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

University of Oxford

Mathematical Institute

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.

GitHub

Why NetGen?

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

Mathematical Institute

Install NetGen using Firedrake scripts

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

PETSc

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

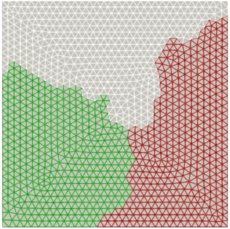
NetGen

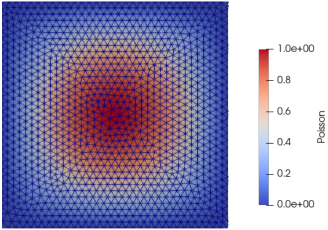
If installing from scratch, use the NetGen fork from the Firdrake project GitHub.

Getting Started – Unstructed Mesh


```python
from firedrake import *
from mpi4py import MPI
from netgen.geom2d import SplineGeometry
import netgen
comm = MPI.COMM_WORLD
if comm.rank == 0:
    geo = SplineGeometry()
    geo.AddRectangle((0, 0), (pi, pi), bc="rect")
    ngmesh = geo.GenerateMesh(maxh=0.1)
    labels = ngmesh.GetBCIDs("rect")
else:
    ngmesh = netgen.libngpy._meshing.Mesh(2)
    labels = None
labels = comm.bcast(labels, root=0)
msh = Mesh(ngmesh)
File("VTK/Poisson2DMesh.pvd").write(msh)
```

Getting Started – Unstructed Mesh





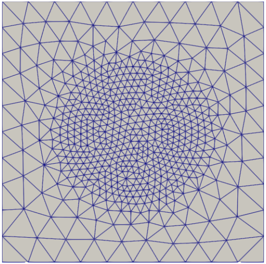
Getting Started – CSG 2D

UNIVERSITY OF OXFORD
Mathematical Institute

```python
from firedrake import *
from netgen.geom2d import SplineGeometry
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, 0.05)
ngmesh = geo.GenerateMesh(maxh=0.2)
msh = Mesh(ngmesh)
File("VTK/CSG2DMesh.pvd").write(msh)
```

Getting Started – CSG 2D

Mathematical Institute



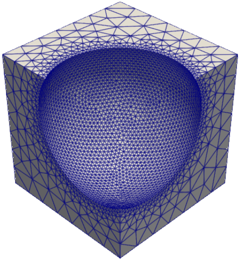
Material Identifiers

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

Getting Started – CSG 3D

UNIVERSITY OF OXFORD
Mathematical Institute

```python
from firedrake import *
from netgen.csg import *
geo = CSGeometry()
brick = OrthoBrick(Pnt(-1,-1,-1),Pnt
    (1,1,1))
sphere = Sphere(Pnt(0.5,0.5,0.5),1)
sphere.maxh(0.05)
geo.Add(brick-sphere)
ngmesh = geo.GenerateMesh(maxh=0.3)
msh = Mesh(ngmesh)
File("VTK/CSG3DMesh.pvd").write(msh)
```



The Open Cascade Technology Kernel

UNIVERSITY OF OXFORD
Mathematical Institute

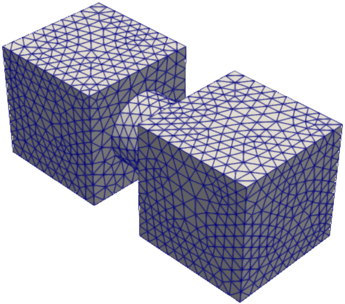
- Basic OCCT objects can be used in NetGen such as: `Box`, `Cylinder`, `Point`, `Segment` and `ArcOfCircle`.
- The `fuse`, `cut` and `common` operations between OCCT objects have been wrapped in NetGen.
- Transformation operations such as `Move` and `Rotate` have also been wrapped into NetGen.

The Open Cascade Technology Kernel

UNIVERSITY OF OXFORD
Mathematical Institute

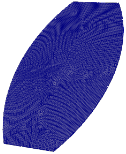
```python
from firedrake import *
from netgen.occ import *
box = Box(Pnt(0,0,0), Pnt(1,1,1))
cyl = Cylinder(Pnt(1,0.5,0.5), X, r=0.3, h=0.5)
solid1 = box + cyl
solid2 = solid1.Rotate(Axis((0,0,0),Y),180).Move
    ((2.5,0.,1.))
solid = solid2 + solid1
geo = OCCGeometry(solid)
ngmesh = geo.GenerateMesh(maxh=0.1)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

The Open Cascade Technology Kernel



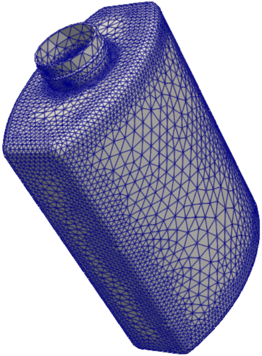
The OCCT Kernel – The Flask Example: I

