

## Advanced Topics from Scientific Computing

TU Berlin Winter 2023/24

Notebook 11

 Jürgen Fuhrmann

# Grid creation and visualization

This notebook shows how to perform grid creation and visualization with the assistance of the packages [ExtendableGrids.jl](#) and [SimplexGridFactory.jl](#) using [Triangulate.jl](#), [TetGen.jl](#) and [Gmsh.jl](#). Visualization in this notebook is done using the [GridVisualize.jl](#) package which itself relies on various backends.

These packages will be used later on by [VoronoiFVM.jl](#) for solving partial differential equations.

```
1 begin
2     using SimplexGridFactory
3     using ExtendableGrids
4     using Triangulate
5     using TetGen
6     using GridVisualize
7     using PlutoVista
8     using PlutoUI
9     import CairoMakie
10    default_plotter!(CairoMakie)
11    CairoMakie.activate!(type="svg")
12 end
```

## 1D grids

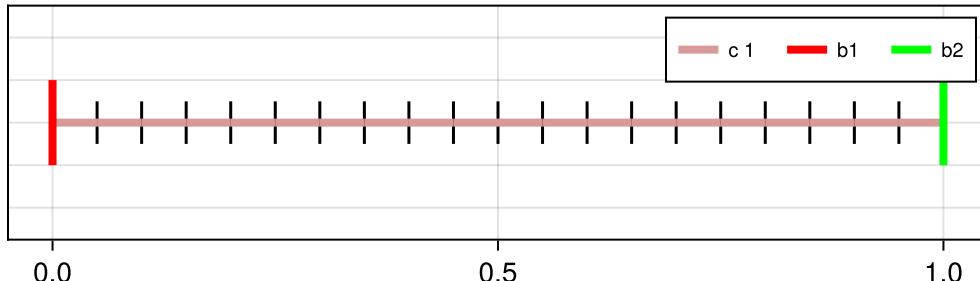
1D grids are created just from arrays of monotonically increasing coordinates using the [simplexgrid](#) method.

```
X1 = 0.0:0.05:1.0
1 X1 = range(0, 1; length = 21)
```

```
g1 = ExtendableGrids.ExtendableGrid{Float64, Int32};
dim: 1 nodes: 21 cells: 20 bfaces: 2
```

```
1 g1 = simplexgrid(x1)
```

We can plot a grid with a method from `GridVisualize.jl`



```
1 gridplot(g1; resolution = (500, 150), legend = :rt)
```

We see some additional information:

- `cellregion`: each grid cell (interval, triangle, tetrahedron) as an integer region marker attached
- `bfaceregion`: boundary faces (points, lines, triangles) have an interger boundary region marker attached

We can also have a look into the grid structure:

```
Dict(XCoordinates => [0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, more ,1.0], Coo
```

```
1 g1.components
```

Components can be accessed via `[ ]`. In fact the keys in the dictionary of components are types.

```
1×21 Matrix{Float64}:
 0.0  0.05  0.1  0.15  0.2  0.25  0.3  0.35 ...  0.7  0.75  0.8  0.85  0.9  0.95  1.0
```

```
1 g1[Coordinates]
```

```
2×20 Matrix{Int32}:
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

```
1 g1[CellNodes]
```

## Modifying region markers

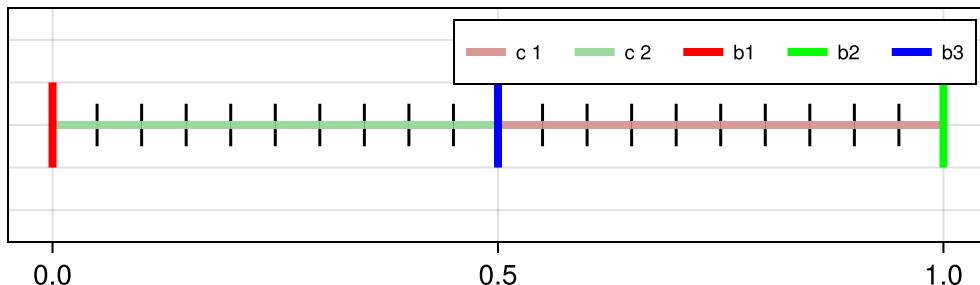
The `simplexgrid` method provides a default distribution of markers, but we would like to be able to change them. This can be done by putting masks on cells or faces (points in 1D):

```
g2 = ExtendableGrids.ExtendableGrid{Float64, Int32};
dim: 1 nodes: 21 cells: 20 bfaces: 2
```

```
1 g2 = deepcopy(g1)
```

```
1 cellmask!(g2, [0.0], [0.5], 2);
```

```
1 bfacemask!(g2, [0.5], [0.5], 3);
```



```
1 gridplot(g2; resolution = (500, 150), legend = :rt)
```

## Creating locally refined grids

For this purpose, we just need to create arrays with the corresponding coordinate values. This can be done programmatically.

Two support methods are provided for this purpose.

0.1

```
1 begin
2 hmin = 0.01;
3 hmax = 0.1;
4 end
```

The `geomspace` method creates an array such that the smallest interval size is `hmin` and the largest interval size is not larger but close to `hmax`, and the interval sizes constitute a geometric sequence.

`X2L` =

```
[0.0, 0.0931551, 0.170501, 0.234722, 0.288044, 0.332316, 0.369076, 0.399597, 0.424939, 0.
```

```
1 X2L = geomspace(0, 0.5, hmax, hmin)
```

`DX2` =

```
[0.0931551, 0.0773463, 0.0642203, 0.0533218, 0.0442729, 0.0367596, 0.0305213, 0.0253417,
```

```
1 DX2 = X2L[2:end] .- X2L[1:(end - 1)]
```

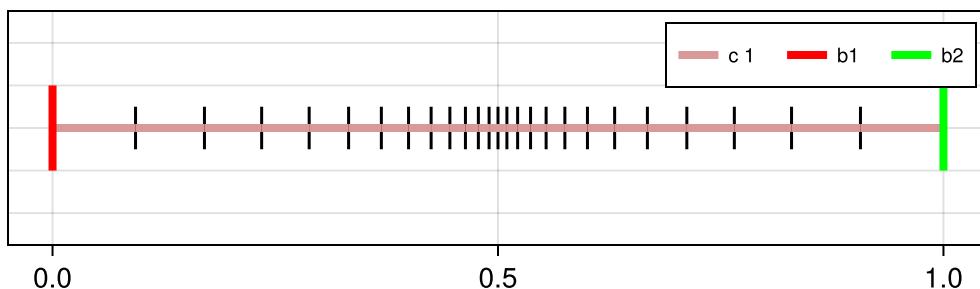
```
[1.20439, 1.20439, 1.20439, 1.20439, 1.20439, 1.20439, 1.20439, 1.20439, 1.20439, 1.20439]
```

```
1 DX2[1:(end - 1)] ./ DX2[2:end]
```

```
X2R =
[0.5, 0.51, 0.522044, 0.536549, 0.55402, 0.575061, 0.600403, 0.630924, 0.667684, 0.711956
1 X2R = geomspace(0.5, 1, hmin, hmax)
```

We can glue these arrays together and create a grid from them:

```
X2 =
[0.0, 0.0931551, 0.170501, 0.234722, 0.288044, 0.332316, 0.369076, 0.399597, 0.424939, 0.
1 X2 = glue(X2L, X2R)
```



