



Proceedings of  
**FEniCS 2021**  
**22–26 March 2021**

**Editors**

Igor Baratta

Jørgen S. Dokken

Chris Richardson

Matthew W. Scroggs

# Contents

<b>Introduction</b>	<b>1</b>
<b>Monday 22 March</b>	<b>2</b>
Patient specific brain simulations with FEniCS and Freesurfer (Vegard Vinje)	2
A finite element model of electric fields in the brain (Vyassa Baratham)	3
Astrophysical tests of gravity using FEniCS (Andrius Tamosiunas)	18
A study of the time independent grade two model in a 2D contraction rheometer (Måns Andersson, Ridgway Scott, and Johan Jansson)	33
Multiscale-in-time modeling of myocardial growth & disease progression (Marc Hirschvogel)	34
External operators in UFL and Firedrake (Nacime Bouziani and David Ham)	51
Linear multipoint constraints in FEniCSx (Jørgen Schartum Dokken, Garth Wells, and Chris Richardson)	68
Implementation of p-multigrid and approximate fast diagonalization methods in Firedrake (Pablo Brubeck and Patrick Farrell)	86
Dynamic composition of solvers for coupled problems in DOLFINx (Martin Řehoř and Jack S. Hale)	87
Additive Schwarz methods for serendipity elements (Jorge Marchena Menendez and Robert Kirby)	88
Domain decomposition of stochastic PDEs using FEniCS (Abhijit Sarkar, Sudhi Sharma, Ajit Desai, Mohammad Khalil, Chris Pettit, and Dominique Poiriel)	89
Towards IEM-FEM coupling for simulation of photoacoustic trace gas sensors (Xiaoyu Wei, Andreas Kloeckner, Robert Kirby, and Peter Coogan)	90
On the reproducibility of numerical experiments with FEniCS (Paul Garlick)	91
mgis.fenics Part I: Coupling MFront and FEniCS for complex solid mechanics simulations (Thomas Helfer, Jérémy Bleyer, Raffaele Russo, and Tamara Dancheva)	100
mgis.fenics Part II: Cosserat media in small deformation with mgis.fenics (Tamara Dancheva, Unai Alonso, Michael Barton, Jérémy Bleyer, Thomas Heffler, and Raffaele Russo)	121
<b>Tuesday 23 March</b>	<b>137</b>
FFCx code generation for expressions (Michal Habera and Andreas Zilian)	137
Explicit dual space representation in UFL (India Marsden, David A Ham, and Reuben Nixon-Hill)	138
Turning FEniCS inside out (Chris Richardson)	151
Making point sets first class in UFL and Firedrake (Reuben W. Nixon-Hill, Daniel Shapero, Colin J. Cotter, and David A. Ham)	164
dolfiny: Convenience wrappers for DOLFINx (Andreas Zilian and Michal Habera)	183
Back to Basix: Construction of arbitrary order finite element DOF maps on polygonal and polyhedral cell meshes (Matthew Scroggs, Jørgen S. Dokken, Chris Richardson, and Garth N. Wells)	193
hIPPYlib-MUQ: Scalable Markov chain Monte Carlo sampling methods for large-scale Bayesian inverse problems governed by PDEs (Ki-Tae Kim, Umberto Villa, Matthew Parno, Noemi Petra, Youssef Marzouk, and Omar Ghattas)	221
Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs (Federico Pichi, Francesco Ballarin, Gianluigi Rozza, and Jan S. Hesthaven)	222
Run-time from 300 years to 300 min: Lessons learned in large-scale modeling in FEniCS (Abhinav Gupta, U Meenu Krishnan, Rajib Chowdhury, and Anupam Chakrabarti)	241



