Netgen Meets Firedrake
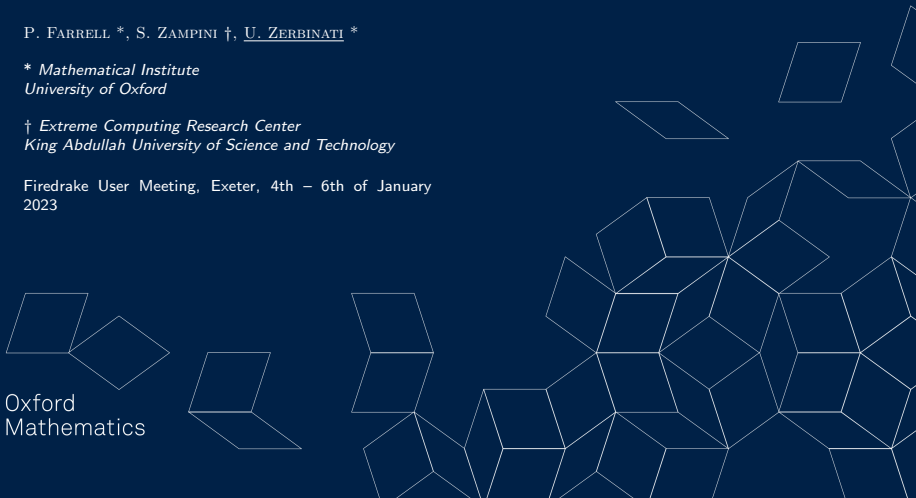
P. Farrell *, S. Zampini †, U. Zerbinati *

* *Mathematical Institute*
University of Oxford

† *Extreme Computing Research Center*
King Abdullah University of Science and Technology

Oxford
Mathematics

Solving A Partial Differential Equation

When solving a partial differential equation the following macro steps can be identified:

- ▶ Geometrical modelling,
- ▶ Meshing using an advancing front method,
- ▶ Discretising a Partial Differential Equation (**PDE**),
- ▶ Solving the linear or nonlinear system arsing from the discretisation.

We aim to allow the Firedrake user to do all the steps above described in a single script.

NetGen is an advancing front 2D/3D-mesh generator, with many interesting features. Among the most important:

- ▶ Python wrapping (through PyBind11),
- ▶ Multiple ways of describing the geometry to be meshed, i.e. its builtin **Constructive Solid Geometry** (**CSG**) and the **Open Cascade Technology** (**OCCT**) geometry kernel,
- ▶ Supports mesh refinement (also anisotropic mesh refinement).

Getting Started – Installing NetGen

Install NetGen using Firedrake scripts

```
python3 firedrake-install --netgen
python3 firedrake-update --netgen
```

PETSc

If you are using an external PETSc installation, such installation should be updated to include commit 654059db.
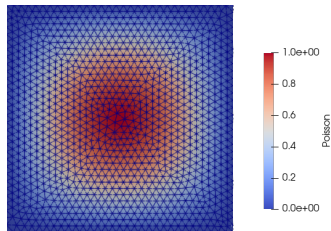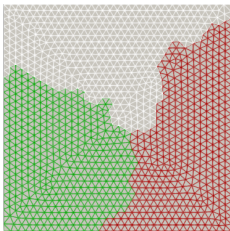
NetGen

If installing from scratch use the NetGen repository from the Firdrake project GitHub.

```python
1 from firedrake import *
2 from mpi4py import MPI
3 from netgen.geom2d import SplineGeometry
4 import netgen
5 comm = MPI.COMM_WORLD
6 if comm.rank == 0:
7     geo = SplineGeometry()
8     geo.AddRectangle((0, 0), (pi, pi), bc="rect")
9     ngmesh = geo.GenerateMesh(maxh=0.1)
10    labels = ngmesh.GetBCIDs("rect")
11 else:
12    ngmesh = netgen.libngpy._meshing.Mesh(2)
13    labels = None
14 labels = comm.bcast(labels, root=0)
15 msh = Mesh(ngmesh)
16 File("VTK/Poisson2DMesh.pvd").write(msh)
```

