UNIVERSITY OF OXFORD

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, Oxford, 18th of September 2024

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

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

UNIVERSITY OF OXFORD
Mathematical Institute

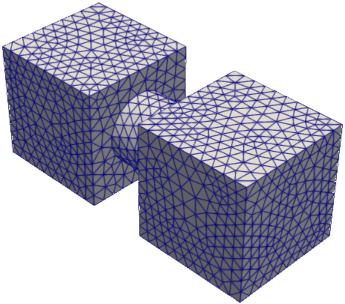
- Basic OCCT objects can be used in NetGen such as: `Box`, `Cylinder`, `Point`, `Segment` and `ArcOfCircle`.
- The `fuse`, `cut` and `common` operations between OCCT objects have been wrapped in NetGen.
- Transformation operations such as `Move` and `Rotate` have also been wrapped into NetGen.

Opencascade via NETGEN: 3D Geometries

UNIVERSITY OF OXFORD
Mathematical Institute

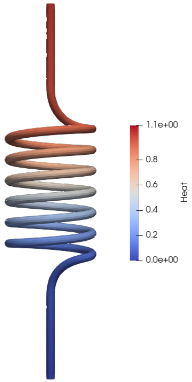
```python
from firedrake import *
from netgen.occ import *
box = Box(Pnt(0,0,0), Pnt(1,1,1))
cyl = Cylinder(Pnt(1,0.5,0.5), X, r=0.3, h=0.5)
solid1 = box + cyl
solid2 = solid1.Rotate(Axis((0,0,0),Y),180).Move((2.5,0.,1.))
solid = solid2 + solid1
geo = OCCGeometry(solid)
ngmesh = geo.GenerateMesh(maxh=0.1)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

Opencascade via NETGEN: 3D Geometries



Linear Refinement Multigrid

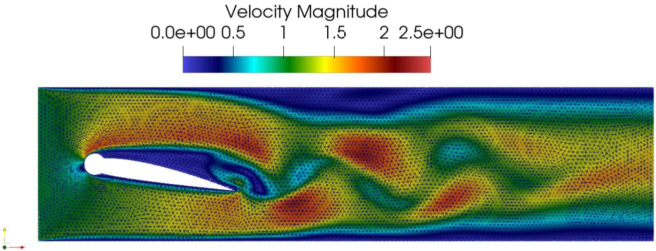

```
msh = Mesh(Mesh(ngmsh).curve_field(3))
hierarchy = MeshHierarchy(msh, 2)
V = FunctionSpace(hierarchy[-1], "CG", 1)
u,v  = TrialFunction(V), TestFunction(V)
a,L = dot(grad(u), grad(v))*dx, 1*v*dx
bcsI=DirichletBC(V,1,ngmsh.GetBCIDs("I"))
bcsO=DirichletBC(V,0.,ngmsh.GetBCIDs("O"))
u = Function(V)
parameters = {"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

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

```
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**.

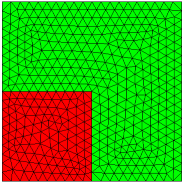
```
pip install --upgrade --pre netgen-mesher
https://gitlab.com/UZerbinati1/petsc.git fork/uz/petsc4pyplex
```

Mesh Labels

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)
```

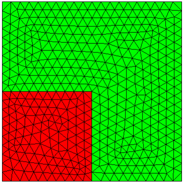


Mesh Labels

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)
```



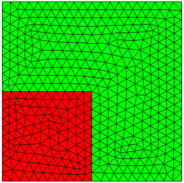
```
assert(abs(assemble(u*dx(mesh.labels[(2, "inner")]))
    -1) < 1e-10)
assert(abs(assemble(u*dx(mesh.labels[(2, "outer")]))
    -3) < 1e-10)
```

Mesh Labels

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)
```



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

Vanilla Embedded Trefftz Framework

Mathematical Institute

$$V_h = \{v_h \in \mathbb{P}^k(\mathcal{T}_h) \ : \ \mathcal{A}v_h = 0\}$$

Paul Stocker

Christoph Lehrenfeld

Vanilla Embedded Trefftz Framework

Mathematical Institute

$$V_h = \{v_h \in \mathbb{P}^k(\mathcal{T}_h) \ : \ \mathcal{A}v_h = 0\}$$

$$\mathcal{A} = \begin{bmatrix} — & u_1 & — \\ & \vdots & \\ — & u_n & — \end{bmatrix} \left[\begin{array}{c|c} \Sigma & 0 \\ \hline 0 & \varepsilon \end{array}\right] \begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix}$$

