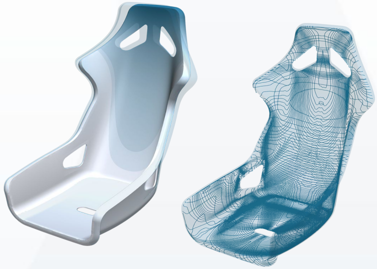
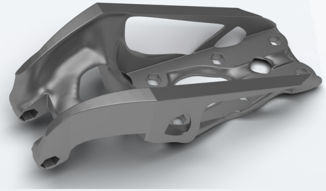
Berlin - Luxembourg - London - Munich





Dynamic composition of solvers for coupled problems in DOLFINx

Martin Řehoř[1,2], **Jack S. Hale**[2]

[1] *Rafinex S.à r.l., Luxembourg*
[2] *Institute of Computational Engineering, University of Luxembourg*

FEniCS 2021
22 March 2021

Project RIFLE

Industrial fellowship | In cooperation with University of Luxembourg | Funded by FNR

rafinex

Description

RIFLE *is* about *Robust Incompressible FLow solver Enhancement*

High-fidelity physics + FEM-based discretizations + Robust preconditioning strategies

Targets

RIFLE *aims* at design problems involving fluids and (a combination of) related phenomena

- ➔ heat transfer in fluids (convection in industrial boilers)
- ➔ non-Newtonian behavior (polymer extrusion, injection molding, AM)
- ➔ multiphase flows (glue deposition, float glass forming)

RIFLE "*hits the bullseye*" with any solutions obtained in the context of stochastic topology optimization

UNIVERSITÉ DU LUXEMBOURG

Fonds National de la Recherche Luxembourg

FEniCS PROJECT

Composable solvers

Motivation

Objectives

- ➔ Exploit **PETSc**'s multiphysics capabilities [1] within the application code using **DOLFINx** from Python
- ➔ Easy implementation of custom preconditioners, such as pressure-convection-diffusion (PCD) approximation for the incompressible Navier-Stokes equations [2]

Strategy

Motivated by the approaches originally implemented in **FENaPack** [3] and **Firedrake** [4]

a coupled system of PDEs with 3 unknown fields

↓ discretization (+ linearization)

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

↓ configuration

PETSc.KSP object

References

[1] J. Brown, M. G. Knepley, D. A. May, L. C. McInnes, and B. Smith, "Composable Linear Solvers for Multiphysics," in 2012 11th International Symposium on Parallel and Distributed Computing, Munich, Germany, Jun. 2012, pp. 55–62, doi: 10.1109/ISPDC.2012.16.

[2] J. Blechta, "Towards efficient numerical computation of flows of non-Newtonian fluids", 2019, PhD Thesis, Charles university, Faculty of Mathematics and Physics, Mathematical Institute of Charles University, Prague, Czech Republic, url: https://dspace.cuni.cz/handle/20.500.11956/108384.

[3] J. Blechta and M. Řehoř, FENaPack 2018.1.0 (FEniCS Navier-Stokes preconditioning package). Zenodo, 2018, doi: 10.5281/ZENODO.1308015.

[4] R. C. Kirby and L. Mitchell, "Solver Composition Across the PDE/Linear Algebra Barrier," SIAM J. Sci. Comput., vol. 40, no. 1, pp. C76–C98, 2017, doi: 10.1137/17M1133208.

Composable solvers

Block assembly & FieldSplit preconditioner

Support for block systems in **DOLFINx**:

```
A = dolfinx.fem.assemble_matrix_nest(a)
b = dolfinx.fem.assemble_vector_nest(L)
# or
A = dolfinx.fem.assemble_matrix_block(a)
b = dolfinx.fem.assemble_vector_block(L, a)
```

`FieldSplit` preconditioner from **PETSc** is designed for the construction of block solvers using

- ➔ block relaxation ($n \times n$ systems)
- ➔ block factorization (2×2 systems)

with block decomposition based on index sets

- ➔ provided in `PETSc.DM` object (not supported)
- ➔ <u>passed directly to `FieldSplit` preconditioner</u>

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

Configuration of the preconditioner instance:

```
pc.setFieldSplitFields(...)
# or
pc.setFieldSplitIS(...)
```

Configuration via PETSc options:

```
-pc_fieldsplit_0_fields 0
-pc_fieldsplit_1_fields 1
-pc_fieldsplit_2_fields 2
```

block Jacobi

$$\begin{bmatrix} ksp(A_{00}, Ap_{00}) & 0 & 0 \\ 0 & ksp(A_{11}, Ap_{11}) & 0 \\ 0 & 0 & ksp(A_{22}, Ap_{22}) \end{bmatrix}$$

Composable solvers

Changing the configuration

Changes in the code when switching from block Jacobi to a block factorization based on Schur complements (2×2 required):

