Flexible computational abstractions for complex preconditioners

Lawrence Mitchell[1,*]

P. E. Farrell (Oxford) R. C. Kirby (Baylor) M. G. Knepley (Buffalo) F. Wechsung (Oxford)

July 2, 2019

[1]Department of Computer Science, Durham University
*lawrence.mitchell@durham.ac.uk

Software setting

Firedrake (`www.firedrakeproject.org`) [...] *is an automated system for the solution of partial differential equations using the finite element method.*

- Finite element problems specified with *embedded* domain specific language, UFL from the FEniCS project.
- *Runtime* compilation to optimised, low-level (C) code.
- PETSc for meshes and (algebraic) solvers.

`arXiv:1501.01809 [cs.MS]`

Advert

3rd Firedrake user meeting is in Durham 26 & 27 September 2019.

`www.firedrakeproject.org/firedrake_19.html`

UFL makes it easy to write down many complex PDEs

Rayleigh-Bénard convection

$$-\Delta u + u \cdot \nabla u + \nabla p + \frac{\mathrm{Ra}}{\mathrm{Pr}} \hat{g} T = 0$$

$$\nabla \cdot u = 0$$

$$-\frac{1}{\mathrm{Pr}} \Delta T + u \cdot \nabla T = 0$$

Newton

$$\begin{bmatrix} F & B^T & M_1 \\ C & 0 & 0 \\ M_2 & 0 & K \end{bmatrix} \begin{bmatrix} \delta u \\ \delta p \\ \delta T \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

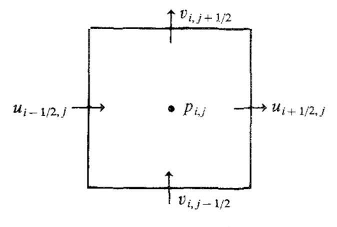
```
from firedrake import *
mesh = ...
V = VectorFunctionSpace(mesh, "CG", 2)
W = FunctionSpace(mesh, "CG", 1)
Z = V * W * W
Ra = Constant(200)
Pr = Constant(6.18)
upT = Function(Z)
u, p, T = split(upT)
v, q, S = TestFunctions(Z)
bcs = ...

F = (inner(grad(u), grad(v))
   + inner(dot(grad(u), u), v)
   - inner(p, div(v))
   + (Ra/Pr)*inner(T*g, v)
   + inner(div(u), q)
   + inner(dot(grad(T), u), S)
   + (1/Pr) * inner(grad(T), grad(S)))*dx

solve(F == 0, upT, bcs=bcs)
```

What about the solver?

Some motivating schemes

Coupled multigrid for Stokes/Navier–Stokes

In the SCGS scheme four velocites and one pressure corresponding to one finite difference node are simultaneously updated by inverting a (small) matrix of equations.

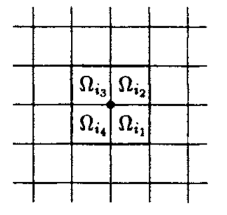


Vanka (1986)

Some motivating schemes

p-independent preconditioners for elliptic problems

[Each subspace is generated from] $V_i^p = V^p \cap H_0^1(\Omega_i')$ *where* Ω_i' *is the open square centered at the ith vertex*

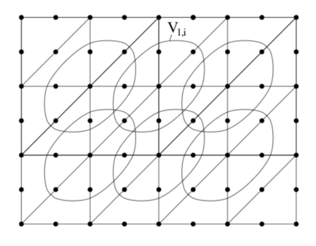


Pavarino (1993)

Some motivating schemes

Multigrid for nearly incompressible elasticity

The suggested smoother is a block Jacobi smoother, which takes care of the kernel [...]. These kernel basis functions are captured by subspaces $V_{l,i}$ as shown

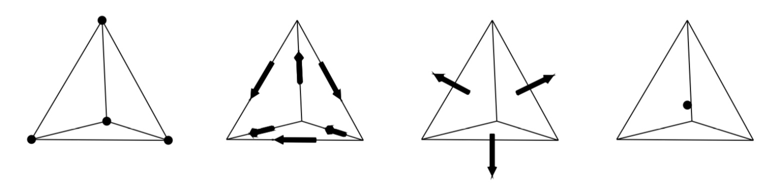


Schöberl (1999)

Some motivating schemes

Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$

To define the Schwarz smoothers, we can use a decomposition of V_h into local patches consisting of all elements surrounding either an edge or a vertex.

