Netgen Meets Firedrake

University of Oxford
Mathematical Institute

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

NETGEN is an advancing front 2D/3D-mesh generator, with many interesting features.

- The geometry we intend to mesh can be described by **Constructive Solid Geometry** (CSG), in particular we can use **Opencascade** to describe our geometry.
- It is able to construct isoparametric meshes, which conform to the geometry.

Joachim Scöberl

ngsPETSc – Firedrake

UNIVERSITY OF OXFORD
Mathematical Institute

ngsPETSc provides new capabilities to **Firedrake** such as:

- Access to all Netgen generated linear meshes and high order meshes.
- Splits for macro elements, such as Alfeld splits and Powell-Sabin splits (even on curved geometries).
- Adaptive mesh refinement capabilities, that conform to the geometry.
- High order mesh hierarchies for multigrid solvers.
- Polygonal discontinuous Galerkin support.

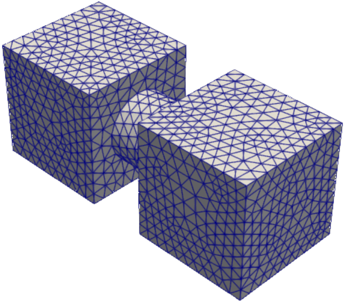
The Open Cascade Technology Kernel

- 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.

Opencascade via NETGEN: 3D Geometries

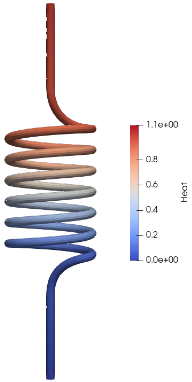

```python
from firedrake import *
from netgen.occ import *
box = Box(Pnt(0,0,0), Pnt(1,1,1))
cyl = Cylinder(Pnt(1,0.5,0.5), X, r=0.3, h=0.5)
solid1 = box + cyl
solid2 = solid1.Rotate(Axis((0,0,0),Y),180).Move
    ((2.5,0.,1.))
solid = solid2 + solid1
geo = OCCGeometry(solid)
ngmesh = geo.GenerateMesh(maxh=0.1)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

Opencascade via NETGEN: 3D Geometries



Linear Refinement Multigrid

```
msh = Mesh(Mesh(ngmsh).curve_field(3))
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)
```



Geometric Conforming Multigrid

University of Oxford Mathematical Institute

ngsPETSc allows us to create a hierarchy of curved meshes for multigrid solvers.

```python
mesh = Mesh(ngmesh)
nh = MeshHierarchy(mesh, 2, netgen_flags={"degree":
    [1, 2, 3]})
mesh = nh[-1]
```

Geometric Conforming Multigrid 3D

University of Oxford
Mathematical Institute

3D Multigrid

The same capabilities are available in 3D, if you have the latest version of Netgen and *DMPlexGetRedundantDM* exposed in your **petesc4py**.

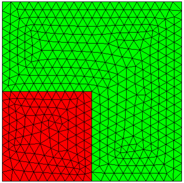
```
pip install --upgrade --pre netgen-mesher
https://gitlab.com/UZerbinati1/petsc.git fork/uz/petsc4pyplex
```

Mesh Labels

UNIVERSITY OF OXFORD
Mathematical Institute

ngsPETSc now provide better mesh labeling capabilities.

```
wp = WorkPlane()
inner = wp.Rectangle(1,1).Face()
inner.name = "inner"
outer = wp.Rectangle(2,2).Face()
outer.name = "outer"
outer = outer - inner
shape = Glue([inner, outer])
shape.edges.name = "rect"
geo = OCCGeometry(shape, dim=2)
```

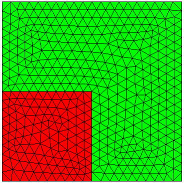


Mesh Labels

UNIVERSITY OF OXFORD
Mathematical Institute

ngsPETSc now provide better mesh labeling capabilities.

```
wp = WorkPlane()
inner = wp.Rectangle(1,1).Face()
inner.name = "inner"
outer = wp.Rectangle(2,2).Face()
outer.name = "outer"
outer = outer - inner
shape = Glue([inner, outer])
shape.edges.name = "rect"
geo = OCCGeometry(shape, dim=2)
```

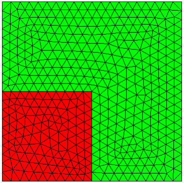


```
assert(abs(assemble(u*dx(mesh.labels[(2, "inner")]))
    -1) < 1e-10)
assert(abs(assemble(u*dx(mesh.labels[(2, "outer")]))
    -3) < 1e-10)
```

Mesh Labels

ngsPETSc now provide better mesh labeling capabilities.

```
wp = WorkPlane()
inner = wp.Rectangle(1,1).Face()
inner.name = "inner"
outer = wp.Rectangle(2,2).Face()
outer.name = "outer"
outer = outer - inner
shape = Glue([inner, outer])
shape.edges.name = "rect"
geo = OCCGeometry(shape, dim=2)
```

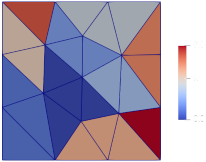


```
V = FunctionSpace(mesh, "DG", 1)
bc = DirichletBC(V, Constant(1), mesh.labels[(1, "inner")])
```

Polygonal Discontinuous Galerkin

Mathematical Institute

```
Rectangle = WorkPlane().Rectangle(1,1).Face()
geo = OCCGeometry(Rectangle, dim=2)
ngmesh = geo.GenerateMesh(maxh=0.3)
mesh = Mesh(ngmesh)
polymesh = dumbAggregation(mesh)
```



```
aDG = inner(grad(u),grad(v))* dx
aDG+=inner((alpha*order**2/(h("+")+h("-")))*jump(u),jump(v))*dS
aDG+=inner(-mean_dudn,jump(v))*dS-inner(mean_dvdn,jump(u))*dS
aDG+=alpha*order**2/h*inner(u,v)*ds
aDG+=-inner(dot(n,grad(u)),v)*ds-inner(dot(n,grad(v)),u)*ds
```

Polygonal Discontinuous Galerkin

```python
f = Function(V).interpolate(exp(x)*sin(y))
L = alpha*order**2/h*inner(f,v)*ds - inner(dot(n,grad(v)),f)*ds

agg_embd = AggregationEmbedding(V, mesh, polymesh)
appctx = {"trefftz_embedding": agg_embd}
uDG = Function(V)
solve(aDG == L, uDG, solver_parameters={"ksp_type":"python","ksp_python_type":trefftz_ksp},appctx=appctx)
```