ngsPETSc: NETGEN meets PETSc

P. E. Farrell*, S. Zampini†, U. Zerbinati*

* *Mathematical Institute*
University of Oxford

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

28 International Conference on Domain Decomposition Methods, 29th January 2024, KAUST

Oxford Mathematics

University of Oxford Mathematical Institute

Overview

- Reynolds and pressure robust Hood-Taylor discretisation with high-order mesh.
- Reynolds robust geometric multigrid on curved meshes.
- Easy implementation:

All codes are available on Github: `https://github.com/UZerbinati/DD28`

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

PETSc

PETSc stands for Portable, Extensible Toolkit for Scientific Computation, is a library for the scalable (parallel) solution of scientific applications modeled by partial differential equations (PDEs).

- **PETSc KSP** provides access to extremly efficent Krylov solvers.
- **PETSc SNES** provides access to extremly efficent non-linear solvers, with line-searching and trust region capabilities.

Stefano Zampini

ngsPETSc – NETGEN/NGSolve

Mathematical Institute

ngsPETSc is an interface between NETGEN/NGSolve and **PETSc**. In particular, **ngsPETSc** provides new capabilities to **NETGEN/NGSolve** such as:

- Access to all linear solver capabilities of **KSP**.
- Access to all preconditioning capabilities of **PC**.
- Access to all non-linear solver capabilities of **SNES**.
- Access to all mesh refinement capabilities of **DMPLEX**.

PETSc DMPlex

PETSc DMPlex handles unstructured grids using the generic **PETSc** interface for hierarchy and multi-physics.

- **PETSc DMPlex** provides a wide variety of primitive mesh operations such as: *meet*, *closure*, *cone*, etc $+$
- **PETSc DMPlex** provides a wide variety of mesh refinement operations such as: *uniform refinement*, *Alfeld refinement*, *box refinement*, etc

Matthew G. Knepley

Firedrake is an automated system for the solution of partial differential equations using the finite element method (FEM).

- Variational formulation can be easily defined using the **UFL** language.
- Wide class of finite elements are available, including $H(div)$, $H(curl)$, H^1 and H^2.
- Provides access to **PETSc** linear solvers and non-linear solvers.

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.

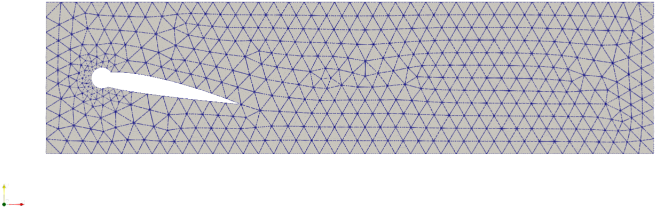
Examples – Opencascade via NETGEN

Mathematical Institute

```
n = 140
profile = "2412"
xNACA = naca(profile, n, False, False)[0]
yNACA = naca(profile, n, False, False)[1]
pnts = [Pnt(xNACA[i], yNACA[i], 0) for i in range(len(
    xNACA))]
spline = SplineApproximation(pnts)
airfoil = Face(Wire(spline)).Move((0.3,0.5,0)).Rotate(
    Axis((0.3,0.5,0), Z), -10)
circle = Circle(Pnt(0.37,0.5),0.07).Face()
shape = (Rectangle(4, 1).Face()-airfoil-circle)
shape.edges.name="wall"
shape.edges.Min(X).name="inlet"
```

Examples – Opencascade via NETGEN


```
shape.edges.Max(X).name="outlet"
geo = OCCGeometry(shape, dim=2)
ngmesh = geo.GenerateMesh(maxh=0.1)
```



Stokes flow – Weak formulation

UNIVERSITY OF OXFORD
Mathematical Institute

Find $(\boldsymbol{u}, p) \in V \times Q$ such that

$$\begin{aligned} \nu \int_\Omega \varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) - \int_\Omega p \nabla \cdot \boldsymbol{v} &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} && \forall \boldsymbol{v} \in V \\ - \int_\Omega q \nabla \cdot \boldsymbol{u} &= 0 && \forall q \in Q \end{aligned}$$

where V and Q are the velocity and pressure spaces respectively, i.e. $V = H_0^1(\Omega)^2$ and $Q = L_0^2(\Omega)$.

