geom2d

November 1, 2022

1 4.1.1 Geometries in 2D

We have to import the `SplineGeometry` class from the `geom2d` module to be able to generate two-dimensional geometries. After importing the module we can create a new `SplineGeometry`.

[ ]:
```
# import netgen.gui
from ngsolve import Mesh
from ngsolve.webgui import Draw
from netgen.geom2d import SplineGeometry
geo = SplineGeometry()
```

Now we can use one the predefined objects (Rectangle,Circle) or generate our own geometry with lines or rational splines of 2nd order.

1.1 Predefined geometries

First we use the predefined ones and add a rectangle and a circle to our geometry with the boundary conditions `rectangle` and `circle`.

[ ]:
```
geo.AddRectangle((-1,-1),(1,1),bc="rectangle")
```

[ ]:
```
geo.SetDomainMaxH(2, 0.02)
ngmesh = geo.GenerateMesh(maxh=0.1)
Draw (Mesh(ngmesh))
```

To get a proper geometry we have to set domain numbers for the domain on the left side of the curve and for the one in its right side. In this case the curves are parametrized in a mathematical positive sense. Additionally we can use `SetMaterial` to identify the domains with names.

[ ]:
```
geo = SplineGeometry()
geo.AddRectangle(p1=(-1,-1),
                 p2=(1,1),
                 bc="rectangle",
                 leftdomain=1,
                 rightdomain=0)
geo.AddCircle(c=(0,0),
              r=0.5,
              bc="circle",
              leftdomain=2,
              rightdomain=1)
geo.SetMaterial (1, "outer")
geo.SetMaterial (2, "inner")
```

[ ]:
```
geo.SetDomainMaxH(2, 0.02)
ngmesh = geo.GenerateMesh(maxh=0.1)
Draw (Mesh(ngmesh))
```

1.2 Using lines and splines

We define a new geometry and write a list of points we want to use for out geometry and add them to geo geometry.

[ ]:
```
geo = SplineGeometry()

pnts =[(0,0),
       #(0,0,0.05), # define a local mesh refinement for one point
       (1,0),
       (1,0.5),
       (1,1),
       (0.5,1),
       (0,1)]

p1,p2,p3,p4,p5,p6 = [geo.AppendPoint(*pnt) for pnt in pnts]
```

Then we define the curves which define our geometry and add them to the geometry using `Append`.

[ ]:
```
curves = [[["line",p1,p2],"bottom"],
          [["line",p2,p3],"right"],
          [["spline3",p3,p4,p5],"curve"],
          [["line",p5,p6],"top"],
          [["line",p6,p1],"left"]]

[geo.Append(c,bc=bc) for c,bc in curves]
```

[ ]:
```
ngmesh = geo.GenerateMesh(maxh=0.2)
```

[ ]:
```
# from ngsolve import *
# mesh = Mesh(ngmesh)
# mesh.Curve(3)
```

Additonally to the boundary condition one can set a maximal mesh size for a whole curve with an optional argument `maxh`.

[ ]:
```
geo = SplineGeometry()

p1,p2,p3,p4,p5,p6 = [geo.AppendPoint(*pnt) for pnt in pnts]
```

```
geo.Append(["line",p1,p2],maxh=0.02)
geo.Append(["line",p2,p4])
geo.Append(["line",p4,p6])
geo.Append(["line",p6,p1])

ngmesh = geo.GenerateMesh(maxh=0.2)#,quad_dominated=True)
Draw (Mesh(ngmesh))
```

1.3 Periodic geometries

The following example shows how construct a geometry for a periodic L_2 finite element space.

Again we start with adding the points to the geometry. In this case the points of a hexagon. For the first three segments we save the return value (the line number) of `geo.Append`. Now we can use those line numbers to identify each of the last three segments with the opposite, already added one. This identification is done with the optional argument `copy`. The meshing algorithm then just copies the boundary mesh to the opposite segment. Thus the segments have to have the same orientation.

[ ]:
```
from math import pi, cos, sin
geo = SplineGeometry()
pnums = [ geo.AddPoint(cos(phi),sin(phi)) for phi in [x*pi/3 for x in range(6)]␣
  ↪]
l1 = geo.Append(["line", 0, 1], leftdomain=1, rightdomain=0, bc="upperRight")
l2 = geo.Append(["line", 1, 2], leftdomain=1, rightdomain=0, bc="upperCenter")
l3 = geo.Append(["line", 2, 3], leftdomain=1, rightdomain=0, bc="upperLeft")
geo.Append(["line", 0, 5], leftdomain=0, rightdomain=1, bc="lowerRight", copy =␣
  ↪l3)
geo.Append(["line", 5, 4], leftdomain=0, rightdomain=1, bc="lowerCenter", copy␣
  ↪= l2)
geo.Append(["line", 4, 3], leftdomain=0, rightdomain=1, bc="lowerLeft", copy =␣
  ↪l1)
ngmesh = geo.GenerateMesh(maxh=0.1)
```

[ ]:
```
```