Piola-mapped finite elements in Firedrake for linear elasticity and Stokes flow (Francis Aznaran, Patrick Farrell, and Robert Kirby) . . . . .	258
UFL to GPU: Generating near roofline actions kernels (Kaushik Kulkarni and Andreas Kloeckner)	302
A second order scheme to compute geometric interfaces with applications in microfluids (Stephan Schmidt, Melanie Gräßer, and Hans-Joachim Schmid) . . . . .	303
Stochastic topology optimisation for robust and manufacturable designs (Johannes Neumann) .	304
Computing multiple solutions of topology optimisation problems with FEniCS (Ioannis Papadopoulos, Patrick Farrell, and Thomas Surowiec) . . . . .	305
Shape optimization in coupled fluid-structure systems using Multiphenics (Harisankar Ramaswamy, Saikat Dey, and Assad Oberai) . . . . .	306
ATOMiCS: topology optimization using FEniCS and OpenMDAO (Jiyao Yan, Ru Xiang, David Kamensky, and John Hwang) . . . . .	307
<b>Wednesday 24 March</b>	<b>308</b>
Simple and sharp: Error estimates of Bank–Weiser type in the FEniCS Project (Jack S. Hale, Raphael Bulle, Alexei Lozinski, Stéphane P. A. Bordas, and Franz Chouly) . . . . .	308
Local <i>a posteriori</i> error estimates for the spectral fractional Laplacian (Raphaël Bulle, Stéphane P. A. Bordas, Franz Chouly, Jack S. Hale, and Alexei Lozinski) . . . . .	336
Making linearized methods in fluid mechanics available to a broader audience with FEniCS (Thomas Ludwig Kaiser, Chuhan Wang, Kilian Oberleithner, and Lutz Lesschaff) . . . . .	369
Predicting sound absorption in additively manufactured porous materials using multiscale simulations in FEniCS (Kamil C. Opiela and Tomasz G. Zieliński) . . . . .	370
Modelling of solidification problems with FEniCS (Florian Arbes, Kent-Andre Mardal, Øyvind Jensen, and Jørgen S. Dokken) . . . . .	371
Digital Math: Solving the reproducibility crisis in the digital era (Johan Jansson, Måns Andersson, Linde Van Beers, Ridgway Scott, and Claes Johnson) . . . . .	372
Coupling of non-matching isogeometric shells, with applications in aerospace structures (Han Zhao, Xiangbei Liu, John T. Hwang, and David Kamensky) . . . . .	373
Generating layer-adapted meshes using mesh PDEs (Róisín Hill and Niall Madden) . . . . .	374
CutFEM-style methods in FEniCSx: CAD and level sets (August Johansson and Vibeke Skytt) .	401
Finite elements on accelerators: an experience using FEniCSx and SYCL (Igor Baratta, Chris Richardson, and Garth Wells) . . . . .	413
Working with complex meshes: The mesh processing pipeline (U Meenu Krishnan, Abhinav Gupta, and Rajib Chowdhury) . . . . .	430
<b>Thursday 25 March</b>	<b>439</b>
Numerical investigation of the interaction of two electrolytic drops under an external electric field (Shyam Sunder Yadav) . . . . .	439
FESTIM, a modelling code for hydrogen transport in materials for nuclear fusion applications (Rémi Delaporte-Mathurin, Etienne Hodille, Floriane Leblond, Jonathan Mougennot, Yann Charles, Christian Grisolia, and James Dark) . . . . .	458
Interfacing AceGEN and FEniCS for advanced constitutive models (Jakub Lengiewicz, Michal Habera, Andreas Zilian, and Stéphane Bordas) . . . . .	474
FEniCS-preCICE: Coupling FEniCS to other simulation software (Ishaan Desai, Benjamin Rodenberg, Richard Hertrich, Alexander Jaust, and Benjamin Uekermann) . . . . .	475
Motion of synthetic microswimmers at low Reynolds numbers with FEniCS (Roberto Ausas, Stevens Paz, Paula Alvez da Silva, and Gustavo Buscaglia) . . . . .	487
Development of an open-source-based framework for multiphysical crystal growth simulations (Arved Enders-Seidlitz, Josef Pal, and Kaspars Dadzis) . . . . .	488
Coupling of thermomechanics with electromagnetism in FEniCS (Bilen Emek Abali) . . . . .	502
Implementation of a nonlinear anisotropic image denoising model in FEniCS (Abderrazzak Boufala and El Mostafa Kalmoun) . . . . .	518
Bistatic polarimetric through-wall SAR public release data set (Daniel Andre and Richard Sabbiers)	519
Nonlocal UFL: Finite elements for Helmholtz equations with a nonlocal boundary condition (Benjamin Sepanski, Robert Kirby, and Andreas Kloeckner) . . . . .	520