```
1 gridplot(simplexgrid(X2); resolution = (500, 150), legend = :rt)
```

## Plotting functions

We assume that functions can be represented by their node values and plotted via their piecewise linear interpolants. E.g. they could come from some simulation.

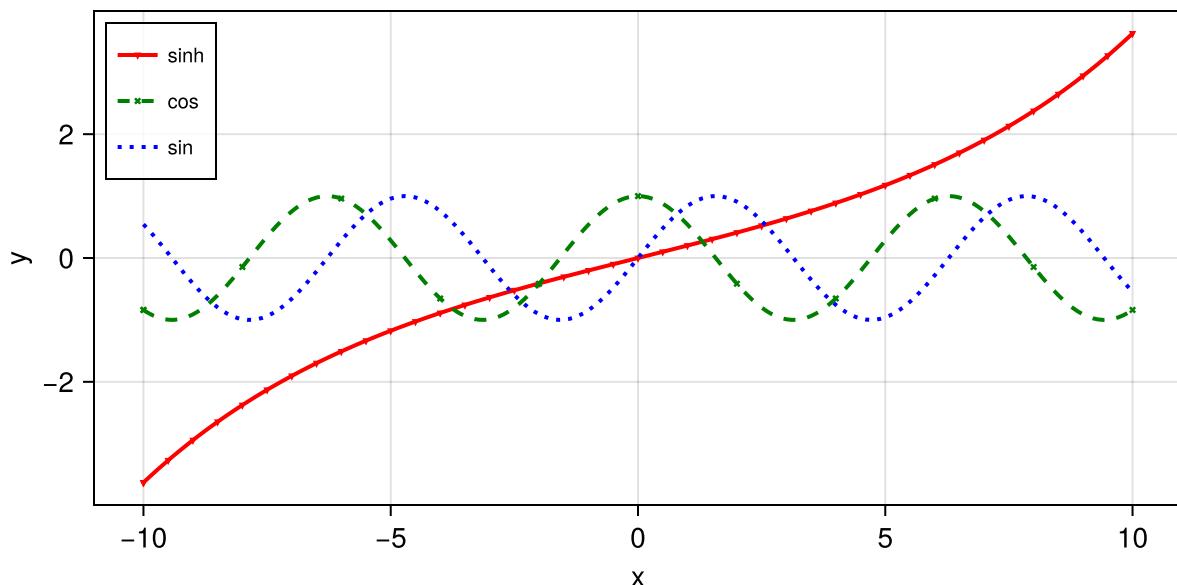
```
g1d2 = ExtendableGrids.ExtendableGrid{Float64, Int32};
dim: 1 nodes: 201 cells: 200 bfaces: 2
```

```
1 g1d2 = simplexgrid(range(-10, 10; length = 201))
```

```
fsin =
[0.544021, 0.457536, 0.366479, 0.271761, 0.174327, 0.0751511, -0.0247754, -0.124454, -0.224454, -0.324454, -0.424454, -0.524454, -0.624454, -0.724454, -0.824454, -0.924454, -0.999693, -0.992225, -0.930426, -0.889191, -0.839072]
1 fsin = map(sin, g1d2)
```

```
fcos =
[-0.839072, -0.889191, -0.930426, -0.962365, -0.984688, -0.997172, -0.999693, -0.992225, -0.930426, -0.889191, -0.839072]
1 fcos = map(cos, g1d2)
```

