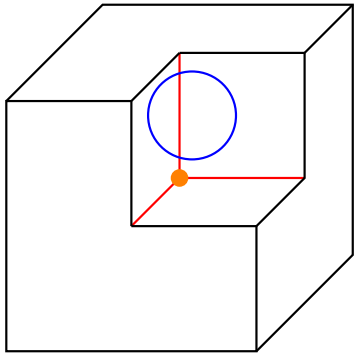
Situation:

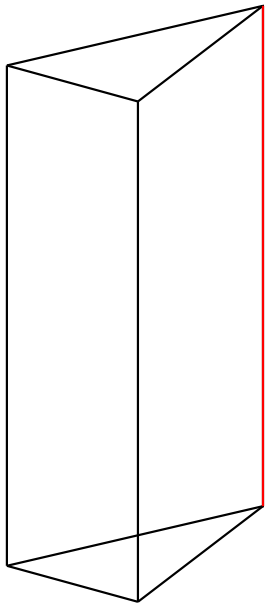


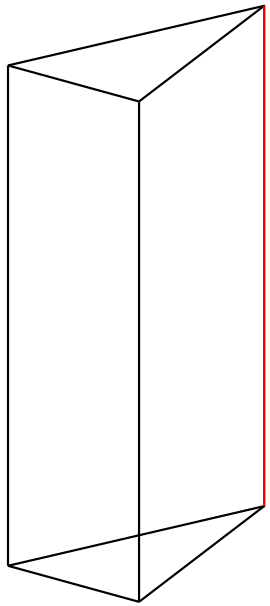
Subdomain only with prisms

graded vs. uniform

Situation:



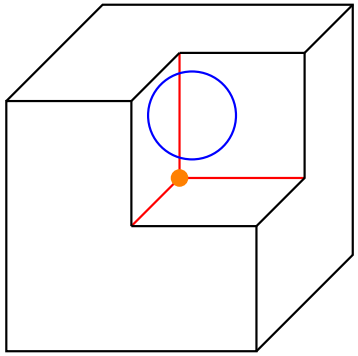


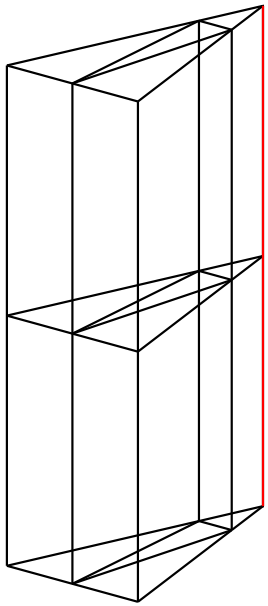


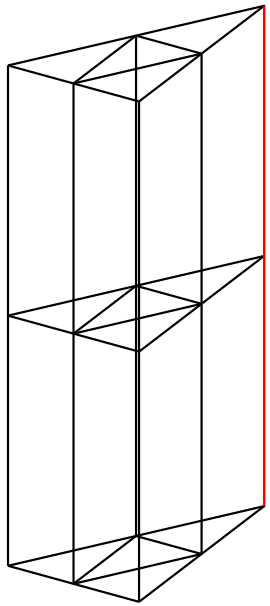
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:





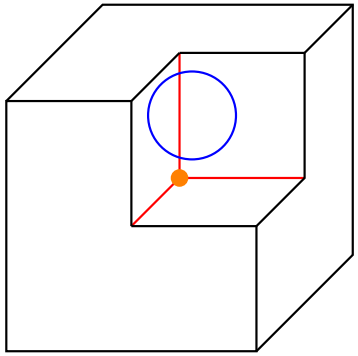


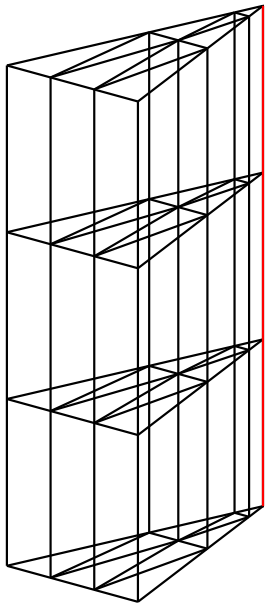
our method vs. standard method

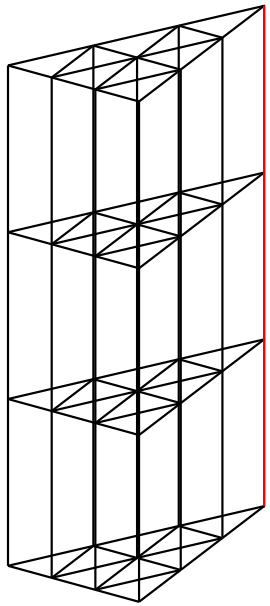
Subdomain only with prisms

graded vs. uniform

Situation:





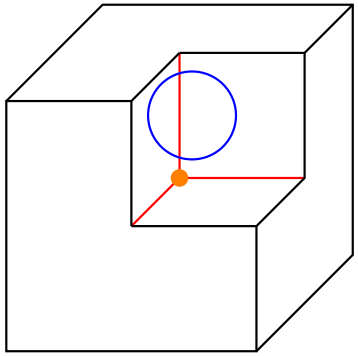


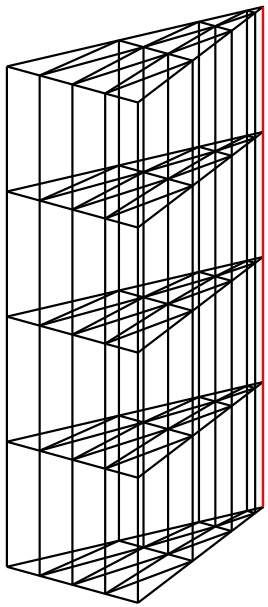
our method vs. standard method

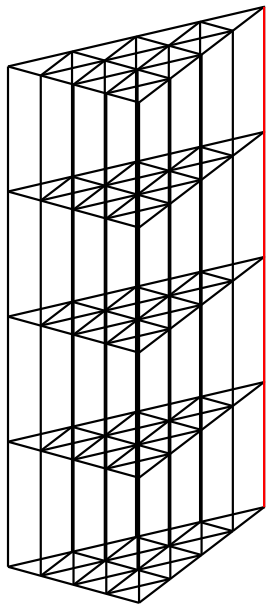
Subdomain only with prisms

graded vs. uniform

Situation:



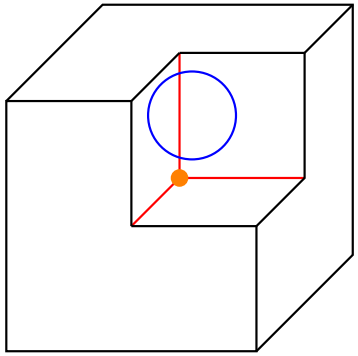


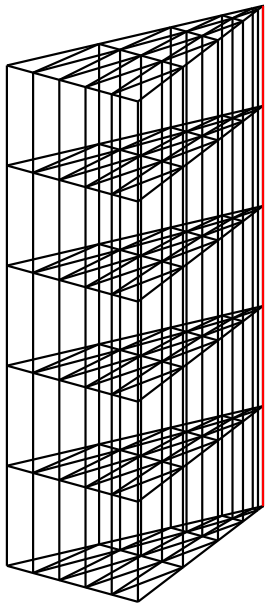


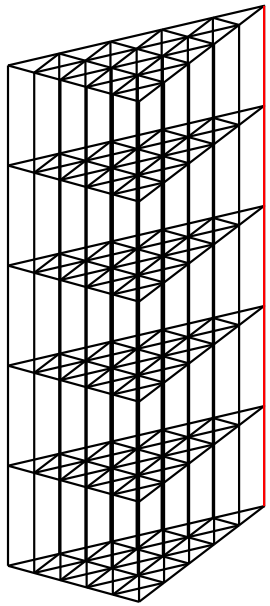
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



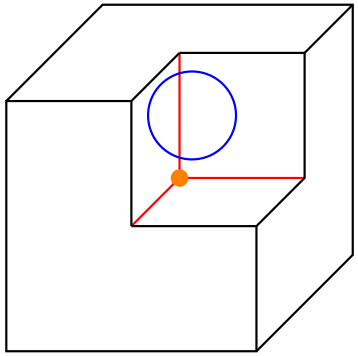


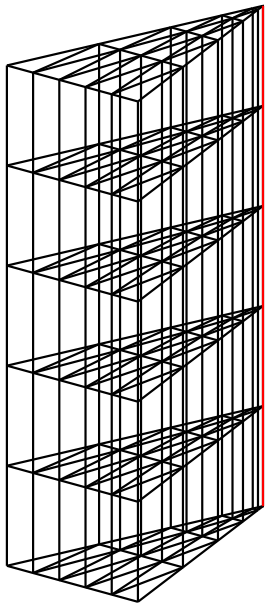


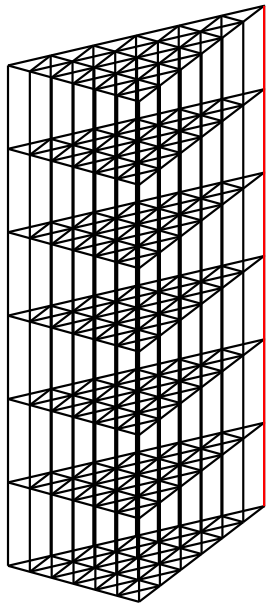
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



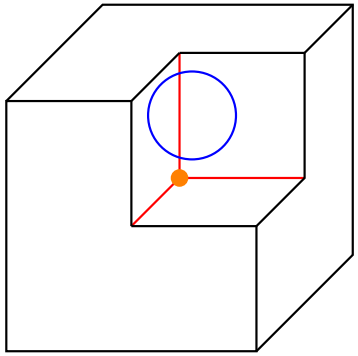


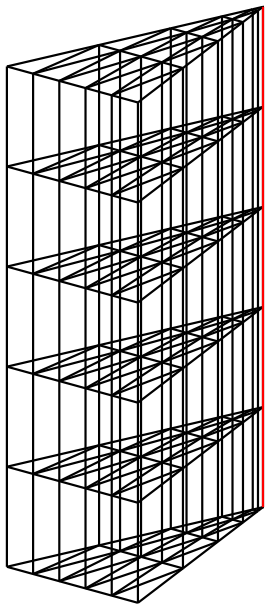


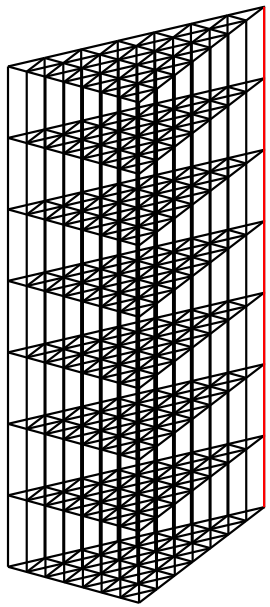
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



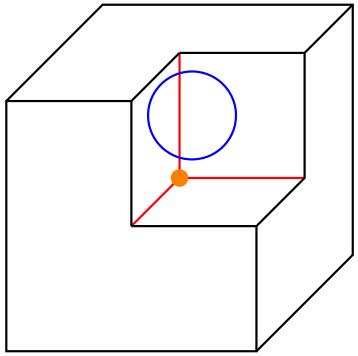


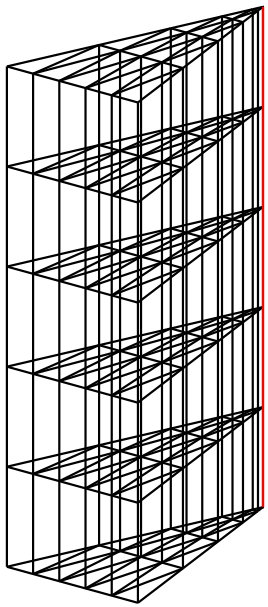


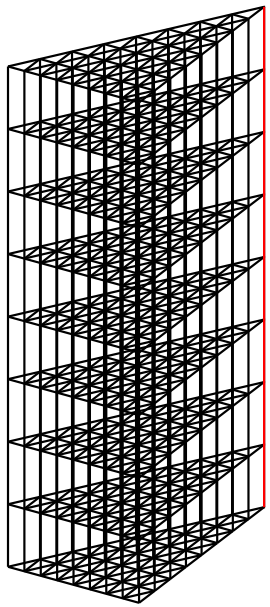
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



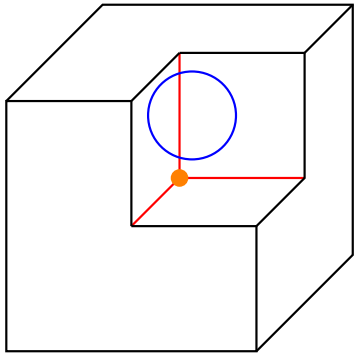


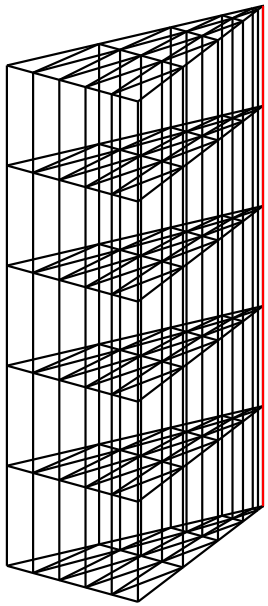


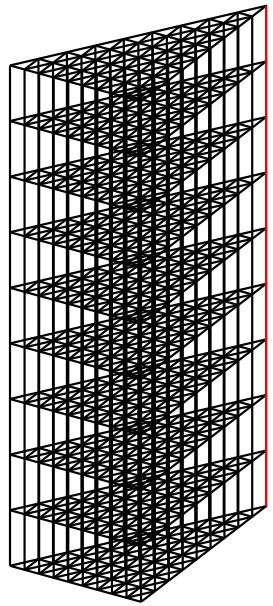
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



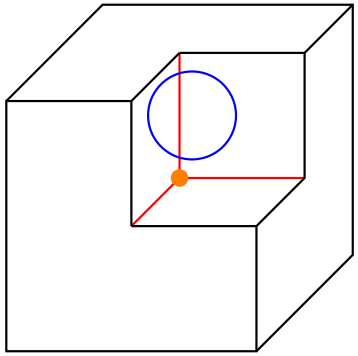


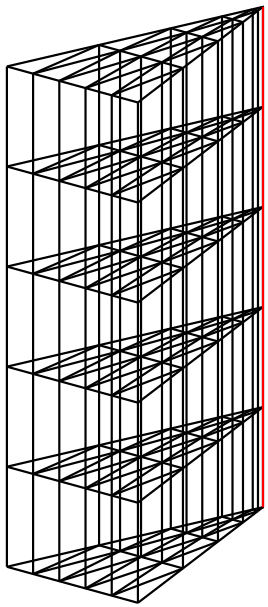


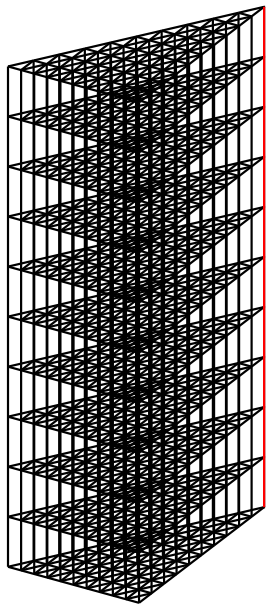
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



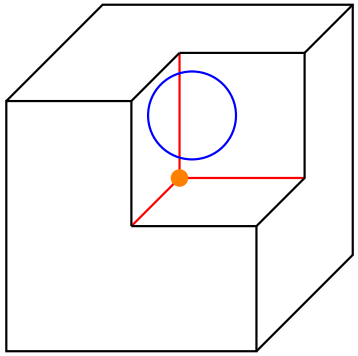


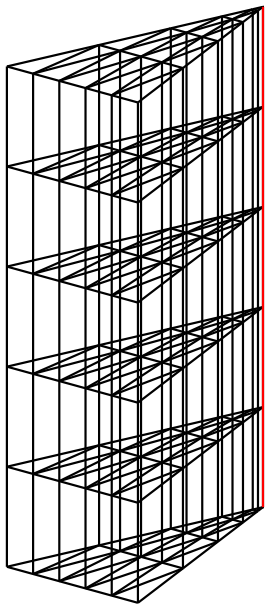


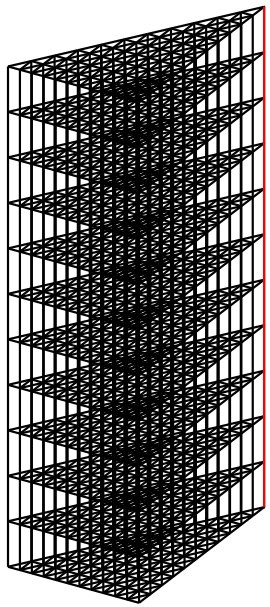
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



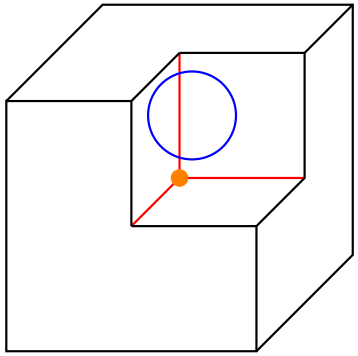


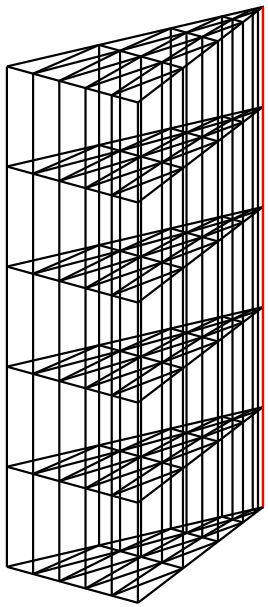


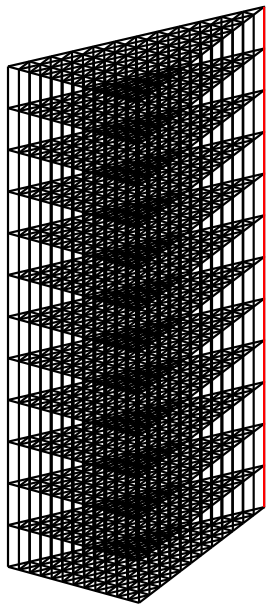
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:





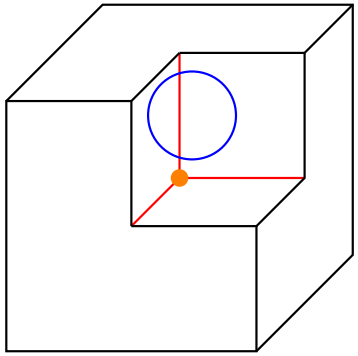


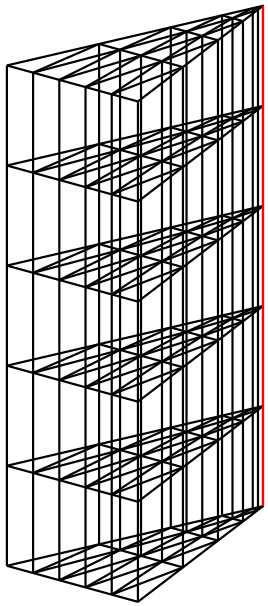
our method vs. standard method

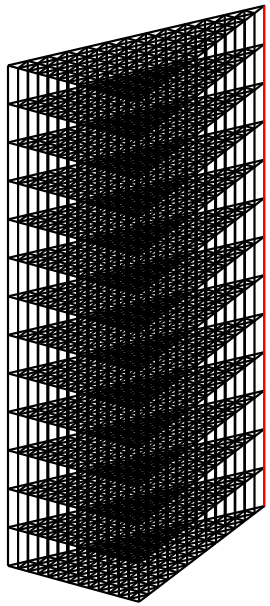
Subdomain only with prisms

graded vs. uniform

Situation:



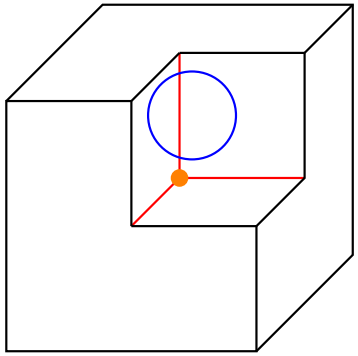


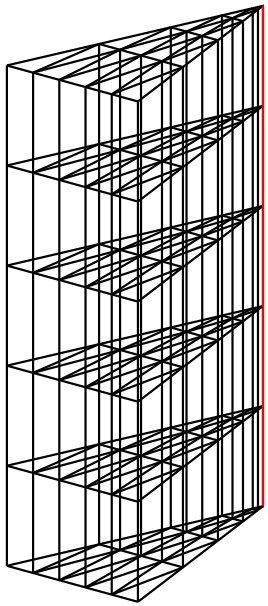


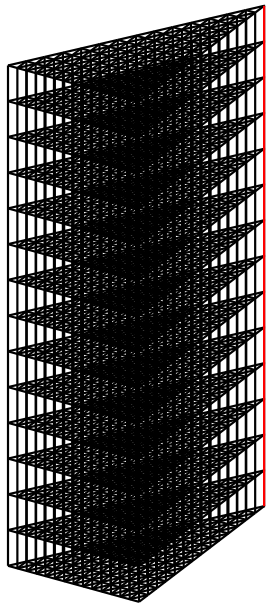
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



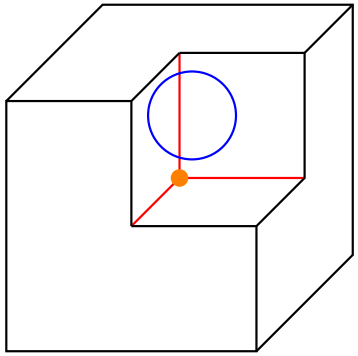


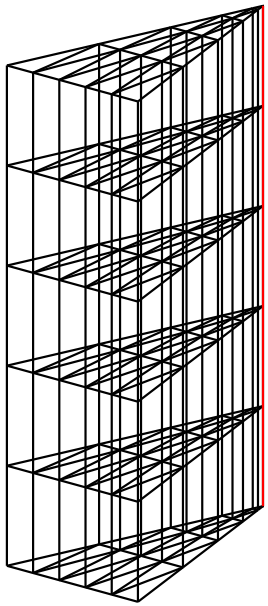


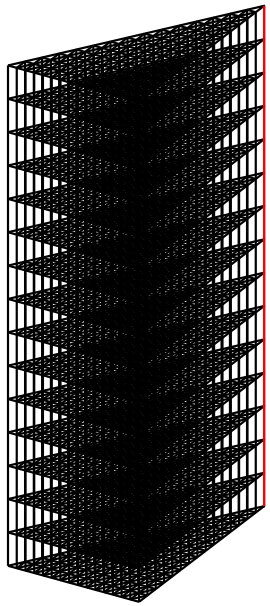
our method vs. standard method

Subdomain only with prisms
graded vs. uniform

Situation:



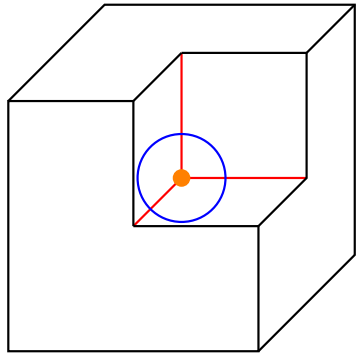




our method vs. standard method

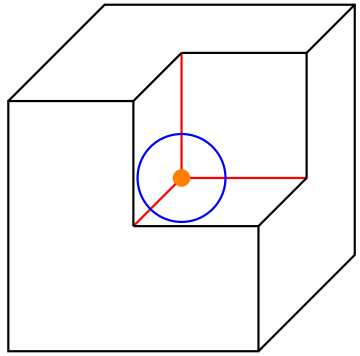
Subdomain with prisms, pyramids and tetrahedra