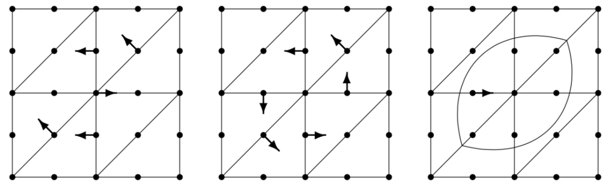


Arnold, Falk, and Winther (2000)

Some motivating schemes

An augmented Lagrangian approach to the Oseen problem

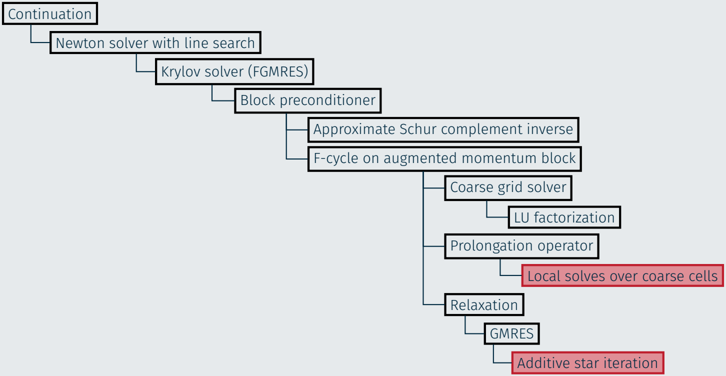
We use a block Gauss-Seidel method [...] based on the decomposition $V_h = \sum_{i=0}^{l} V_i$. [...For] P2-P0 finite elements the natural choice is to gather nodel DOFs for velocity inside ovals [around a vertex]



Benzi and Olshanskii (2006)

Some motivating schemes

Augmented Lagrangian for 3D Navier–Stokes



Farrell, Mitchell, and Wechsung (2018)

Unifying observation

Smoothers all use block Jacobi/G-S with problem-specific choice of blocks.

- Decompose space (usually) based on some mesh decomposition
- Build and solve little problems on the resulting patches
- Combine additively or multiplicatively

Unifying observation

Smoothers all use block Jacobi/G-S with problem-specific choice of blocks.

- Decompose space (usually) based on some mesh decomposition
- Build and solve little problems on the resulting patches
- Combine additively or multiplicatively

Challenge

Want to do this inside block preconditioners, and as a multigrid smoother.

Not sufficient to specify dof decomposition on a (single) global matrix.

PCPATCH

PCPATCH

Requirements

- Want *flexible* PC $\Rightarrow$ change decomposition easily
- Need to nest inside more complex solvers

PCPATCH

Requirements

- Want *flexible* PC $\Rightarrow$ change decomposition easily
- Need to nest inside more complex solvers

Idea

- Separate topological decomposition from algebraic operators
- User only provides topological description of patches
- Ask discretisation library to make the operators once decomposition is obtained

PCPATCH

Idea

- Separate topological decomposition from algebraic operators
- User only provides topological description of patches
- Ask discretisation library to make the operators once decomposition is obtained

Library support

- PETSc (`mcs.anl.gov/petsc`): `DMPlex` + `PetscDS`

  `-pc_type patch`
- Firedrake (`www.firedrakeproject.org`):

  `-pc_type python -pc_python_type firedrake.PatchPC`

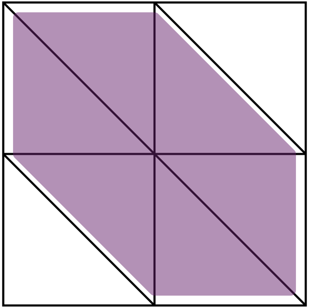
  `-snes_type python -snes_python_type firedrake.PatchSNES`

Describing patches

- `DMPlex` associates dofs with topological entities in mesh
- A patch is defined by a set of these entities, `PCPATCH` determines the dofs that correspond to them
- Adjacency relations defined using topological queries: often `star` and `closure`

