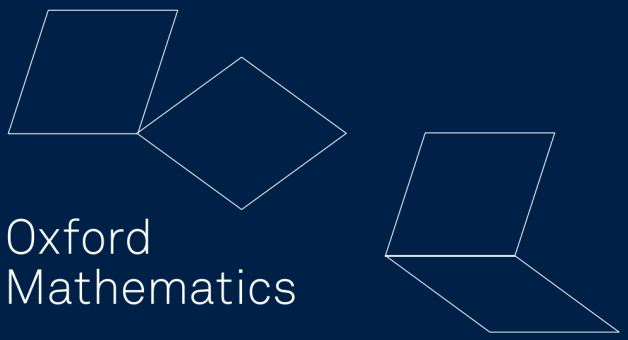
Firedrake-NETGEN Integration: New Features

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, Great Missenden, 13th of September 2023



Oxford Mathematics

Mathematical Institute

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.

Why NETGEN?

UNIVERSITY OF OXFORD
Mathematical Institute

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 **adaptive mesh refinement** (also anisotropic mesh refinement).
- Supports **high-order** meshes for curved geometries.

Getting Started – Installing NETGEN

UNIVERSITY OF OXFORD
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 you are interested in **anisotropic mesh refinement**, please install, `https://github.com/UZerbinati/netgen.git`

Firedrake-NETGEN Integration: Previous Version

UNIVERSITY OF OXFORD
Mathematical Institute

Some of the old features of the Firedrake-NETGEN integration are:

- Describing the geometry to be meshed using **Open Cascade Technology** geometry kernel.
- Using **NETGEN** as a mesh generator in Firedrake, for **linear** meshes.
- Marking subregions of the geometry for **finer meshing**.
- **Adaptive mesh refinement**, this feature can be accessed using the `refine_marked_elements` method.

ngsPETSc – A new component of NGSolve

UNIVERSITY OF OXFORD
Mathematical Institute

ngsPETSc is a new component of the NGSolve library, which allows to use PETSc as a linear algebra backend for NGSolve. In particular thanks to the **NETGEN-DMPlex** interface it is possible NETGEN mesh in Firedrake.

- Less code to maintain on the Firedrake side and additional features for the **NETGEN-DMPlex** interface.
- No need to install the full NGSolve library, for most of the features here presented only NETGEN will suffice.

Firedrake-NETGEN Integration: New Features

UNIVERSITY OF OXFORD
Mathematical Institute

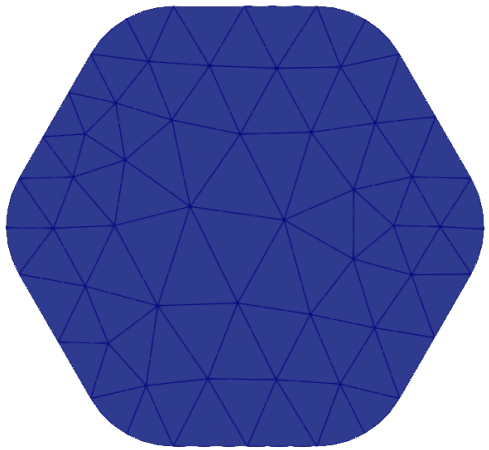
Some of the new features of the Firedrake-NETGEN integration are:

- Using **NETGEN** as a mesh generator in Firedrake, for **high-order** meshes.
- Support for **anisotropic** mesh refinement, using NETGEN `ZRefinement` and `HPRefinement` methods.
- Using **PETSc Transformation** for **Alfeld** and **Powell-Sabin** splits and **quadrilateral** meshes.

Open Cascade – High Order Meshes

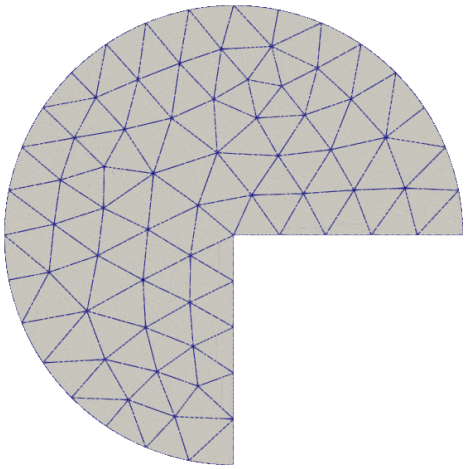
UNIVERSITY OF OXFORD
Mathematical Institute

```
wp = WorkPlane()
if comm.rank == 0:
    for i in range(6):
        wp.Line(0.6).Arc(0.4, 60)
    shape = wp.Face()
    ngmesh = OCCGeometry(shape,dim=2)
    .GenerateMesh(maxh=1.)
else:
    ngmesh = netgen.libngpy._meshing.
    Mesh(2)
import firedrake as fd
mesh = fd.Mesh(fd.Mesh(ngmesh).
    curve_field(3))
fd.File("VTK/wp.pvd").write(mesh)
```