```
fsinh =
[-3.62686, -3.55234, -3.47923, -3.40752, -3.33718, -3.26816, -3.20046, -3.13403, -3.06886, -2.999693, -2.92225, -2.839072, -2.766479, -2.693761, -2.620937, -2.547991, -2.474837, -2.401563, -2.327176, -2.252671, -2.177056, -2.101326, -2.025476, -1.949506, -1.873416, -1.797206, -1.720876, -1.644426, -1.567856, -1.491166, -1.414356, -1.337426, -1.260376, -1.183206, -1.105916, -1.028496, -0.950946, -0.873266, -0.795456, -0.717516, -0.639446, -0.561246, -0.482916, -0.404456, -0.325866, -0.247146, -0.1683, -0.089396, 0.0098146, -0.089396, -0.1683, -0.247146, -0.325866, -0.404456, -0.482916, -0.561246, -0.639446, -0.717516, -0.795456, -0.873266, -0.950946, -0.105916, -0.183206, -0.260376, -0.337426, -0.414356, -0.491166, -0.567856, -0.644426, -0.720876, -0.797206, -0.873416, -0.949506, -1.025476, -1.101326, -1.177056, -1.252671, -1.327176, -1.401563, -1.474837, -1.547991, -1.620937, -1.693761, -1.766479, -1.839072, -1.911506, -1.983706, -2.055676, -2.127416, -2.198896, -2.269126, -2.338996, -2.407526, -2.475786, -2.543716, -2.611466, -2.678916, -2.745996, -2.812676, -2.878916, -2.944786, -3.010376, -3.075676, -3.140676, -3.205376, -3.269676, -3.333676, -3.397376, -3.460776, -3.523876, -3.586676, -3.649176, -3.711476, -3.773576, -3.835476, -3.897176, -3.958676, -4.019976, -4.081076, -4.141976, -4.202676, -4.263176, -4.323476, -4.383576, -4.443476, -4.503176, -4.562676, -4.622076, -4.681276, -4.739376, -4.796376, -4.852276, -4.907076, -4.960776, -5.013376, -5.065876, -5.117276, -5.168576, -5.219776, -5.269876, -5.319876, -5.369776, -5.419576, -5.469276, -5.518876, -5.568376, -5.617776, -5.667176, -5.716476, -5.765676, -5.814776, -5.863776, -5.912676, -5.961476, -6.009176, -6.056776, -6.104276, -6.151676, -6.198976, -6.246176, -6.293276, -6.339276, -6.385176, -6.429976, -6.474676, -6.519276, -6.563776, -6.607176, -6.650476, -6.693676, -6.736776, -6.779776, -6.822676, -6.865476, -6.908176, -6.950776, -6.993276, -7.035676, -7.077976, -7.120176, -7.162276, -7.204276, -7.246176, -7.287976, -7.329676, -7.371276, -7.412776, -7.454176, -7.495476, -7.536676, -7.577776, -7.618776, -7.659676, -7.700476, -7.741176, -7.781776, -7.822276, -7.862676, -7.902976, -7.943176, -7.983276, -8.023276, -8.063176, -8.102976, -8.142676, -8.182276, -8.221776, -8.261176, -8.300476, -8.339676, -8.378776, -8.417776, -8.456676, -8.495476, -8.534176, -8.572776, -8.611276, -8.649676, -8.687976, -8.726176, -8.764276, -8.802276, -8.839276, -8.876176, -8.912976, -8.949676, -8.986276, -9.022776, -9.059176, -9.095476, -9.131676, -9.167776, -9.203776, -9.239676, -9.275476, -9.311176, -9.346776, -9.382276, -9.417676, -9.452476, -9.487176, -9.521776, -9.556376, -9.590876, -9.625376, -9.659776, -9.694176, -9.728476, -9.762676, -9.796876, -9.830976, -9.864976, -9.898876, -9.932676, -9.966376, -9.999976, -10.033476, -10.066876, -10.100176, -10.133376, -10.166476, -10.199476, -10.232376, -10.265276, -10.297976, -10.330676, -10.363276, -10.395776, -10.428176, -10.460476, -10.492676, -10.524776, -10.556776, -10.588676, -10.620476, -10.652176, -10.683776, -10.715276, -10.746676, -10.777976, -10.809176, -10.840276, -10.871276, -10.902276, -10.933176, -10.963976, -10.994676, -11.025276, -11.055876, -11.086476, -11.116976, -11.147476, -11.177876, -11.208276, -11.238576, -11.268876, -11.299176, -11.329376, -11.359576, -11.389676, -11.419776, -11.449876, -11.479876, -11.509876, -11.539876, -11.569876, -11.599876, -11.629876, -11.659876, -11.689876, -11.719876, -11.749876, -11.779876, -11.809876, -11.839876, -11.869876, -11.899876, -11.929876, -11.959876, -11.989876, -12.019876, -12.049876, -12.079876, -12.109876, -12.139876, -12.169876, -12.199876, -12.229876, -12.259876, -12.289876, -12.319876, -12.349876, -12.379876, -12.409876, -12.439876, -12.469876, -12.499876, -12.529876, -12.559876, -12.589876, -12.619876, -12.649876, -12.679876, -12.709876, -12.739876, -12.769876, -12.799876, -12.829876, -12.859876, -12.889876, -12.919876, -12.949876, -12.979876, -13.009876, -13.039876, -13.069876, -13.099876, -13.129876, -13.159876, -13.189876, -13.219876, -13.249876, -13.279876, -13.309876, -13.339876, -13.369876, -13.399876, -13.429876, -13.459876, -13.489876, -13.519876, -13.549876, -13.579876, -13.609876, -13.639876, -13.669876, -13.699876, -13.729876, -13.759876, -13.789876, -13.819876, -13.849876, -13.879876, -13.909876, -13.939876, -13.969876, -13.999876, -14.029876, -14.059876, -14.089876, -14.119876, -14.149876, -14.179876, -14.209876, -14.239876, -14.269876, -14.299876, -14.329876, -14.359876, -14.389876, -14.419876, -14.449876, -14.479876, -14.509876, -14.539876, -14.569876, -14.599876, -14.629876, -14.659876, -14.689876, -14.719876, -14.749876, -14.779876, -14.809876, -14.839876, -14.869876, -14.899876, -14.929876, -14.959876, -14.989876, -15.019876, -15.049876, -15.079876, -15.109876, -15.139876, -15.169876, -15.199876, -15.229876, -15.259876, -15.289876, -15.319876, -15.349876, -15.379876, -15.409876, -15.439876, -15.469876, -15.499876, -15.529876, -15.559876, -15.589876, -15.619876, -15.649876, -15.679876, -15.709876, -15.739876, -15.769876, -15.799876, -15.829876, -15.859876, -15.889876, -15.919876, -15.949876, -15.979876, -16.009876, -16.039876, -16.069876, -16.099876, -16.129876, -16.159876, -16.189876, -16.219876, -16.249876, -16.279876, -16.309876, -16.339876, -16.369876, -16.399876, -16.429876, -16.459876, -16.489876, -16.519876, -16.549876, -16.579876, -16.609876, -16.639876, -16.669876, -16.699876, -16.729876, -16.759876, -16.789876, -16.819876, -16.849876, -16.879876, -16.909876, -16.939876, -16.969876, -16.999876, -17.029876, -17.059876, -17.089876, -17.119876, -17.149876, -17.179876, -17.209876, -17.239876, -17.269876, -17.299876, -17.329876, -17.359876, -17.389876, -17.419876, -17.449876, -17.479876, -17.509876, -17.539876, -17.569876, -17.599876, -17.629876, -17.659876, -17.689876, -17.719876, -17.749876, -17.779876, -17.809876, -17.839876, -17.869876, -17.899876, -17.929876, -17.959876, -17.989876, -18.019876, -18.049876, -18.079876, -18.109876, -18.139876, -18.169876, -18.199876, -18.229876, -18.259876, -18.289876, -18.319876, -18.349876, -18.379876, -18.409876, -18.439876, -18.469876, -18.499876, -18.529876, -18.559876, -18.589876, -18.619876, -18.649876, -18.679876, -18.709876, -18.739876, -18.769876, -18.799876, -18.829876, -18.859876, -18.889876, -18.919876, -18.949876, -18.979876, -19.009876, -19.039876, -19.069876, -19.099876, -19.129876, -19.159876, -19.189876, -19.219876, -19.249876, -19.279876, -19.309876, -19.339876, -19.369876, -19.399876, -19.429876, -19.459876, -19.489876, -19.519876, -19.549876, -19.579876, -19.609876, -19.639876, -19.669876, -19.699876, -19.729876, -19.759876, -19.789876, -19.819876, -19.849876, -19.879876, -19.909876, -19.939876, -19.969876, -19.999876, -20.029876, -20.059876, -20.089876, -20.119876, -20.149876, -20.179876, -20.209876, -20.239876, -20.269876, -20.299876, -20.329876, -20.359876, -20.389876, -20.419876, -20.449876, -20.479876, -20.509876, -20.539876, -20.569876, -20.599876, -20.629876, -20.659876, -20.689876, -20.719876, -20.749876, -20.779876, -20.809876, -20.839876, -20.869876, -20.899876, -20.929876, -20.959876, -20.989876, -21.019876, -21.049876, -21.079876, -21.109876, -21.139876, -21.169876, -21.199876, -21.229876, -21.259876, -21.289876, -21.319876, -21.349876, -21.379876, -21.409876, -21.439876, -21.469876, -21.499876, -21.529876, -21.559876, -21.589876, -21.619876, -21.649876, -21.679876, -21.709876, -21.739876, -21.769876, -21.799876, -21.829876, -21.859876, -21.889876, -21.919876, -21.949876, -21.979876, -22.009876, -22.039876, -22.069876, -22.099876, -22.129876, -22.159876, -22.189876, -22.219876, -22.249876, -22.279876, -22.309876, -22.339876, -22.369876, -22.399876, -22.429876, -22.459876, -22.489876, -22.519876, -22.549876, -22.579876, -22.609876, -22.639876, -22.669876, -22.699876, -22.729876, -22.759876, -22.789876, -22.819876, -22.849876, -22.879876, -22.909876, -22.939876, -22.969876, -22.999876, -23.029876, -23.059876, -23.089876, -23.119876, -23.149876, -23.179876, -23.209876, -23.239876, -23.269876, -23.299876, -23.329876, -23.359876, -23.389876, -23.419876, -23.449876, -23.479876, -23.509876, -23.539876, -23.569876, -23.599876, -23.629876, -23.659876, -23.689876, -23.719876, -23.749876, -23.779876, -23.809876, -23.839876, -23.869876, -23.899876, -23.929876, -23.959876, -23.989876, -24.019876, -24.049876, -24.079876, -24.109876, -24.139876, -24.169876, -24.199876, -24.229876, -24.259876, -24.289876, -24.319876, -24.349876, -24.379876, -24.409876, -24.439876, -24.469876, -24.499876, -24.529876, -24.559876, -24.589876, -24.619876, -24.649876, -24.679876, -24.709876, -24.739876, -24.769876, -24.799876, -24.829876, -24.859876, -24.889876, -24.919876, -24.949876, -24.979876, -25.009876, -25.039876, -25.069876, -25.099876, -25.129876, -25.159876, -25.189876, -25.219876, -25.249876, -25.279876, -25.309876, -25.339876, -25.369876, -25.399876, -25.429876, -25.459876, -25.489876, -25.519876, -25.549876, -25.579876, -25.609876, -25.639876, -25.669876, -25.699876, -25.729876, -25.759876, -25.789876, -25.819876, -25.849876, -25.879876, -25.909876, -25.939876, -25.969876, -25.999876, -26.029876, -26.059876, -26.089876, -26.119876, -26.149876, -26.179876, -26.209876, -26.239876, -26.269876, -26.299876, -26.329876, -26.359876, -26.389876, -26.419876, -26.449876, -26.479876, -26.509876, -26.539876, -26.569876, -26.599876, -26.629876, -26.659876, -26.689876, -26.719876, -26.749876, -26.779876, -26.809876, -26.839876, -26.869876, -26.899876, -26.929876, -26.959876, -26.989876, -27.019876, -27.049876, -27.079876, -27.109876, -27.139876, -27.169876, -27.199876, -27.229876, -27.259876, -27.289876, -27.319876, -27.349876, -27.379876, -27.409876, -27.439876, -27.469876, -27.499876, -27.529876, -27.559876, -27.589876, -27.619876, -27.649876, -27.679876, -27.709876, -27.739876, -27.769876, -27.799876, -27.829876, -27.859876, -27.889876, -27.919876, -27.949876, -27.979876, -28.009876, -28.039876, -28.069876, -28.099876, -28.129876, -28.159876, -28.189876, -28.219876, -28.249876, -28.279876, -28.309876, -28.339876, -28.369876, -28.399876, -28.429876, -28.459876, -28.489876, -28.519876, -28.549876, -28.579876, -28.609876, -28.639876, -28.669876, -28.699876, -28.729876, -28.759876, -28.789876, -28.819876, -28.849876, -28.879876, -28.909876, -28.939876, -28.969876, -29.009876, -29.039876, -29.069876, -29.099876, -29.129876, -29.159876, -29.189876, -29.219876, -29.249876, -29.279876, -29.309876, -29.339876, -29.369876, -29.399876, -29.429876, -29.459876, -29.489876, -29.519876, -29.549876, -29.579876, -29.609876, -29.639876, -29.669876, -29.699876, -29.729876, -29.759876, -29.789876, -29.819876, -29.849876, -29.879876, -29.909876, -29.939876, -29.969876, -29.999876, -30.029876, -30.059876, -30.089876, -30.119876, -30.149876, -30.179876, -30.209876, -30.239876, -30.269876, -30.299876, -30.329876, -30.359876, -30.389876, -30.419876, -30.449876, -30.479876, -30.509876, -30.539876, -30.5
```