Describing patches

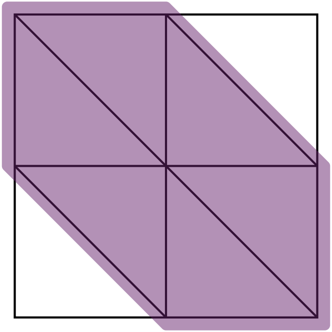
- `DMPlex` associates dofs with topological entities in mesh
- A patch is defined by a set of these entities, `PCPATCH` determines the dofs that correspond to them
- Adjacency relations defined using topological queries: often `star` and `closure`



`star(vertex)`

Describing patches

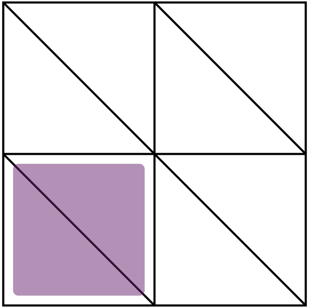
- `DMPlex` associates dofs with topological entities in mesh
- A patch is defined by a set of these entities, `PCPATCH` determines the dofs that correspond to them
- Adjacency relations defined using topological queries: often `star` and `closure`



`closure(star(vertex))`

Describing patches

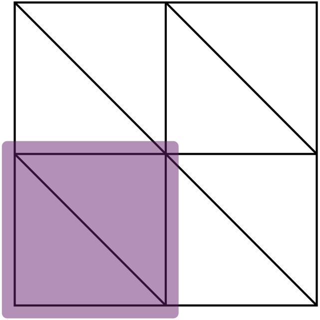
- `DMPlex` associates dofs with topological entities in mesh
- A patch is defined by a set of these entities, `PCPATCH` determines the dofs that correspond to them
- Adjacency relations defined using topological queries: often `star` and `closure`



`star(edge)`

Describing patches

- `DMPlex` associates dofs with topological entities in mesh
- A patch is defined by a set of these entities, `PCPATCH` determines the dofs that correspond to them
- Adjacency relations defined using topological queries: often `star` and `closure`



`closure(star(edge))`

Describing patches

- Each patch defined by set of mesh entities

Builtin

Specify patches by selecting:

1. Mesh entities $\{p_i\}$ to iterate over (e.g. vertices, cells)
2. Adjacency relation that gathers points in patch

`star` entities in star(p_i)

`vanka` entities in closure(star(p_i))

User-defined

1. Custom adjacency relation (e.g. “only vertices in `closure o star`”)
2. List of patches, plus iteration order $\Rightarrow$ line-/plane-smoothers

Patch assembly

- If we just want homogeneous Dirichlet, can use list of dofs to select from assembled global operator

✗ Doesn't work for other transmission conditions

✗ Doesn't work for nonlinear smoothers

⇒ Callback interface to get PDE library to assemble on each patch

Callbacks

```
/* Patch Jacobian */
UserComputeOp(PC, Vec state, Mat operator, Patch patch, void *userctx);
/* Patch Residual */
UserComputeF(PC, Vec state, Vec residual, Patch patch, void *userctx);
```

Examples

Which subspace to choose?

- For symmetric problems, can use kernel decomposition theorem of Schöberl (1999) and Lee, Wu, Xu, and Zikatanov (2007)
- Key challenge is to find a decomposition $\{V_i\}$ such that every u in the kernel $\mathcal{N}$ can be written as $u = \sum_i u_i$ with $u_i \in V_i \cap \mathcal{N}$.

Characterising the kernel

Appropriate discrete de Rham complexes can help us finding the support of a basis for $\mathcal{N}$.

$H(\mathrm{div})$ multigrid in 2D (Arnold, Falk, and Winther 1997)

$$\text{Find } u \in V \subset H(\mathrm{div}) \text{ s.t.} \quad (u, v)_{L^2} + \gamma(\mathrm{div}\, u, \mathrm{div}\, v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$$

L^2 de Rham complex

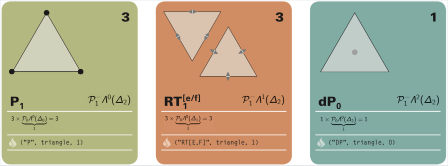
$$H^1 \xrightarrow{\mathrm{grad}^\perp} H(\mathrm{div}) \xrightarrow{\mathrm{div}} L^2$$

