



Mathematical
Institute

Netgen Meets Firedrake

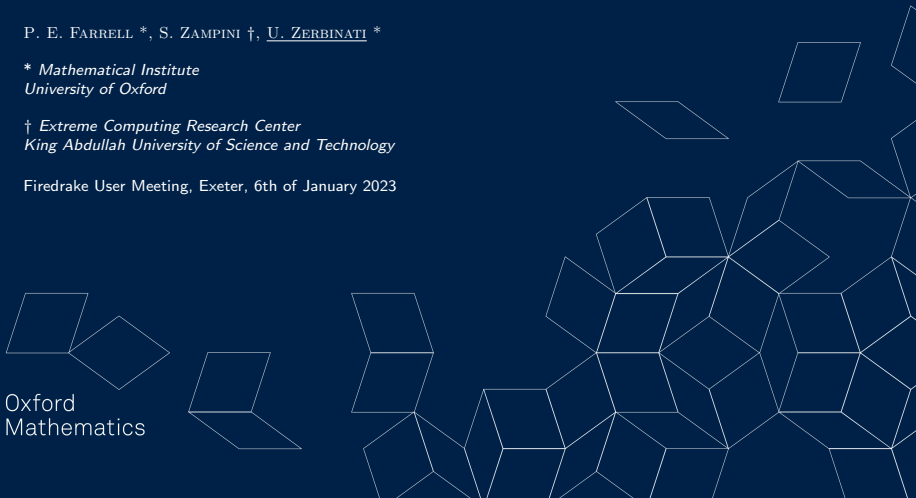
P. E. FARRELL ^{*}, S. ZAMPINI [†], U. ZERBINATI ^{*}

^{*} *Mathematical Institute
University of Oxford*

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

Firedrake User Meeting, Exeter, 6th of January 2023

Oxford
Mathematics



Solving A Partial Differential Equation

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

- ▶ Geometrical modelling,
- ▶ Meshing,
- ▶ Discretising a PDE,
- ▶ Solving the linear or nonlinear system.

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.

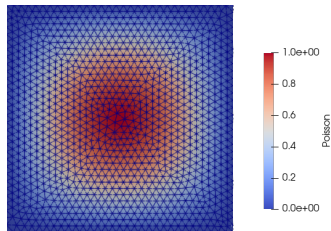
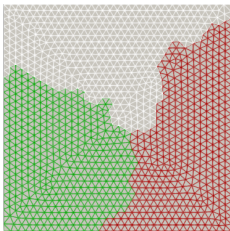
NetGen

If installing from scratch, use the NetGen fork from the Firedrake project GitHub.

Getting Started – Unstructured Mesh

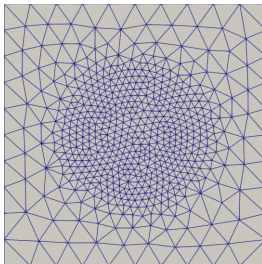
```
1 from firedrake import *
2 from mpi4py import MPI
3 from netgen.geom2d import SplineGeometry
4 import netgen
5 comm = MPI.COMM_WORLD
6 if comm.rank == 0:
7     geo = SplineGeometry()
8     geo.AddRectangle((0, 0), (pi, pi), bc="rect")
9     ngmesh = geo.GenerateMesh(maxh=0.1)
10    labels = ngmesh.GetBCIDs("rect")
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 – Unstructured 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)
```

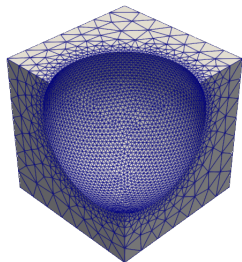


Material Identifiers

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

Getting Started – CSG 3D

```
1 from firedrake import *
2 from netgen.csg import *
3 geo = CSGeometry()
4 brick = OrthoBrick(Pnt(-1,-1,-1),Pnt
    (1,1,1))
5 sphere = Sphere(Pnt(0.5,0.5,0.5),1)
6 sphere.maxh(0.05)
7 geo.Add(brick-sphere)
8 ngmesh = geo.GenerateMesh(maxh=0.3)
9 msh = Mesh(ngmesh)
10 File("VTK/CSG3DMesh.pvd").write(msh)
```