```
1 let
2     vis = GridVisualizer(; resolution = (600, 300), legend = :lt )
3
4     scalarplot!(vis, g1d2, fsinh; label = "sinh", markershape = :dtriangle, color =
:red, markevery = 5, clear = false)
5
6     scalarplot!(vis, g1d2, fcos; label = "cos", markershape = :xcross, color =
:green, linestyle = :dash, clear = false,
7                         markevery = 20)
8
9     scalarplot!(vis, g1d2, fsin; label = "sin", markershape = :none, color = :blue,
linestyle = :dot, clear = false, markevery = 20)
10
11    reveal(vis)
12 end
```

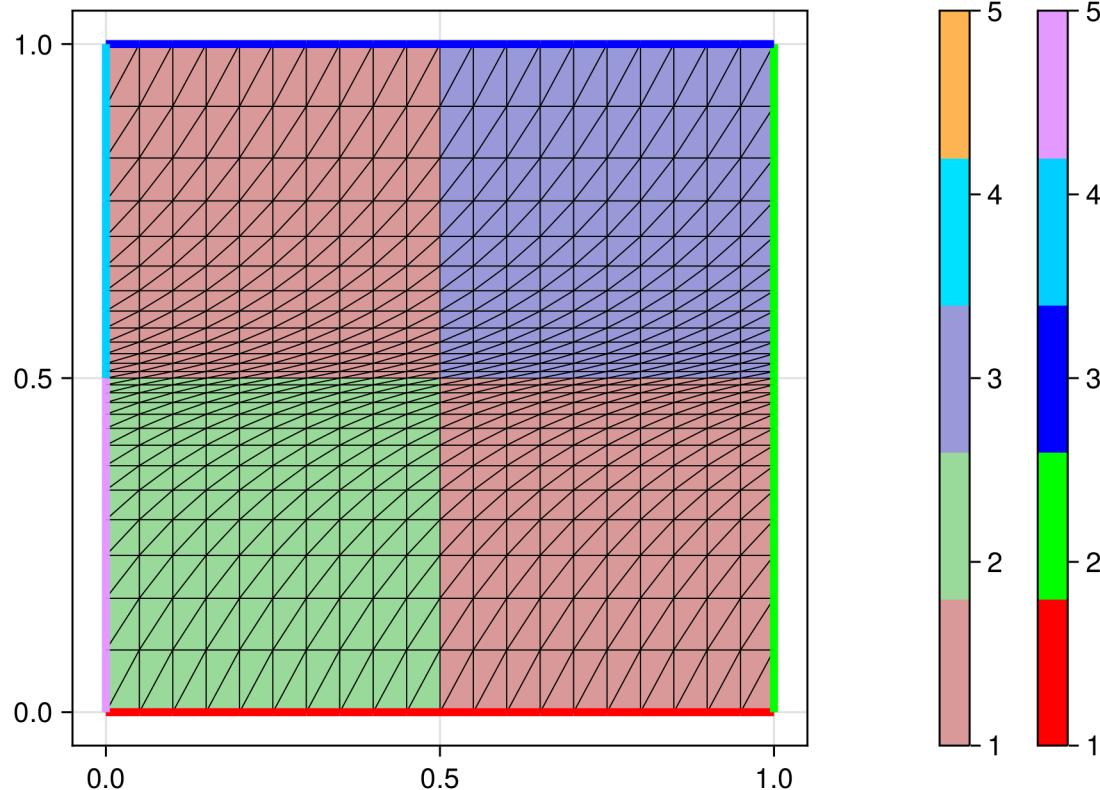
## 2D grids

### Tensor product grids

For 2D tensor product grids, we can again use the `simplexgrid` method and apply the mask methods for modifying cell and boundary region markers.

```
ExtendableGrids.ExtendableGrid{Float64, Int32};  
dim: 2 nodes: 567 cells: 1040 bfaces: 92
```

```
1 begin  
2     g2d1 = simplexgrid(x1, x2)  
3     cellmask!(g2d1, [0.0, 0.0], [0.5, 0.5], 2)  
4     cellmask!(g2d1, [0.5, 0.5], [1.0, 1.0], 3)  
5     bfacemask!(g2d1, [0.0, 0.0], [0.0, 0.5], 5)  
6 end
```



```
1 gridplot(g2d1; resolution = (600, 400), linewidth = 0.5)
```

To interact with the plot when using the PlutoVista backend, you can use the mouse wheel or double touch to zoom, "shift-mouse" to pan, and "alt-mouse-left" or "ctrl-mouse-left" to reset.

We can also have a look into the components of a 2D grid:

```
Dict(XCoordinates => [0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, more ,1.0], Coo
```

```
1 g2d1.components
```

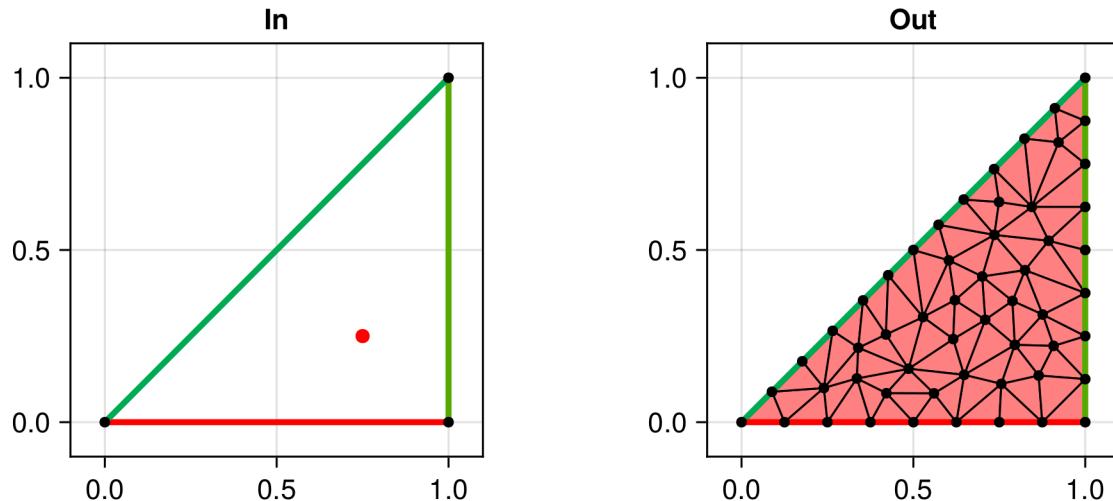
# Unstructured grids

For the triangulation of unstructured grids, we use the mesh generator Triangle via the [Triangulate.jl](#) and [SimplexGridFactory.jl](#) packages.

The later package exports the `SimplexGridBuilder` which shall help to simplify the creation of the input for `Triangulate`.

```
builder2 =  
    SimplexGridBuilder(Triangulate, 3, 1, 1.0, 1.0e-12, [1, 2, 3], [[1, 2], [2, 3], [3, 1]], {  
  
1 builder2 = let  
2     b = SimplexGridBuilder(; Generator = Triangulate)  
3     p1 = point!(b, 0, 0)  
4     p2 = point!(b, 1, 0)  
5     p3 = point!(b, 1, 1)  
6  
7     # Specify outer boundary  
8     facetregion!(b, 1)  
9     facet!(b, p1, p2)  
10    facetregion!(b, 2)  
11    facet!(b, p2, p3)  
12    facetregion!(b, 3)  
13    facet!(b, p3, p1)  
14  
15    cellregion!(b, 1)  
16    regionpoint!(b, 0.75, 0.25)  
17  
18    options!(b; maxvolume = 0.01)  
19    b  
20 end
```

We can plot the current state of the builder (in the moment this works only with PyPlot):

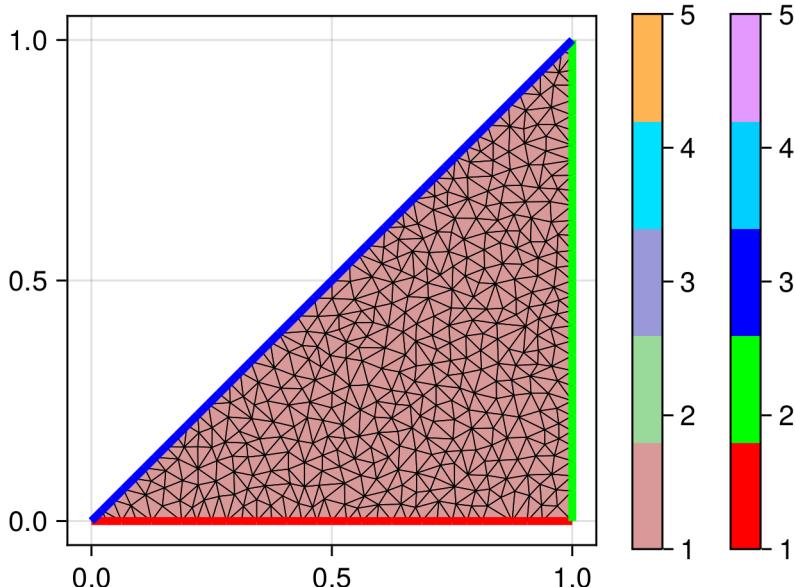


```
1 builderplot(builder2; Plotter = CairoMakie)
```

Found 'resolution' in the theme when creating a 'Scene'. The 'resolution' keyword for 'Scene's and 'Figure's has been deprecated. Use 'Figure(; size = ...)' or 'Scene(; size = ...)' instead, which better reflects that this is a unitless size and not a pixel resolution. The key could also come from 'set\_theme!' calls or related theming functions.