Stokes flow – Inf-sup condition

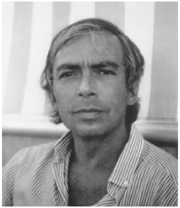
Mathematical Institute

We can also look for a discrete solution, i.e. find $(\boldsymbol{u}_h, p_h) \in V_h \times Q_h$ such that

$$\nu \int_\Omega \varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) - \int_\Omega p \nabla \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}$$
$$- \int_\Omega q \nabla \cdot \boldsymbol{u} = 0$$

for all $(\boldsymbol{v}, q) \in V_h \times Q_h$.

$$\inf_{q \in Q_h} \sup_{\boldsymbol{v} \in V_h} \frac{\int_\Omega q \nabla \cdot \boldsymbol{v}}{\|q\|_{L^2} \|\boldsymbol{v}\|_{H^1}} \geq \beta > 0$$



Franco Brezzi

Stokes flow – UFL


```python
u, p = TrialFunctions(Z)
v, q = TestFunctions(Z)
nu = Constant(1e-3)
a = (nu*inner(eps(u), eps(v)) - p * div(v) - div(u) * q)*dx
L = inner(Constant((0, 0)), v) * dx
```

Stokes flow – Scott–Vogelius element

UNIVERSITY OF OXFORD
Mathematical Institute

We can use the Scott–Vogelius pair, which is a mixed finite element of order k for the velocity and order $k-1$ for the pressure. Such an element is inf-sup stable for $k \geq 2$, under certain assumptions on the mesh. Such pair is **divergence-free**.

When $k=2$ we need **Alfeld splits**.

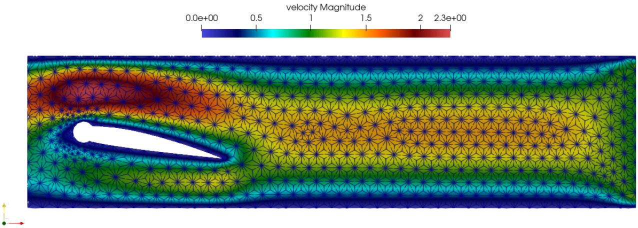
```
geo = OCCGeometry(shape, dim=2)
ngmesh = geo.GenerateMesh(maxh=0.1)
ngmesh.SplitAlfeld()
```

```
V = VectorFunctionSpace(mesh, "CG", 2)
W = FunctionSpace(mesh, "DG", 1)
Z = V * W
```

Ridgway Scott

Stokes flow – Scott–Vogelius

```
bcs = [DirichletBC(Z.sub(0), inflowoutflow,
    labelsInlet),
       DirichletBC(Z.sub(0), zero(2), labelsWall)]
nullspace = MixedVectorSpaceBasis(Z, [Z.sub(0),
    VectorSpaceBasis(constant=True)])
solve(a == L, sol0, bcs=bcs, solver_parameters=
    paramsLU)
```

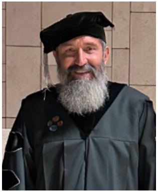


Stokes flow – Hood–Taylor element

Another element pair we will use is the Hood–Taylor pair, which has **no restrictions** on the mesh in two dimensions.

```
V = VectorFunctionSpace(mesh, "CG", 2)
W = FunctionSpace(mesh, "CG", 1)
Z = V * W
```

We lose the point-wise divergence-free property! This is not an issue because the same would happen for Scott-Vogelius on curved meshes.



