# Template for VoronoiFVM notebook

v1.0, 2024-11-17

```julia
begin
    using VoronoiFVM # The VoronoiFVM package
    using ExtendableGrids# Manage grids, create rectangular grids
    using SimplexGridFactory, Triangulate# Create grids with general geometry
    using GridVisualize, CairoMakie # Visualization
    CairoMakie.activate!(type="png")
    GridVisualize.default_plotter!(CairoMakie)
    using PlutoUI# Sliders etc.
end
```

## Table of Contents

```julia
PlutoUI.TableOfContents()
```

# Grid Creation

We create grids in domains $\Omega$ which have disjoint boundary parts $\Gamma_1$, $\Gamma_2$.

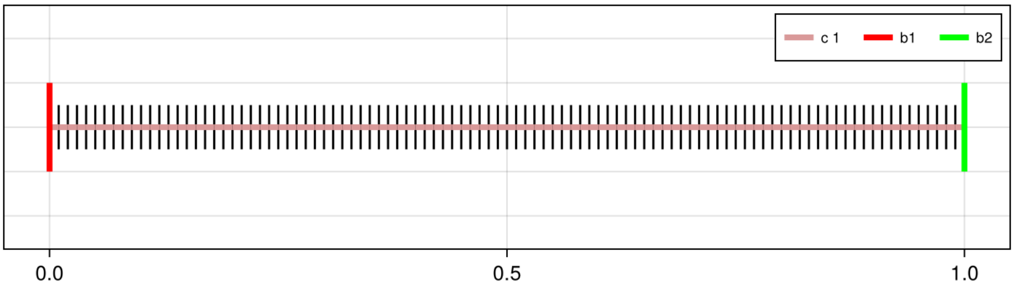
## 1D Grid

```
X = 0.0:0.01:1.0
```

```julia
X=range(0,1,length=101)
```

```
grid1d = ExtendableGrids.ExtendableGrid{Float64, Int32}
              dim =       1
           nnodes =     101
           ncells =     100
          nbfaces =       2
```

```julia
grid1d=simplexgrid(X)
```



```julia
gridplot(grid1d, size=(700,200), legend=:rt)
```

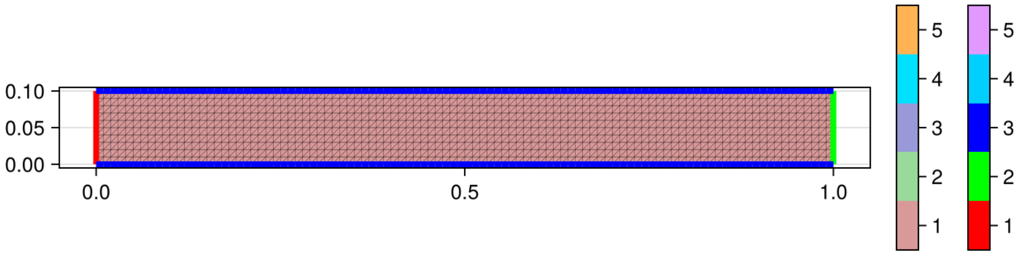
## 2D rectangular grid

```
Y = 0.0:0.01:0.1
```

```julia
Y=range(0,0.1, length=11)
```

```
ExtendableGrids.ExtendableGrid{Float64, Int32}
      dim =        2
   nnodes =     1111
   ncells =     2000
  nbfaces =      220
```

```
begin
    grid2d=simplexgrid(X,Y)
    bfacemask!(grid2d,[0,0], [1,1], 3, allow_new=false)
    bfacemask!(grid2d,[0,0], [0,1], 1)
    bfacemask!(grid2d,[1,0], [1,1], 2)
end

```

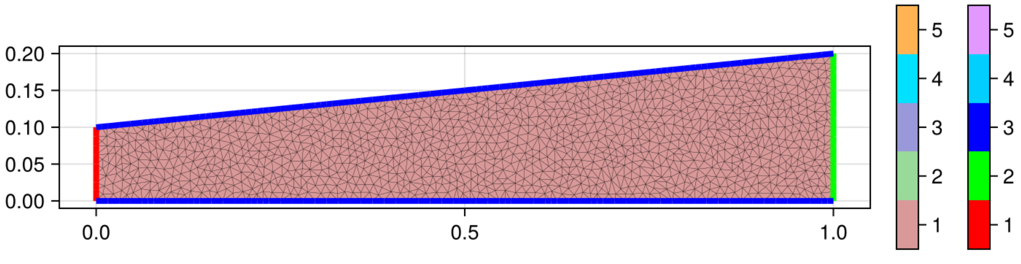


```
gridplot(grid2d, size=(700,200), linewidth=0.1)
```