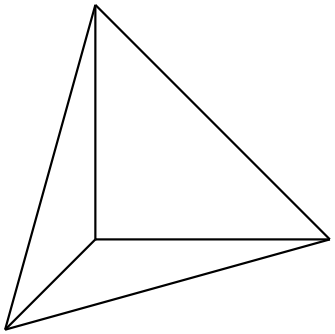
Situation:



Subdomain with prisms, pyramids and tetrahedra

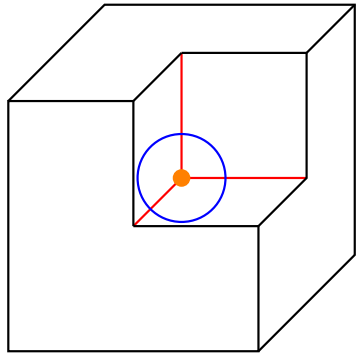
Situation:

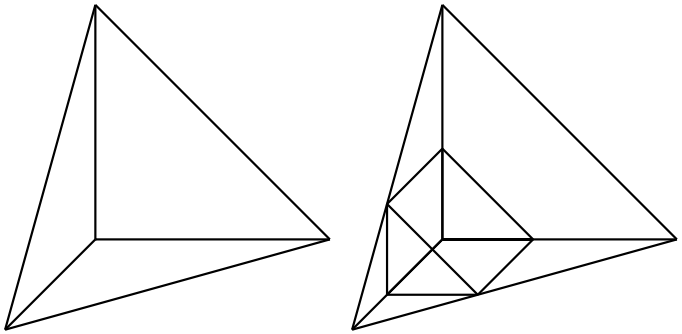




Subdomain with prisms, pyramids and tetrahedra

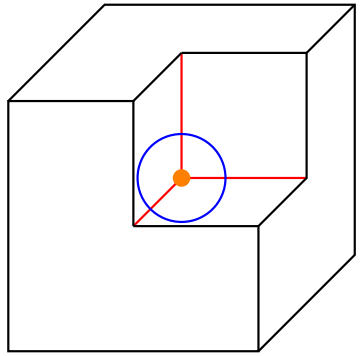
Situation:

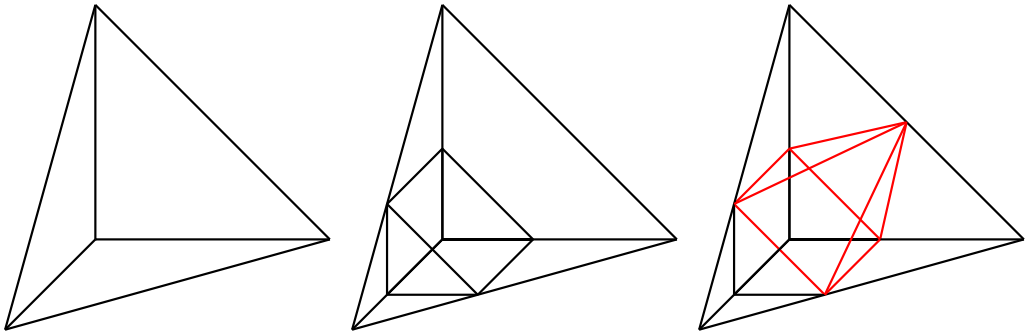




Subdomain with prisms, pyramids and tetrahedra

Situation:

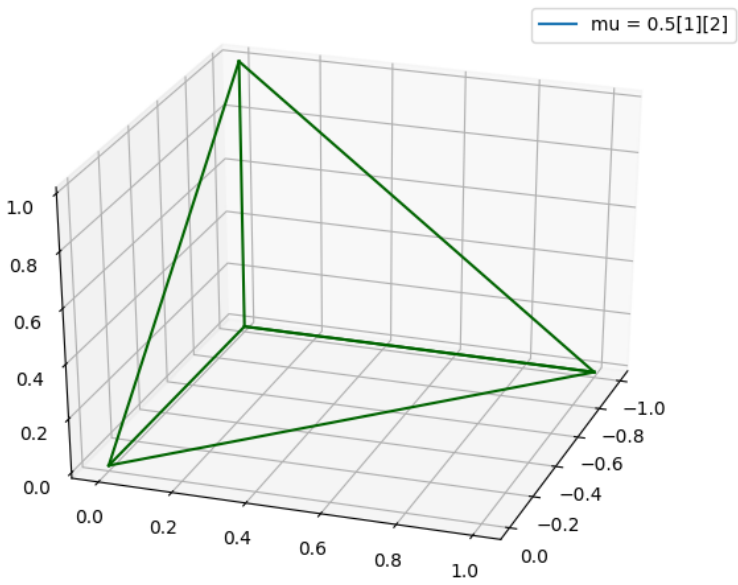


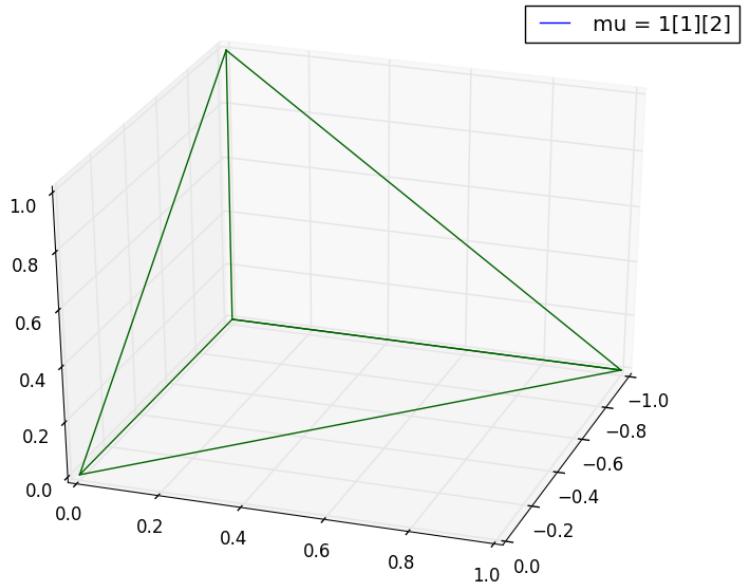


Subdomain with prisms, pyramids and tetrahedra

1 segments per edge. $1 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

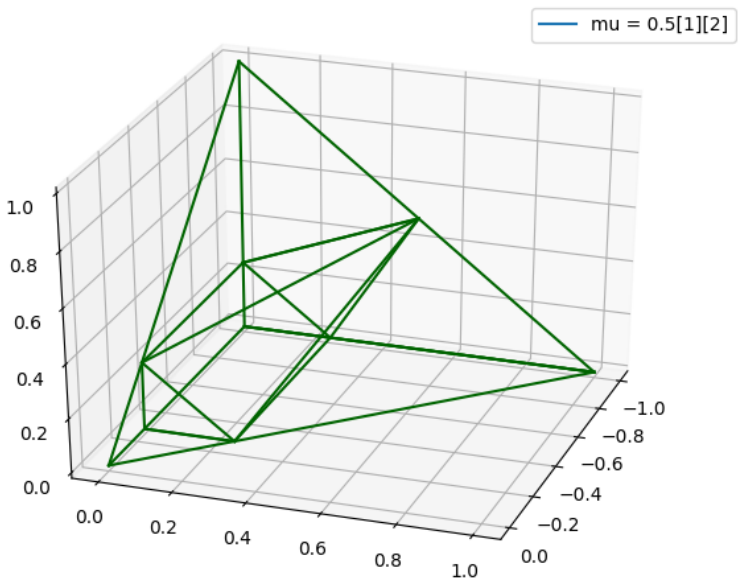


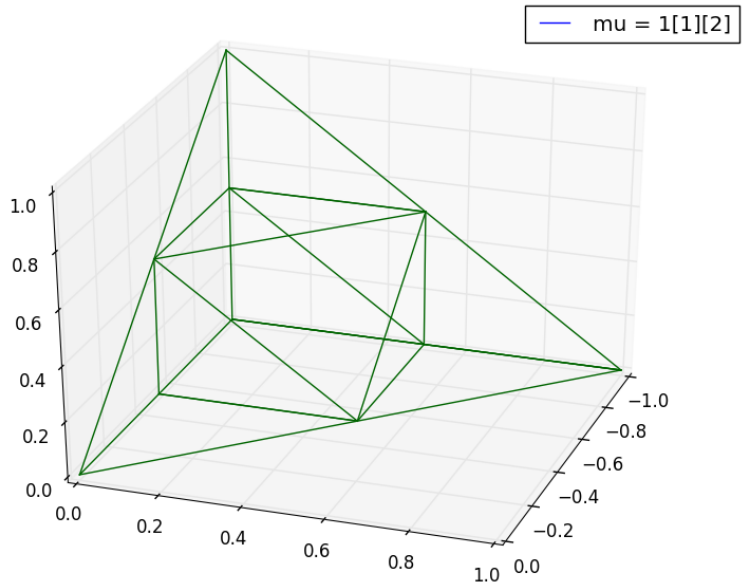


Subdomain with prisms, pyramids and tetrahedra

2 segments per edge. $2 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

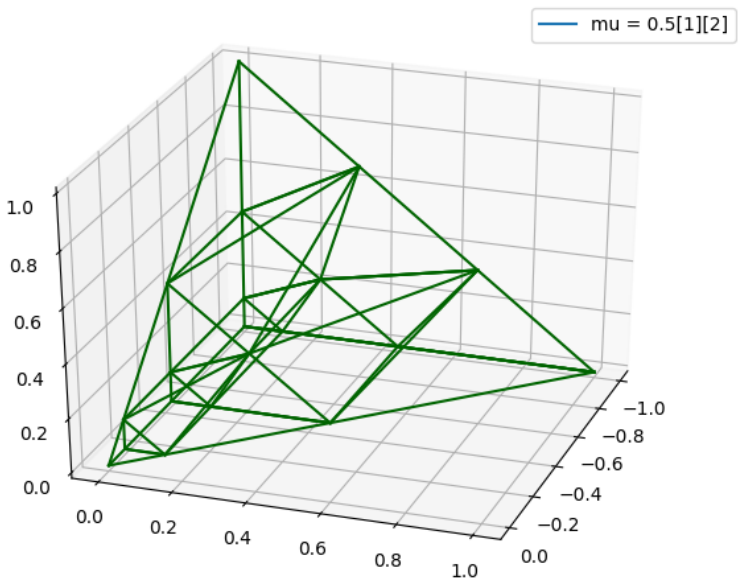


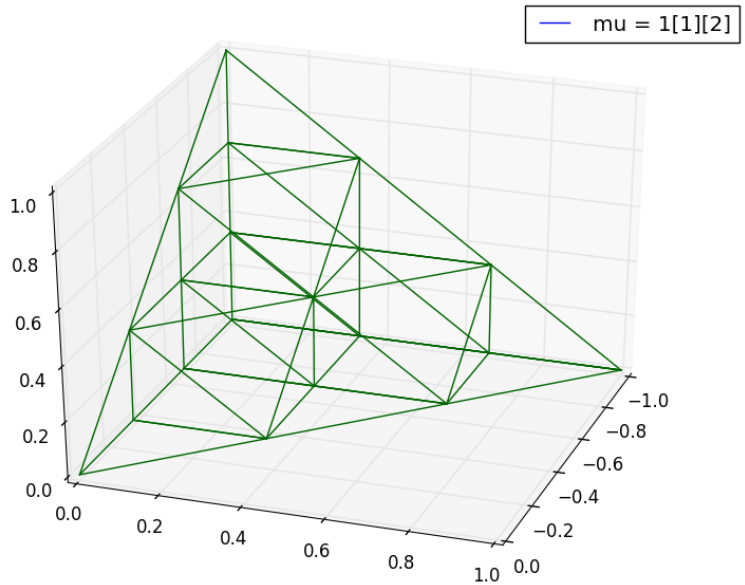


Subdomain with prisms, pyramids and tetrahedra

3 segments per edge. $3 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

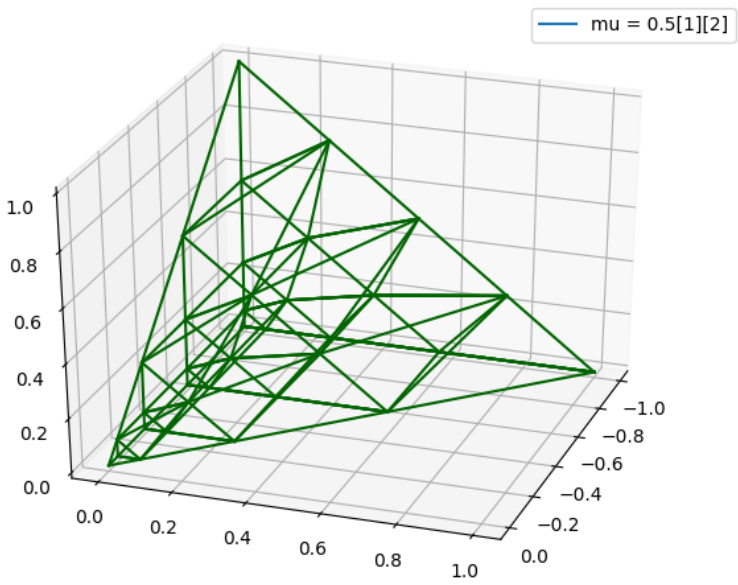


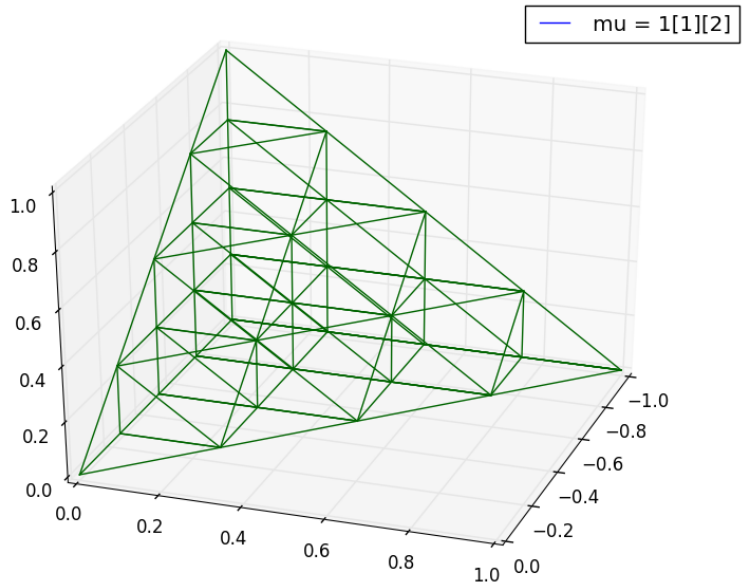


Subdomain with prisms, pyramids and tetrahedra

4 segments per edge. $4 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

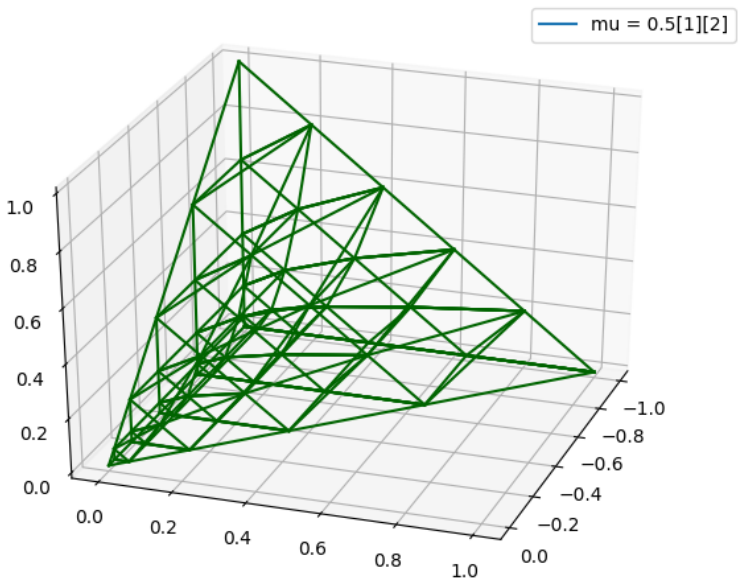


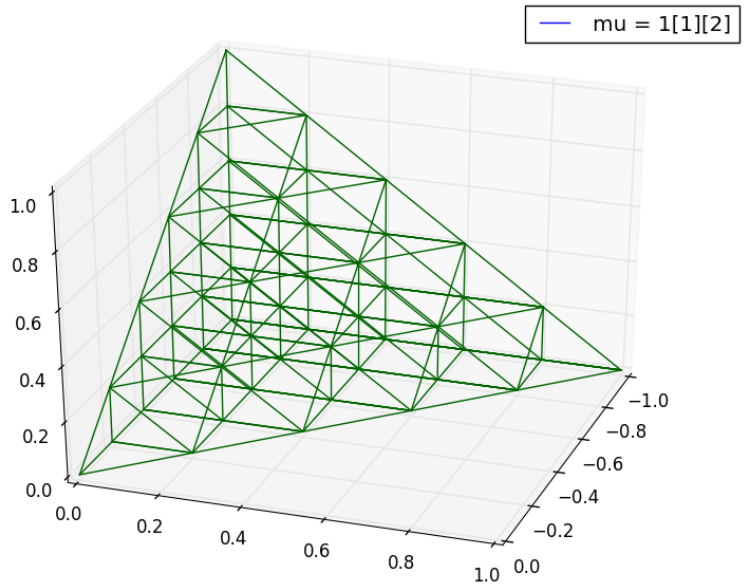


Subdomain with prisms, pyramids and tetrahedra

5 segments per edge. $5 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

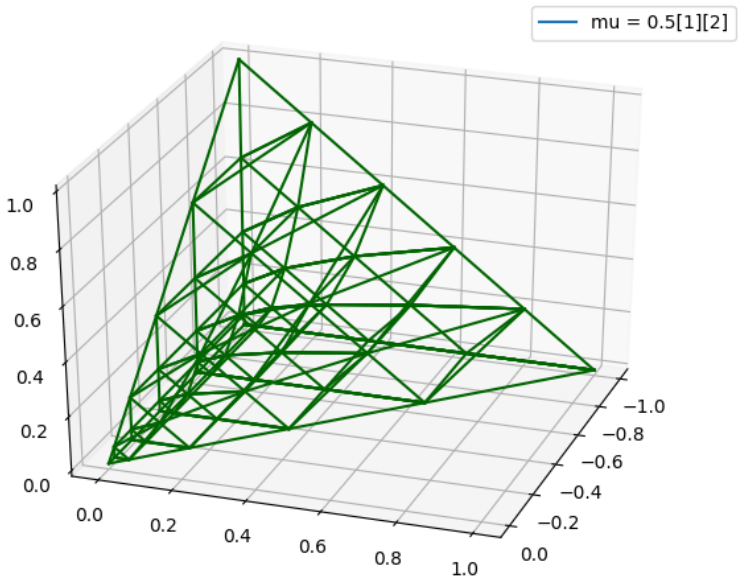


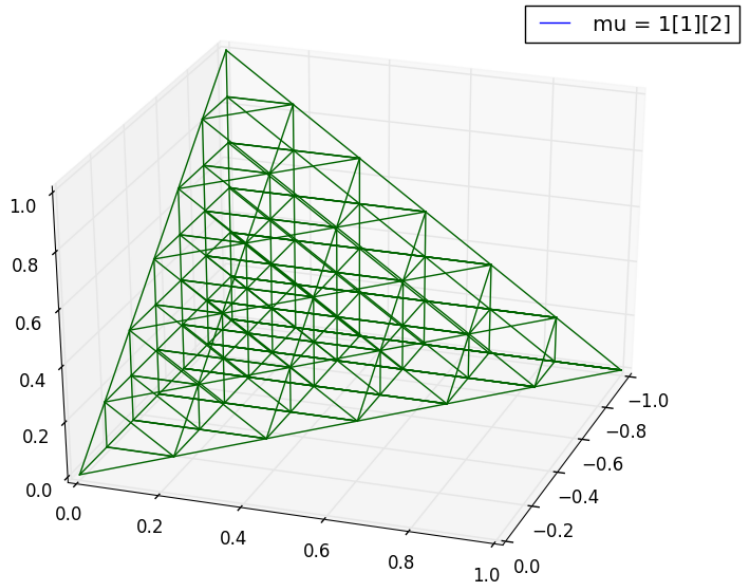


Subdomain with prisms, pyramids and tetrahedra

6 segments per edge. $6 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

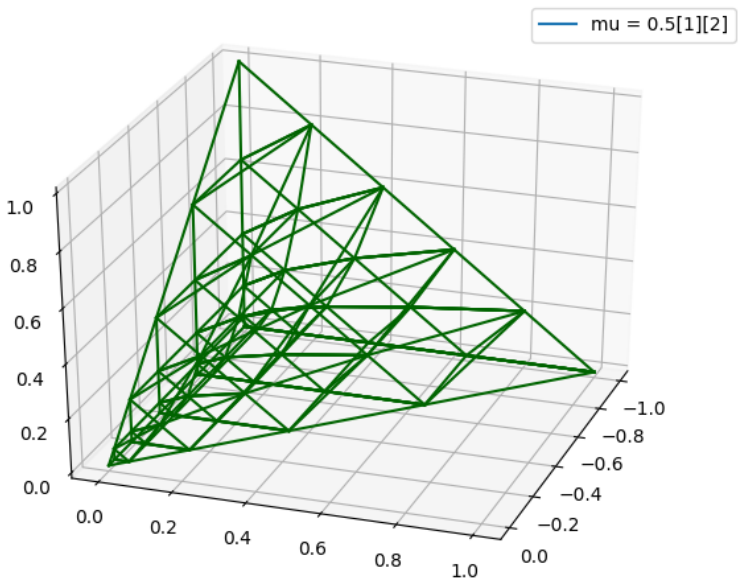


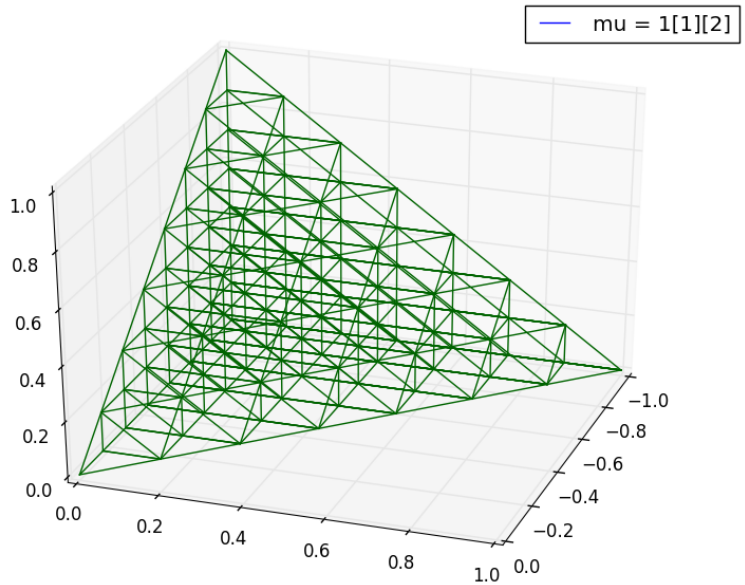


Subdomain with prisms, pyramids and tetrahedra

7 segments per edge. $7 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

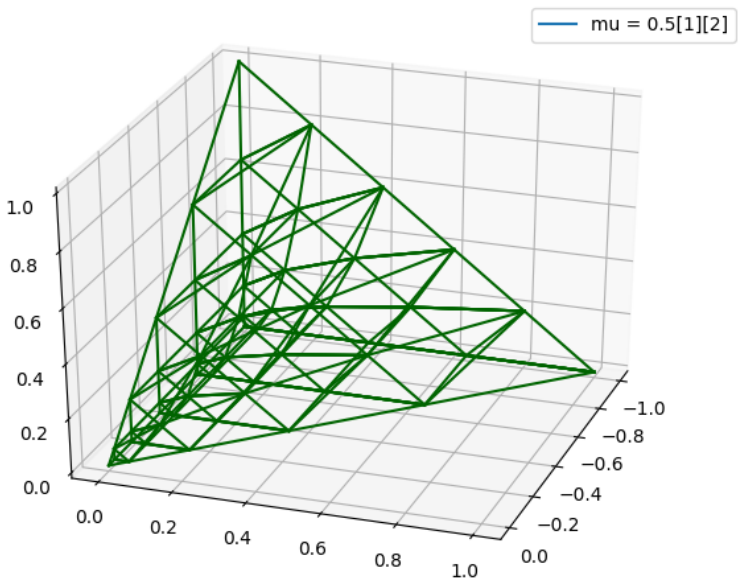


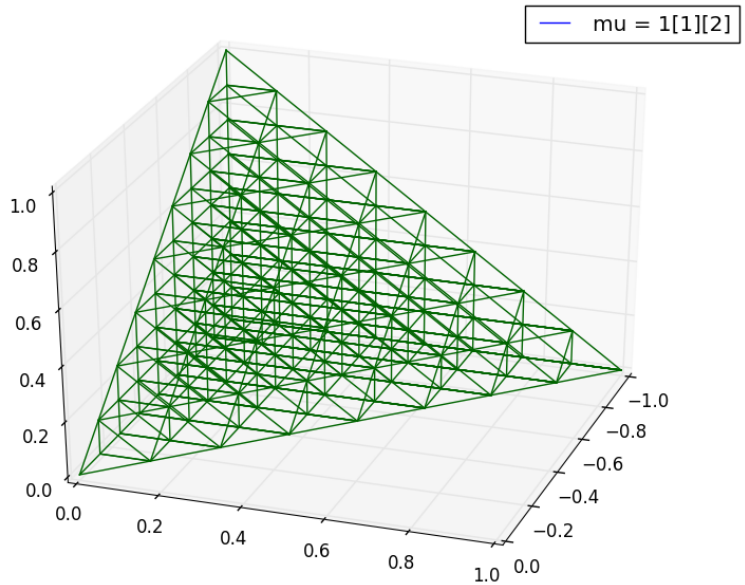


Subdomain with prisms, pyramids and tetrahedra

8 segments per edge. $8 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

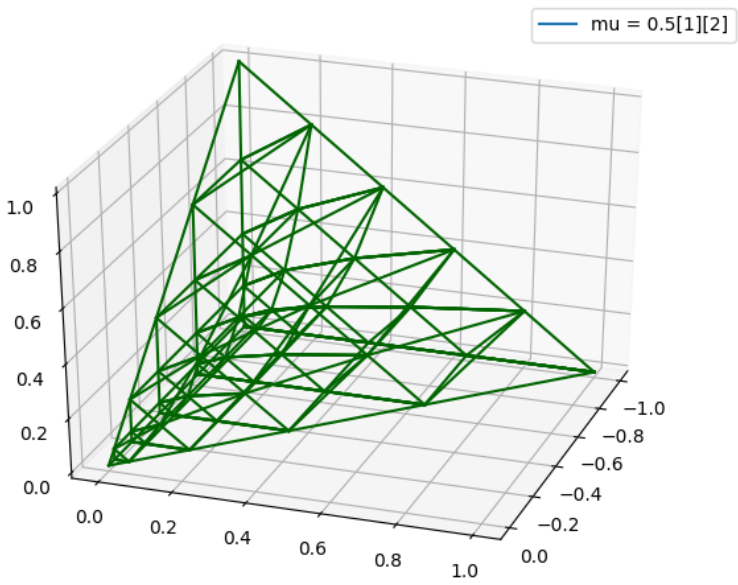


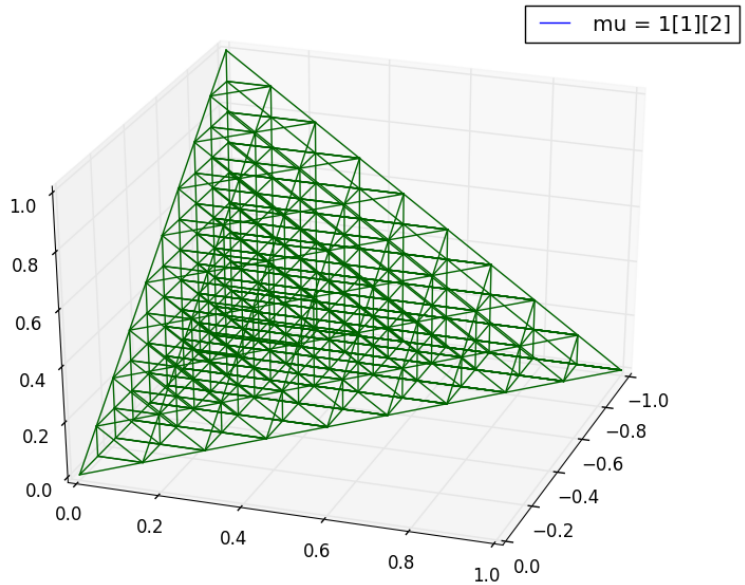


Subdomain with prisms, pyramids and tetrahedra

9 segments per edge. $9 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

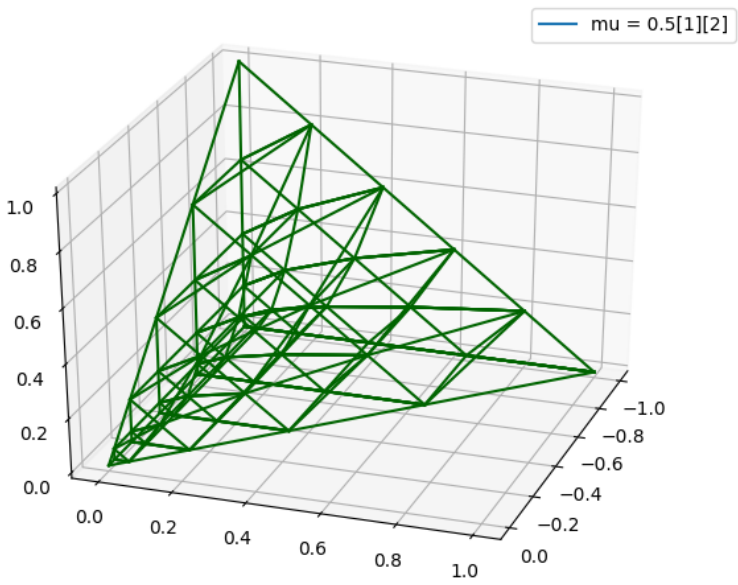


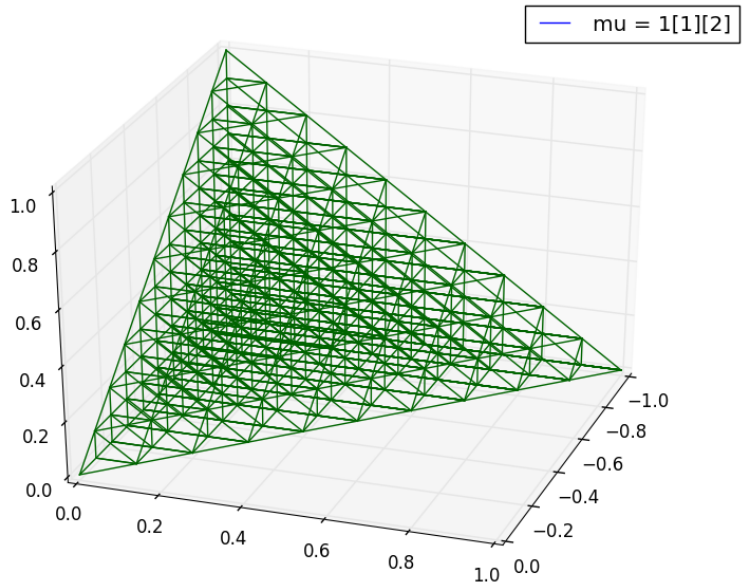


Subdomain with prisms, pyramids and tetrahedra

10 segments per edge. $10 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

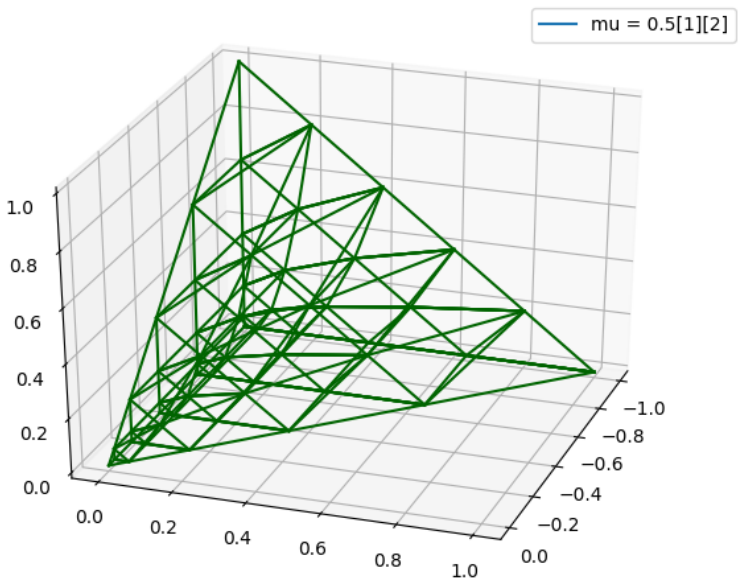


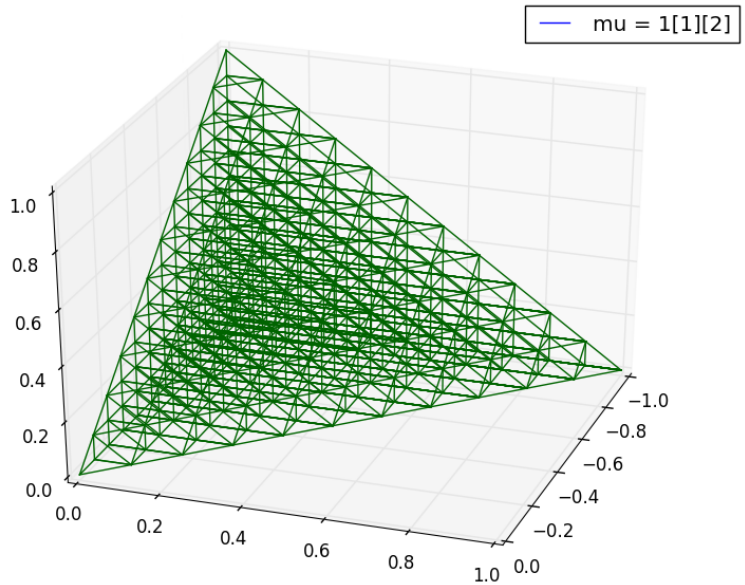


Subdomain with prisms, pyramids and tetrahedra

11 segments per edge. $11 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

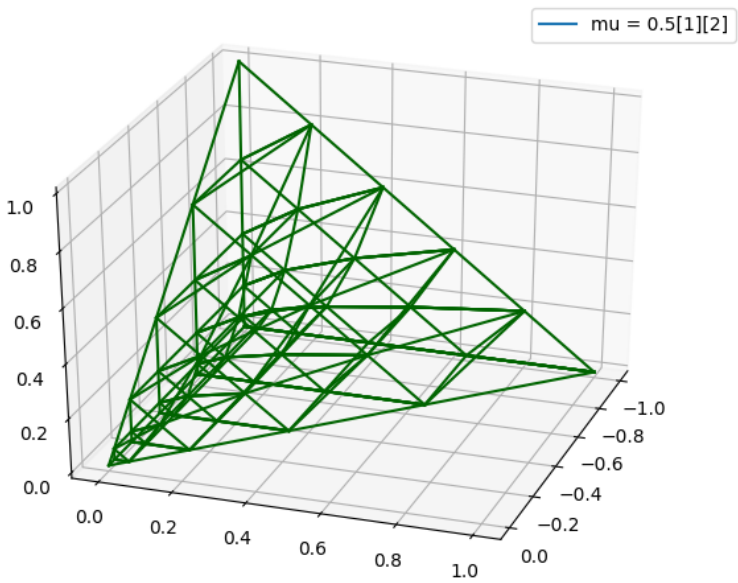


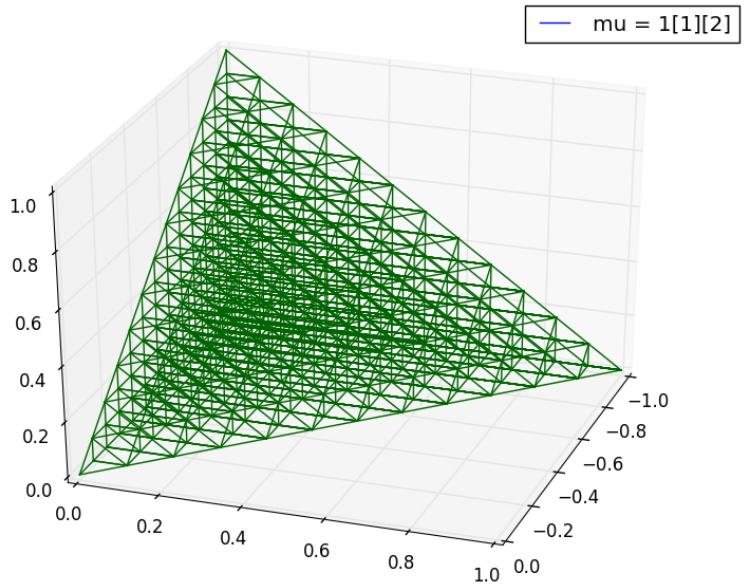


Subdomain with prisms, pyramids and tetrahedra

12 segments per edge. $12 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

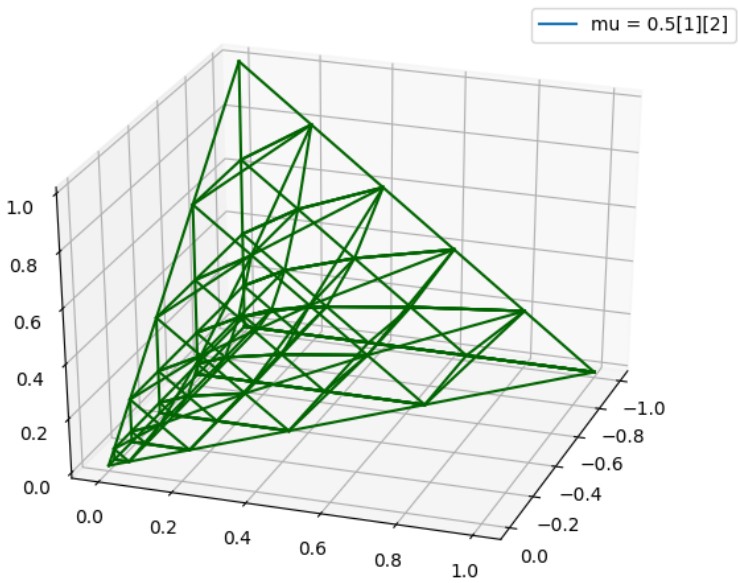


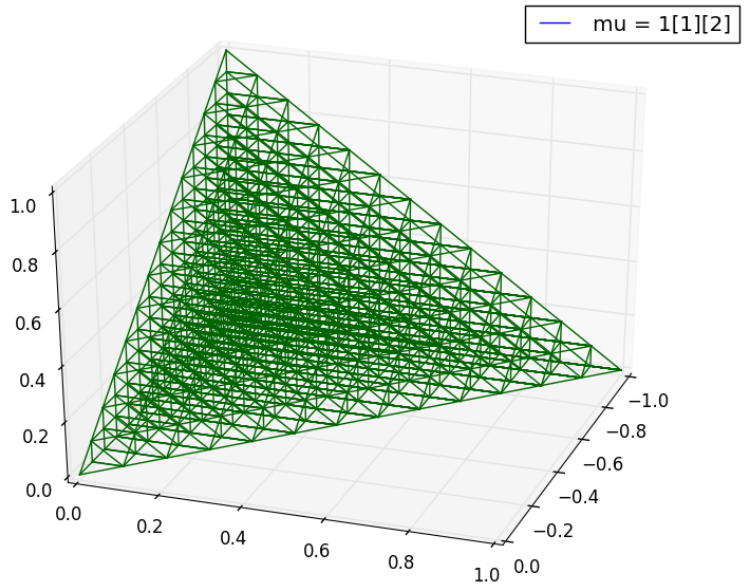


Subdomain with prisms, pyramids and tetrahedra

13 segments per edge. $13 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

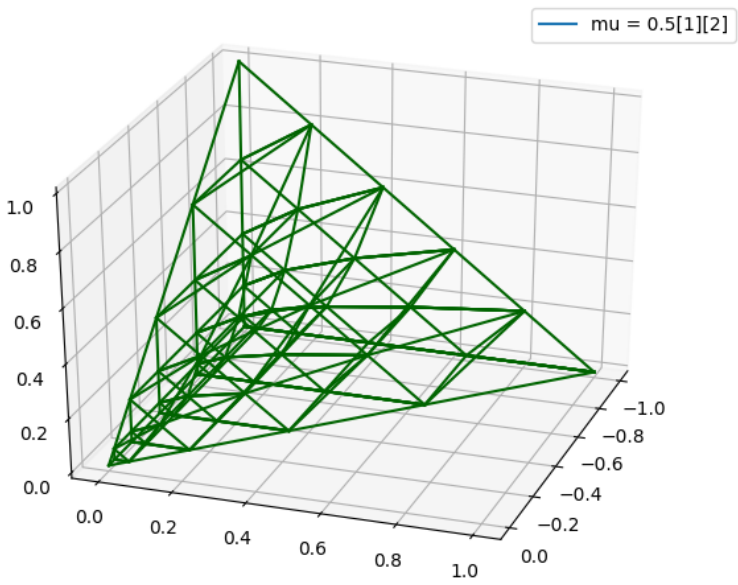


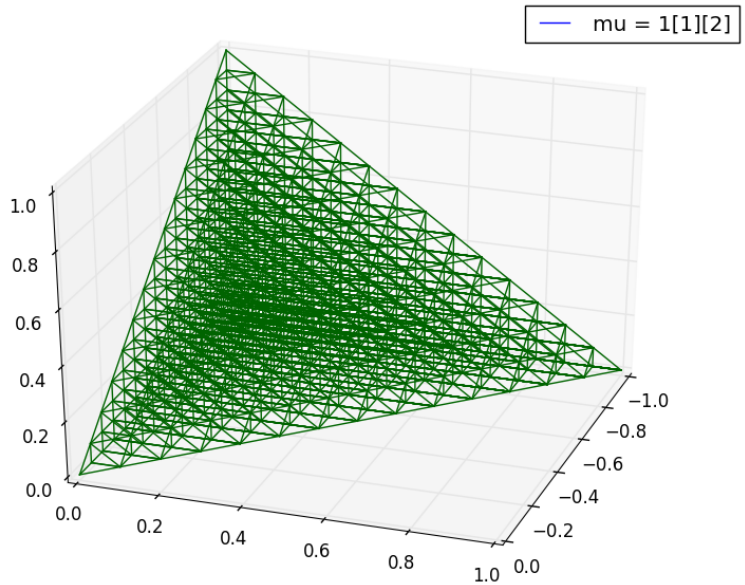


Subdomain with prisms, pyramids and tetrahedra