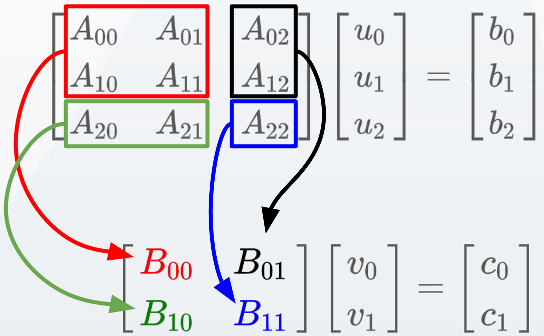
```
# [...]

# FE = ufl.MixedElement([FE0, FE1, FE2]) # !! (block Jacobi)
FE = ufl.MixedElement([ufl.MixedElement([FE0, FE1]), FE2])
space = dolfinx.fem.FunctionSpace(mesh, FE)

# [...] # !! different splitting of test and trial functions
A = dolfinx.fem.assemble_matrix_nest(a)
# [...]

ises, _ = A.getNestISes()
fields = [(str(i), iset) for i, iset in enumerate(ises)]
pc.setFieldSplitIS(*fields)

# [...] # !! other changes when postprocessing the solution
```



Preferred setup (**not supported by default**):

```
-pc_fieldsplit_0_fields 0,1
-pc_fieldsplit_1_fields 2
```

Question: What if we want to use another FieldSplit for the combined B_{00} block?

Composable solvers

Implementation & Usage

```
# [...]

FE = ufl.MixedElement([FE0, FE1, FE2])
space = dolfinx.fem.FunctionSpace(mesh, FE)

# [...]
A = create_splittable_matrix_block(a) # A.getType() == "python" 
A.assemble()
# [...]

opts = PETSc.Options()
opts["pc_type"] = "python"
opts["pc_python_type"] = "fenics_pctools.WrappedPC"
opts["wrapped_pc_type"] = "fieldsplit"
opts["wrapped_pc_fieldsplit_0_fields"] = 0
opts["wrapped_pc_fieldsplit_1_fields"] = 1
opts["wrapped_pc_fieldsplit_2_fields"] = 2

# ises, _ = A.getPythonContext().ISes
# fields = [(str(i), iset) for i in enumerate(ises)]
# pc.setFieldSplitIS(*fields) # isinstance(pc, WrappedPC) 

# [...]
```

`SplittableMatrixBlock` context

- ➔ wraps PETSc.MAT object assembled as a block matrix in **DOLFINx**
- ➔ keeps index sets
- ➔ keeps PDE-level info (forms, bcs, etc.)
- ➔ knows how to extract a submatrix given a combination of block indices

`WrappedPC` preconditioner

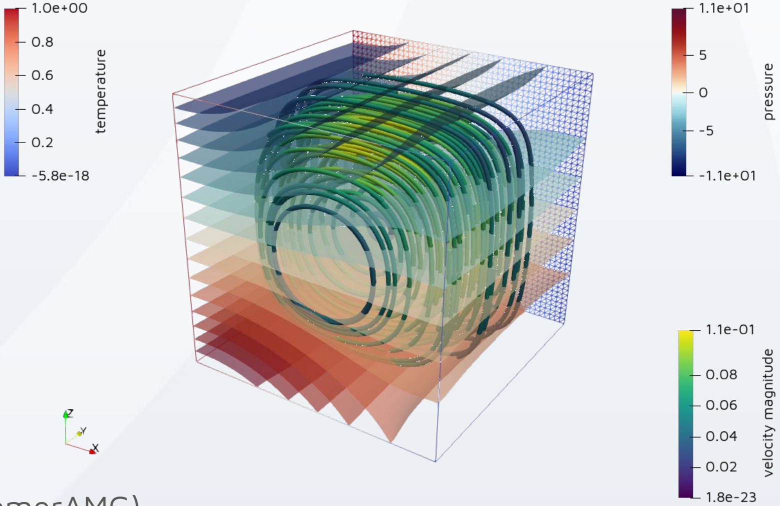
- ➔ works with splittable block matrices
- ➔ wraps `PETSc.PC` object that interacts with the wrapped matrix

Preliminary results

Rayleigh-Bénard convection [4]

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

$$\begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$$



The **nonlinear** problem is solved using the Newton method with

- ➔ outer linear iteration: FGMRES with block Gauss-Seidel
- ➔ temperature block: GMRES with algebraic multigrid (Hypre BoomerAMG)
- ➔ Navier-Stokes block: GMRES with a lower Schur complement factorization, where for the Schur complement we use the PCD approximation from [2]

References

[2] J. Blechta, "Towards efficient numerical computation of flows of non-Newtonian fluids", 2019, PhD Thesis, Charles university, Faculty of Mathematics and Physics, Mathematical Institute of Charles University, Prague, Czech Republic, url: https://dspace.cuni.cz/handle/20.500.11956/108384.