Getting Started – Unstructed Mesh

Getting Started – CSG 2D

```
1 from firedrake import *
2 from netgen.geom2d import SplineGeometry
3 geo = SplineGeometry()
4 geo.AddRectangle(p1=(-1,-1),p2=(1,1),bc="rectangle",
5     leftdomain=1,rightdomain=0)
6 geo.AddCircle(c=(0,0),r=0.5,bc="circle",
7     leftdomain=2,rightdomain=1)
8 geo.SetMaterial (1, "outer")
9 geo.SetMaterial (2, "inner")
10 geo.SetDomainMaxH(2, 0.05)
11 ngmesh = geo.GenerateMesh(maxh=0.2)
12 msh = Mesh(ngmesh)
13 File("VTK/CSG2DMesh.pvd").write(msh)
```
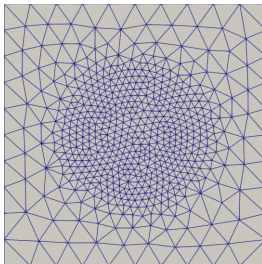
Material Identifiers

The material identifier specified above are carried to the DMPLEX as CELL_SETS_LABEL.

The Open Cascade Technology Kernel

▶ Basic OCCT objects can be used in NetGen such as: `Box`, `Cylinder`, `Point`, `Segment` and `ArcOfCircle`.

▶ The `fuse`,`cut` and `common` operations between OCCT objects have been wrapped in NetGen.

▶ Transformation operations such as `Move` and `Rotate` have also been wrapped into NetGen

```
1 from firedrake import *
2 from netgen.occ import *
3 box = Box(Pnt(0,0,0), Pnt(1,1,1))
4 cyl = Cylinder(Pnt(1,0.5,0.5), X, r=0.3, h=0.5)
5 solid1 = box+cyl
6 solid2 = solid1.Rotate(Axis((0,0,0),Y),180).Move
      ((2.5,0.,1.))
7 solid = solid2+solid1
8 geo = OCCGeometry(solid)
9 ngmesh = geo.GenerateMesh(maxh=0.1)
10 msh = Mesh(ngmesh)
11 File("VTK/OCC.pvd").write(msh)
```