```python
from firedrake import *
from netgen.occ import *
H, W, T  = 70.,50.,30.
P1,P2=Pnt(-W/2,0,0),Pnt(-W/2,-T/4,0)
P3,P4=Pnt(0,-T/2.,0),Pnt(W/2.,-T/ 4.,0)
P5=Pnt(W/2.,0, 0)
S1=Segment(P1, P2)
A=ArcOfCircle(P2, P3, P4)
S2=Segment(P4, P5)
wire=Wire([S1, A, S2])
mwire=wire.Mirror(Axis((0,0,0),X))
wire=Wire([wire,mwire])
base=Face(wire).Extrude(0.3*Z)
msh=Mesh(OCCGeometry(base).GenerateMesh(
    maxh=0.5))
File("VTK/OCCBottleBase.pvd").write(msh)
```



The OCCT Kernel – The Flask Example: II


```
f = Face(wire)
f.bc("bottom")
body=f.Extrude(H*Z)
body=body.MakeFillet(body.edges,T/12)
neckAx=Axes(body.faces.Max(Z).center,Z)
NeckR,NeckH=T/4,H/10
neck=Cylinder(neckAx,NeckR,NeckH)
body=body+neck
fmax=body.faces.Max(Z)
thickbody=body.MakeThickSolid([fmax],-T/50,1.e-3)
msh=Mesh(OCCGeometry(thickbody).GenerateMesh(maxh=3.))
File("VTK/OCCBottleBody.pvd").write(msh)
```



The OCCT Kernel – Linear Elasticity: I


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

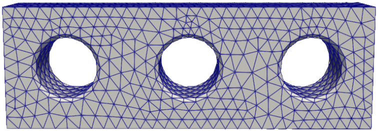
The OCCT Kernel – Linear Elasticity: II

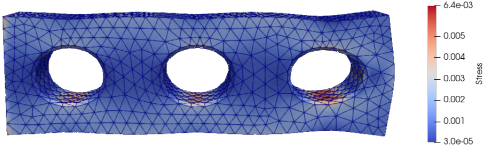
UNIVERSITY OF OXFORD
Mathematical Institute

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

The OCCT Kernel – Linear Elasticity: III

University of Oxford
Mathematical Institute





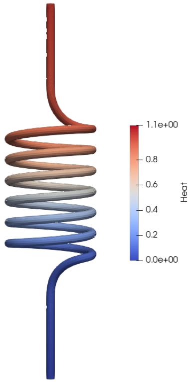
The OCCT Kernel – Multigrid: I

UNIVERSITY OF OXFORD
Mathematical Institute

```python
from firedrake import *
from netgen.occ import *
cyl = Cylinder((0,0,0), Z, r=0.01, h=0.03).faces[0]
heli = Edge(Segment((0,0), (12*pi, 0.03)), cyl)
ps,vs = heli.start, heli.start_tangent
pe,ve = heli.end, heli.end_tangent
e1 = Segment((0,0,-0.03), (0,0,-0.01))
c1 = BezierCurve( [(0,0,-0.01), (0,0,0), ps-vs, ps])
e2 = Segment((0,0,0.04), (0,0,0.06))
c2 = BezierCurve( [pe, pe+ve, (0,0,0.03), (0,0,0.04)])
spiral = Wire([e1, c1, heli, c2, e2])
circ = Face(Wire([Circle((0,0,-0.03), Z, 0.001)]))
coil = Pipe(spiral, circ)
coil.faces.maxh, coil.faces.name = 0.1, "coilbnd"
coil.faces.Max(Z).name,coil.faces.Min(Z).name="I","O"
ngmsh = OCCGeometry(coil).GenerateMesh(maxh=0.3)
```

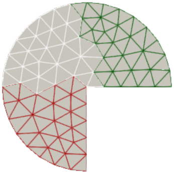
The OCCT Kernel – Multigrid: II


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



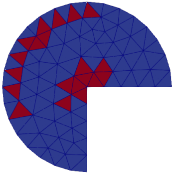
Mesh Refinement – Adaptive Mesh Refinement: I

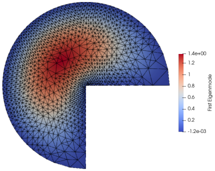
```
tolerance = 1e-16
max_iterations = 15
exact = 3.375610652693620492628**2
geo=SplineGeometry()
pnts=[(0, 0), (1, 0), (1, 1),(0, 1),
     (-1, 1), (-1, 0),(-1, -1), (0, -1)]
P = [geo.AppendPoint(*pnt) for pnt in pnts]
L,B=["line","spline3"],["line","curve"]
curves = [[[L[0],P[0],P[1]],B[0]],
          [[L[1],P[1],P[2],P[3]],B[1]],
          [[L[1],P[3],P[4],P[5]],B[1]],
          [[L[1],P[5],P[6],P[7]],B[1]],
          [[L[0],P[7],P[0]],B[0]]]
[geo.Append(c, bc=bc) for c, bc in curves]
```



Mesh Refinement – Adaptive Mesh Refinement: II

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





Conclusion

Mathematical Institute

- 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.