$H(\mathrm{div})$ multigrid in 2D (Arnold, Falk, and Winther 1997)

$$\text{Find } u \in V \subset H(\mathrm{div}) \text{ s.t. } \quad (u, v)_{L^2} + \gamma(\operatorname{div} u, \operatorname{div} v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$$

L^2 de Rham complex

$$H^1 \xrightarrow{\mathrm{grad}^\perp} H(\mathrm{div}) \xrightarrow{\mathrm{div}} L^2$$



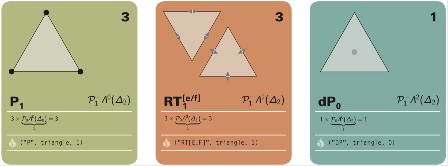
femtable.org

$H(\mathrm{div})$ multigrid in 2D (Arnold, Falk, and Winther 1997)

Find $u \in V \subset H(\mathrm{div})$ s.t. $\quad (u, v)_{L^2} + \gamma(\operatorname{div} u, \operatorname{div} v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$

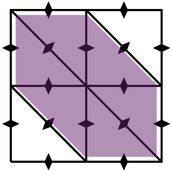
L^2 de Rham complex

$$H^1 \xrightarrow{\mathrm{grad}^\perp} H(\mathrm{div}) \xrightarrow{\mathrm{div}} L^2$$



femtable.org

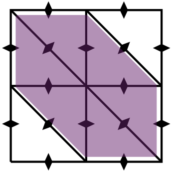
- Exact sequence: $\ker(\mathrm{div}) = \mathrm{range}(\mathrm{grad}^\perp)$
- Need patches containing support of the P_k basis functions $\Rightarrow$ star around vertices



$H(\mathrm{div})$ multigrid in 2D (Arnold, Falk, and Winther 1997)

Find $u \in V \subset H(\mathrm{div})$ s.t. $\quad (u, v)_{L^2} + \gamma(\mathrm{div}\, u, \mathrm{div}\, v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$

```
-ksp_type cg
-pc_type mg
-mg_levels_
   -pc_type python
   -pc_python_type firedrake.PatchPC
   -patch_
      -pc_patch_construct_dim 0
      -pc_patch_construct_type star
```



Smoother \ γ	0	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
Point-Jacobi ($k = 1$)	11	27	49	68	86	103
Point-Jacobi ($k = 2$)	10	45	71	93	113	134
Block-Jacobi ($k = 1$)	6	11	12	12	12	12
Block-Jacobi ($k = 2$)	7	8	8	8	8	8

Table 1: Iteration counts for multigrid preconditioned CG using RT_k elements.

$H(\mathrm{div})$ and $H(\mathrm{curl})$ multigrid in 3D (Arnold, Falk, and Winther 2000)

$$\text{Find } u \in V \subset H(\mathbf{curl}) \text{ s.t.} \quad (u, v)_{L^2} + \gamma(\mathbf{curl}\, u, \mathbf{curl}\, v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$$

L^2 de Rham complex

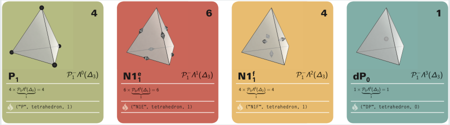
$$H^1 \xrightarrow{\mathrm{grad}} H(\mathrm{curl}) \xrightarrow{\mathrm{curl}} H(\mathrm{div}) \xrightarrow{\mathrm{div}} L^2$$

$H(\mathrm{div})$ and $H(\mathrm{curl})$ multigrid in 3D (Arnold, Falk, and Winther 2000)

Find $u \in V \subset H(\mathrm{curl})$ s.t. $\quad (u, v)_{L^2} + \gamma(\mathrm{curl}\, u, \mathrm{curl}\, v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$

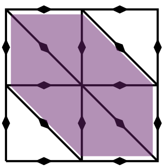
L^2 de Rham complex

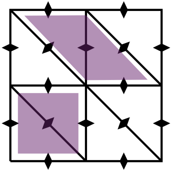
$$H^1 \xrightarrow{\text{grad}} H(\mathrm{curl}) \xrightarrow{\text{curl}} H(\mathrm{div}) \xrightarrow{\text{div}} L^2$$



femtable.org

- Exact sequence:
  $\ker(\mathrm{curl}) = \mathrm{range}(\mathrm{grad})$,
  $\ker(\mathrm{div}) = \mathrm{range}(\mathrm{curl})$
- $H(\mathrm{curl})$: star around vertices
- $H(\mathrm{div})$: star around edges