[4] R. C. Kirby and L. Mitchell, "Solver Composition Across the PDE/Linear Algebra Barrier," SIAM J. Sci. Comput., vol. 40, no. 1, pp. C76–C98, 2017, doi: 10.1137/17M1133208.

Preliminary results

Rayleigh-Bénard convection

Weak scaling

~30,000 DOF / process

DOF ($\times 10^6$)	MPI processes	Nonlinear iterations	Linear iterations	Navier-Stokes iterations	Temperature iterations	Time to solution (s)
0.7405	24	2	8	120 (15)	50 (6.2)	8.62
1.488	48	2	8	123 (15.4)	49 (6.1)	9.05
2.793	96	2	8	121 (15.1)	50 (6.2)	8.88
5.769	192	2	9	134 (14.9)	56 (6.2)	10.3
11.66	384	2	9	139 (15.4)	56 (6.2)	12.6
23.39	768	2	9	141 (15.7)	56 (6.2)	12.4

70% efficiency

Strong scaling

max ~500,000
min ~30,000
DOF / process

DOF ($\times 10^6$)	MPI processes	Nonlinear iterations	Linear iterations	Navier-Stokes iterations	Temperature iterations	Time to solution (s)
14.09	28	2	8	111 (13.9)	42 (5.2)	140
14.09	56	2	8	110 (13.8)	42 (5.2)	69.7
14.09	112	2	9	131 (14.6)	56 (6.2)	41.3
14.09	224	2	9	133 (14.8)	56 (6.2)	22.3
14.09	448	2	9	143 (15.9)	56 (6.2)	12.7

70% efficiency

The experiments presented in this work were carried out using the HPC facilities of the University of Luxembourg [5].

References

[5] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an Academic HPC Cluster: The UL Experience," in Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014), Bologna, Italy, Jul. 2014, pp. 959–967, url: http://hdl.handle.net/10993/16622

Conclusion & Further steps

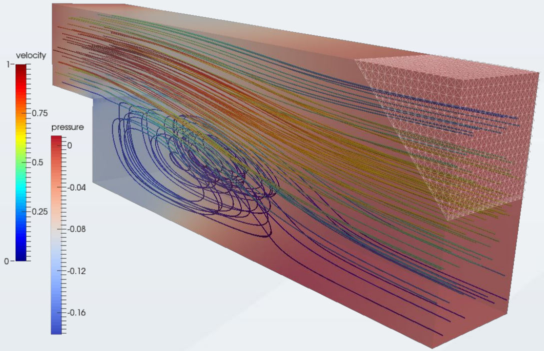
Dynamic composition of block solvers using **DOLFINx**

➔ is implemented entirely in **petsc4py**

➔ gives satisfactory scaling results for decent-sized problems on HPC infrastructures

➔ is currently used to test various configurations of iterative solvers in the context of stabilized viscoelastic flows

➔ opens the possibility to implement matrix-free methods




```
# Configuration of PCD preconditioner for NS equations
-snes_type newtonls
-snes_linesearch_type basic
-snes_monitor
-snes_converged_reason
-snes_rtol 1.0e-09
-snes_max_it 25

-ksp_converged_reason
-ksp_rtol 1e-10
-ksp_max_it 1000
-ksp_type gmres
-ksp_gmres_restart 150
-ksp_pc_side right
-pc_type python
-pc_python_type fenics_pctools.WrappedPC

-prefix_push wrapped_
-pc_type fieldsplit
-pc_fieldsplit_type schur
-pc_fieldsplit_schur_fact_type upper
-pc_fieldsplit_schur_precondition user
-pc_fieldsplit_0_fields 0
-pc_fieldsplit_1_fields 1
-fieldsplit_0_ksp_type preonly
-fieldsplit_0_pc_type python
-fieldsplit_0_pc_python_type fenics_pctools.WrappedPC
-fieldsplit_0_wrapped_pc_type lu
-fieldsplit_0_wrapped_pc_factor_mat_solver_type mumps
-fieldsplit_1_ksp_type preonly
-fieldsplit_1_pc_type python
-fieldsplit_1_pc_python_type fenics_pctools.PCDPC_vY
-fieldsplit_1_pcd_Mp_ksp_type preonly
-fieldsplit_1_pcd_Mp_pc_type lu
-fieldsplit_1_pcd_Mp_pc_factor_mat_solver_type mumps
-fieldsplit_1_pcd_Ap_ksp_type preonly
-fieldsplit_1_pcd_Ap_pc_type lu
-fieldsplit_1_pcd_Ap_pc_factor_mat_solver_type mumps
-prefix_pop
```

THANK YOU FOR YOUR ATTENTION!

ACKNOWLEDGEMENTS

The present work is supported by the National Research Fund, Luxembourg in the frame of the Industrial Fellowship project RIFLE (13754363).

The experiments presented in this work were carried out using the HPC facilities of the University of Luxembourg.

CONTACT

www.rafinex.com

Conflict of interest: Jack S. Hale declares that he has a family member working at Rafinex S.à r.l.