## 2D general grid

```
ExtendableGrids.ExtendableGrid{Float64, Int32}
      dim =        2
   nnodes =     1259
   ncells =     2329
  nbfaces =      187
```

```
begin
    builder = SimplexGridBuilder(; Generator = Triangulate)

    p1 = point!(builder, 0, 0)
    p2 = point!(builder, 1, 0)
    p3 = point!(builder, 1, 0.2)
    p4 = point!(builder, 0, 0.1)

    facetregion!(builder, 3)
    facet!(builder, p1, p2)
    facet!(builder, p3, p4)
    facetregion!(builder, 1)
    facet!(builder, p1,p4)
    facetregion!(builder, 2)
    facet!(builder, p2,p3)

    options!(builder; maxvolume = 1.0e-4)

    ggrid2d = simplexgrid(builder)


end
```



```
gridplot(ggrid2d, size=(700,200), linewidth=0.1)
```

# Problem description

Regard the following system of reacting species with the reaction

$$u_1 \leftrightharpoons 2u_2$$

in the time interval $[0, t_{end}]$

$$\begin{aligned}
\partial_t u_1 - \nabla \cdot D_1 \nabla u_1 + R(u_1, u_2) &= 0\\
\partial_t u_2 - \nabla \cdot D_2 \nabla u_2 - R(u_1, u_2) &= 0\\
R(u1, u2) &= kp + u_1 - km - u_2^2\\
u_1|_{t=0} = u_2|_{t=0} &= 0\\
u_1|_{\Gamma_1} &= 1\\
D_2 \partial_n u_2|_{\Gamma_1} &= 0\\
D_1 \partial_n u_1|_{\Gamma_2} &= 0\\
u_2|_{\Gamma_2} &= 0
\end{aligned}$$

```
myflux (generic function with 1 method)
```

```julia
function myflux(y,u, edge, data)
    (;D_1, D_2)= data
    y[1]= D_1*(u[1,1]- u[1,2])
    y[2]= D_2*(u[2,1]- u[2,2])
end
```

```
myreaction (generic function with 1 method)
```

```julia
function myreaction(y, u, node, data)
    (;kp, km) = data
    R=kp*u[1] - km*u[2]^2
    y[1]=R
    y[2]=-R
end

```

```
mystorage (generic function with 1 method)
```

```julia
function mystorage(y,u, node, data)
    y[1]=u[1]
    y[2]=u[2]
end

```

```
mybc (generic function with 1 method)
```

```julia
function mybc(y,u, bnode, data)
    boundary_dirichlet!(y,u,bnode, species=1, value=1, region=1)
    boundary_dirichlet!(y,u,bnode, species=2, value=0, region=2)
end

```

```
mydata = ▶(kp = 1.0, km = 0.1, D_1 = 1, D_2 = 0.1)
```

```julia
mydata= (kp=1.0, km =0.1, D_1=1, D_2=0.1)
```

```
mygrid = ExtendableGrids.ExtendableGrid{Float64, Int32}
            dim =       2
         nnodes =    1259
         ncells =    2329
        nbfaces =     187
```

```julia
mygrid=ggrid2d
```

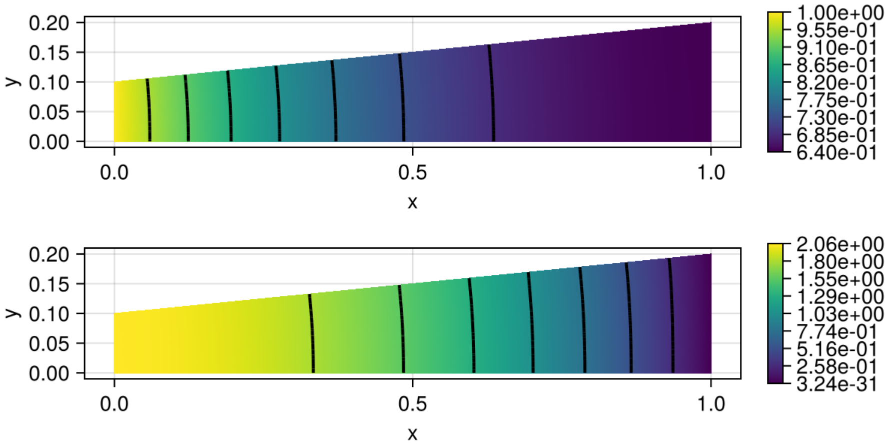
```
mysystem =
VoronoiFVM.System{Float64, Float64, Int32, Int64, Matrix{Int32}}(
  grid = ExtendableGrids.ExtendableGrid{Float64, Int32}(dim=2, nnodes=1259, ncells=232
  nbfaces=187),
  physics = Physics(data=@NamedTuple{kp::Float64, km::Float64, D_1::Int64,
  D_2::Float64}, flux=myflux, storage=mystorage, reaction=myreaction, breaction=mybc,
  num_species = 2)
```

```julia
mysystem=VoronoiFVM.System(mygrid; flux=myflux, reaction=myreaction,
storage=mystorage, breaction=mybc, data=mydata, species=[1,2])
```