14 segments per edge. $14 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

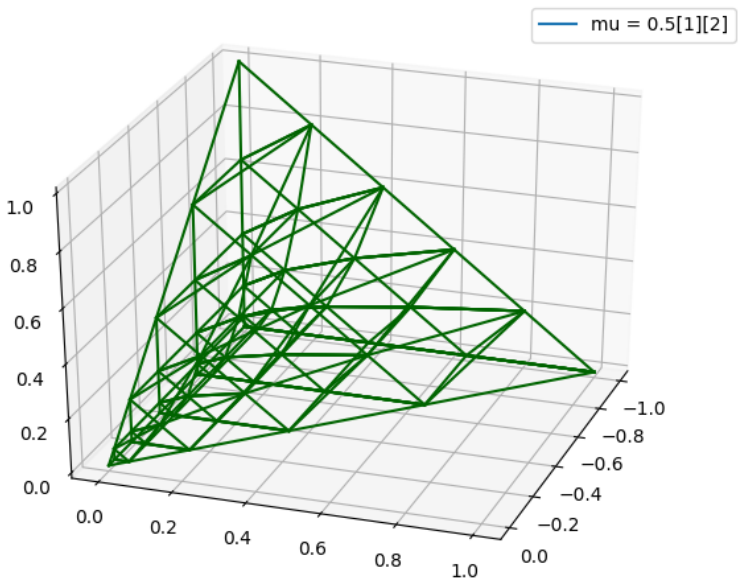


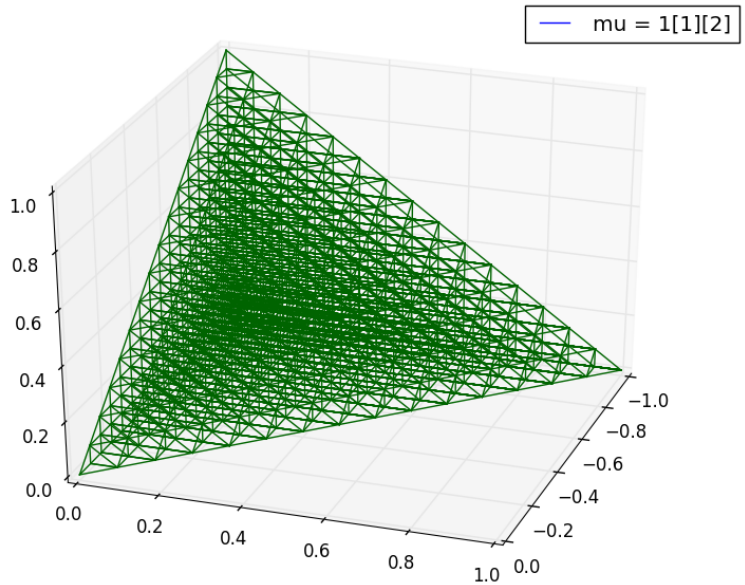


Subdomain with prisms, pyramids and tetrahedra

15 segments per edge. $15 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

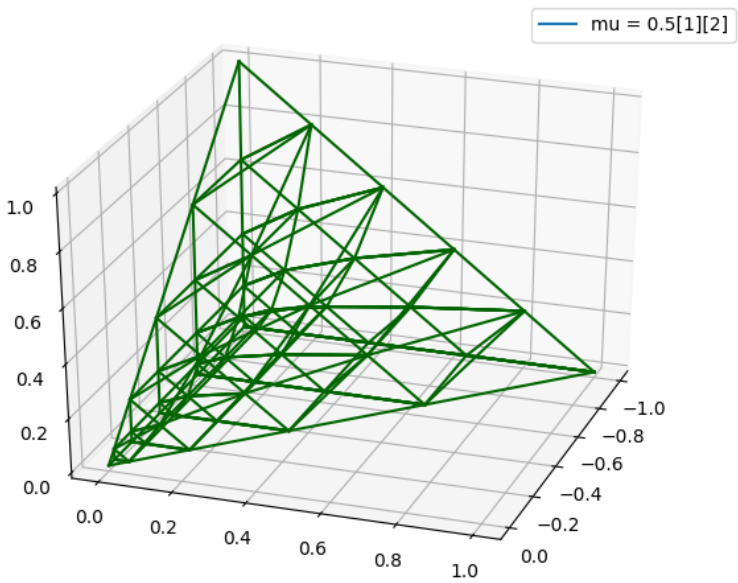


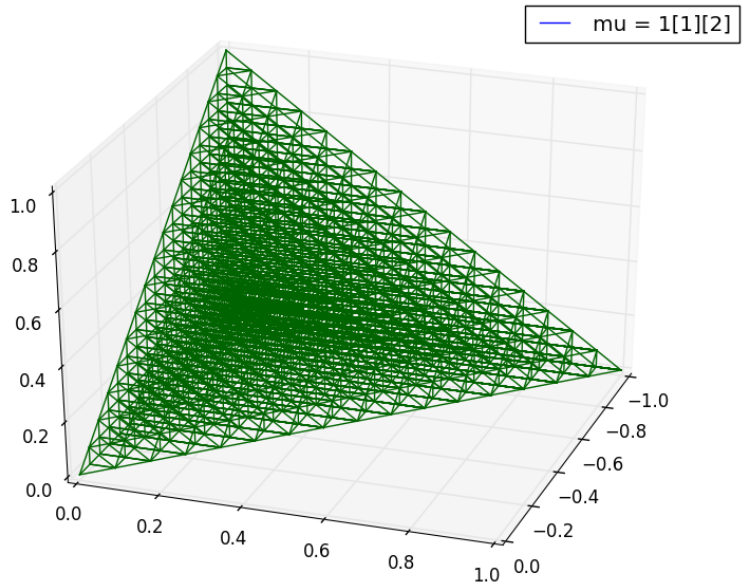


Subdomain with prisms, pyramids and tetrahedra

16 segments per edge. $16 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

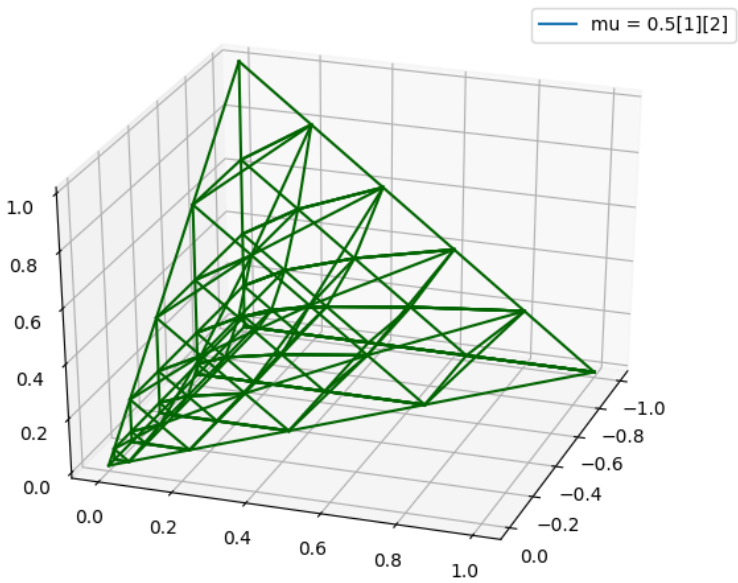


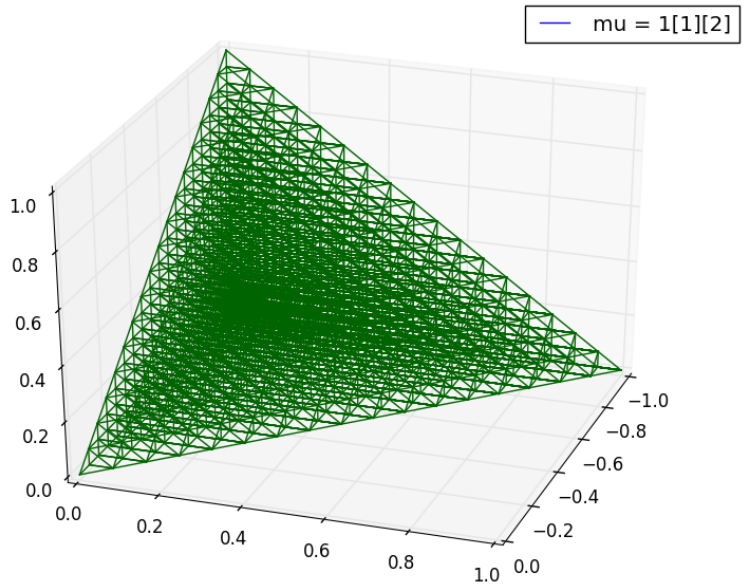


Subdomain with prisms, pyramids and tetrahedra

17 segments per edge. $17 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

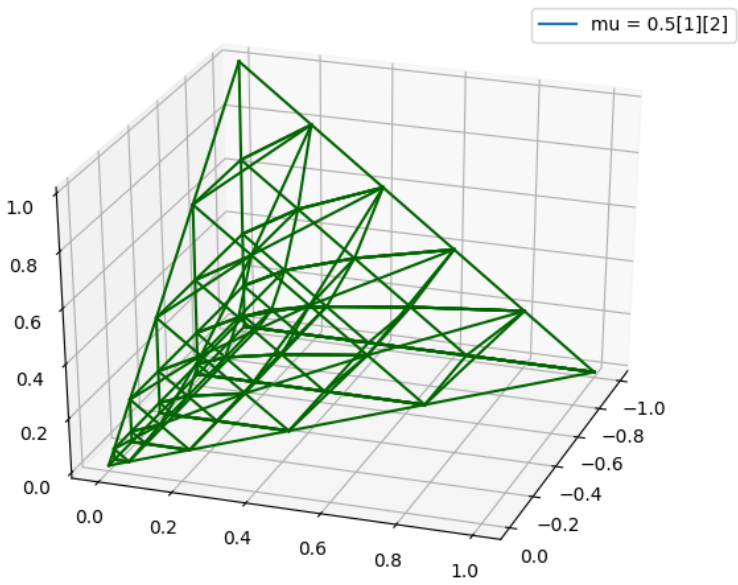


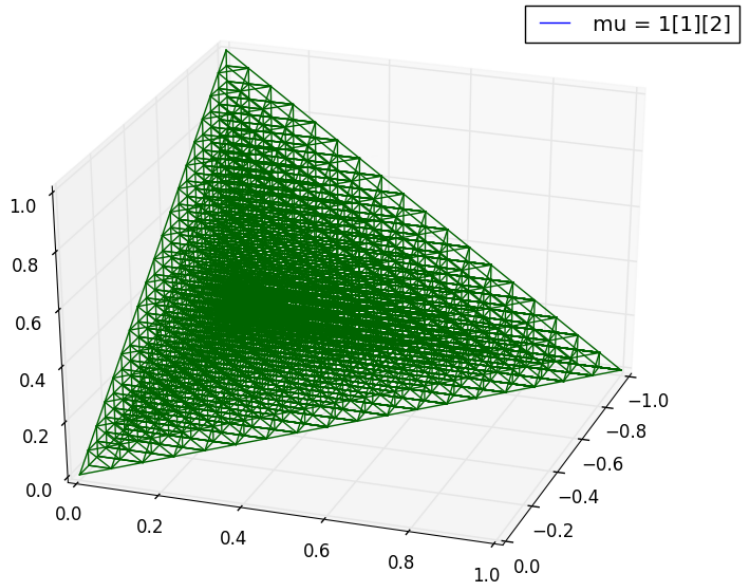


Subdomain with prisms, pyramids and tetrahedra

18 segments per edge. $18 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

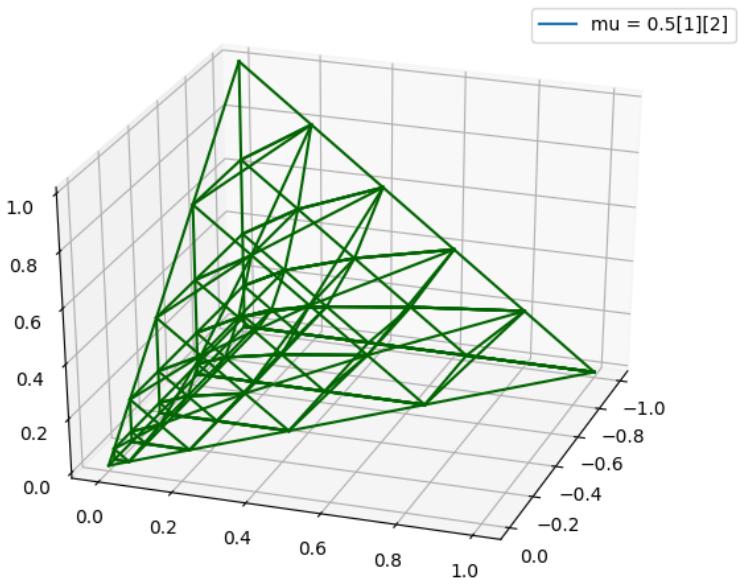


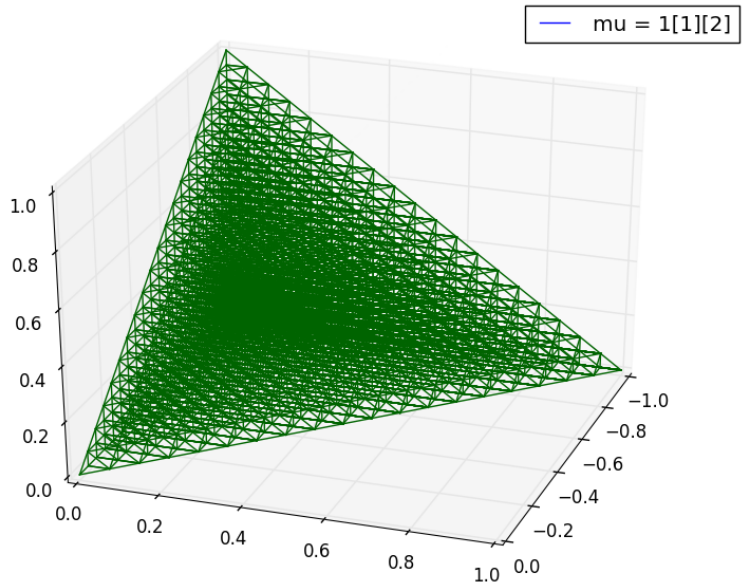


Subdomain with prisms, pyramids and tetrahedra

19 segments per edge. $19 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

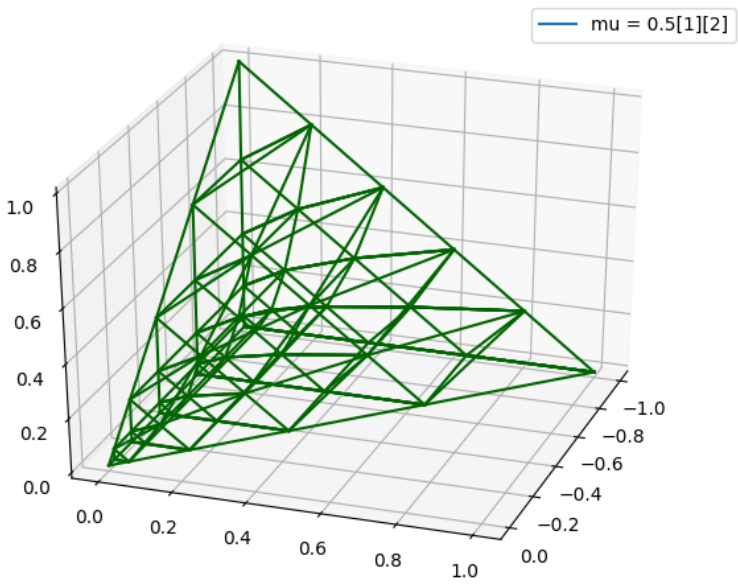


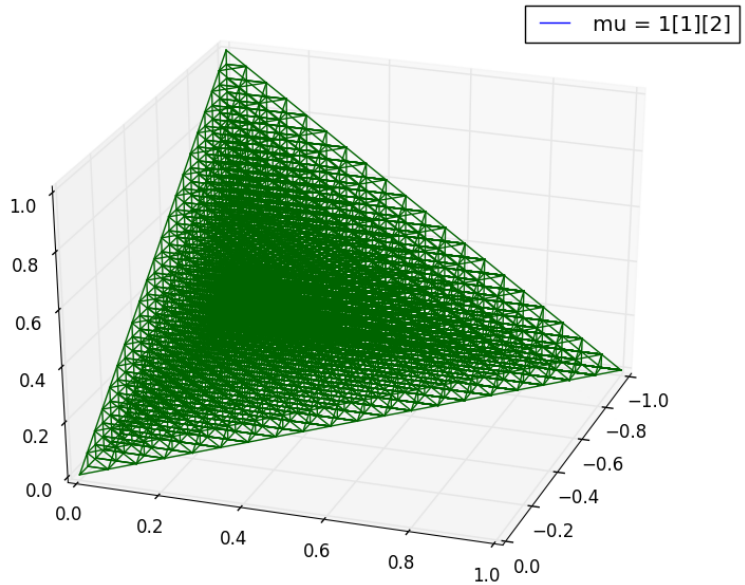


Subdomain with prisms, pyramids and tetrahedra

20 segments per edge. $20 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

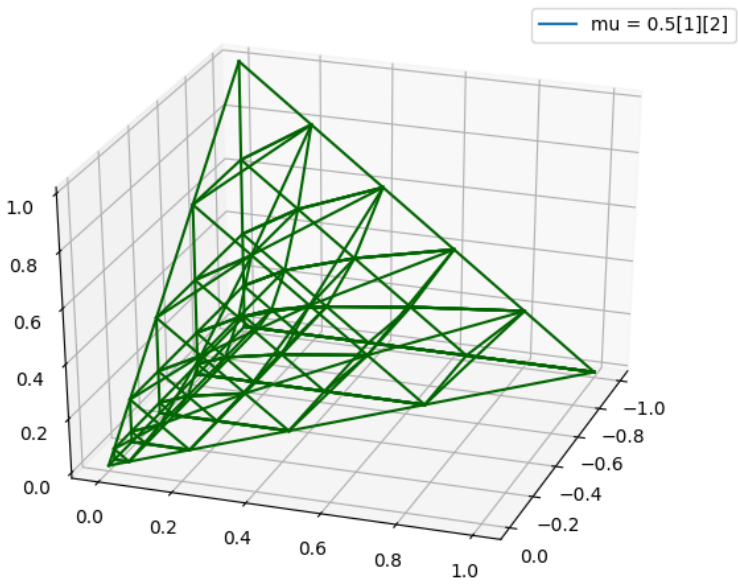


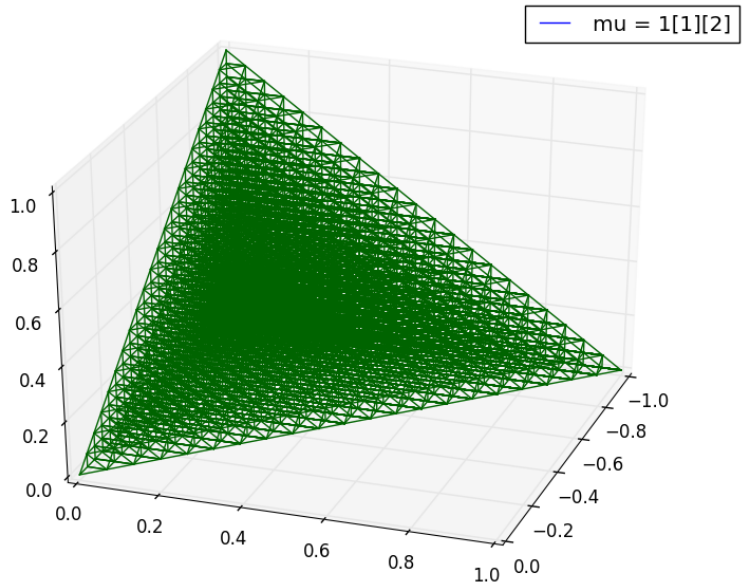


Subdomain with prisms, pyramids and tetrahedra

21 segments per edge. $21 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$

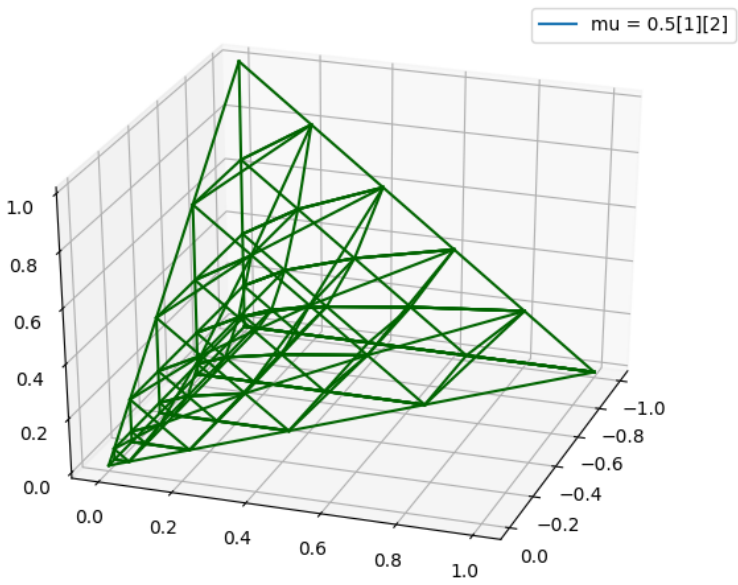


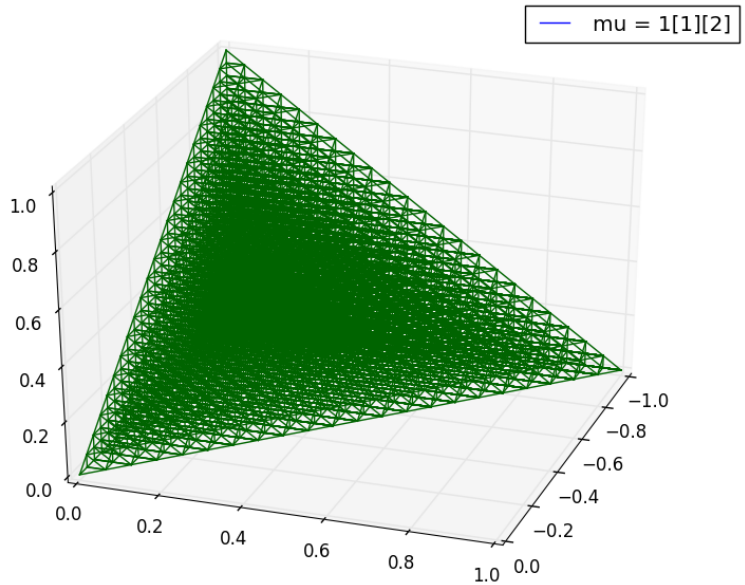


Subdomain with prisms, pyramids and tetrahedra

22 segments per edge. $22 + 1$ nodes per edge.

$h_i = h_i(E), \quad 1 \leqslant i \leqslant 3.$





Where . . .

$$h_1, h_2 \sim \begin{cases} (1/n)^{\frac{1}{\mu}} & \text{if } d(E, A_S) = 0 \\ (1/n)\, d(E, A_S)^{1-\mu} & \text{if } 0 < d(E, A_S) < 1 \end{cases}$$

$$h_3 \sim \begin{cases} (1/n)^{\frac{1}{\nu}} & \text{if } d(E, S) = 0 \\ (1/n)\, d(E, S)^{1-\nu} & \text{if } 0 < d(E, S) < 1 \end{cases}$$

Where . . .

$$h_1, h_2 \sim \begin{cases} (1/n)^{\frac{1}{\mu}} & \text{if } d(E, A_S) = 0 \\ (1/n)\, d(E, A_S)^{1-\mu} & \text{if } 0 < d(E, A_S) < 1 \end{cases}$$

$$h_3 \sim \begin{cases} (1/n)^{\frac{1}{\nu}} & \text{if } d(E, S) = 0 \\ (1/n)\, d(E, S)^{1-\nu} & \text{if } 0 < d(E, S) < 1 \end{cases}$$

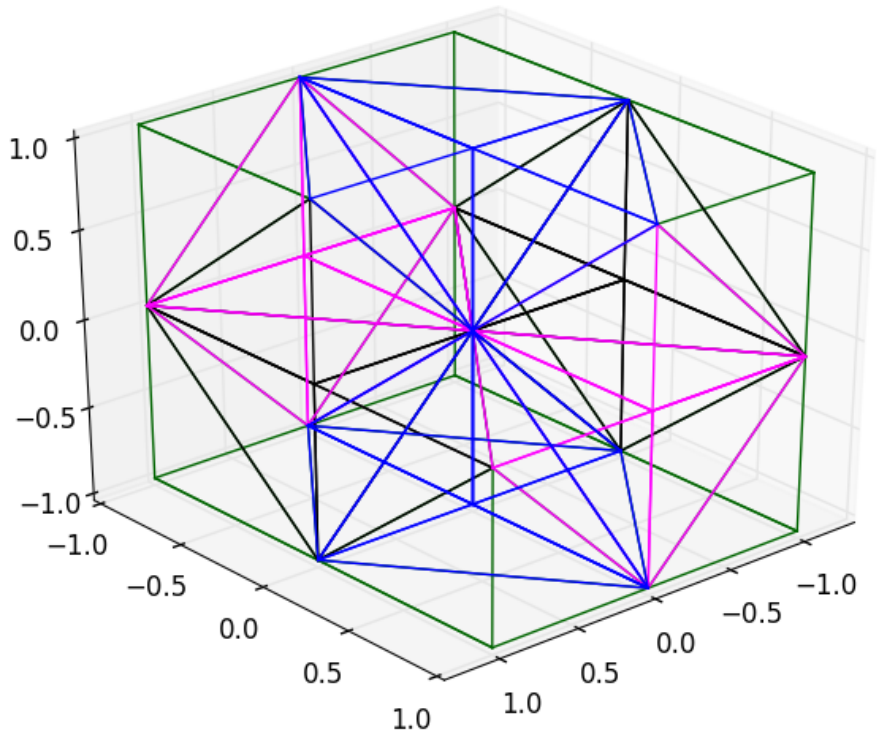
μ y ν parameters of the theory

Observaciones

- linear algebra: `NumPy`.
- singularities may be stronger or weaker in different regions of Ω.
- the meshing process allows for different parameters of refinement in different macro–elements.

Meshing algorithm
Example $[-1,1]^3 \setminus (0,1)^3$.

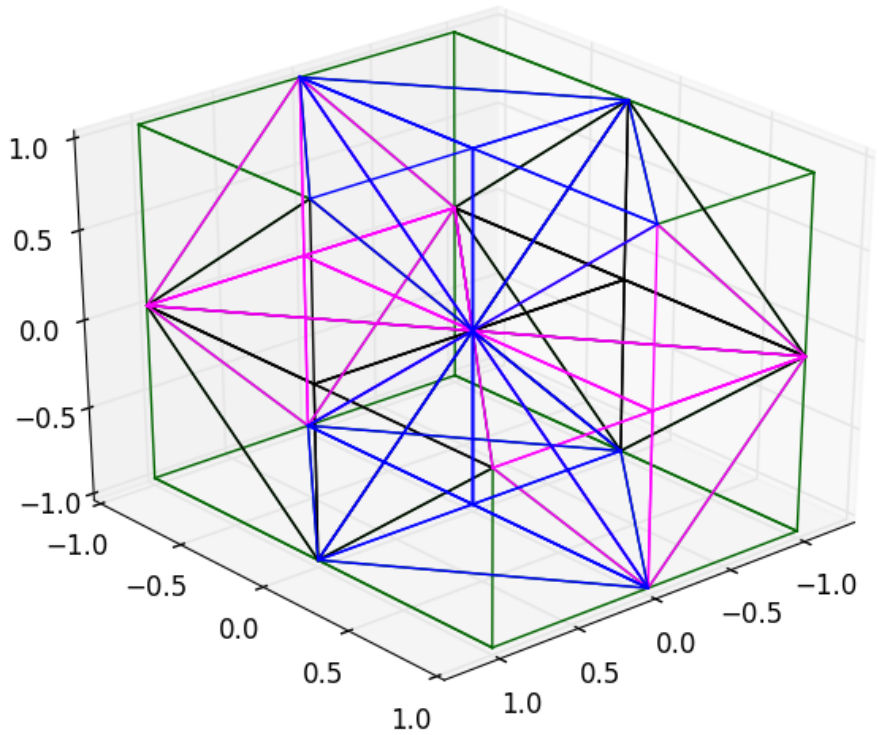
- https://github.com/alexisjawtu/mesher



Meshing algorithm
Example $[-1,1]^3 \setminus (0,1)^3$.