Open Cascade – High Order Meshes

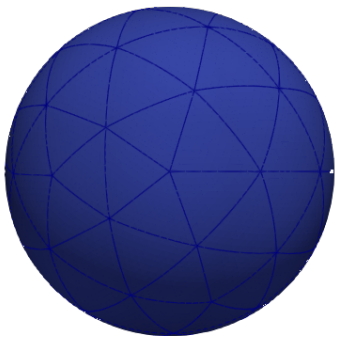
```
wp = WorkPlane()
if comm.rank == 0:
    for i in range(6):
        wp.Line(0.6).Arc(0.4, 60)
    shape = wp.Face()
    ngmesh = OCCGeometry(shape,dim=2)
    .GenerateMesh(maxh=1.)
else:
    ngmesh = netgen.libngpy._meshing.
    Mesh(2)
import firedrake as fd
mesh = fd.Mesh(fd.Mesh(ngmesh).
    curve_field(3))
fd.File("VTK/wp.pvd").write(mesh)
```



Open Cascade 3D – High Order Meshes

UNIVERSITY OF OXFORD
Mathematical Institute

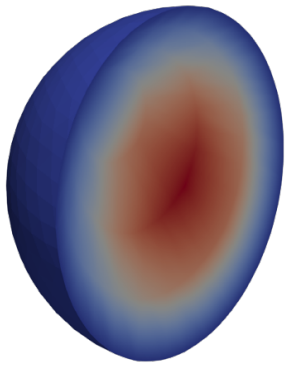
```
comm = MPI.COMM_WORLD
if comm.rank == 0:
    shape = Sphere(Pnt(0,0,0), 1)
    ngmesh = OCCGeometry(shape,dim=3)
    .GenerateMesh(maxh=1.)
else:
    ngmesh = netgen.libngpy._meshing.
    Mesh(3)
import firedrake as fd
mesh = fd.Mesh(fd.Mesh(ngmesh).
    curve_field(3))
fd.File("VTK/sphere.pvd").write(u)
```



Open Cascade 3D – High Order Meshes

Mathematical Institute

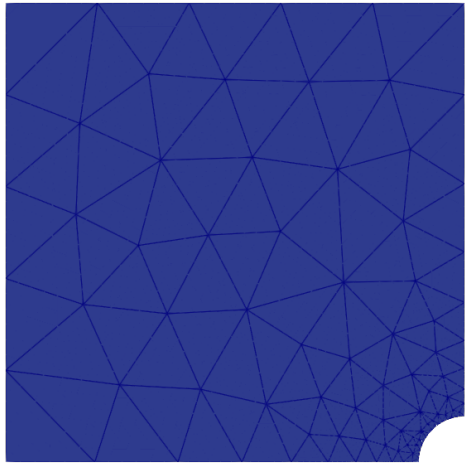
```
comm = MPI.COMM_WORLD
if comm.rank == 0:
    shape = Sphere(Pnt(0,0,0), 1)
    ngmesh = OCCGeometry(shape,dim=3)
    .GenerateMesh(maxh=1.)
else:
    ngmesh = netgen.libngpy._meshing.
    Mesh(3)
import firedrake as fd
mesh = fd.Mesh(fd.Mesh(ngmesh).
    curve_field(3))
fd.File("VTK/sphere.pvd").write(u)
```



Constructive Solid Geometry – High Order Meshes

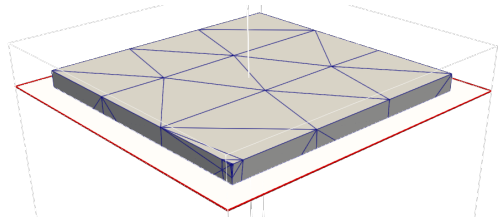
UNIVERSITY OF OXFORD
Mathematical Institute