The Open Cascade Technology Kernel
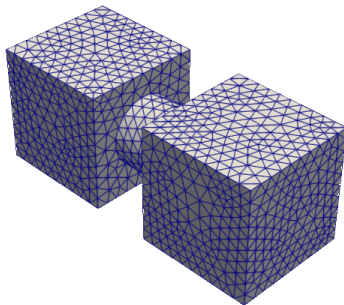
The OCCT Kernel – The Flask Example: I


```
1  from firedrake import *
2  from netgen.occ import *
3  H, W, T  = 70.,50.,30.
4  P1,P2=Pnt(-W/2,0,0),Pnt(-W/2,-T/4,0)
5  P3,P4=Pnt(0,-T/2.,0),Pnt(W/2.,-T/ 4.,0)
6  P5=Pnt(W/2.,0, 0)
7  S1=Segment(P1, P2)
8  A=ArcOfCircle(P2, P3, P4)
9  S2=Segment(P4, P5)
10 wire=Wire([S1, A, S2])
11 mwire=wire.Mirror(Axis((0,0,0),X))
12 wire=Wire([wire,mwire])
13 base=Face(wire).Extrude(0.3*Z)
14 msh=Mesh(OCCGeometry(base).GenerateMesh(
        maxh=0.5))
15 File("VTK/OCCBottleBase.pvd").write(msh)
```
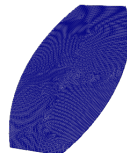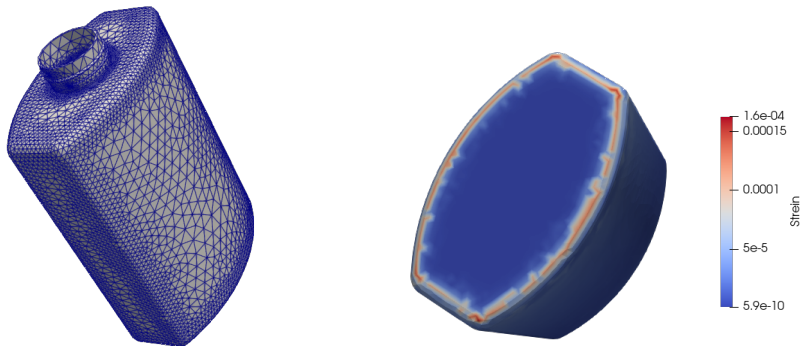
```
1 f = Face(wire)
2 f.bc("bottom")
3 body=f.Extrude(H*Z)
4 body=body.MakeFillet(body.edges,T
      /12)
5 neckAx=Axes(body.faces.Max(Z).
      center,Z)
6 NeckR,NeckH=T/4,H/10
7 neck=Cylinder(neckAx,NeckR,NeckH)
8 body=body+neck
9 fmax=body.faces.Max(Z)
10 thickbody=body.MakeThickSolid([fmax
      ],-T/50,1.e-3)
11 msh=Mesh(OCCGeometry(thickbody).
      GenerateMesh(maxh=3.))
12 File("VTK/OCCBottleBody.pvd").write
      (msh)
```
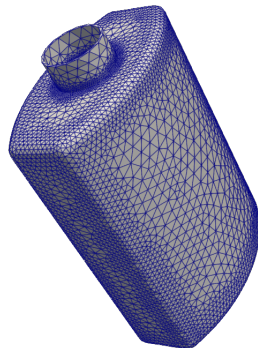
```
1  cyl1=Cylinder(neckAx, NeckR*0.99,1).faces[0]
2  cyl2=Cylinder(neckAx, NeckR*1.05,1).faces[0]
3  aPnt=Pnt(2*pi,NeckH/2.0)
4  aDir=Dir(2*pi,NeckH/4.0)
5  anAx2d=gp_Ax2d(aPnt, aDir)
6  aMajor,aMinor=2*pi,NeckH/10
7  arc1=Ellipse(anAx2d,aMajor,aMinor).Trim(0,pi)
8  arc2=Ellipse(anAx2d,aMajor,aMinor/4).Trim(0,pi)
9  seg=Segment(arc1.start, arc1.end)
10 wire1=Wire([Edge(arc1,cyl1),Edge(seg,cyl1)])
11 wire2=Wire([Edge(arc2,cyl2),Edge(seg,cyl2)])
12 threading=ThruSections([wire1,wire2])
13 bottle=thickbody+threading
14 ngmsh = OCCGeometry(bottle).GenerateMesh(maxh=3.)
15 msh = Mesh(ngmsh)
```
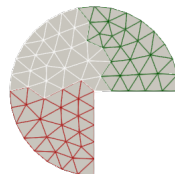
The OCCT Kernel – The Flask Example: III

```
1 tolerance = 1e-16
2 max_iterations = 15
3 exact = 3.375610652693620492628**2
4 geo=SplineGeometry()
5 pnts=[(0, 0), (1, 0), (1, 1),(0, 1),
6      (-1, 1), (-1, 0),(-1, -1), (0, -1)]
7 P = [geo.AppendPoint(*pnt) for pnt in
      pnts]
8 L,B=["line","spline3"],["line","curve"]
9 curves = [[[L[0],P[0],P[1]],B[0]],
10           [[L[1],P[1],P[2],P[3]],B[1]],
11           [[L[1],P[3],P[4],P[5]],B[1]],
12           [[L[1],P[5],P[6],P[7]],B[1]],
13           [[L[0],P[7],P[0]],B[0]]]
14 [geo.Append(c, bc=bc) for c, bc in
      curves]
```
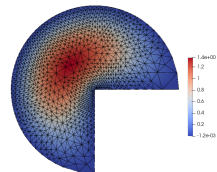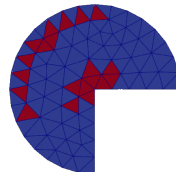
Mesh Refinement – Adaptive Mesh Refinement: II

```python
1  if comm.rank == 0:
2      ngmsh = geo.GenerateMesh(maxh=0.2)
3      labels=sum([ngmsh.GetBCIDs(label)
       for label in ["line","curve"]], [])
4  else:
5      ngmsh=netgen.libngpy._meshing.Mesh
       (2)
6      labels = None
7  msh = Mesh(ngmsh)
8  labels = comm.bcast(labels, root=0)
9  for i in range(max_iterations):
10     lam, uh, V = Solve(msh,labels)
11     mark = Mark(msh, uh, lam)
12     msh = msh.Refine(mark)
13     File("VTK/PacManAdp.pvd").write(uh,
       mark)
14 assert(abs(lam-exact)<1e-2)
```

- ▶ Mesh **hierarchy awareness**, using Firedrake `HierarchyBase` class.
- ▶ Support for **anisotropic** mesh refinement, using NetGen `ZRefinement` and `HPRefinement` methods.
- ▶ Support for NetGen **high order** mesh in Firedrake.
- ▶ Support for MFEM **GLVIS** mesh and solution live display, thanks to PETSc-GLVIS interface.