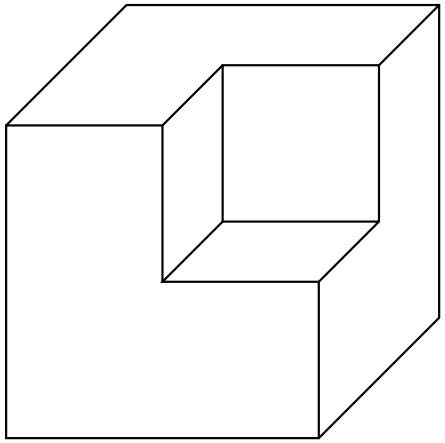
- https://github.com/alexisjawtu/mesher

  First partition in macro–tetrahedra or macro–prisms Λ_ℓ.

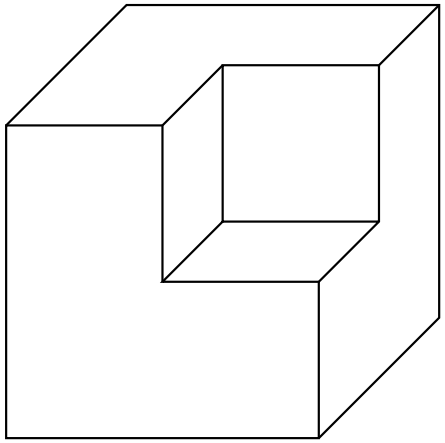


Meshing algorithm
Description/example Fichera Domain

- $[-1,1]^3 - (0,1)^3$

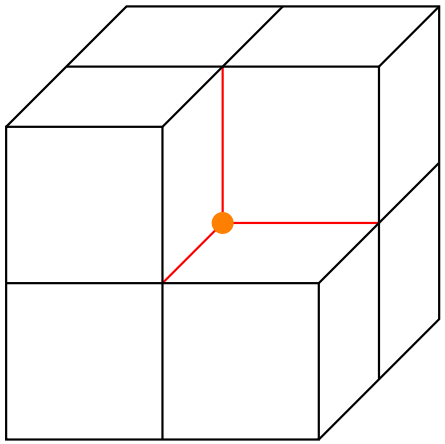


- $[-1,1]^3 - (0,1)^3$

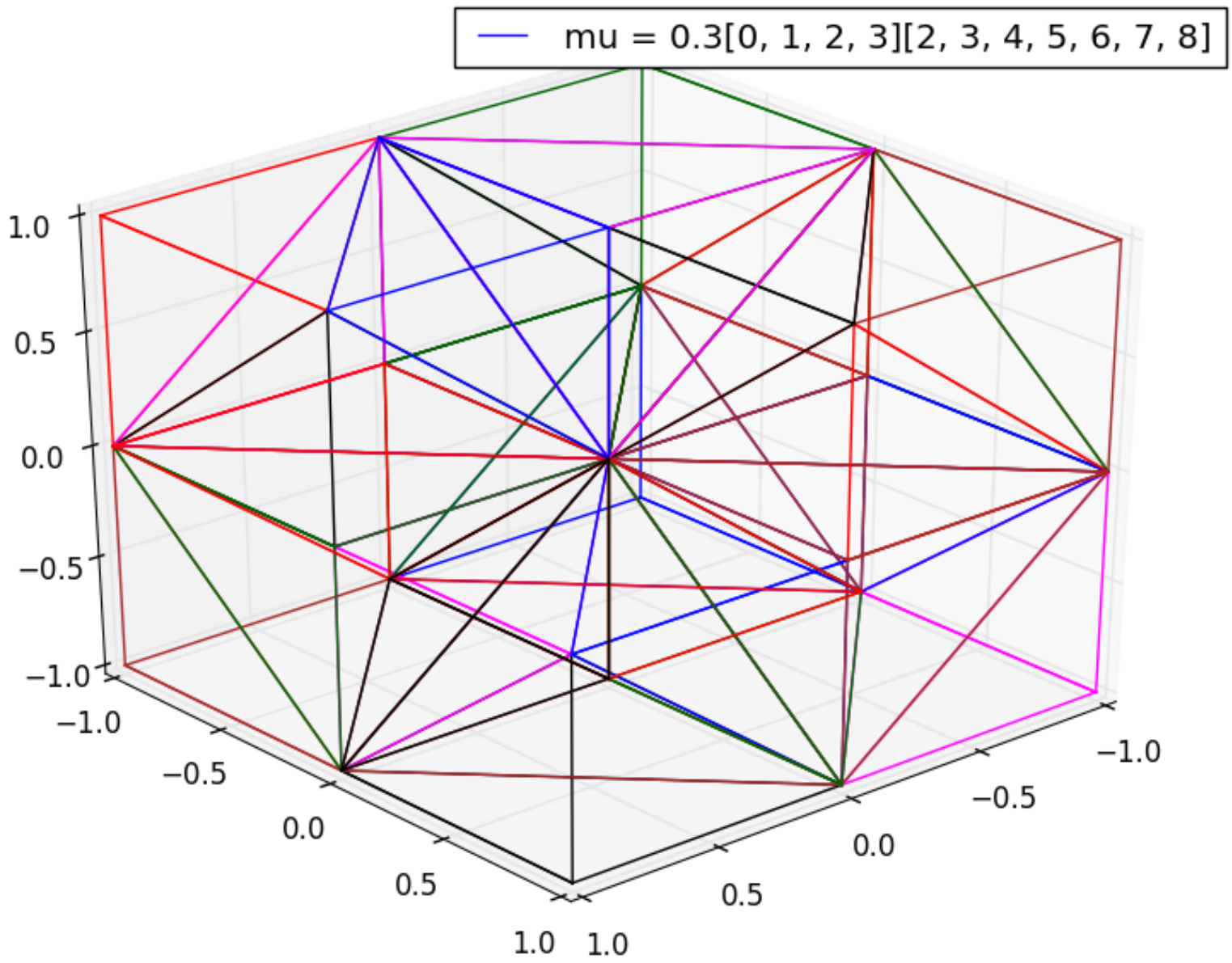


- $\Omega = \cup_{\ell=1}^{N} \overline{\Lambda_\ell}$ (tetrahedra or prisms).

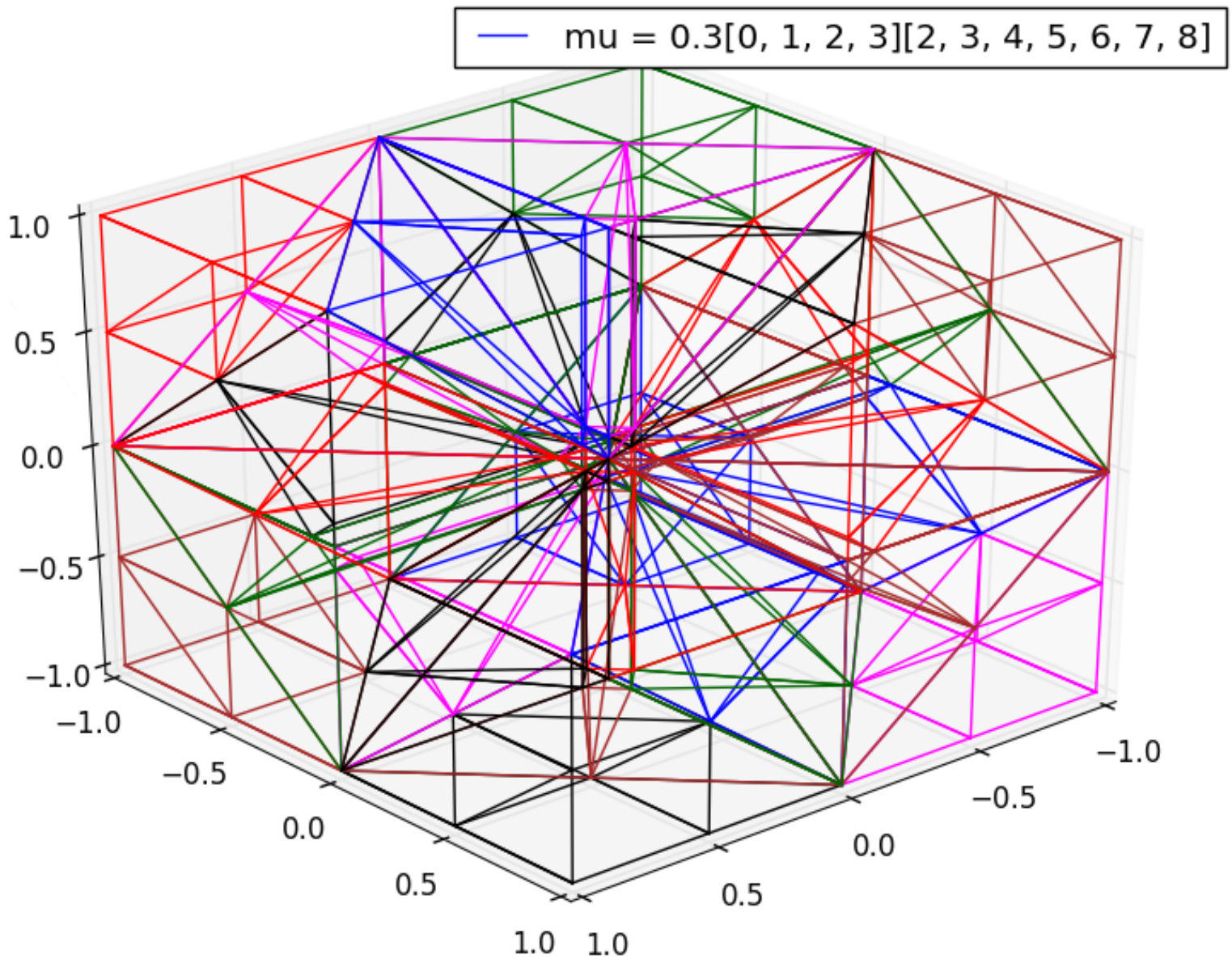
- Into 7 cubes:



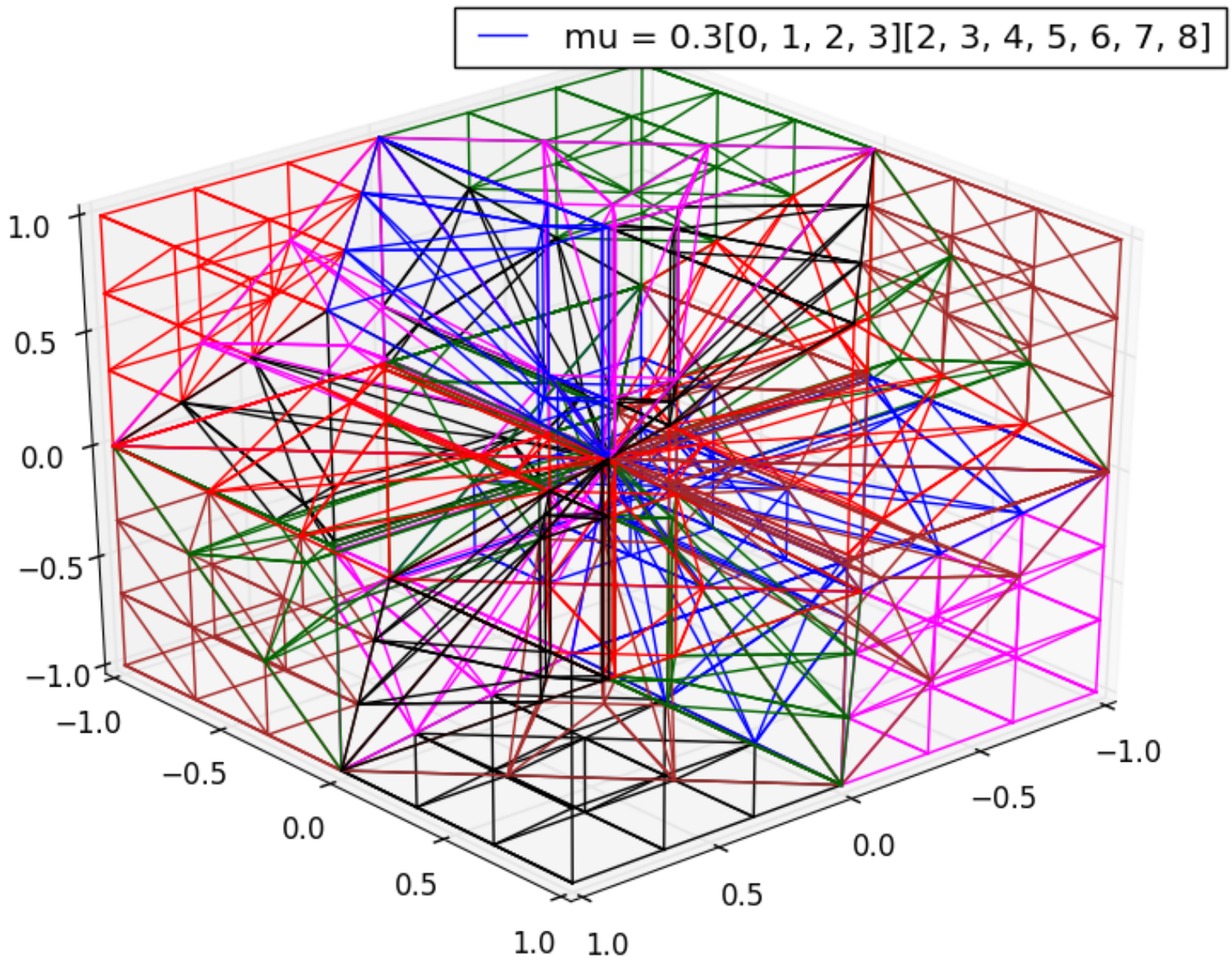
$\Omega \subseteq \mathbb{R}^3$
Fichera Domain



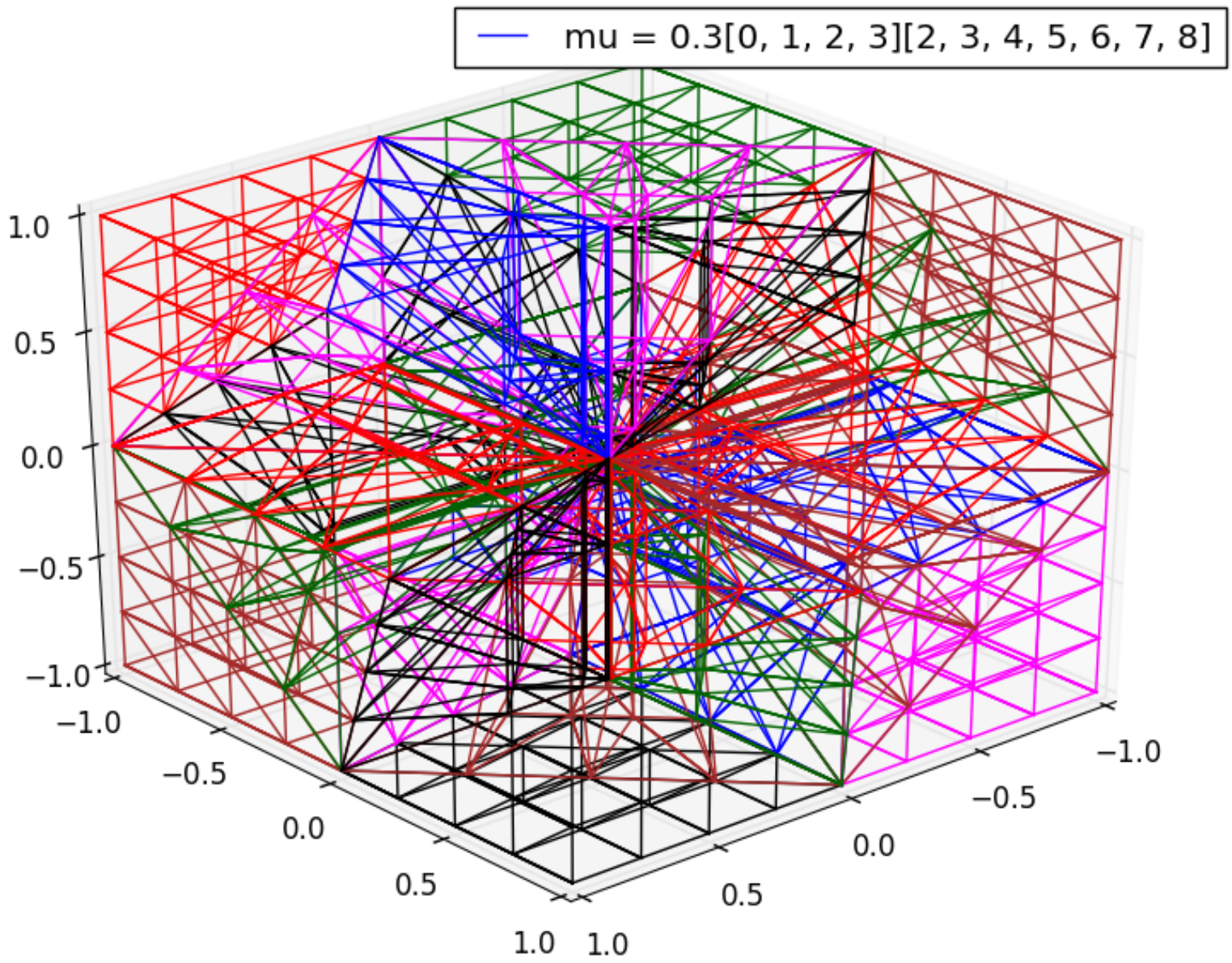
$\Omega \subseteq \mathbb{R}^3$
Fichera Domain



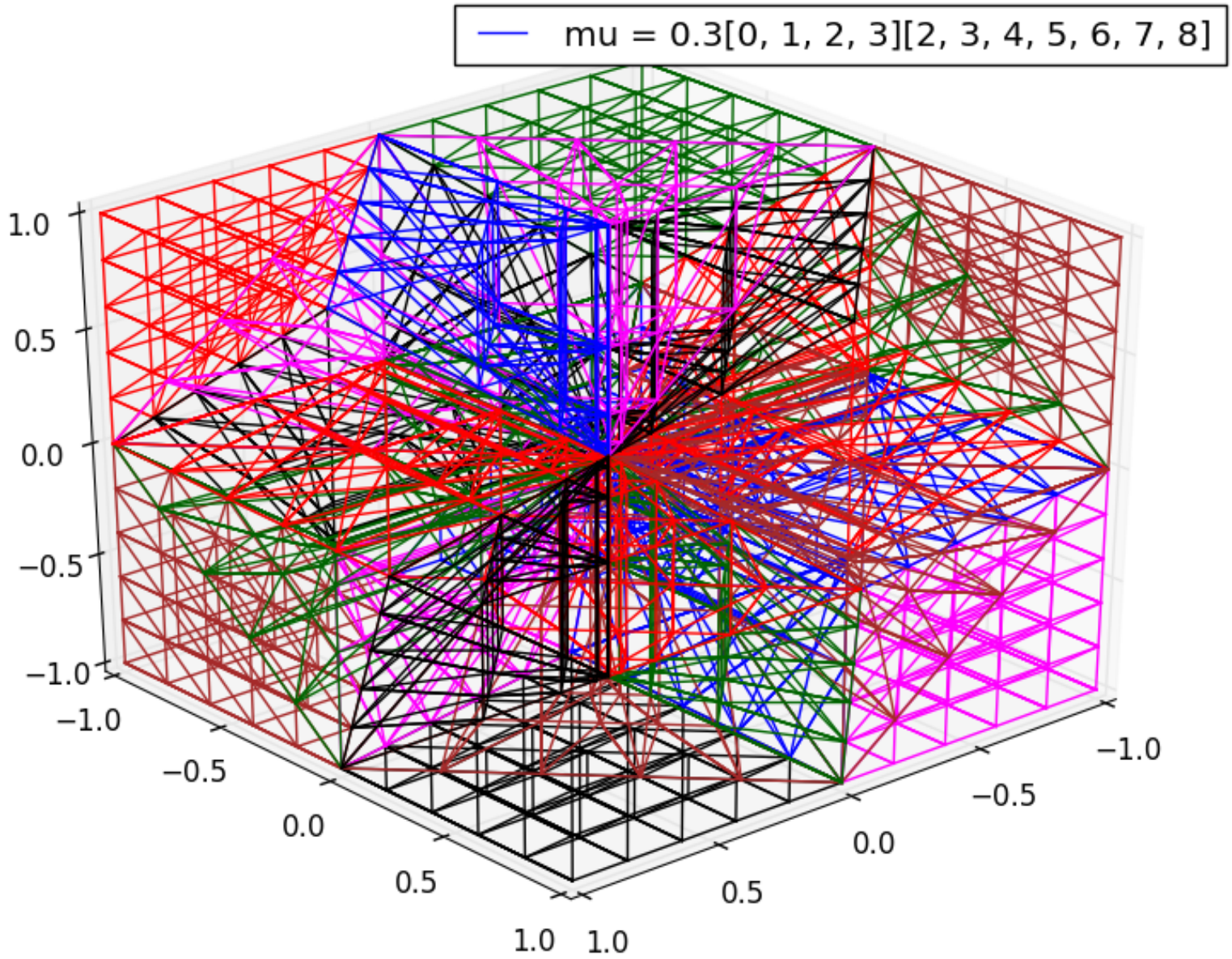
$\Omega \subseteq \mathbb{R}^3$
Fichera Domain



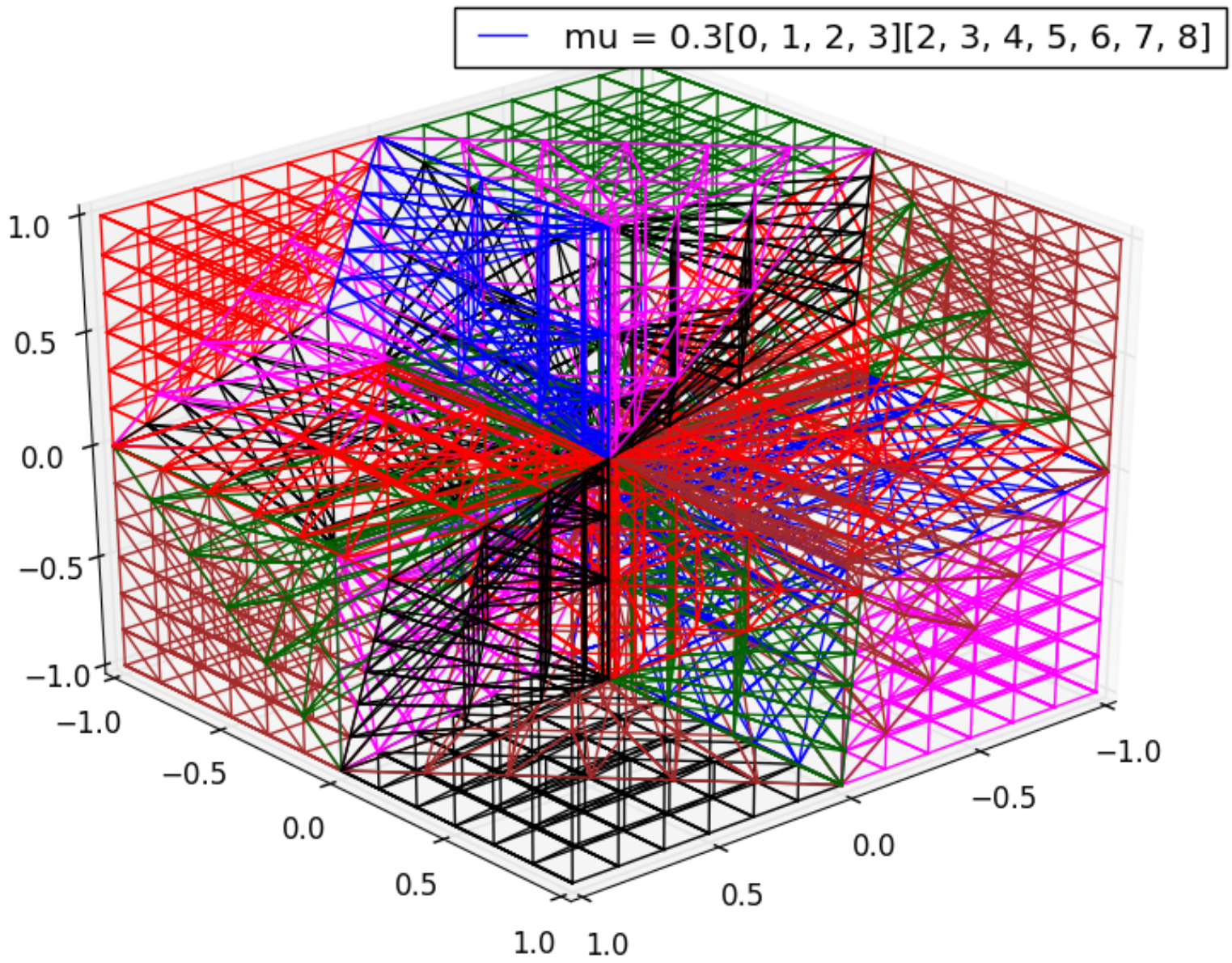
$\Omega \subseteq \mathbb{R}^3$
Fichera Domain