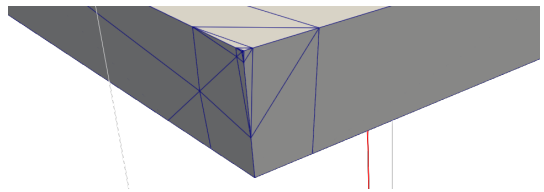
```
1  if comm.rank == 0:
2      geo = CSG2d()
3      circle = Circle(center=(1,1),
       radius=0.1, bc="curve").Maxh(0.01)
4      rect = Rectangle(pmin=(0,1), pmax
       =(1,2), bottom="bottom", left="
       left", top="top", right="right")
5      geo.Add(rect-circle)
6      ngmesh = geo.GenerateMesh(maxh
       =0.2)
7  else:
8      ngmesh = netgen.libngpy._meshing.
       Mesh(2)
9  import firedrake as fd
10 mesh = fd.Mesh(fd.Mesh(ngmesh).
       curve_field(3))
```



Anisotropic Mesh Refinement – Singular Vertex

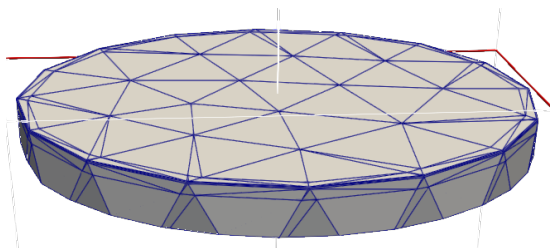
```
1  if comm.rank == 0:
2      cube = Box((0,0,0), (1,1,1))
3      cube.vertices.Max(X+Y+Z).hpref=3
4      cube.vertices.Max(X+Y-Z).hpref=3
5      geo = OCCGeometry(cube)
6      mesh = ngs.Mesh(geo.GenerateMesh(
       maxh=0.4))
7      mesh.RefineHP(2)
8      ngmesh = mesh.ngmesh
9  else:
10     ngmesh = netgen.libngpy._meshing.
       Mesh(3)
11 import firedrake as fd
12 mesh = fd.Mesh(ngmesh,netgen_flags={"
       purify_to_tets": True})
```





Anisotropic Mesh Refinement – Singular Edge

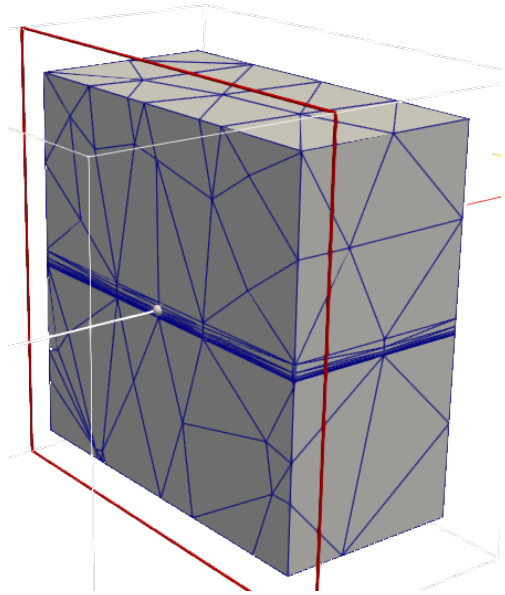
```
comm = MPI.COMM_WORLD
if comm.rank == 0:
    cyl= Cylinder((0,0,0), Z, r=1, h=3)
    cyl.edges.Max(Z).hpref=1
    geo = OCCGeometry(cyl)
    mesh = ngs.Mesh(geo.GenerateMesh(maxh=0.4))
    mesh.RefineHP(2)
    ngmesh = mesh.ngmesh
else:
    ngmesh = netgen.libngpy._meshing.Mesh(3)
import firedrake as fd
mesh = fd.Mesh(ngmesh,netgen_flags={"purify_to_tets": True})
```



Anisotropic Mesh Refinement – Z Refinement

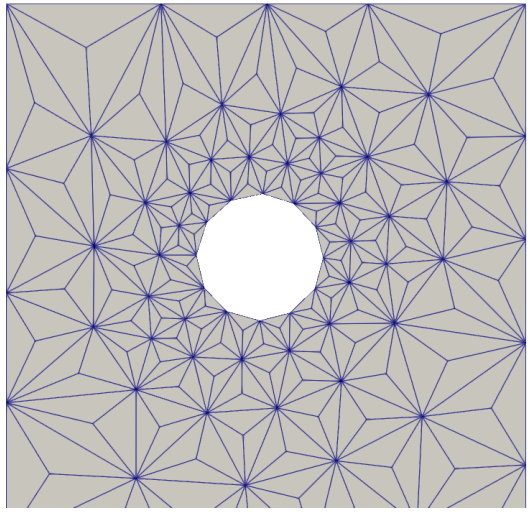
Mathematical Institute

```
1       geo = CSGeometry()
2       box = OrthoBrick(Pnt(0,0,0),Pnt
        (1,1,1))
3       top = Plane(Pnt(0,0,0.52),Vec
        (0,0,1))
4       bot = Plane(Pnt(0,0,0.48),Vec
        (0,0,-1))
5       plate = box * top * bot
6       geo.Add((box-top).mat("air"))
7       geo.Add(plate.mat("plate"))
8       geo.Add((box-bot).mat("air"))
9       slices = [2**(-i) for i in
        reversed(range(1,6))]
10      geo.CloseSurfaces(bot,top,slices)
11      ngmesh = geo.GenerateMesh(maxh
        =0.3)
12      ZRefinement(ngmesh,geo)
```