# Stationary solution

```
mysol = 2×1259 VoronoiFVM.DenseSolutionArray{Float64, 2}:
         1.0       0.640308      0.639961      …  0.640139    0.640587  0.640777  0.640427
         2.06278   4.23622e-31   3.24012e-31      0.0808262   0.169523  0.192866  0.145453
```

```
mysol=solve(mysystem)
```



```
let
    vis=GridVisualizer(size=(600,300),layout=(2,1), legend=:rt)
    scalarplot!(vis[1,1],mygrid, mysol[1,:], label="u_1")
    scalarplot!(vis[2,1],mygrid, mysol[2,:], label="u_2")
    reveal(vis)
end
```

# Transient solution

```
tend = 10
```

```
tend=10
```

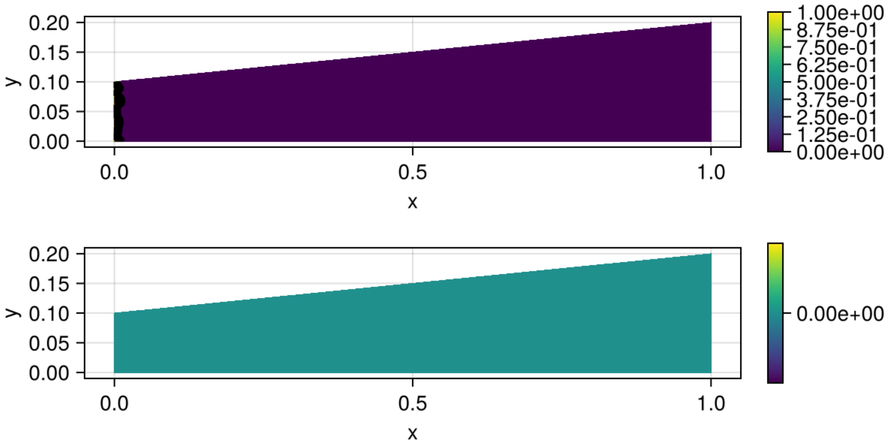
```
myinival =
2×1259 VoronoiFVM.DenseSolutionArray{Float64, 2}:
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
myinival=unknowns(mysystem, inival=0)
```

```
mytsol =
t: 64-element Vector{Float64}:
  0.0
  0.001
  0.0010999999999999999
  0.0012199999999999978
  0.0013639999999999963
  0.0015367999999999944
  0.0017441599999999922
  ⋮
  6.0509416850392705
  6.999044549788927
  7.749283412341695
  8.499522274894463
  9.249761137447232
 10.0
u: 64-element Vector{Matrix{Float64}}:
 [1.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
 [1.0 3.179554553889352e-14 … 6.549149473352493e-14 4.583693869516127e-14; 0.000796157
 [1.0 3.532838393210388e-14 … 7.276832748169428e-14 5.0929931883512466e-14; 0.00087457
 [1.0 4.014589083193617e-14 … 8.269128122919793e-14 5.787492259490045e-14; 0.00096767
 [1.0 4.689940517749544e-14 … 9.660196405280116e-14 6.761089088189293e-14; 0.00107857
 [1.0 5.6696572990202296e-14 … 1.1678187143713848e-13 8.173463597907736e-14; 0.0012110
 [1.0 7.152878102316569e-14 … 1.4733280106623088e-13 1.031169711080399e-13; 0.0013694
 ⋮
 [1.0 0.633752815899025 … 0.6342201060700844 0.6338714022418487; 1.9553106929208774 4.
 [1.0 0.6360593054252665 … 0.6365274112354683 0.6361781332608476; 1.9943507826733005 4
 [1.0 0.6373502688656448 … 0.6378188149917686 0.6374692276648308; 2.0156534312501324 4
 [1.0 0.6382560286235159 … 0.6387248734531001 0.6383750766527738; 2.0303324345889306 4
 [1.0 0.638887819387511 … 0.6393568673921701 0.6390069283169636; 2.0404427704759933 4.
 [1.0 0.6393267051792006 … 0.6397958917865866 0.639445855758309; 2.047404444557198 4.2
```

```
mytsol=solve(mysystem; inival=myinival, times=(0,tend), Δt=1.0e-4,
force_first_step=true)
```

0.0



```
let
    u=mytsol(myt)
    vis=GridVisualizer(size=(600,300),layout=(2,1), legend=:rt)
    scalarplot!(vis[1,1],mygrid, u[1,:], label="u_1")
    scalarplot!(vis[2,1],mygrid, u[2,:], label="u_2")
    reveal(vis)
end
```