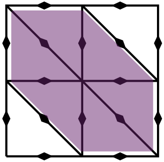




$H(\text{curl})$ multigrid in 3D (Arnold, Falk, and Winther 2000)

Find $u \in V \subset H(\text{curl})$ s.t. $\quad (u, v)_{L^2} + \gamma(\text{curl}\, u, \text{curl}\, v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$

```
-ksp_type cg
-pc_type mg
-mg_levels_
   -pc_type python
   -pc_python_type firedrake.PatchPC
   -patch_
      -pc_patch_construct_dim 0
      -pc_patch_construct_type star
```



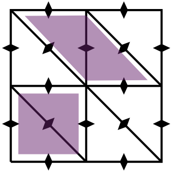
Smoother \ γ	0	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
Point-Jacobi ($k = 1$)	10	48	85	120	150	180
Point-Jacobi ($k = 2$)	22	115	211	293	370	446
Block-Jacobi ($k = 1$)	9	16	18	18	18	18
Block-Jacobi ($k = 2$)	9	12	12	12	12	12

Table 2: Iteration counts for multigrid preconditioned CG using Nedelec edge-elements of the first kind.

$H(\mathrm{div})$ multigrid in 3D (Arnold, Falk, and Winther 2000)

Find $u \in V \subset H(\mathrm{div})$ s.t. $\quad (u, v)_{L^2} + \gamma(\mathrm{div}\, u, \mathrm{div}\, v)_{L^2} = (f, v)_{L^2} \quad \forall v \in V.$

```
-ksp_type cg
-pc_type mg
-mg_levels_
   -pc_type python
   -pc_python_type firedrake.PatchPC
   -patch_
      -pc_patch_construct_dim 1
      -pc_patch_construct_type star
```



Smoother \ γ	0	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
Point-Jacobi ($k = 1$)	11	63	109	146	180	221
Point-Jacobi ($k = 2$)	26	180	366	531	687	844
Block-Jacobi ($k = 1$)	12	30	36	36	37	37
Block-Jacobi ($k = 2$)	11	17	17	17	17	17

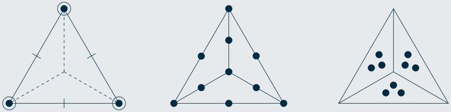
Table 3: Iteration counts for multigrid preconditioned CG using Nedelec face-elements of the first kind.

Nearly incompressible elasticity

$$\text{Find } u \in V \subset H^1 \text{ s.t.} \quad (\operatorname{grad} u, \operatorname{grad} v) + \gamma(\operatorname{div} u, \operatorname{div} v) = (f, v) \quad \forall v \in V.$$

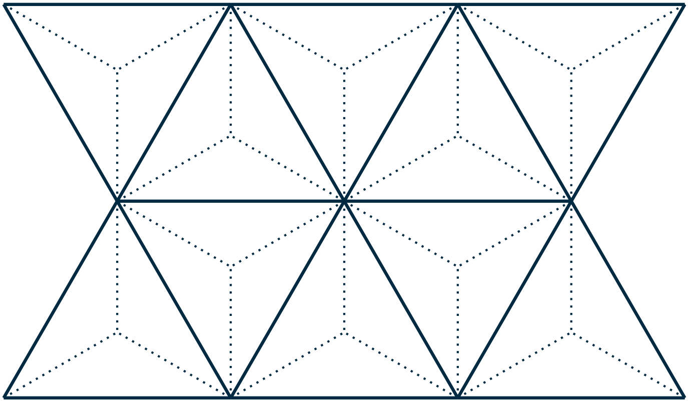
2D Stokes complex

$$H^2 \xrightarrow{\operatorname{grad}^\perp} H^1 \xrightarrow{\operatorname{div}} L^2$$