```
grid2d2 = ExtendableGrids.ExtendableGrid{Float64, Int32};  
dim: 2 nodes: 449 cells: 793 bfaces: 103
```

```
1 grid2d2 = simplexgrid(builder2; maxvolume = 0.001)
```



```
1 gridplot(grid2d2; resolution = (400, 300), linewidth = 0.5)
```

# More complicated grids

More complicated grids include:

- local refinement
- interior boundaries
- different region markers
- holes

The particular way to describe these things is due to Jonathan Shewchuk and his mesh generator `Triangle` via its Julia wrapper package `Triangulate.jl`.

## Local refinement

```
refinement_center = [0.8, 0.2]
1 refinement_center = [0.8, 0.2]
```

For local refinement, we define a function, which is able to tell if a triangle is to be refined ("unsuitable") or can be kept as it is.

The function measures the distance between the refinement center and the triangle barycenter. We require that the area increases with the distance from the refinement center.

```
1 function unsuitable(x1, y1, x2, y2, x3, y3, area)
2     bary_x = (x1 + x2 + x3) / 3.0
3     bary_y = (y1 + y2 + y3) / 3.0
4     dx = bary_x - refinement_center[1]
5     dy = bary_y - refinement_center[2]
6     qdist = dx^2 + dy^2
7     area > 0.1 * max(1.0e-2, qdist)
8 end;
```

## Interior boundaries

Interior boundaries are described in a similar as exterior ones - just by facets connecting points.

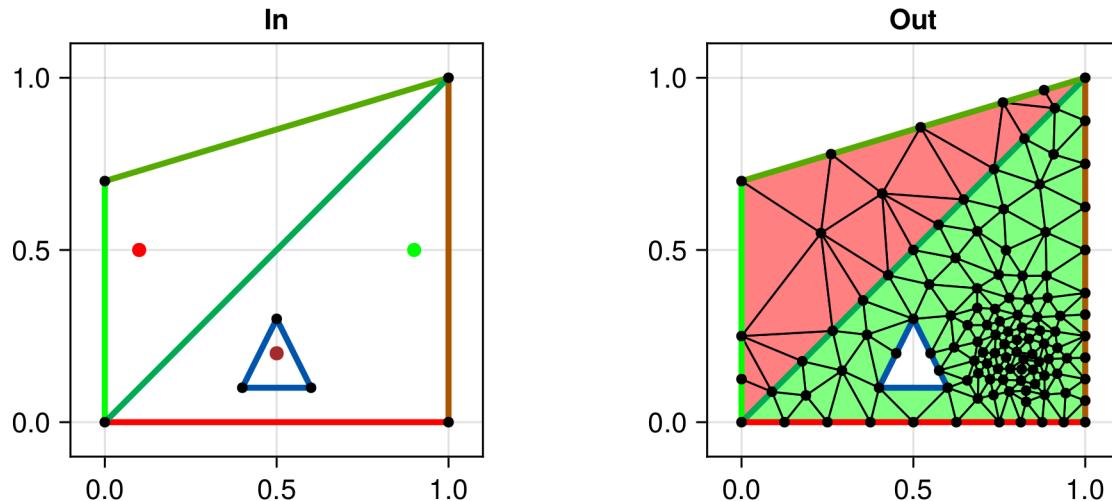
## Subregions

Subregions are defined as regions surrounded by interior boundaries. By placing a "region point" into such a region and specifying a "region number", we can set the cell region marker for all triangles created in the subregion.

## Holes

Holes are defined in a similar way as subregions, but a "hole point" is places into the place which shall become the hole.

```
1 builder3 = let
2     b = SimplexGridBuilder(; Generator = Triangulate, tol = 1.0e-10)
3
4     # Specify points
5     p1 = point!(b, 0, 0)
6     p2 = point!(b, 1, 0)
7     p3 = point!(b, 1, 1)
8     p4 = point!(b, 0, 0.7)
9
10    # Specify outer boundary
11    facetregion!(b, 1)
12    facet!(b, p1, p2)
13    facetregion!(b, 2)
14    facet!(b, p2, p3)
15    facetregion!(b, 3)
16    facet!(b, p3, p4)
17    facetregion!(b, 4)
18    facet!(b, p1, p4)
19
20    # Activate unsuitable callback
21    options!(b; unsuitable = unsuitable)
22
23    # Specify interior boundary
24    facetregion!(b, 5)
25    facet!(b, p1, p3)
26
27    # Coarse elements in upper left region #1
28    cellregion!(b, 1)
29    maxvolume!(b, 0.1)
30    regionpoint!(b, 0.1, 0.5)
31
32    # Fine elements in lower right region #2
33    cellregion!(b, 2)
34    maxvolume!(b, 0.01)
35    regionpoint!(b, 0.9, 0.5)
36
37    # Hole
38    hp1 = point!(b, 0.4, 0.1)
39    hp2 = point!(b, 0.6, 0.1)
40    hp3 = point!(b, 0.5, 0.3)
41    holepoint!(b, 0.5, 0.2)
42    facetregion!(b, 6)
43    facet!(b, hp1, hp2)
44    facet!(b, hp2, hp3)
45    facet!(b, hp3, hp1)
46
47    b
48 end;
```



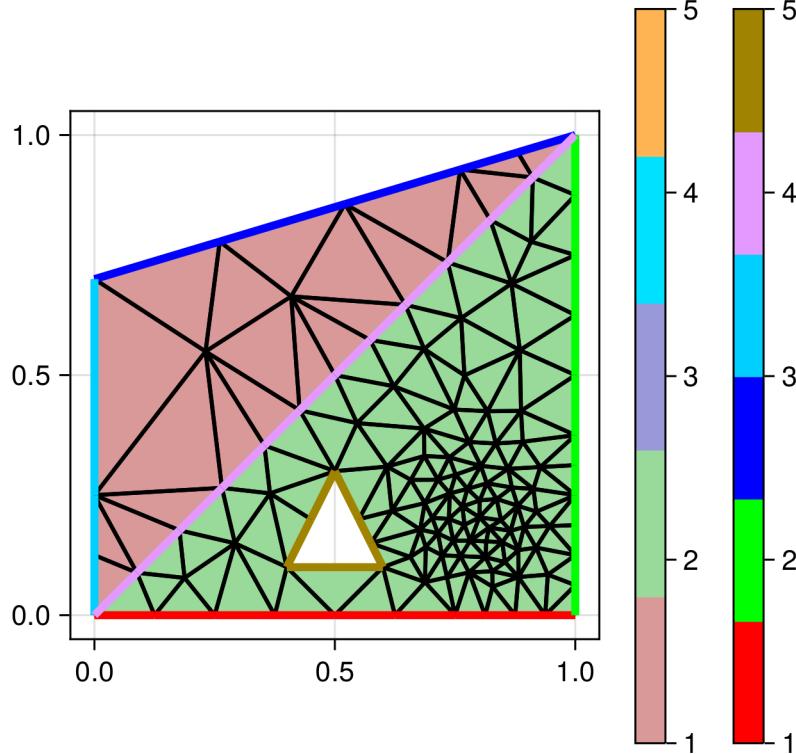
```
1 builderplot(builder3; Plotter = CairoMakie)
```

Found 'resolution' in the theme when creating a 'Scene'. The 'resolution' keyword for 'Scene's and 'Figure's has been deprecated. Use 'Figure(; size = ...)' or 'Scene(; size = ...)' instead, which better reflects that this is a unitless size and not a pixel resolution. The key could also come from 'set\_theme!' calls or related theming functions.

## Create a simplex grid from the builder