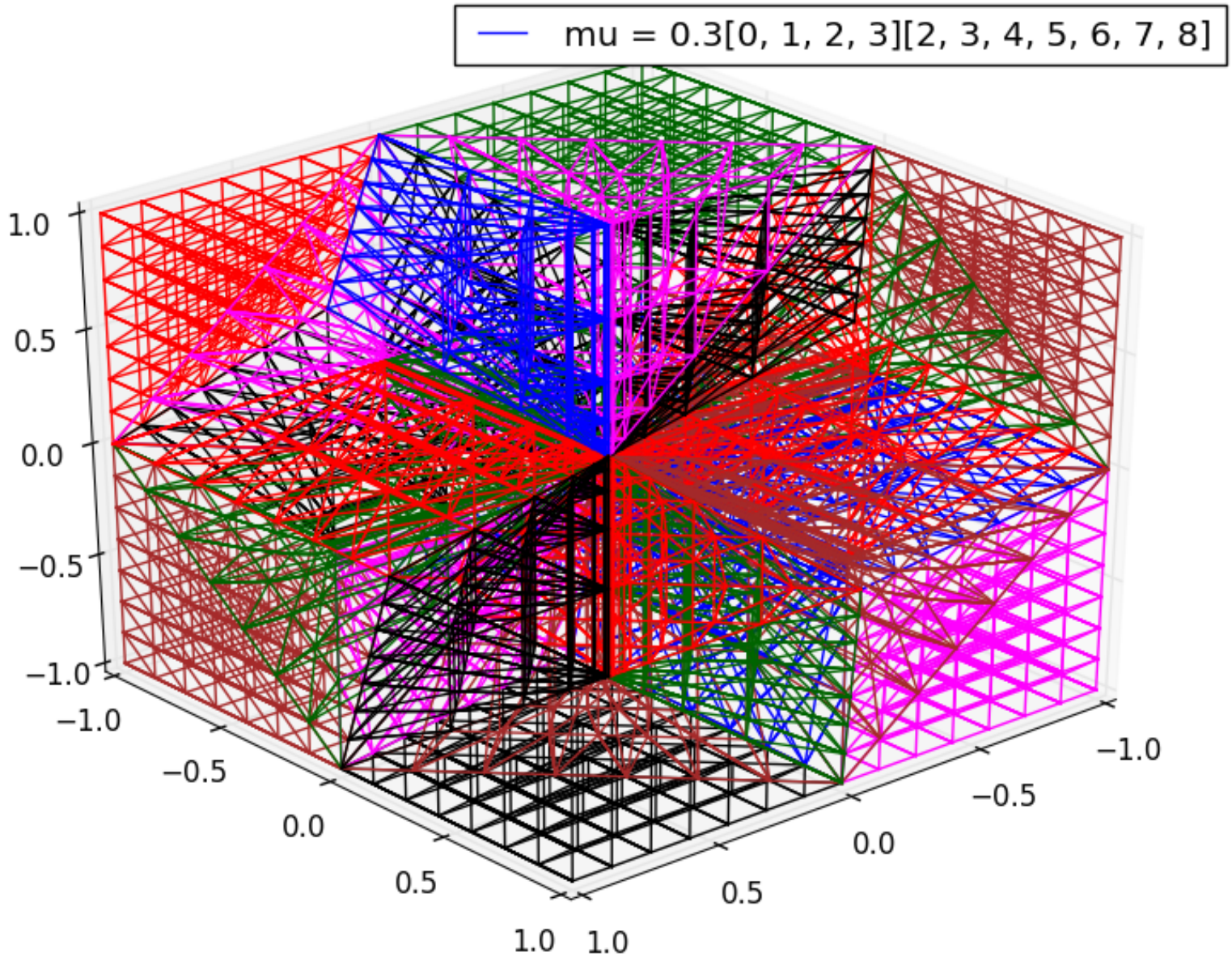


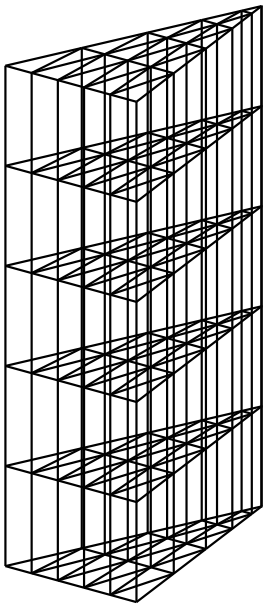
$\Omega \subseteq \mathbb{R}^3$
Fichera Domain

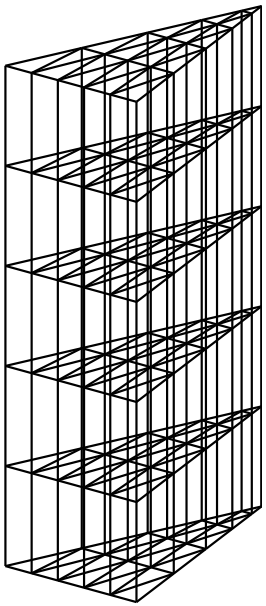


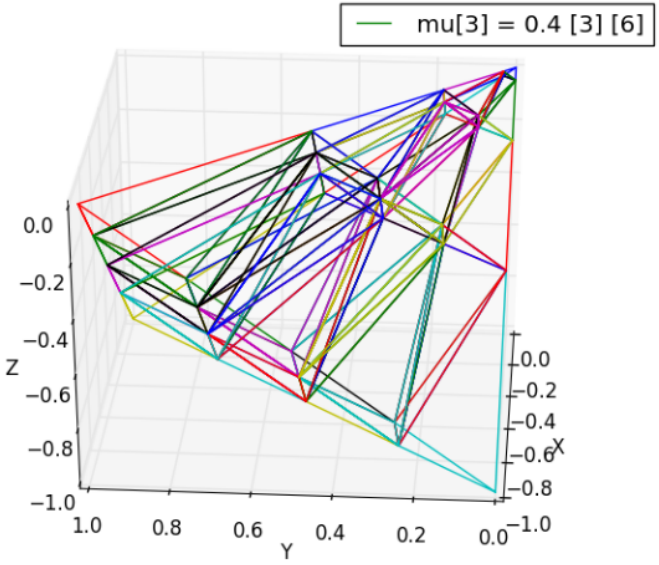
$\Omega \subseteq \mathbb{R}^3$
Fichera Domain

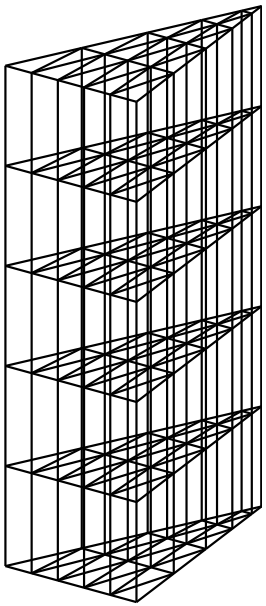


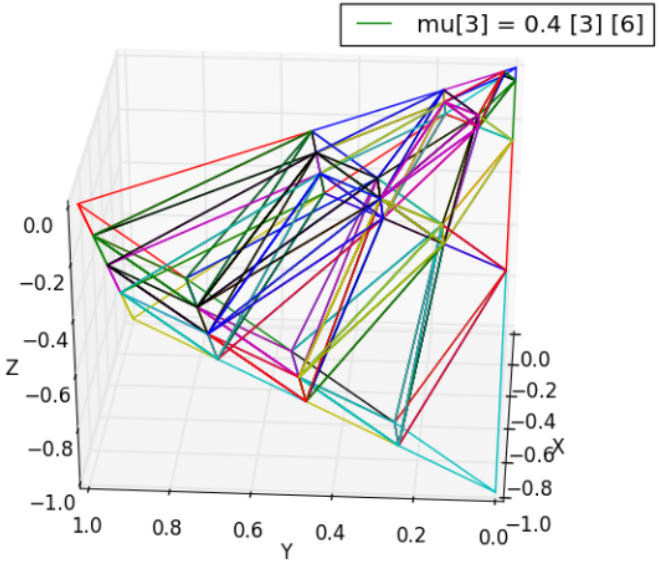

mu = 0.3[0, 1, 2, 3][2, 3, 4, 5, 6, 7, 8]
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.0
0.5
0.0
−0.5
−1.0

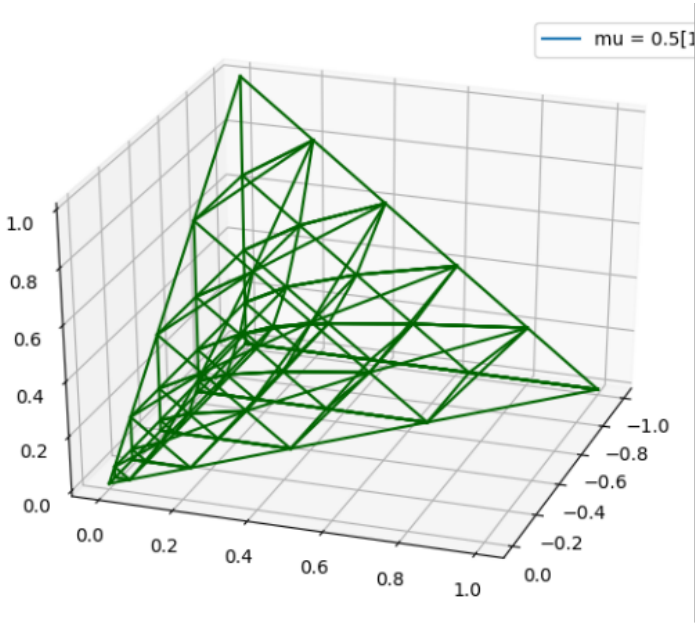


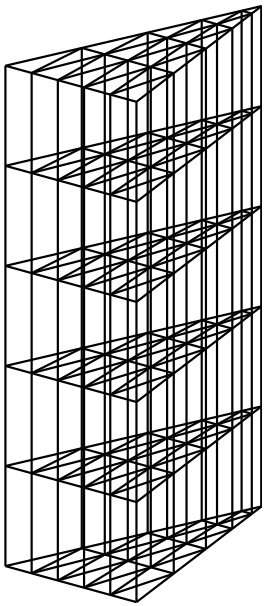


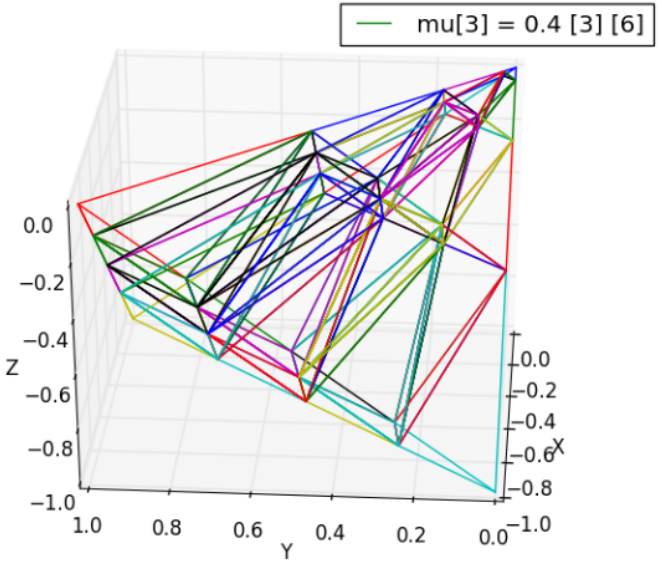

mu[3] = 0.4 [3] [6]
0.0
−0.2
−0.4
Z
−0.6
−0.8
−1.0
1.0
0.8
0.6
Y
0.4
0.2
0.0
0.0
−0.2
−0.4
−0.6
X
−0.8
−1.0




mu[3] = 0.4 [3] [6]
0.0
−0.2
−0.4
Z
−0.6
−0.8
−1.0
1.0
0.8
0.6
Y
0.4
0.2
0.0
0.0
−0.2
−0.4
−0.6
X
−0.8
−1.0


mu = 0.5[1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
−1.0
−0.8
−0.6
−0.4
−0.2
0.0




mu[3] = 0.4 [3] [6]
0.0
−0.2
−0.4
Z
−0.6
−0.8
−1.0
1.0
0.8
0.6
Y
0.4
0.2
0.0
0.0
−0.2
−0.4
−0.6
X
−0.8
−1.0

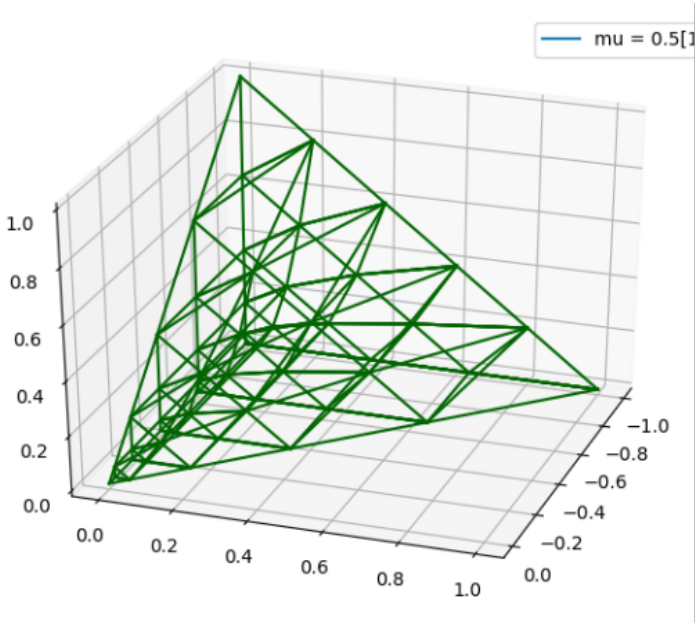

mu = 0.5[1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
−1.0
−0.8
−0.6
−0.4
−0.2
0.0

Table: Local complexity of the mesher

	std. tetrahedra	combined
Nmbr. of edges	$\frac{(n-1)n(7n+4)}{6}$	$\frac{(n-1)n(2n+5)}{3}$

Table: Local complexity of the mesher

	std. tetrahedra	combined
Nmbr. of edges	$\frac{(n-1)n(7n+4)}{6}$	$\frac{(n-1)n(2n+5)}{3}$
leading term	$\frac{7}{6}n^3$	$\frac{2}{3}n^3$

Table: Local complexity of the mesher

	std. tetrahedra	combined
Nmbr. of edges	$\frac{(n-1)n(7n+4)}{6}$	$\frac{(n-1)n(2n+5)}{3}$
leading term	$\frac{7}{6}n^3$	$\frac{2}{3}n^3$
Nmbr. of faces	$\frac{(n-1)(5n^2-9n+6)}{2}$	$\frac{(n-1)(5n^2+5n-6)}{6}$

Table: Local complexity of the mesher

	std. tetrahedra	combined
Nmbr. of edges	$\frac{(n-1)n(7n+4)}{6}$	$\frac{(n-1)n(2n+5)}{3}$
leading term	$\frac{7}{6}n^3$	$\frac{2}{3}n^3$
Nmbr. of faces	$\frac{(n-1)(5n^2-9n+6)}{2}$	$\frac{(n-1)(5n^2+5n-6)}{6}$
leading term	$\frac{5}{2}n^3$	$\frac{5}{6}n^3$

Table: Local complexity of the mesher

	std. tetrahedra	combined
Nmbr. of edges	$\frac{(n-1)n(7n+4)}{6}$	$\frac{(n-1)n(2n+5)}{3}$
leading term	$\frac{7}{6}n^3$	$\frac{2}{3}n^3$
Nmbr. of faces	$\frac{(n-1)(5n^2-9n+6)}{2}$	$\frac{(n-1)(5n^2+5n-6)}{6}$
leading term	$\frac{5}{2}n^3$	$\frac{5}{6}n^3$
Nmbr. of elements	$(n-1)^3$	$\frac{(n-1)n(2n-1)}{6}$

Table: Local complexity of the mesher

	std. tetrahedra	combined
Nmbr. of edges	$\frac{(n-1)n(7n+4)}{6}$	$\frac{(n-1)n(2n+5)}{3}$
leading term	$\frac{7}{6}n^3$	$\frac{2}{3}n^3$
Nmbr. of faces	$\frac{(n-1)(5n^2-9n+6)}{2}$	$\frac{(n-1)(5n^2+5n-6)}{6}$
leading term	$\frac{5}{2}n^3$	$\frac{5}{6}n^3$
Nmbr. of elements	$(n-1)^3$	$\frac{(n-1)n(2n-1)}{6}$
leading term	n^3	$\frac{2}{6}n^3$

other good features
why did I do it?

- This is deterministic.

other good features
why did I do it?

- This is deterministic.
- The whole thing is of a recursive nature

other good features
why did I do it?

- This is deterministic.
- The whole thing is of a recursive nature
- This is free (beer/freedom).

other good features
why did I do it?

- This is deterministic.
- The whole thing is of a recursive nature
- This is free (beer/freedom).
- Commercial meshers use heuristics

other good features
why did I do it?

- This is deterministic.
- The whole thing is of a recursive nature
- This is free (beer/freedom).
- Commercial meshers use heuristics
- Commercial meshers give degenerate polyhedra (Volume 0)

other good features
why did I do it?

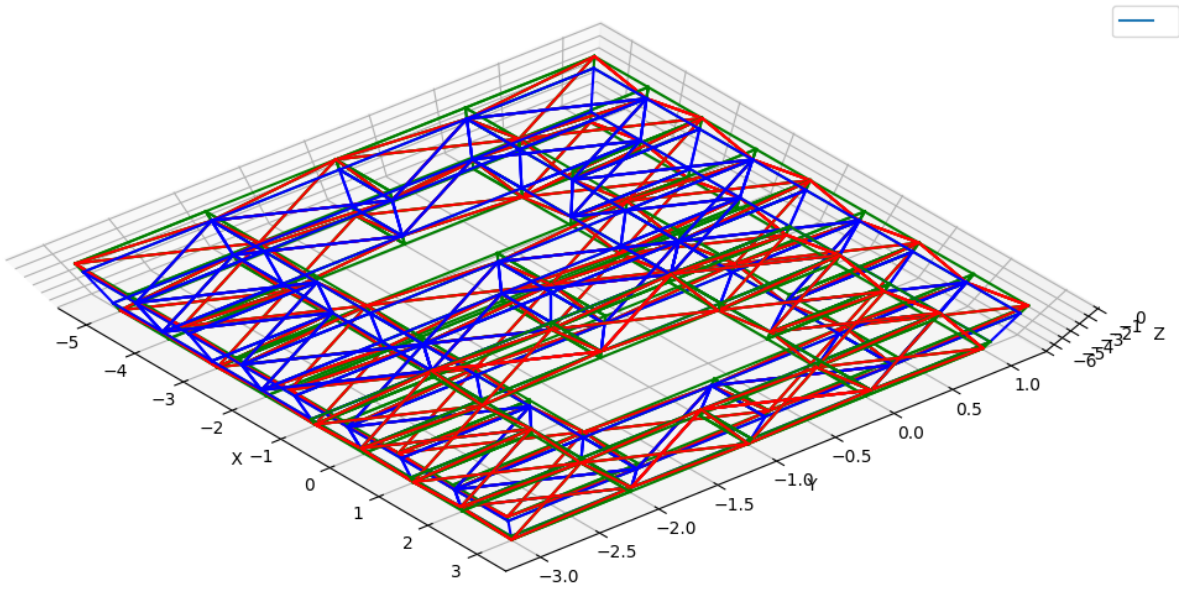
- This is deterministic.
- The whole thing is of a recursive nature
- This is free (beer/freedom).
- Commercial meshers use heuristics
- Commercial meshers give degenerate polyhedra (Volume 0)
- This is fun.

other good features
why did I do it?

- This is deterministic.
- The whole thing is of a recursive nature
- This is free (beer/freedom).
- Commercial meshers use heuristics
- Commercial meshers give degenerate polyhedra (Volume 0)
- This is fun.
- I was disobedient.