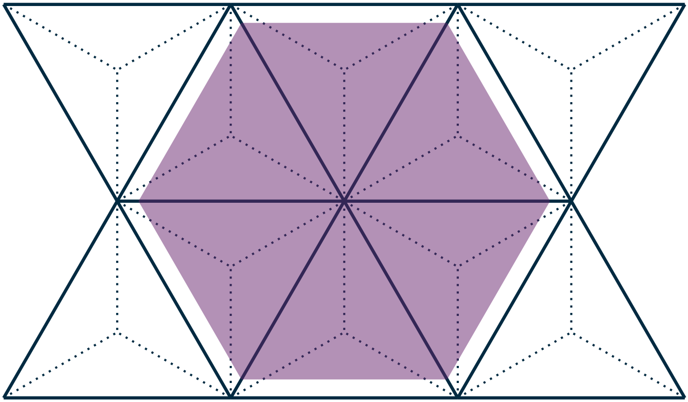


- Decomposition must capture $\ker \operatorname{div} = \operatorname{range} \operatorname{grad}^\perp$.
- Support of HCT element is on "macro" mesh $\Rightarrow$ `MacroStar`

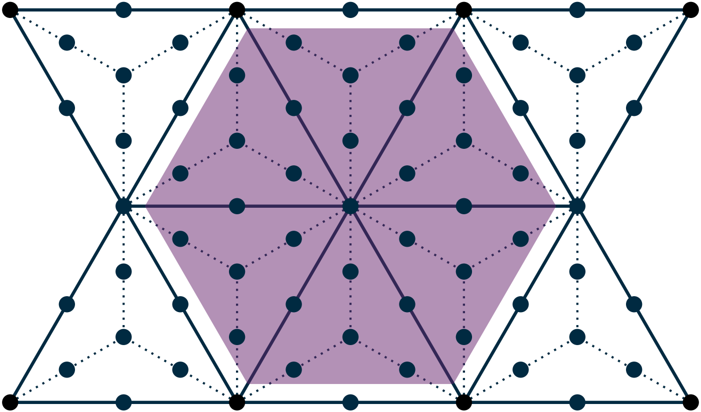
MacroStar



MacroStar



MacroStar



MacroStar

```
-ksp_type cg
-pc_type mg
-mg_levels_
  -pc_type python
  -pc_python_type firedrake.PatchPC
    -patch_
      -pc_patch_construct_dim 0
      -pc_patch_construct_type python
      -pc_patch_construct_python_type MacroStar
```

Just need to write custom adjacency to construct patch around each vertex

MacroStar

```
-ksp_type cg
-pc_type mg
-mg_levels_
  -pc_type python
  -pc_python_type firedrake.PatchPC
    -patch_
      -pc_patch_construct_dim 0
      -pc_patch_construct_type python
      -pc_patch_construct_python_type MacroStar
```

Just need to write custom adjacency to construct patch around each vertex

```python
class MacroStar(OrderedRelaxation):
    def callback(self, dm, vertex):
        if dm.getLabelValue("MacroVertices", vertex) != 1:
            return None
        s = list(self.star(dm, vertex))
        closures = list(chain(*(self.closure(dm, e) for e in s)))
        want = [v for v in closures if dm.getLabelValue("MacroVertices", v) != 1]
        star = list(chain(*(self.star(dm, v) for v in want)))
        return s + star
```

Vanka for Stokes

Find $(u, p) \in V \times Q \subset (H^1)^d \times L^2$ s.t.

$$(\operatorname{grad} u, \operatorname{grad} v) - (p, \operatorname{div} v) - (\operatorname{div} u, q) = (f, v) \quad \forall (v, q) \in V \times Q.$$

Vanka patch

Solve simultaneously for (u, p) on each pressure dof, gathering those velocity dofs that couple to the pressure dof.

- **P2-P0: loop over cells, gather closure of star**
- P2-P1: loop over vertices, gather closure of star

Vanka for Stokes

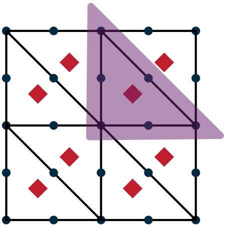
Find $(u, p) \in V \times Q \subset (H^1)^d \times L^2$ s.t.

$$(\operatorname{grad} u, \operatorname{grad} v) - (p, \operatorname{div} v) - (\operatorname{div} u, q) = (f, v) \quad \forall (v, q) \in V \times Q.$$

Vanka patch

Solve simultaneously for (u, p) on each pressure dof, gathering those velocity dofs that couple to the pressure dof.