```
grid2d3 = ExtendableGrids.ExtendableGrid{Float64, Int32};  
dim: 2 nodes: 117 cells: 199 bfaces: 47
```

```
1 grid2d3 = simplexgrid(builder3)
```



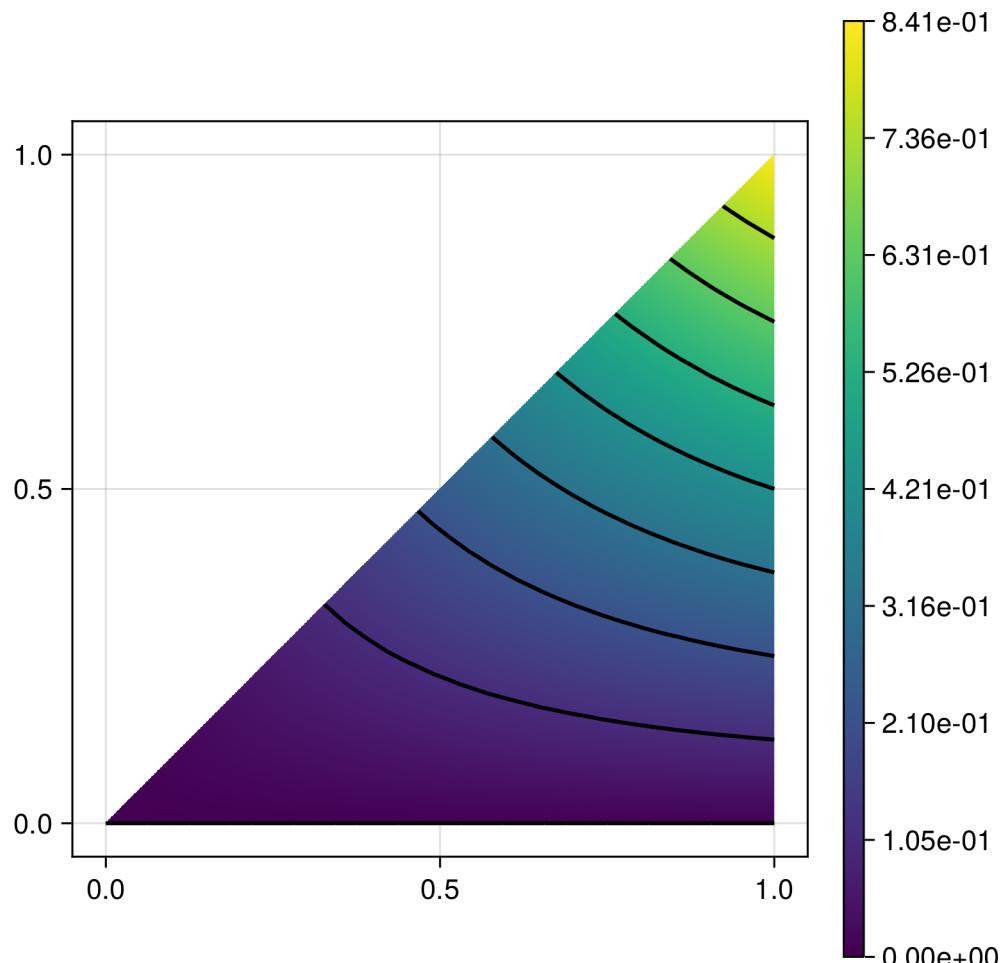
```
1 gridplot(grid2d3; resolution = (400, 400))
```

## Plotting of functions

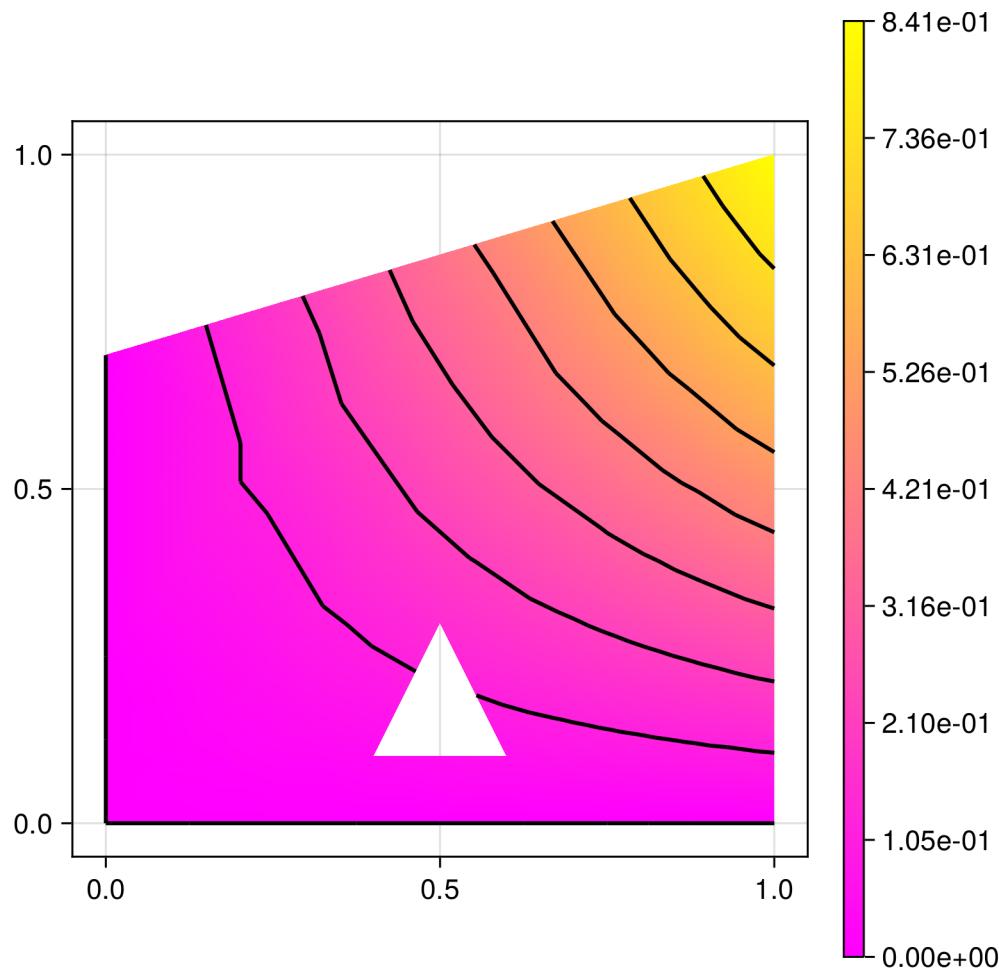
Functions defined on the nodes of a triangular grid can be seen as piecewise linear functions from the P1 finite element space defined by the triangulation.

```
fsin2 =
[0.0, 0.0, 0.841471, 0.38939, 0.420735, 0.631103, 0.603703, 0.467345, 0.720622, 0.736287,
1 fsin2 = map((x, y) -> sin(x) * y, grid2d2)
```

```
fsin3 =
[0.0, 0.0, 0.841471, 0.0, 0.0399334, 0.0599, 0.14776, 0.0, 0.122412, 0.0, 0.0, 0.38939, 0
1 fsin3 = map((x, y) -> sin(y) * x, grid2d3)
```