PETSc Transformation – Alfeld Split

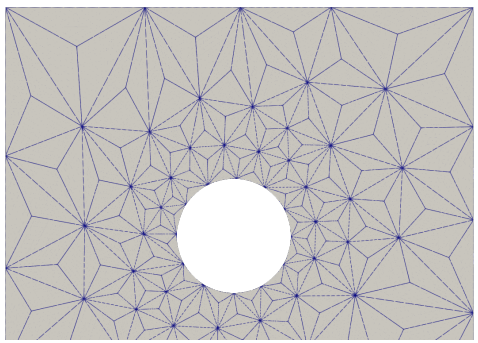
```
1  if comm.rank == 0:
2      shape = Rectangle(2,0.41).Circle
       (0.2,0.2,0.05).Reverse().Face()
3      ngmesh = OCCGeometry(shape,dim=2)
       .GenerateMesh(maxh=0.5)
4  else:
5      ngmesh = netgen.libngpy._meshing.
       Mesh(2)
6  import firedrake as fd
7  transform = PETSc.DMPlexTransform().
       create(comm=PETSc.COMM_WORLD)
8  transform.setType(PETSc.
       DMPlexTransformType.REFINEALFELD)
9  mesh = fd.Mesh(ngmesh,netgen_flags={"
       transform":transform})
10 mesh = fd.Mesh(mesh.curve_field(3))
```



NETGEN Transformation - Curved Alfeld Split

Mathematical Institute

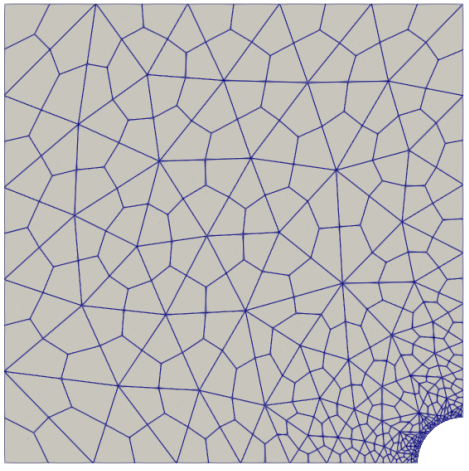
```
1  if comm.rank == 0:
2      shape = Rectangle(2,0.41).Circle
       (0.2,0.2,0.05).Reverse().Face()
3      ngmesh = OCCGeometry(shape,dim=2)
       .GenerateMesh(maxh=0.5)
4      mesh = ngs.Mesh(ngmesh)
5      mesh.SplitElements_Alfeld()
6      ngmesh = mesh.ngmesh
7  else:
8      ngmesh = netgen.libngpy._meshing.
       Mesh(2)
9  import firedrake as fd
10 mesh = fd.Mesh(ngmesh)
11 mesh = fd.Mesh(mesh.curve_field(3))
```



PETSc Transformation – Quadrilateral Split

UNIVERSITY OF OXFORD
Mathematical Institute

```
1  if comm.rank == 0:
2      geo = CSG2d()
3      circle = Circle(center=(1,1),
       radius=0.1, bc="curve").Maxh(0.01)
4      rect = Rectangle(pmin=(0,1), pmax
       =(1,2))
5      geo.Add(rect-circle)
6      ngmesh = geo.GenerateMesh(maxh
       =0.2)
7  else:
8      ngmesh = netgen.libngpy._meshing.
       Mesh(2)
9  import firedrake as fd
10 mesh = fd.Mesh(ngmesh,netgen_flags={"
       quad": True})
```



Future work

- Improve support for **quadrilateral elements**, and consider **hexahedral elements**.
- Mesh **hierarchy awareness**, using Firedrake `HierarchyBase` class.
- Support **"snap back"**, to OCCT geometry, thanks to PETSc OCC awareness.
- Support for MFEM **GLVIS** mesh and solution live display, thanks to PETSc-GLVIS interface.

Thank You for your attention !

`https://github.com/UZerbinati/Firedrake23`