Plasma modelling using FEniCS and FEDM (Aleksandar P. Jovanovic, Detlef Loffhagen, and Markus M. Becker) . . . . .	577
Implementation of fluid-structure interactions for rigid body motion in FEniCS using immersed finite element method (Chayut Teeraratkul and Debanjan Mukherjee) . . . . .	598
Direct FEM Simulation for air pollution dispersion in building aerodynamics (Linde van Beers, Johan Jansson, and Måns Andersson) . . . . .	599
A tracer’s sojourn in a compressible velocity field (Nate Sime, Cian Wilson, and Peter van Keken) . . . . .	600
<b>Friday 26 March</b>	<b>601</b>
A two-level nonlinear beam model using adjoints (Marco Morandini) . . . . .	601
Interoperability with automatic differentiation libraries through NumPy interface to FEniCS (Ivan Yashchuk) . . . . .	642
Developing an automatized optimization problem in FEniCS for parameter determination of metamaterials (Navid Shekarchizadeh and Alberto Maria Bersani) . . . . .	660
Total generalized variation for piecewise constant functions (Lukas Baumgärnter, Stephan Schmidt, Roland Herzog, Ronny Bergmann, and José Vidal-Núñez) . . . . .	680
Efficient Hessian computation in deterministic and Bayesian inverse problems (Daniel I Gendin and Paul Barbone) . . . . .	681
Dissipative materials models with Materiaux and FEniCS (Matthias Rambauser, Dipayan Mukherjee, and Konstantinos Danas) . . . . .	693
Reduced order methods for optimal flow control: FEniCS-based applications (Maria Strazzullo, Francesco Ballarin, and Gianluigi Rozza) . . . . .	694
A generic FEniCS-framework for moment approximations of Boltzmann’s equation (Edilbert Christhuraj and Manuel Torrilhon) . . . . .	710
Hybridized discontinuous Galerkin methods for the Stokes and Navier–Stokes equations in FEniCSx: non-simplex cells and curved geometries (Joseph P. Dean, Sander Rhebergen, Chris N. Richardson, and Garth N. Wells) . . . . .	722
Semismooth Newton method for Bingham flow (Alexei Gazca) . . . . .	742
Non-intrusive reduced order modeling of linear poroelasticity in heterogeneous porous media (Teeratrorn Kadeethum, Francesco Ballarin, and Nikolaos Bouklas) . . . . .	765
Consensus ADMM for inverse problems governed by multiple PDE models (Luke Lozenski and Umberto Villa) . . . . .	783
Automating the formulation and resolution of convex variational problems with the fenics_optim package (Jeremy Bleyer) . . . . .	806
Generating high-order time stepping methods (Robert Kirby, Jorge Marchena-Menendez, and Patrick Farrell) . . . . .	831

# Introduction

The FEniCS 2021 conference was held online in March 2021, using a combination of Zoom and Gather Town. This document collects the abstracts and slides for the talks presented at FEniCS 2021.

The following picture was taking during the Gather Town evening session on Thursday 25 March.



## Prizes

Three prizes were given at FEniCS 2021. These were awarded to:

- Best talk by a PhD student or undergraduate: Rémi Delaporte-Mathurin
- Best talk by a PhD student or undergraduate (runner up): India Marsden
- Best talk by a postdoc: Marc Hirschvogel

The prizes were kindly provided by Rafinex.