```
1 scalarplot(grid2d2, fsin2; label = "grid2d2")
```



```
1 scalarplot(grid2d3, fsin3; label = "grid2d3", colormap = :spring, isolines = 10)
```

## 3D Grids

### Tensor product grids

Please note that "masking" is not yet implemented. Furthermore, PyPlot visualization is slow, with GLMakie or PlutoVista it is way faster.

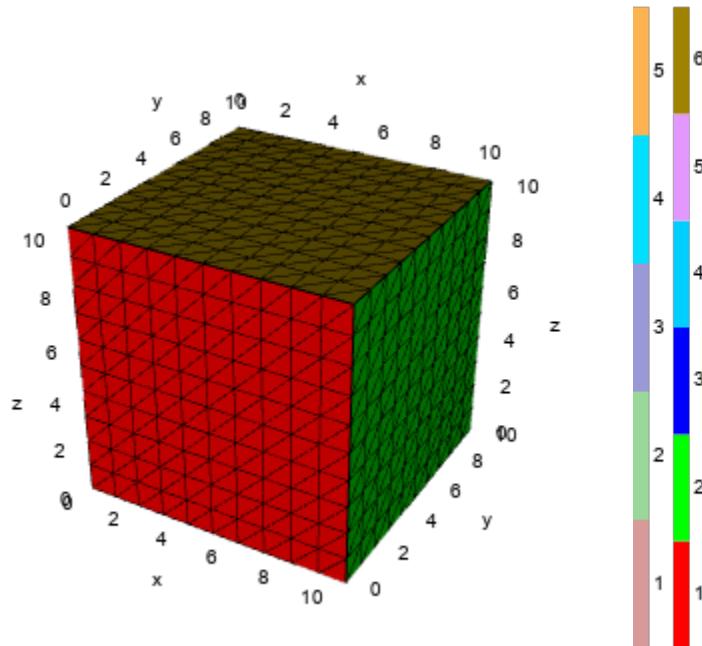
```
X3 = 0.0:1.01:10.1  
1 X3 = range(0.0, 10.1; length = 11)
```

```
grid3d1 = ExtendableGrids.ExtendableGrid{Float64, Int32};  
dim: 3 nodes: 1331 cells: 6000 bfaces: 1200
```

```
1 grid3d1 = simplexgrid(X3, X3, X3)
```

```
func3 =  
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
1 func3 = map((x, y, z) -> sin(x / 2) * cos(y / 2) * z / 10, grid3d1)
```

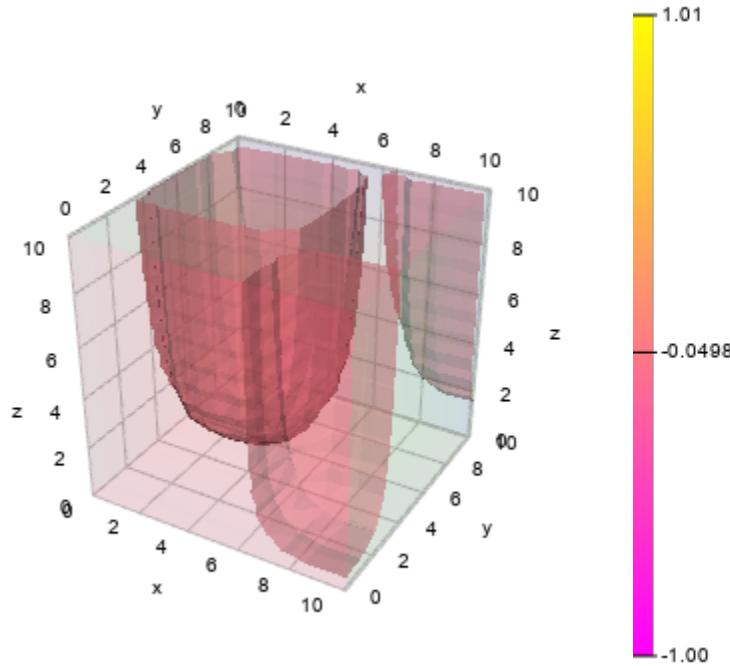
```
p3dg =
```



```
1 p3dg = GridVisualizer(; dim = 3, resolution = (400, 400), Plotter=PlutoVista)
```

```
1 gridplot!(p3dg, grid3d1; zplanes = [zplane], yplanes = [yplane], xplanes =  
[xplane], show = true)
```

```
p3ds =
```



```
1 p3ds = GridVisualizer(; dim = 3, resolution = (400, 400), Plotter=PlutoVista)
```

```
1
2 scalarplot!(p3ds, grid3d1, func3; zplanes = [zplane], yplanes = [yplane], xplanes =
3 [xplane], levels = [flevel], colormap = :spring,
resolution = (200, 200), show = true, levelalpha = 0.5, outlinealpha =
0.1)
```

f=  -0.049773893527225145

x=  10.1

y=  10.1

z=  10.1

mean (generic function with 1 method)

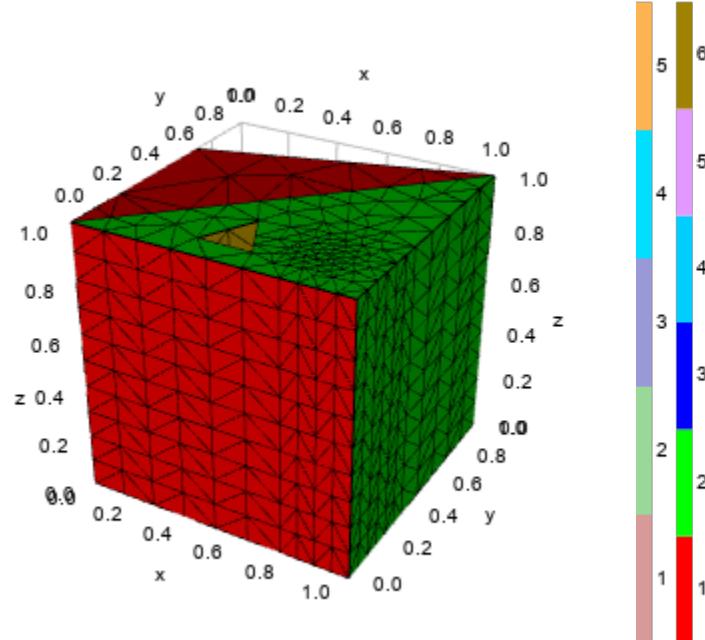
```
1 mean(x) = sum(x) / length(x)
```

extruded\_grid = ExtendableGrids.ExtendableGrid{Float64, Int32};  
dim: 3 nodes: 1287 cells: 5970 bfaces: 1338

```
1 extruded_grid=simplexgrid(grid2d3, 0:0.1:1)
```

16788756 allocations during ExtendableGrids.ITEMTYPE\_CELL volume calculation

```
p3dx =
```



```
1 p3dx = GridVisualizer(; dim = 3, resolution = (400, 400), Plotter=PlutoVista)
```

```
1 gridplot!(p3dx,extruded_grid, Plotter=PlutoVista,zplanes=[zplane2],show=true)
```

z=  1.0

## Unstructured grids

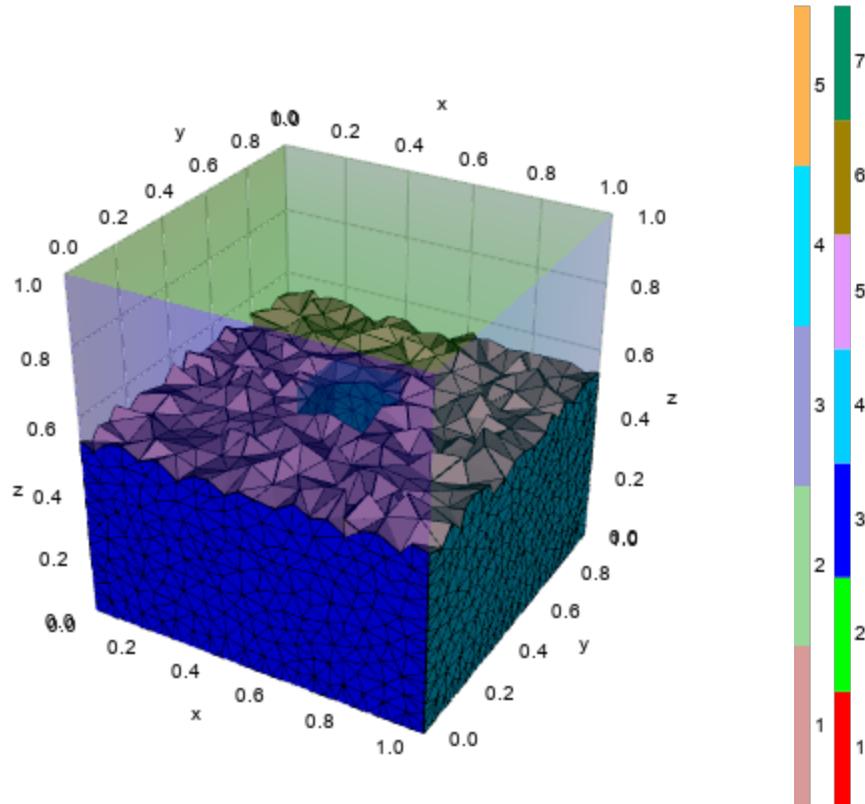
The SimplexGridBuilder API supports creation of three-dimensional grids in way very similar to the 2D case. Just define points with three coordinates and planar (!) facets with at least three points to describe the geometry.

The backend for mesh generation in this case is the [TetGen](#) mesh generator by Hang Si from WIAS Berlin and its Julia wrapper [TetGen.jl](#).