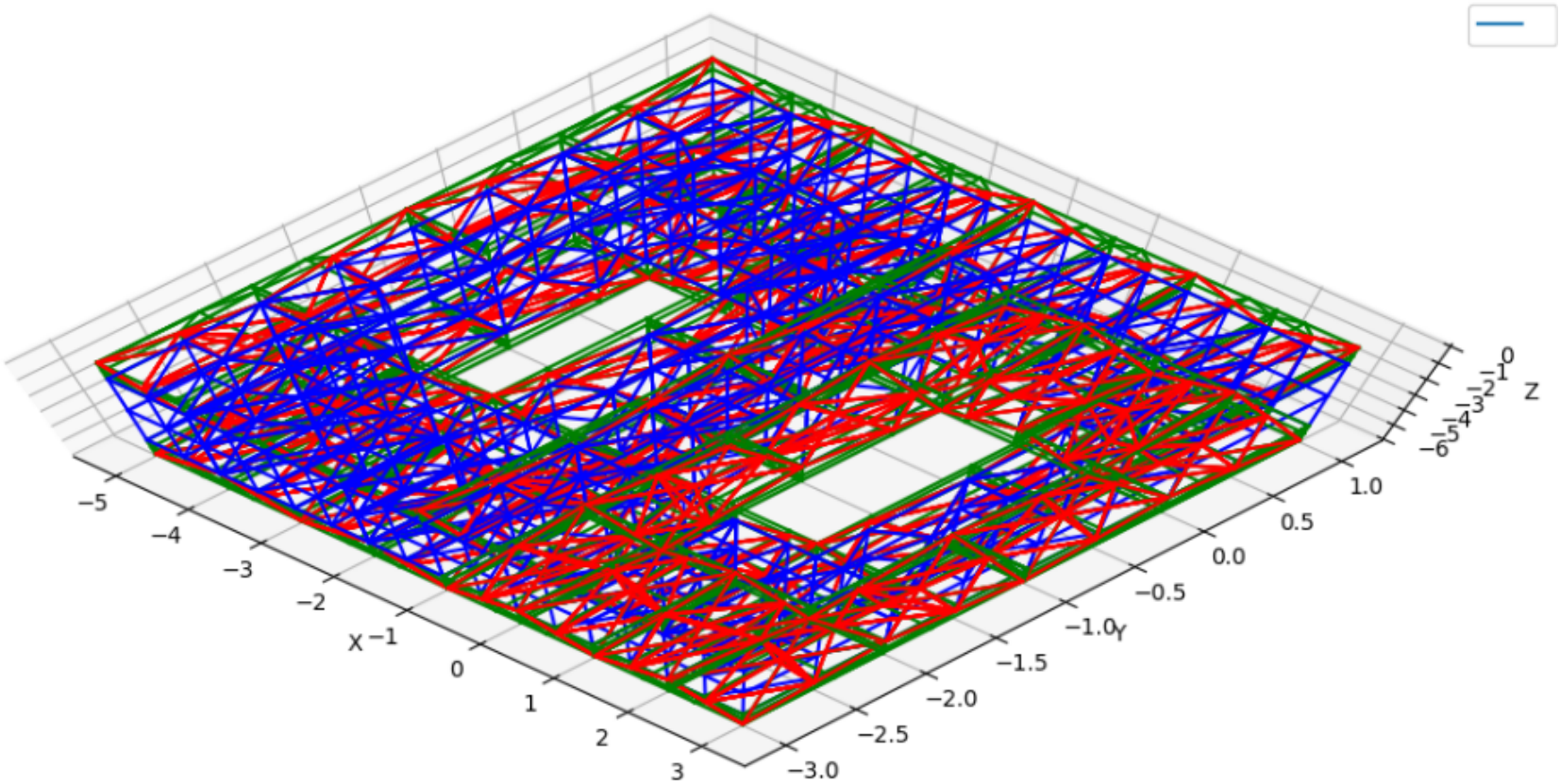
Example

Another domain that is frequent in articles of Numerical Analysis



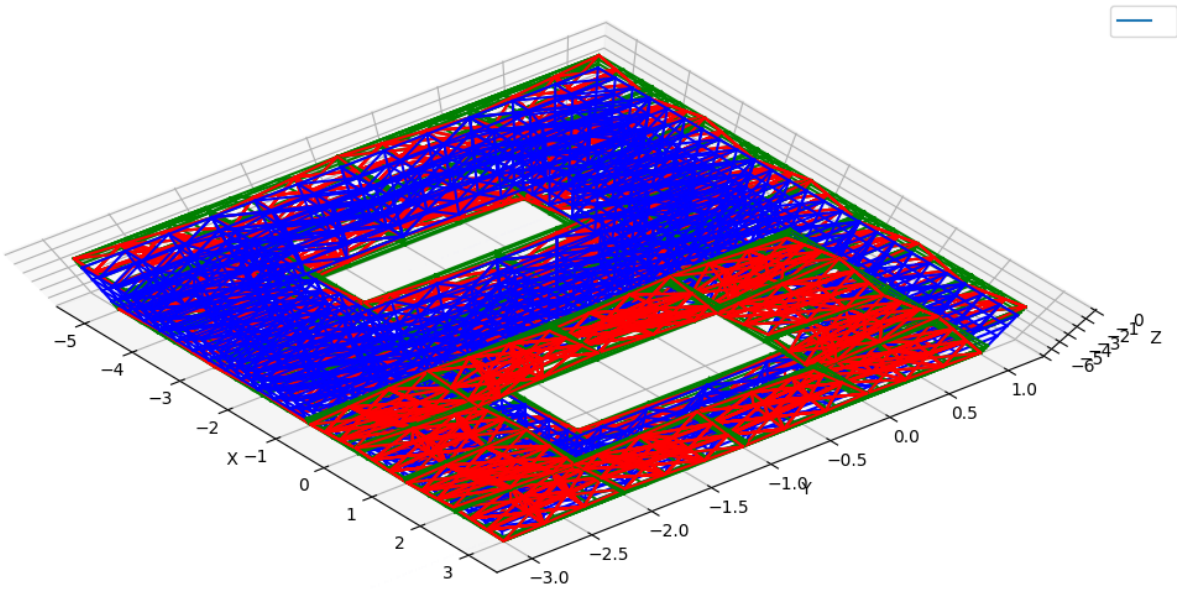
Example

Another domain that is frequent in articles of Numerical Analysis



Example

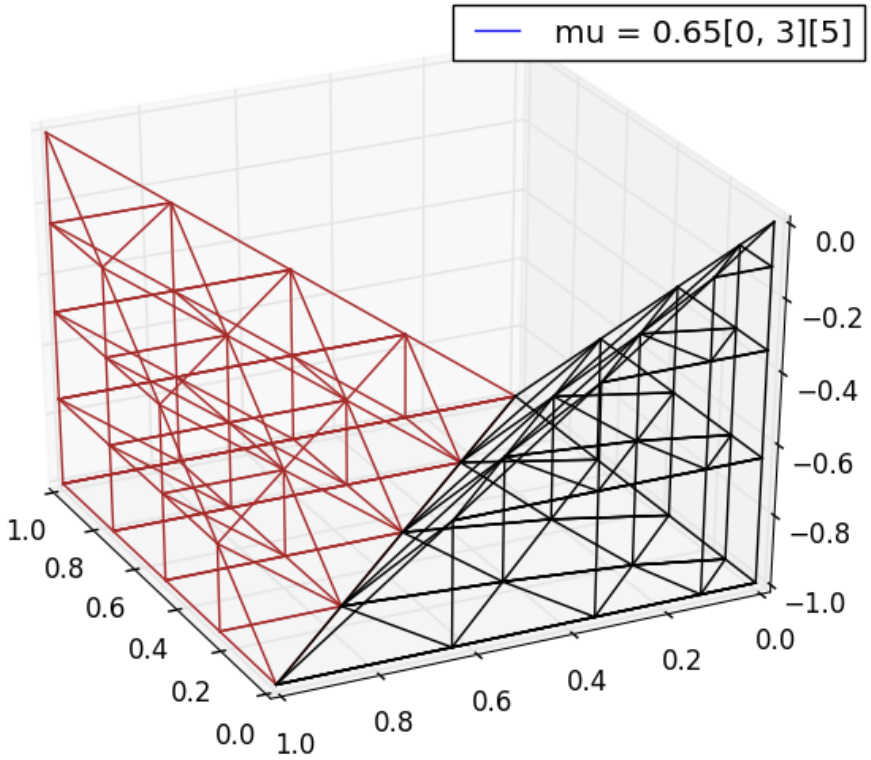
Another domain that is frequent in articles of Numerical Analysis



Conformity

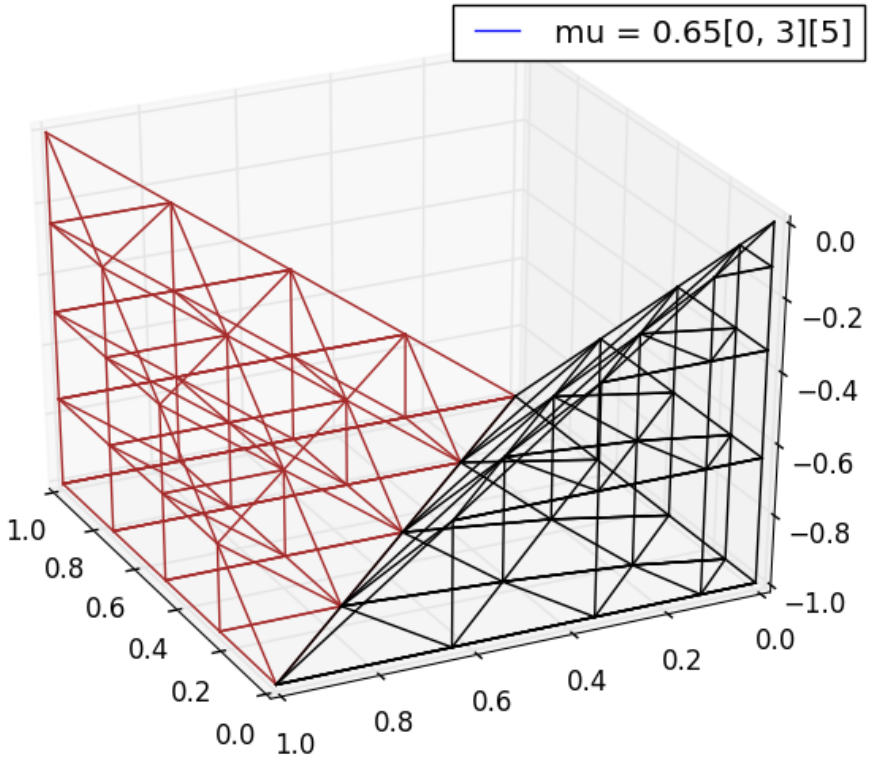
- pictures with GnuPlot, real time examples with mayavi

Conformity
by edges



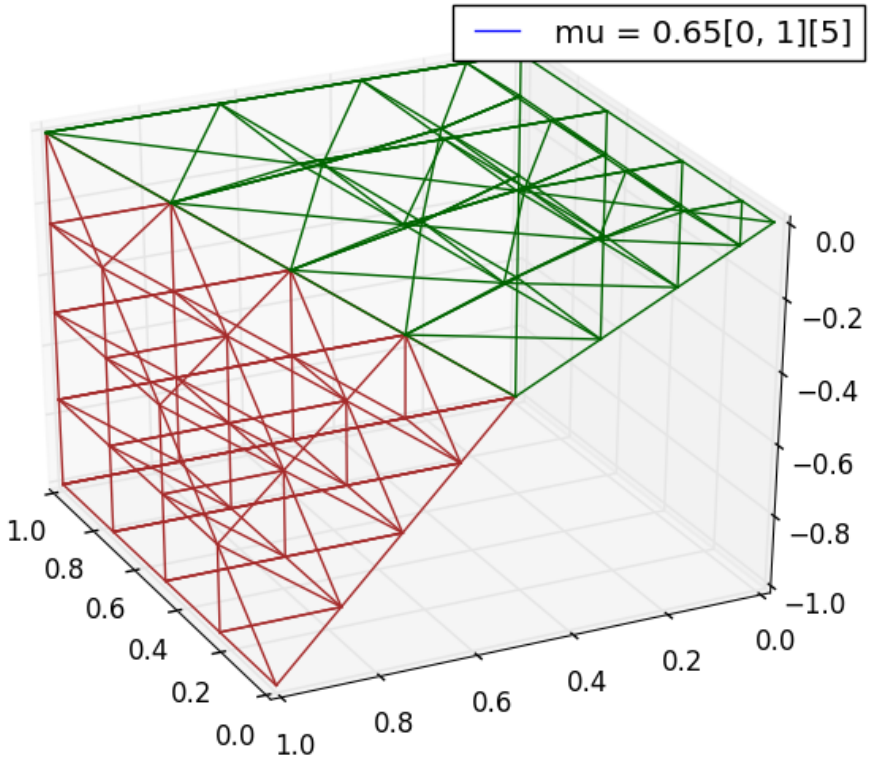
- Conformity achieved grading two neighbour macro–elements.

Conformity
by edges

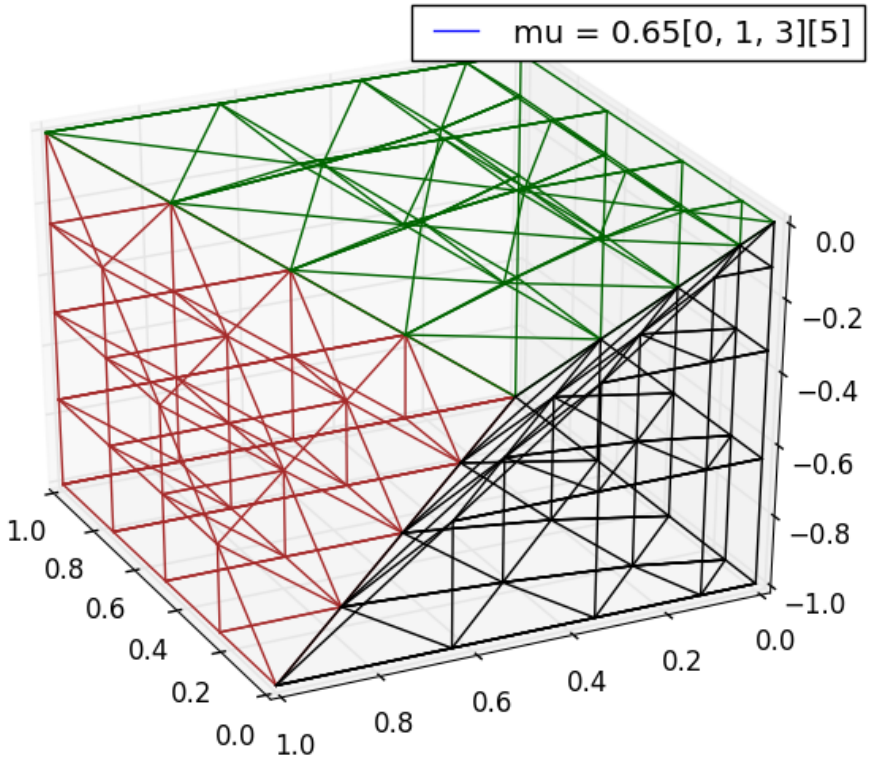


- Conformity achieved grading two neighbour macro–elements.
- grading is independent; it must be consistent.

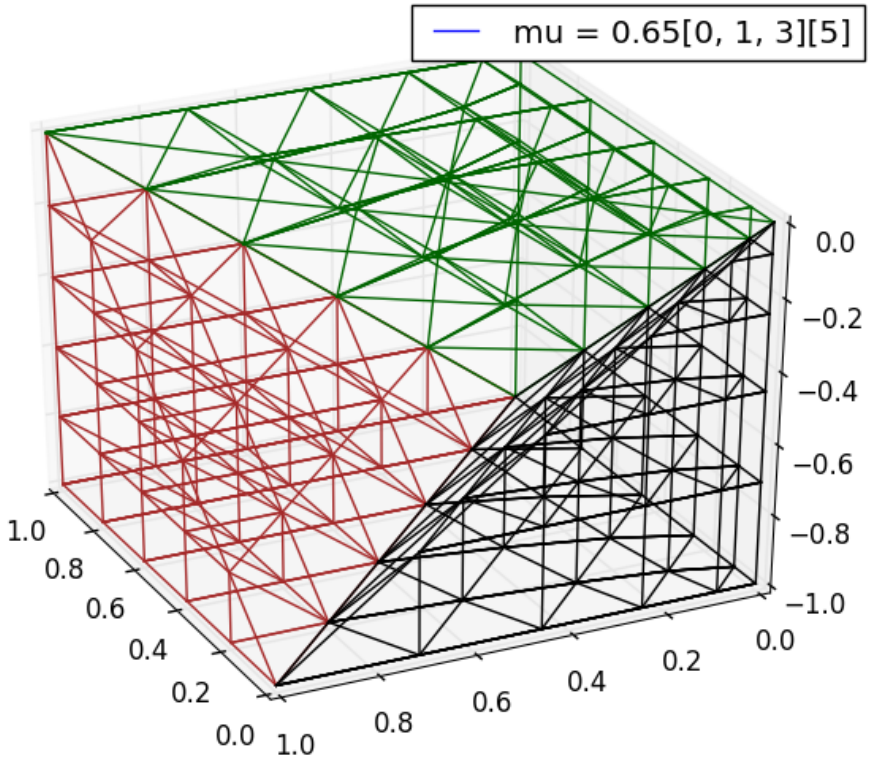
Conformity
by edges



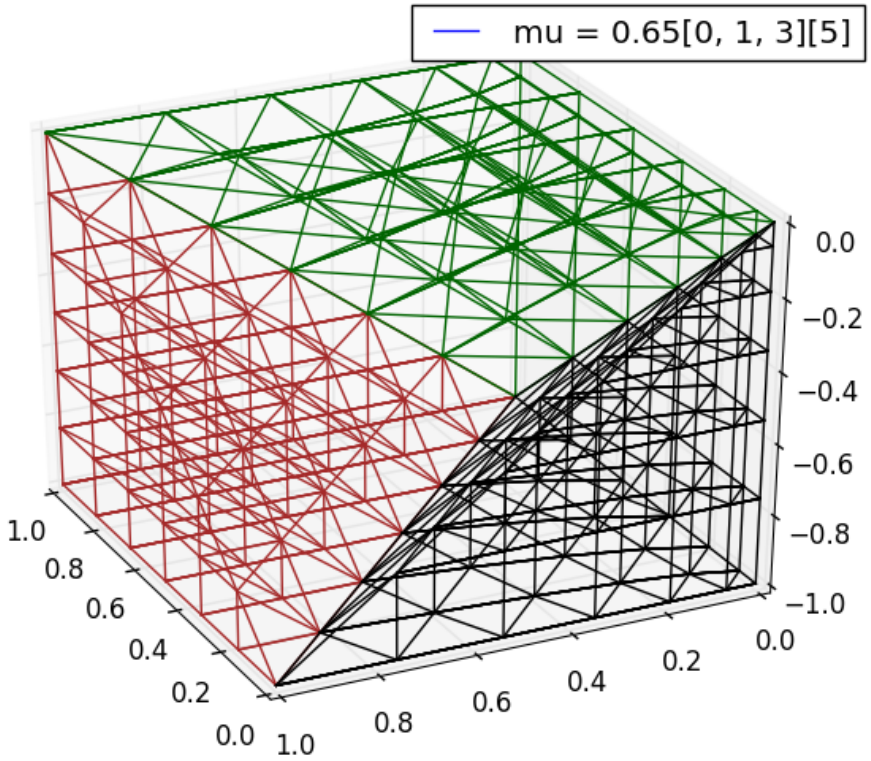
Conformity
by edges



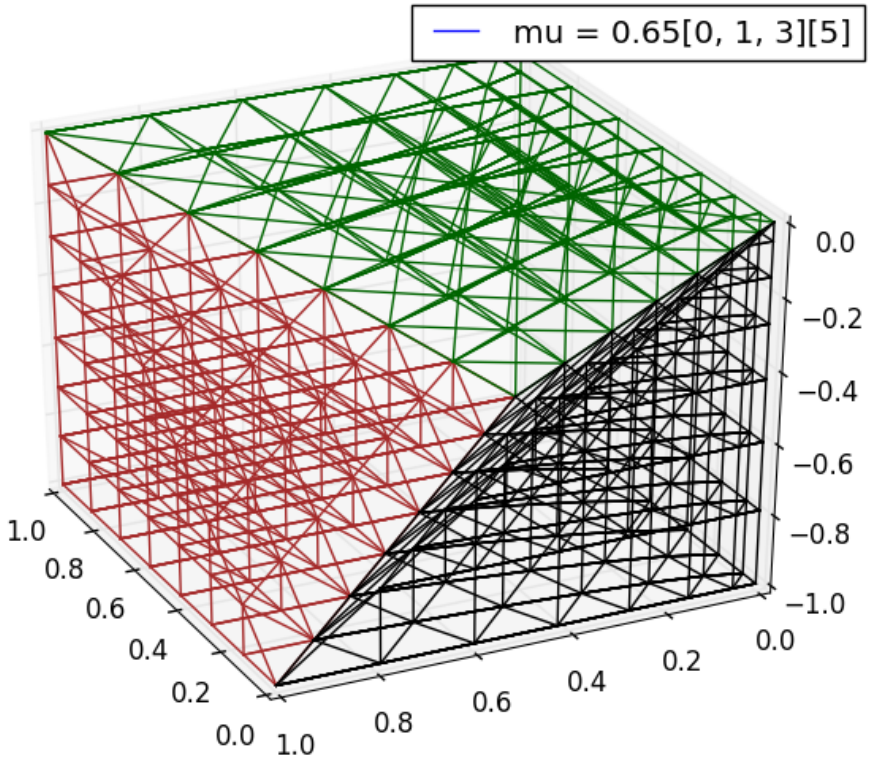
Conformity
by edges



Conformity
by edges



Conformity
by edges



Conformity
by faces

write in a csv:
0,-1, 0, 0,-1, 1, 0,-1, 0, 1, 0, 0, 0,0.65
0,-1, 0, 0,-1,-1, 0,-1, 0, 1, 0, 0, 0,0.65
orig x y sing.

A simple real time example (if time allows us)

(part of the) output

two decimals

partition.ver

```
 1. -1.00  1.00  1.00
 2. -1.00  1.00  0.50
 3. -1.00  1.00  0.00
 4. -1.00  0.50  1.00
 5. -1.00  0.50  0.50
 6. -1.00  0.00  1.00
 7. -0.50  1.00  1.00
 8. -0.50  1.00  0.50
 9. -0.50  0.50  1.00
10.  0.00  1.00  1.00
11. -1.00  0.00  0.00
12. -1.00  0.00  0.34
13. -1.00  0.00  1.00
14. -1.00  0.34  0.00
15. -1.00  0.50  0.50
16. -1.00  1.00  0.00
17. -0.34  0.00  0.00
18. -0.34  0.00  0.34
19. -0.34  0.34  0.00
20.  0.00  0.00  0.00
21.  0.00  0.00  0.00
22. -0.34  0.00  0.34
23. -0.34  0.34  0.00
24.  0.00  0.34  0.34
25. -0.34  0.00  0.34
       ...
```

partition.faces

```
3 1 4 2
3 7 9 8
4 4 1 9 7
4 1 2 7 8
4 2 4 8 9
3 2 5 3
3 2 5 8
3 2 3 8
3 5 3 8
3 4 6 5
3 4 6 9
3 4 5 9
3 6 5 9
3 2 4 5
3 9 8 5
3 7 9 10
3 7 8 10
3 9 8 10
3 11 14 12
3 17 19 18
4 14 11 19 17
4 11 12 17 18
4 12 14 18 19
3 12 5 6
3 12 5 18
   ...
```

partition.ebv

```
6 1 4 2 7 9 8
4 2 5 3 8
4 4 6 5 9
5 2 4 9 8 5
4 7 9 8 10
6 11 14 12 17 19 18
4 12 5 6 18
4 14 3 5 19
5 12 14 19 18 5
4 17 19 18 20
4 20 18 19 24
4 18 6 5 9
4 19 5 3 8
4 24 9 8 10
4 18 19 24 9
4 19 24 9 8
4 18 19 5 9
4 19 5 9 8
   ...
```

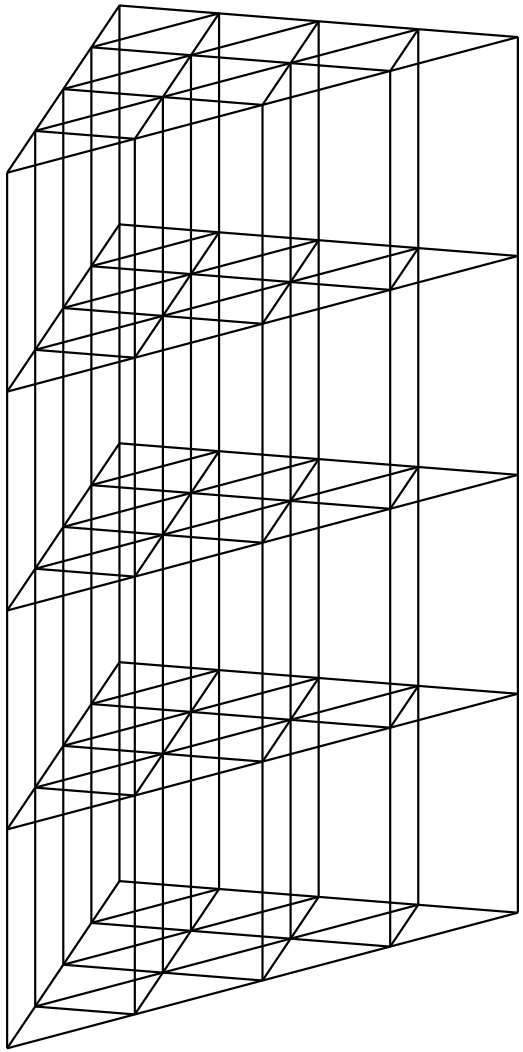
partition.ebf (etc...)

```
2 1 2 3 4 5
0 6 7 8 9 0
0 10 11 12 13 0
1 14 7 15 12 5
0 2 16 17 18 0
2 19 20 21 22 23
0 24 25 26 27 0
0 28 29 30 31 0
1 32 25 33 30 23
0 20 34 35 36 0
0 36 37 38 39 0
0 27 40 41 13 0
0 31 42 43 9 0
0 44 45 46 18 0
0 39 47 48 49 0
0 49 50 51 44 0
0 33 47 41 52 0
0 52 42 51 15 0
   ...
```