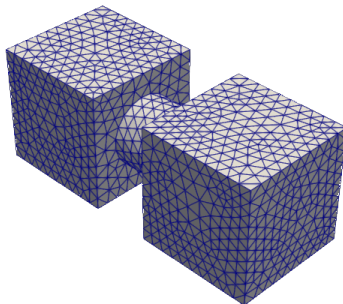


- ▶ Basic OCCT objects can be used in NetGen such as: Box, Cylinder, Point, Segment and ArcOfCircle.
- ▶ The fuse, cut and common operations between OCCT objects have been wrapped in NetGen.
- ▶ Transformation operations such as Move and Rotate have also been wrapped into NetGen.

The Open Cascade Technology Kernel

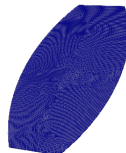
```
1 from firedrake import *
2 from netgen.occ import *
3 box = Box(Pnt(0,0,0), Pnt(1,1,1))
4 cyl = Cylinder(Pnt(1,0.5,0.5), X, r=0.3, h=0.5)
5 solid1 = box+cyl
6 solid2 = solid1.Rotate(Axis((0,0,0),Y),180).Move
    ((2.5,0.,1.))
7 solid = solid2+solid1
8 geo = OCCGeometry(solid)
9 ngmesh = geo.GenerateMesh(maxh=0.1)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

The Open Cascade Technology Kernel



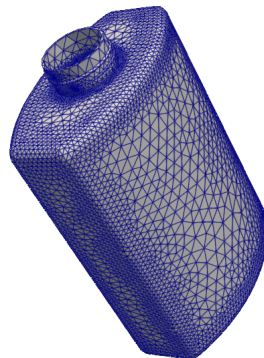
The OCCT Kernel – The Flask Example: I

```
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(OCCTGeometry(base).GenerateMesh(
    maxh=0.5))
15 File("VTK/OCCTBottleBase.pvd").write(msh)
```



The OCCT Kernel – The Flask Example: II

```
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)
```



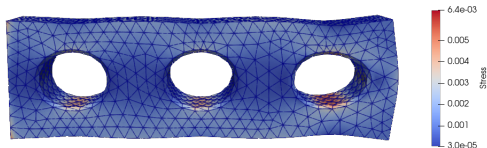
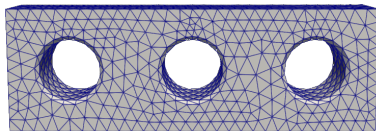
The OCCT Kernel – Linear Elasticity: I

```
1 from firedrake import *
2 from netgen.occ import *
3 box = Box((0,0,0), (3,0.6,1))
4 cyl = sum( [Cylinder((0.5+i,0,0.5), Y, 0.25,0.8) for i
              in range(3)] )
5 box.faces.name,cyl.faces.name="outer","cyl"
6 geo = box-cyl
7 cylboxedges = geo.faces["outer"].edges * geo.faces["
    cyl"].edges
8 cylboxedges.name = "cylbox"
9 geo = geo.MakeChamfer(cylboxedges, 0.03)
10 geo.faces.Min(X).name = "fix"
11 geo.faces.Max(X).name = "force"
12 ngmesh = OCCGeometry(geo).GenerateMesh(maxh=0.1)
13 msh = Mesh(ngmesh)
14 File("VTK/OCCSolidMesh.pvd").write(msh)
```

The OCCT Kernel – Linear Elasticity: II