Daniele Boffi

Fieldsplit Schur preconditioner

UNIVERSITY OF OXFORD
Mathematical Institute

The previous set of equations can be written in matrix form as

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix}$$

We choose as preconditioner the **fieldsplit Schur preconditioner**, i.e.

$$\begin{bmatrix} I & -\hat{A}^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{A}^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B\hat{A}^{-1} & I \end{bmatrix}$$

where S is the **Schur complement**, i.e. $S = -BA^{-1}B^T$.

```
    "fieldsplit_0_ksp_type": "preonly",
    "fieldsplit_0_pc_type": "mg",
    "fieldsplit_1_ksp_type": "preonly",
    "fieldsplit_1_pc_type": "python",
```

Fieldsplit Schur preconditioner – Mass matrix

Thanks to the inf-sup condition we can prove that the Schur complement is spectrally equivalent to the mass matrix, hence we can use as preconditioner:

$$\begin{bmatrix} I & -\hat{A}^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{A}^{-1} & 0 \\ 0 & -\nu\hat{M}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B\hat{A}^{-1} & I \end{bmatrix}$$

where M is the mass matrix.

```
class Mass(AuxiliaryOperatorPC):
    def form(self, pc, test, trial):
        a=1/nu*inner(test, trial)*dx
        bcs = None
        return (a, bcs)
```

Andrew Wathen

Multigrid on curved meshes

UNIVERSITY OF OXFORD
Mathematical Institute

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

```
mesh = Mesh(ngmesh)
from ngsPETSc import NetgenHierarchy
mh = NetgenHierarchy(ngmesh,2, 2)
```

We can then use a multigrid solver to compute $\hat{A}^{-1}$:

```
    "fieldsplit_0_pc_type": "mg",
    "fieldsplit_1_ksp_type": "preonly",
    "fieldsplit_1_pc_type": "python",
    "fieldsplit_1_pc_python_type": "__main__.Mass",
    "fieldsplit_1_aux_pc_type": "bjacobi",
    "fieldsplit_1_aux_sub_pc_type": "icc",
```

Navier-Stokes flow

UNIVERSITY OF OXFORD
Mathematical Institute

A more interesting example is the Navier-Stokes flow, which is a non-linear problem. In particular, we will consider the problem of finding $(\boldsymbol{u}, p) \in H^1(\Omega)^2 \times L^2(\Omega)$ such that

$$\int_\Omega \partial_t \boldsymbol{u} \cdot \boldsymbol{v} + \int_\Omega (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} + \nu \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - \int_\Omega p \nabla \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}$$

$$-\int_\Omega q \nabla \cdot \boldsymbol{u} = 0$$

for all $(\boldsymbol{v}, q) \in H^1(\Omega)^2 \times L^2(\Omega)$.

Navier-Stokes flow – Augmented Lagrangian

We consider an augmented Lagrangian formulation for the discrete problem, i.e. find $(\boldsymbol{u}_h, p_h) \in V_h \times Q_h$ such that

$$\begin{aligned}(\partial_t \boldsymbol{u}, \boldsymbol{v})_0 + (\boldsymbol{u} \cdot \nabla \boldsymbol{u}, \boldsymbol{v})_0 + \nu(\nabla \boldsymbol{u}, \nabla \boldsymbol{v})_0 \\ - (p, \nabla \cdot \boldsymbol{v})_0 + \gamma(\nabla \cdot \boldsymbol{u}, \nabla \cdot \boldsymbol{v})_0 \quad &= (\boldsymbol{f}, \boldsymbol{v})_0\end{aligned}$$

and verifying the weak divergence free constraint $(\nabla \cdot \boldsymbol{u}, q)_0 = 0$, for all $(\boldsymbol{v}, q) \in V_h \times Q_h$.

Navier-Stokes flow – Fieldsplit Schour preconditioner

The linearized version of the Navier-Stokes equations can be written in matrix form as

$$\begin{bmatrix} A + \gamma B^T W B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix}$$

We choose as preconditioner the **fieldsplit Schur preconditioner**, i.e.

$$\begin{bmatrix} I & -\hat{A}_\gamma^{-1} B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{A}_\gamma^{-1} & 0 \\ 0 & \hat{S}_\gamma^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BA_\gamma^{-1} & I \end{bmatrix}$$

$$A_\gamma = A + \gamma B^T W B \qquad S_\gamma = -BA_\gamma^{-1} B^T.$$

In this case, we notice that $S_\gamma \sim -(\nu + \gamma)^{-1} M$, but $\mathcal{S} \not\sim \nu^{-1} M$.