Paul Stocker

Christoph Lehrenfeld

Vanilla Embedded Trefftz Framework

$$V_h = \{v_h \in \mathbb{P}^k(\mathcal{T}_h) \ : \ \mathcal{A}v_h = 0\}$$

$$\mathcal{A} = \left[\begin{array}{c} U^* \\ \hline U_0 \end{array}\right] \left[\begin{array}{c|c} \Sigma & 0 \\ \hline 0 & \varepsilon \end{array}\right] \left[\begin{array}{c|c} V_* & V_0 \end{array}\right]$$

Paul Stocker

Christoph Lehrenfeld

Vanilla Embedded Trefftz Framework

University of Oxford
Mathematical Institute

$$V_h = \{v_h \in \mathbb{P}^k(\mathcal{T}_h) \ : \ \mathcal{A}v_h = 0\}$$

$$\mathcal{A} = \left[\begin{array}{c} U^* \\ \hline U_0 \end{array}\right] \left[\begin{array}{c|c} \Sigma & 0 \\ \hline 0 & \varepsilon \end{array}\right] \left[\begin{array}{c|c} V_* & V_0 \end{array}\right]$$

$$V_0^T K V_0 \vec{U} = V_0^T \vec{F}$$

Paul Stocker

Christoph Lehrenfeld

Vanilla Embedded Trefftz Framework

UNIVERSITY OF OXFORD
Mathematical Institute

$$V_h = \{v_h \in \mathbb{P}^k(\mathcal{T}_h) \; : \; \mathcal{A} v_h = 0\}$$

$$\mathcal{A} = \left[\begin{array}{c} U^* \\ \hline U_0 \end{array}\right] \left[\begin{array}{c|c} \Sigma & 0 \\ \hline 0 & \varepsilon \end{array}\right] \left[\begin{array}{c|c} V_* & V_0 \end{array}\right]$$

$$V_0^T K V_0 \vec{U} = V_0^T \vec{F}$$

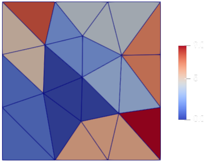
Lehrenfeld C, Stocker P. Embedded Trefftz discontinuous Galerkin methods. Int J Numer Methods Eng. 2023; 124(17): 3637-3661. doi: 10.1002/nme.7258

Paul Stocker

Christoph Lehrenfeld

Polygonal Discontinuous Galerkin


```
geo = OCCGeometry(Rectangle, dim=2)
ngmesh = geo.GenerateMesh(maxh=0.3)
mesh = Mesh(ngmesh)
polymesh = dumb_aggregation(mesh)
```

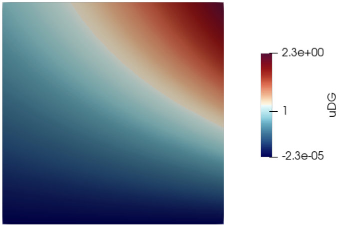


```
aDG = inner(grad(u),grad(v))* dx
aDG += inner((alpha*order**2/(h("+")+h("-")))*jump(u), jump(v))*dS
aDG += inner(-mean_dudn,jump(v))*dS-inner(mean_dvdn, jump(u))*dS
aDG += alpha*order**2/h*inner(u,v)*ds
aDG += -inner(dot(n,grad(u)),v)*ds -inner(dot(n,grad(v)),u)*ds
```

Polygonal Discontinuous Galerkin

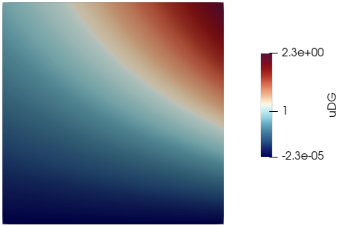
University of Oxford
Mathematical Institute

```
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":"firedrake.trefftz.trefftz_ksp"},appctx=appctx)
```



Polygonal Discontinuous Galerkin


```python
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":"firedrake.trefftz.trefftz_ksp"},appctx=appctx)
```



Post David and Patrick comment

The implementation is now independent of ngsPETSc and can be found in **PR: #3775**.