```
1 V = VectorFunctionSpace(msh, "CG", 3)
2 W = FunctionSpace(msh, "CG", 3)
3 u,v = TrialFunction(V), TestFunction(V)
4 sol = Function(V)
5 lbFix = ngmesh.GetBCIDs("fix")
6 lbF = ngmesh.GetBCIDs("force")[0]
7 bc = DirichletBC(V, 0, lbFix)
8 f = as_vector([1e-3, 0, 0])
9 E, nu, Id = 210.,0.3, Identity(3)
10 mu = Constant((0.5/(1+nu))*E)
11 lambda_ = Constant(E*(nu/((1+nu)*(1-2*nu))))
12 epsilon = lambda u: 0.5*(grad(u) + grad(u).T)
13 sigma = lambda u: lambda_*div(u)*Id + 2*mu*epsilon(u)
14 a = inner(sigma(u), epsilon(v))*dx
15 L = inner(f, v)*ds(lbF)
```


The OCCT Kernel – Linear Elasticity: III



The OCCT Kernel – Multigrid: I

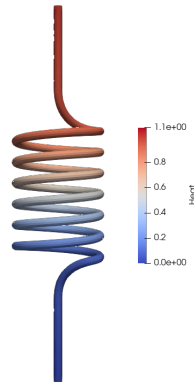
```
1 from firedrake import *
2 from netgen.occ import *
3 cyl = Cylinder((0,0,0), Z, r=0.01, h=0.03).faces[0]
4 heli = Edge(Segment((0,0), (12*pi, 0.03)), cyl)
5 ps,vs = heli.start, heli.start_tangent
6 pe,ve = heli.end, heli.end_tangent
7 e1 = Segment((0,0,-0.03), (0,0,-0.01))
8 c1 = BezierCurve( [(0,0,-0.01), (0,0,0), ps-vs, ps])
9 e2 = Segment((0,0,0.04), (0,0,0.06))
10 c2 = BezierCurve( [pe, pe+ve, (0,0,0.03), (0,0,0.04)])
11 spiral = Wire([e1, c1, heli, c2, e2])
12 circ = Face(Wire([Circle((0,0,-0.03), Z, 0.001)]))
13 coil = Pipe(spiral, circ)
14 coil.faces.maxh, coil.faces.name = 0.1, "coilbnd"
15 coil.faces.Max(Z).name, coil.faces.Min(Z).name="I", "0"
16 ngmsh = OCCGeometry(coil).GenerateMesh(maxh=0.3)
```

The OCCT Kernel – Multigrid: II

```

1 msh = Mesh(ngmsh)
2 hierarchy = MeshHierarchy(msh, 2)
3 V = FunctionSpace(hierarchy[-1], "CG", 1)
4 u,v = TrialFunction(V), TestFunction(V)
5 a,L = dot(grad(u), grad(v))*dx, 1*v*dx
6 bcsI=DirichletBC(V,1,ngmsh.GetBCIDs("I"))
7 bcs0=DirichletBC(V,0.,ngmsh.GetBCIDs("O"))
8 u = Function(V)
9 parameters = {"ksp_type": "preonly", "
    pc_type": "mg",
10    "pc_mg_type": "full", "
    mg_levels_ksp_type": "chebyshev",
11    "mg_levels_ksp_max_it": 2, "
    mg_levels_pc_type": "jacobi"}
12 solve(a==L,u,bcs=[bcsI,bcs0],
    solver_parameters=par)

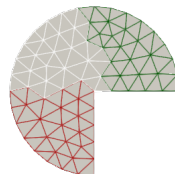
```



Mesh Refinement – Adaptive Mesh Refinement: I

```

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]
```

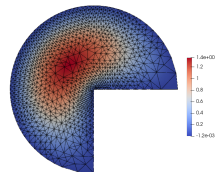
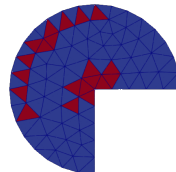


Mesh Refinement – Adaptive Mesh Refinement: II

```

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

```



- ▶ Mesh **hierarchy awareness**, using Firedrake `HierarchyBase` class.
- ▶ Make the implementation works in **complex** arithmetic.
- ▶ 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.