Navier-Stokes flow – Augmented Lagrangian

UNIVERSITY OF OXFORD
Mathematical Institute

- The augmented Lagrangian term helps enforce the divergence-free constraint, and makes the scheme pressure robust.
- We can use as preconditioner

$$\begin{bmatrix} I & -\hat{A}_\gamma^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{A}_\gamma^{-1} & 0 \\ 0 & -(\nu+\gamma)M^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BA_\gamma^{-1} & I \end{bmatrix}$$

- How do we compute $\hat{A}_\gamma^{-1}$ efficiently ? Can we adopt a multigrid approach ?

Subspace correction methods for nearly singular problems

UNIVERSITY OF OXFORD
Mathematical Institute

To compute $\hat{A}_\gamma^{-1}$ we can use a subspace correction method. We decompose the space V_h as follows:

$$V_h = \sum_{i=1} V_i.$$

We consider a coarse space V_H and the projection and injection operators:

$$P_H : V_H \to V_h, \qquad I : V_h \to V_i.$$

We then consider as smoother the two–level additive Schwarz preconditioner defined as:

$$\hat{A}_\gamma^{-1} = P_H A_{\gamma,H}^{-1} P_H^T + \sum_{i=1} I_i A_{\gamma,i}^{-1} I_i^T.$$

Subspace correction methods for nearly singular problems

Robust relaxetion

We need the discrete kernel,

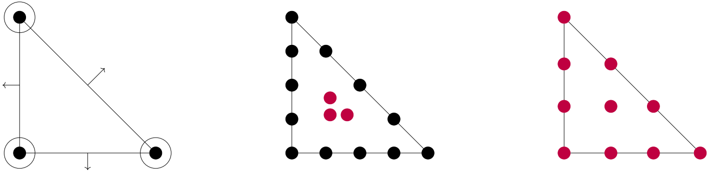
$$\mathbb{K}_h = \{\boldsymbol{v} \in V_h : B\boldsymbol{v} = 0\},$$

to decompose in a **stable** way as follows:

$$\mathbb{K}_h = \sum_{i=1} \mathbb{K}_h \cap V_i.$$

Robust relaxation via FEEC – Hood–Taylor

$$\begin{tikzcd}
0 \arrow[r] & \left[H^2(\Omega)\right]^2 \arrow[r, "\nabla\times"] \arrow[d] & \left[H^1_0(\Omega)\right]^2 \arrow[r, "\nabla\cdot"] \arrow[d] & L^2(\Omega) \arrow[r] \arrow[d] & 0 \\
0 \arrow[r] & \left[\mathbb{P}^5(\mathcal{T}_h)\right]^2 \arrow[r, "\nabla\times"] & \left[\mathbb{P}^4(\mathcal{T}_h)\right]^2 \arrow[r, "\nabla\cdot"] & \mathbb{P}^3_{disc}(\mathcal{T}_h) \arrow[r] & 0
\end{tikzcd}$$



Robust prolongation – Hood–Taylor

University of Oxford Mathematical Institute

Robust prolongation

If the space pair (V_h, Q_h) is inf-sup stable and the meshes are nested, we have a robust prolongation operator defined by:

$$\tilde{P}_H \boldsymbol{u}_H - \tilde{\boldsymbol{u}}_h = \boldsymbol{u}_H - \tilde{\boldsymbol{u}}_h,$$

where

$$a_\gamma(\tilde{\boldsymbol{u}}_h, \tilde{\boldsymbol{v}}_h) = \gamma(\nabla \cdot \tilde{\boldsymbol{u}}_h, \nabla \cdot \tilde{\boldsymbol{v}}_h)_{\pi_{Q_h} \pi_{Q_h}{}^T} \quad \forall \tilde{\boldsymbol{v}}_h \in \tilde{V}_h,$$

where π_{Q_h} is the L^2 projection onto Q_h and $\tilde{V}_h$ is the space of discrete velocity vanishing at the boundary of the coarse cells.

Firedrake' 24

University of Oxford
Mathematical Institute

Join us at the Firedrake user and developer workshop that will be held between 16-18 September 2024 at the University of Oxford.