- **P2-P0: loop over cells, gather closure of star**
- P2-P1: loop over vertices, gather closure of star



```
-ksp_type cg
-pc_type mg
-mg_levels_
   -pc_type python
   -pc_python_type firedrake.PatchPC
   -patch_
      -pc_patch_construct_codim 0
      -pc_patch_construct_type vanka
      -pc_patch_exclude_subspaces 1
```

Vanka for Stokes

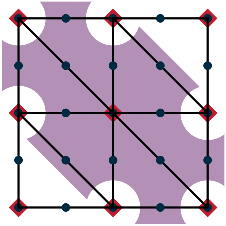
Find $(u, p) \in V \times Q \subset (H^1)^d \times L^2$ s.t.

$$(\operatorname{grad} u, \operatorname{grad} v) - (p, \operatorname{div} v) - (\operatorname{div} u, q) = (f, v) \quad \forall (v, q) \in V \times Q.$$

Vanka patch

Solve simultaneously for (u, p) on each pressure dof, gathering those velocity dofs that couple to the pressure dof.

- P2-P0: loop over cells, gather closure of star
- **P2-P1: loop over vertices, gather closure of star**



```
-ksp_type cg
-pc_type mg
-mg_levels_
   -pc_type python
   -pc_python_type firedrake.PatchPC
   -patch_
      -pc_patch_construct_dim 0
      -pc_patch_construct_type vanka
      -pc_patch_exclude_subspaces 1
      -pc_patch_vanka_dim 0
```

Conclusions

- `PCPATCH` provides simple and flexible interface for subspace correction methods
- Currently works with `DMPlex` + `PetscDS` and Firedrake
- Implements
    - Additive and multiplicative smoothing
    - Simultaneous smoothing of multiple fields: monolithic approaches
    - Partition of unity (or not)
    - Nonlinear relaxation (Firedrake only)

Thanks!

References

- Alnæs, M. S., A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells (2014). "Unified Form Language: A Domain-specific Language for Weak Formulations of Partial Differential Equations". *ACM Trans. Math. Softw.* 40. doi:10.1145/2566630. arXiv: 1211.4047 [cs.MS].
- Arnold, D. N., R. S. Falk, and R. Winther (2000). "Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$". *Numerische Mathematik* 85. doi:10.1007/s002110000137.
- Arnold, D. N., R. S. Falk, and R. Winther (July 1997). "Preconditioning in $H(\mathrm{div})$ and Applications". *Mathematics of Computation* 66. doi:10.1090/S0025-5718-97-00826-0.
- Benzi, M. and M. A. Olshanskii (2006). "An Augmented Lagrangian-Based Approach to the Oseen Problem". *SIAM Journal on Scientific Computing* 28. doi:10.1137/050646421.
- Farrell, P. E., L. Mitchell, and F. Wechsung (2018). *An augmented Lagrangian preconditioner for the 3D stationary incompressible Navier–Stokes equations at high Reynolds number*. To appear in SIAM SISC. arXiv: 1810.03315 [math.NA].
- Lee, Y.-J., J. Wu, J. Xu, and L. Zikatanov (2007). "Robust subspace correction methods for nearly singular systems". *Mathematical Models and Methods in Applied Sciences* 17. doi:10.1142/s0218202507002522.
- Pavarino, L. F. (1993). "Additive Schwarz methods for the p-version finite element method". *Numerische Mathematik* 66. doi:10.1007/BF01385709.
- Rathgeber, F. et al. (2016). "Firedrake: automating the finite element method by composing abstractions". *ACM Transactions on Mathematical Software* 43. doi:10.1145/2998441. arXiv: 1501.01809 [cs.MS].
- Schöberl, J. (1999). "Multigrid methods for a parameter dependent problem in primal variables". *Numerische Mathematik* 84. doi:10.1007/s002110050465.
- Vanka, S. (1986). "Block-implicit multigrid solution of Navier-Stokes equations in primitive variables". *Journal of Computational Physics* 65. doi:10.1016/0021-9991(86)90008-2.