```
1 builder3d = let
2     b = SimplexGridBuilder(; Generator = TetGen)
3
4     p1 = point!(b, 0, 0, 0)
5     p2 = point!(b, 1, 0, 0)
6     p3 = point!(b, 1, 1, 0)
7     p4 = point!(b, 0, 1, 0)
8     p5 = point!(b, 0, 0, 1)
9     p6 = point!(b, 1, 0, 1)
10    p7 = point!(b, 1, 1, 1)
11    p8 = point!(b, 0, 1, 1)
12
13    facetregion!(b, 1)
14    facet!(b, p1, p2, p3, p4)
15    facetregion!(b, 2)
16    facet!(b, p5, p6, p7, p8)
17    facetregion!(b, 3)
18    facet!(b, p1, p2, p6, p5)
19    facetregion!(b, 4)
20    facet!(b, p2, p3, p7, p6)
21    facetregion!(b, 5)
22    facet!(b, p3, p4, p8, p7)
23    facetregion!(b, 6)
24    facet!(b, p4, p1, p5, p8)
25
26    hp1 = point!(b, 0.4, 0.4, 0.4)
27    hp2 = point!(b, 0.6, 0.4, 0.4)
28    hp3 = point!(b, 0.6, 0.6, 0.4)
29    hp4 = point!(b, 0.4, 0.6, 0.4)
30    hp5 = point!(b, 0.4, 0.4, 0.6)
31    hp6 = point!(b, 0.6, 0.4, 0.6)
32    hp7 = point!(b, 0.6, 0.6, 0.6)
33    hp8 = point!(b, 0.4, 0.6, 0.6)
34
35    facetregion!(b, 7)
36    facet!(b, hp1, hp2, hp3, hp4)
37    facet!(b, hp5, hp6, hp7, hp8)
38    facet!(b, hp1, hp2, hp6, hp5)
39    facet!(b, hp2, hp3, hp7, hp6)
40    facet!(b, hp3, hp4, hp8, hp7)
41    facet!(b, hp4, hp1, hp5, hp8)
42    holepoint!(b, 0.5, 0.5, 0.5)
43
44    b
45 end;
```

```
grid3d2 = ExtendableGrids.ExtendableGrid{Float64, Int32};
dim: 3 nodes: 4650 cells: 21311 bfaces: 4890
```

```
1 grid3d2 = simplexgrid(builder3d; maxvolume = 0.0001)
```



```
1 gridplot(grid3d2; zplane = 0.5, azim = 20, elev = 20, linewidth = 0.5, outlinealpha = 0.3, Plotter=PlutoVista)
```

## Gmsh examples

```
1 using Gmsh
```

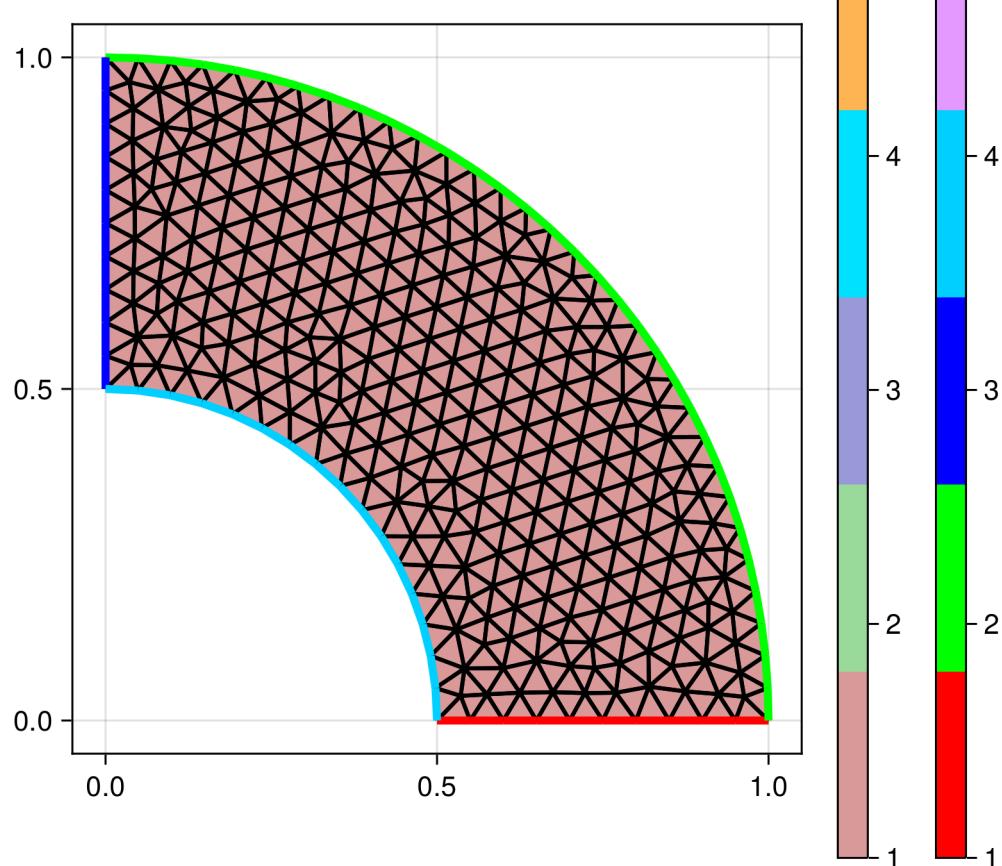
Gmsh is an open source 3D finite element mesh generator with a built-in CAD engine and post-processor.

ExtendableGrids.jl now has an extension which allows to interact with Gmsh using its Julia API

Example adopted from the documentation of Ferrite.jl, a finite element solver package.

## quarter\_ring (generic function with 2 methods)

```
1 function quarter_ring(h=0.05)
2     # Initialize gmsh
3     Gmsh.initialize()
4     gmsh.option.set_number("General.Verbosity", 2)
5
6     # Add the points
7     o = gmsh.model.geo.add_point(0.0, 0.0, 0.0, h)
8     p1 = gmsh.model.geo.add_point(0.5, 0.0, 0.0, h)
9     p2 = gmsh.model.geo.add_point(1.0, 0.0, 0.0, h)
10    p3 = gmsh.model.geo.add_point(0.0, 1.0, 0.0, h)
11    p4 = gmsh.model.geo.add_point(0.0, 0.5, 0.0, h)
12
13    # Add the lines
14    l1 = gmsh.model.geo.add_line(p1, p2)
15    l2 = gmsh.model.geo.add_circle_arc(p2, o, p3)
16    l3 = gmsh.model.geo.add_line(p3, p4)
17    l4 = gmsh.model.geo.add_circle_arc(p4, o, p1)
18
19    # Create the closed curve loop and the surface
20    loop = gmsh.model.geo.add_curve_loop([l1, l2, l3, l4])
21    surf = gmsh.model.geo.add_plane_surface([loop])
22
23    # Synchronize the model
24    gmsh.model.geo.synchronize()
25
26    # Create the physical domains
27    gmsh.model.add_physical_group(1, [l1], -1, "Γ1")
28    gmsh.model.add_physical_group(1, [l2], -1, "Γ2")
29    gmsh.model.add_physical_group(1, [l3], -1, "Γ3")
30    gmsh.model.add_physical_group(1, [l4], -1, "Γ4")
31    gmsh.model.add_physical_group(2, [surf])
32
33    # Generate a 2D mesh
34    gmsh.model.mesh.generate(2)
35
36    grid=simplexgrid_from_gmsh(gmsh.model)
37
38    # Finalize the Gmsh library
39    gmsh.finalize()
40
41    return grid
42 end
```



```
1 gridplot(quarter_ring())
```

# Table of Contents

## Grid creation and visualization

1D grids

  Modifying region markers

  Creating locally refined grids

  Plotting functions

2D grids

  Tensor product grids

  Unstructured grids

  More complicated grids

    Local refinement

    Interior boundaries

    Subregions

    Holes

  Plotting of functions

3D Grids

  Tensor product grids

  Unstructured grids

Gmsh examples