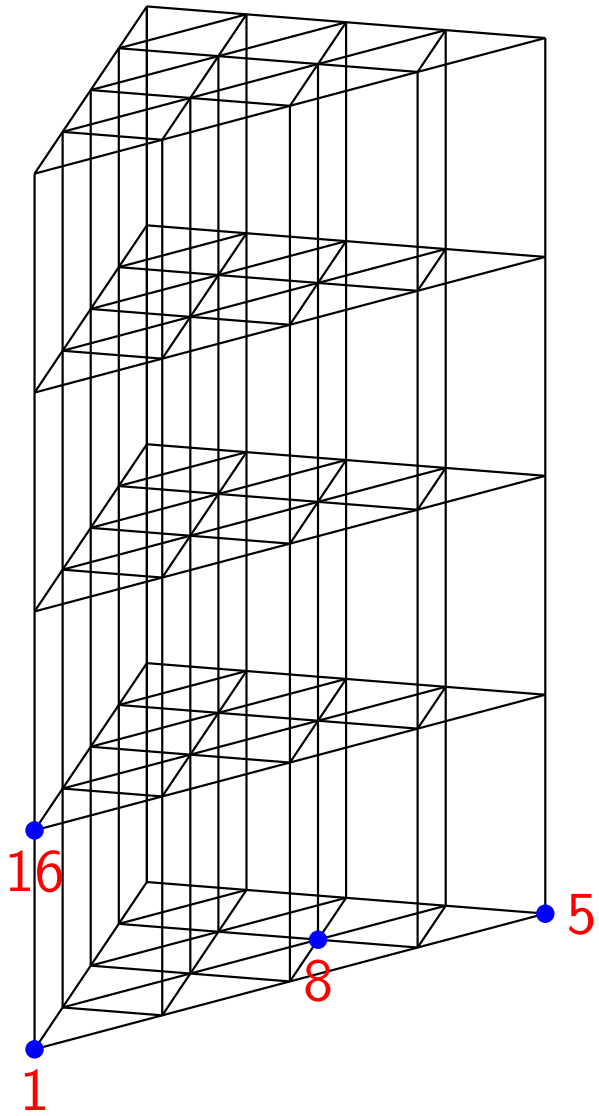
some specific algorithms
building physical points

line nr.	T	v	v	v	v	v	v
1	6	1	2	6	16	17	21
2	6	2	6	7	17	21	22
3	6	2	3	7	17	18	22
4	6	3	7	8	18	22	23
5	6	3	4	8	18	19	23
6	6	4	8	9	19	23	24
7	6	4	5	9	19	20	24
8	6	6	7	10	21	22	25
9	6	7	10	11	22	25	26
10	6	7	8	11	22	23	26
11	6	8	11	12	23	26	27
12	6	8	9	12	23	24	27
13	6	10	11	13	25	26	28
14	6	11	13	14	26	28	29
15	6	11	12	14	26	27	29
16	6	13	14	15	28	29	30
17	6	16	17	21	31	32	36
18	6	17	21	22	32	36	37
19	6	17	18	22	32	33	37
20	6	18	22	23	33	37	38
21	6	18	19	23	33	34	38
22	6	19	23	24	34	38	39
23	6	19	20	24	34	35	39
24	6	21	22	25	36	37	40
25	6	22	25	26	37	40	41
26	6	22	23	26	37	38	41
27	6	23	26	27	38	41	42
28	6	23	24	27	38	39	42
29	6	25	26	28	40	41	43
30	6	26	28	29	41	43	44

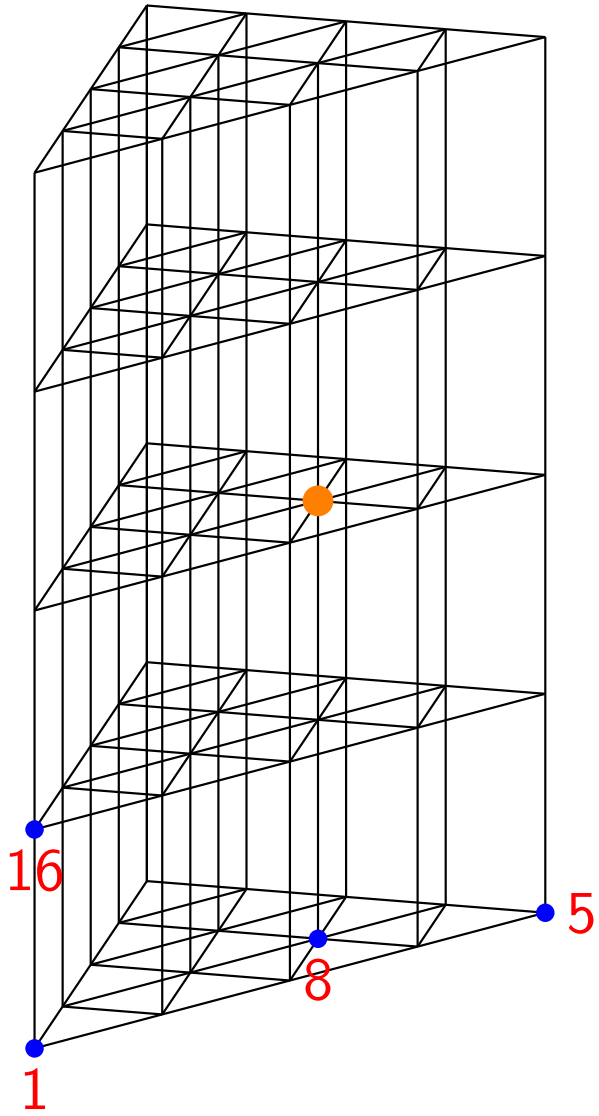
some specific algorithms
building the graph of indexation of the vertices – prismatic case



some specific algorithms
building the graph of indexation of the vertices – prismatic case

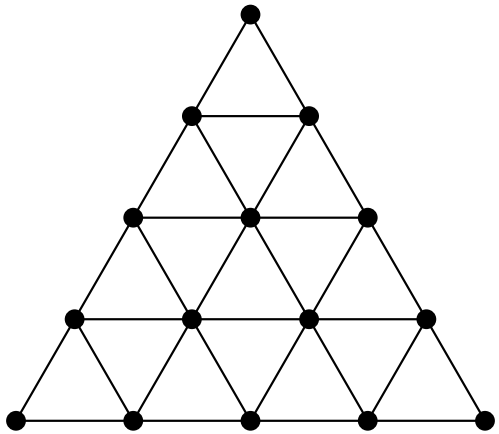


some specific algorithms
building the graph of indexation of the vertices – prismatic case

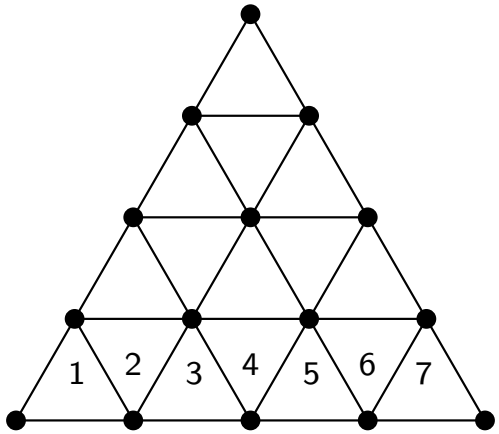


- belongs to 12 elements

How to make an algorithm that passes exactly once per each vertex and records the table?

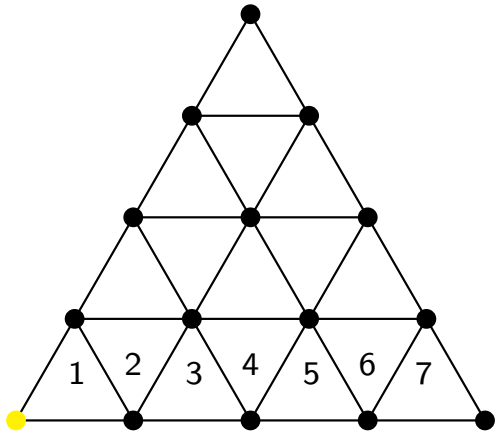


(We mean something like this):



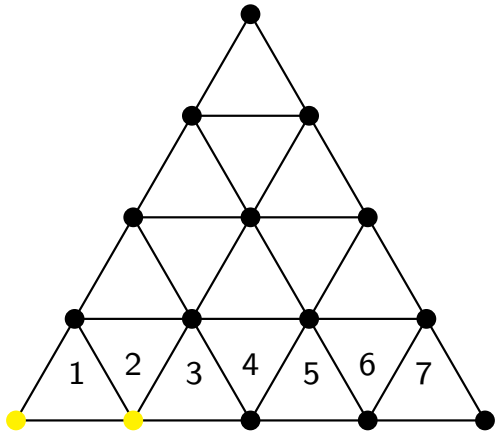
1: * * * * * *
2: * * * * * *
3: * * * * * *
4: * * * * * *
5: * * * * * *
6: * * * * * *
7: * * * * * *

(We mean something like this):



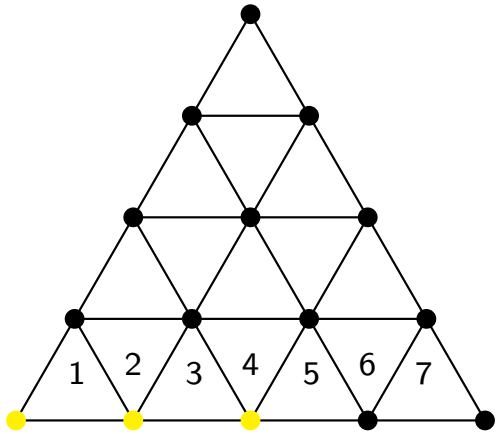
1: 1 * * * * *
2: * * * * * *
3: * * * * * *
4: * * * * * *
5: * * * * * *
6: * * * * * *
7: * * * * * *

(We mean something like this):



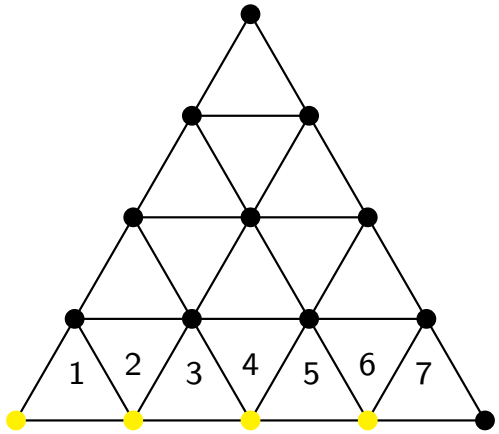
1: 1 2 * * * *
2: 2 * * * * *
3: 2 * * * * *
4: * * * * * *
5: * * * * * *
6: * * * * * *
7: * * * * * *

(We mean something like this):



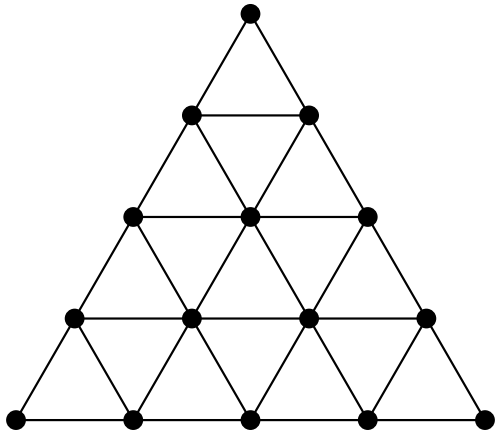
1: 1 2 * * * *
2: 2 * * * * *
3: 2 3 * * * *
4: 3 * * * * *
5: 3 * * * * *
6: * * * * * *
7: * * * * * *

(We mean something like this):



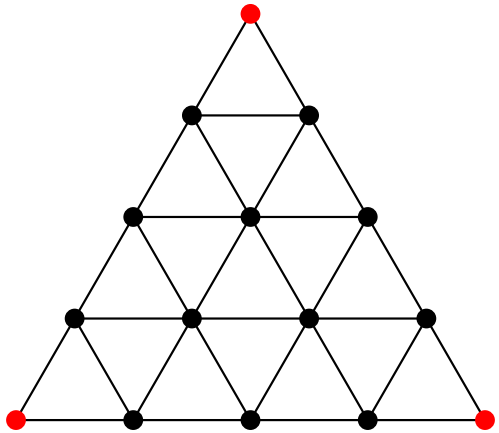
1: 1 2 * * * *
2: 2 * * * * *
3: 2 3 * * * *
4: 3 * * * * *
5: 3 4 * * * *
6: 4 * * * * *
7: 4 * * * * *

The idea:



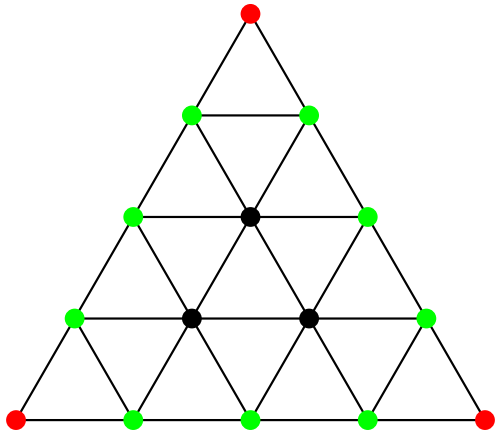
The elements surrounding the nodes may be

The idea:



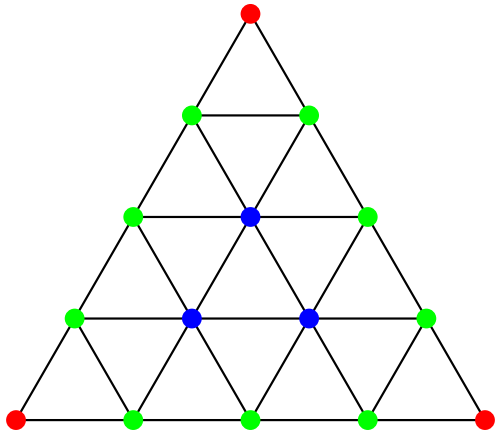
The elements surrounding the nodes may be 1

The idea:



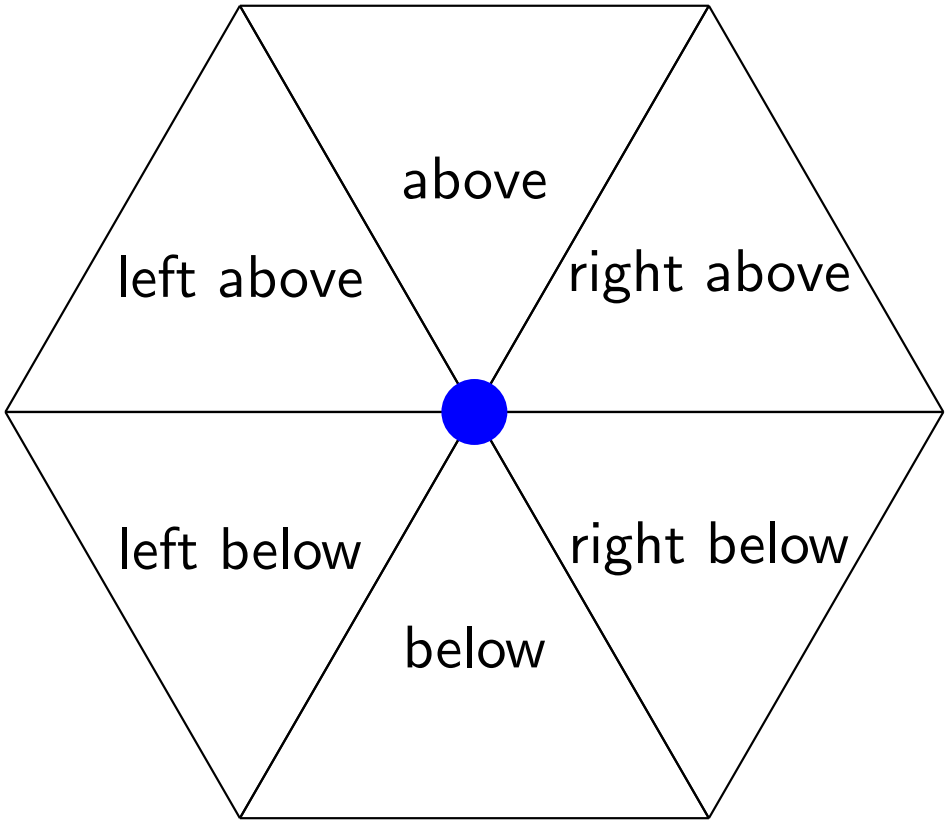
The elements surrounding the nodes may be 1, 3

The idea:

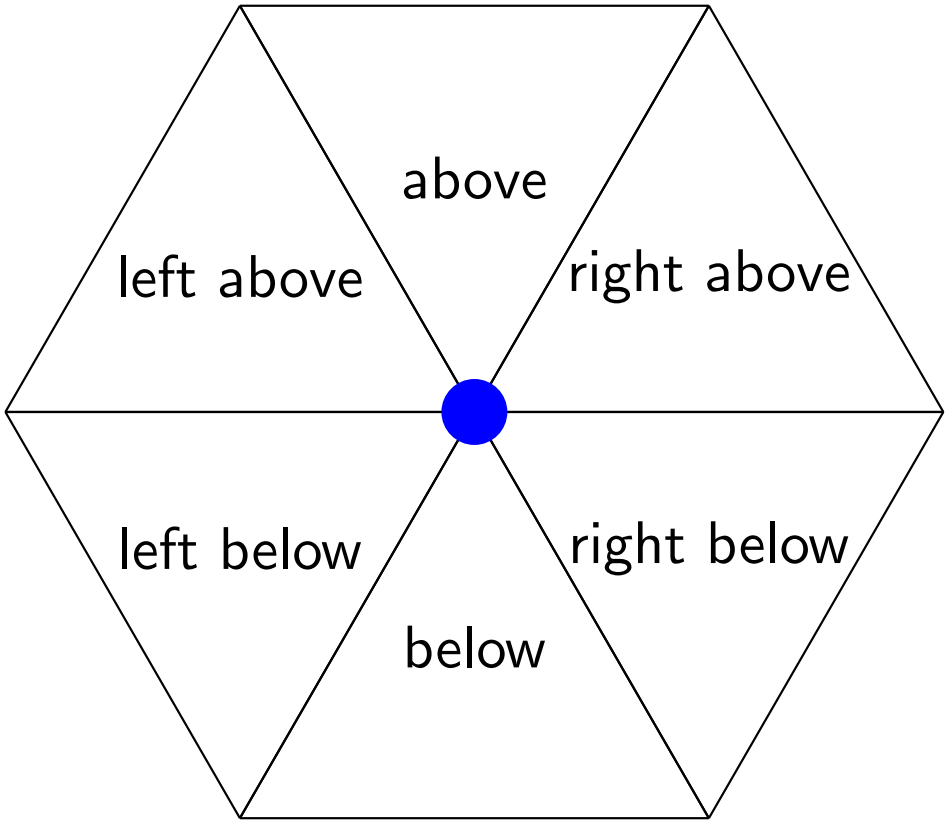


The elements surrounding the nodes may be 1, 3 or 6.

If nodes have incidence 6 (the others are "base cases")

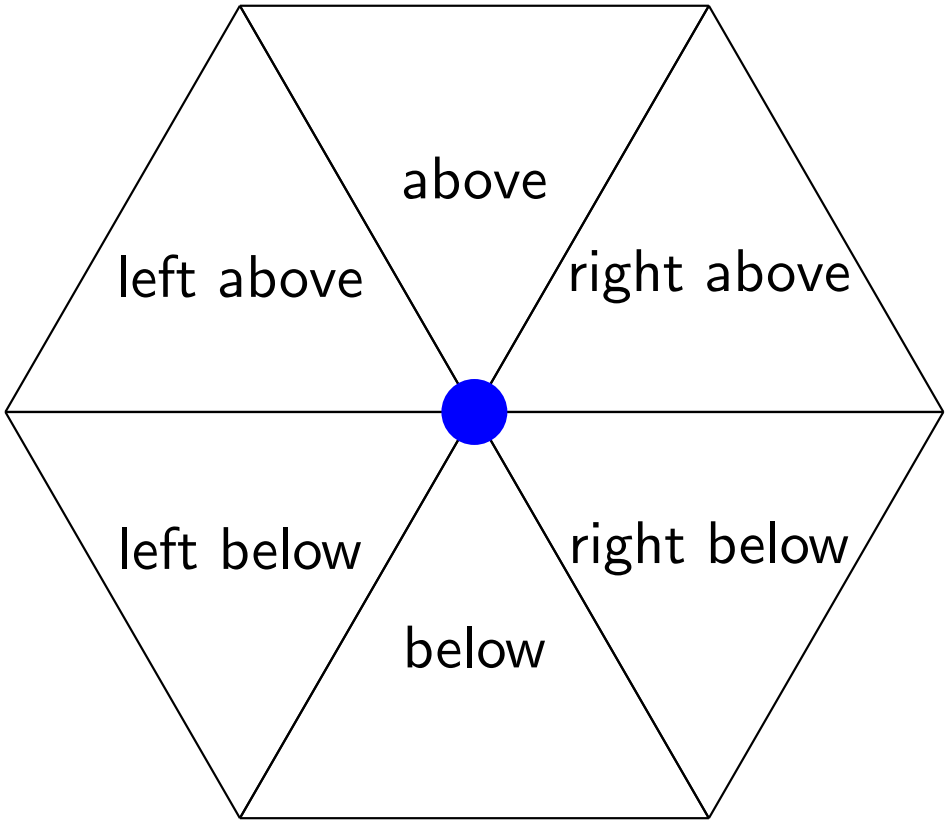


If nodes have incidence 6 (the others are "base cases")



Each one of these six is a function of the current (blue) vertex

If nodes have incidence 6 (the others are "base cases")



Each one of these six is a function of the current (blue) vertexand we increment them iteratively

```
    # ROW HEAD BASE CASE
    current_node += 1
    below       = sum([2*k-1 for k in range(levels,row,-1)]) + 1
    right_below = below + 1
    right_above = below + extra_odd
    assign(current_node, [below,right_below,right_above])
    # INDUCTIVE MIDDLE STEPS. START HEXAGONS
    step = 1
    while step < row - 1:  # row 'row' has 'row' nodes
        current_node += 1
        left_below  = sum([2*k-1 for k in range(levels,row,-1)]) \
                      + step*2
        below       = left_below + 1
        right_below = below + 1
        left_above  = sum([2*k-1 for k in range(levels,row,-1)]) \
                      + extra_odd + (2*step-1)
        above       = left_above + 1
        right_above = above + 1
        assign(current_node, [left_below,below,right_below,\
                      left_above,above,right_above])
        step += 1
    # ROW TAIL BASE CASE
    current_node += 1
    below      = sum([2*k-1 for k in range(levels,row,-1)]) \
                 + extra_odd
    left_below = below - 1
    left_above = left_below + extra_odd - 1
    assign(current_node, [left_below,below,left_above])
    row -= 1
# TWO UPPER ROWS
```

Another example, if time

convex combinations to produce the nodes

```
mesh.macroel_tetrahedra()
```

Thank You...

- ... Python

Thank You...

- ... Python
- NumPy,

Thank You...

- ... Python
- NumPy,
- mayavi and matplotlib,

Thank You...

- ... Python
- NumPy,
- mayavi and matplotlib,
- TiKz and Pgf,

Thank You...

- ... Python
- NumPy,
- mayavi and matplotlib,
- TiKz and Pgf,
- Brasil,

Thank You...

- ... Python
- NumPy,
- mayavi and matplotlib,
- TiKz and Pgf,
- Brasil,
- and PyConBr '19!

The End.