# Patient specific brain simulations with FEniCS and Freesurfer

Vegard Vinje, Simula Research Laboratory, Norway

22 March 2021

Patient-specific simulations of physical processes within the brain requires detailed grid construction from image data, typically MR-images. In this talk I will present the pipeline developed by our group on how to go from MRI images to finite element simulations. In many aspects of neuroscience there is particular focus on distinct regions within the brain. The new pipeline allows for efficient analysis of different brain regions and a more robust comparison with experimental results found in the literature.

Freesurfer is a software developed to identify the brain tissue found in the MR images as well as segmenting it into several different subregions. Each subregion will be assigned a unique number according to a look-up-table with the corresponding name of the region. This information can be translated into FEniCS Meshfunctions with the Python package SVMTK (Figure). With the Meshfunction, we may assign different physical parameters to different regions of the brain. In addition, analysis and visualization of different regions in post-processing may also be performed.

In this talk I show the general usage of this pipeline along with a few examples of relevant simulations for transport phenomena within the brain.

---

You can cite this talk as:

Vegard Vinje. "Patient specific brain simulations with FEniCS and Freesurfer". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 2. doi: 10.6084/m9.figshare.14494593.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/vinje.html>.

# A finite element model of electric fields in the brain

Vyassa Baratham, University of California, Berkeley, United States

22 March 2021

In recent years, simulation of neural tissue at subcellular resolution has become an important tool for understanding the electrophysiological signals that neuroscientists and physicians record from the brain. These simulations use precise chemistry and physics to generate true-to-life predictions of neural activity, but employ simplifying assumptions to propagate that activity from neuron to recording electrode, a problem which could be solved exactly using finite element modeling. In this talk, I will summarize the biophysics of neural simulation, present the widely-used simplified model of electric field propagation in the brain, and share my ideas for implementing a more exact model in FEniCS. I welcome any feedback or tips on my proposed approach. I believe that a freely available implementation of this problem in an open source FEM solver is an important next step in the development of neural simulators. We must understand in detail how electric fields propagate through neural tissue before we can develop neural interfaces capable of reading and stimulating the brain at high resolution.

---

You can cite this talk as:

Vyassa Baratham. "A finite element model of electric fields in the brain". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 3–17. doi: 10.6084/m9.figshare.14494656.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/baratham.html>.

# A finite element model of electric fields in the brain

Vyassa Baratham

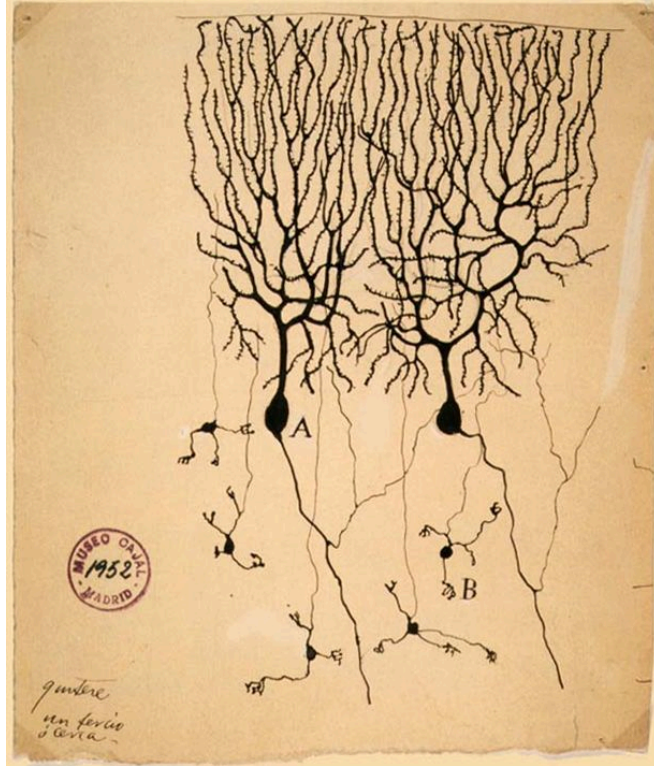
[vbaratham@berkeley.edu](mailto:vbaratham@berkeley.edu)

PhD Candidate  
University of California, Berkeley

22 Mar 2021  
FEniCS 2021

# The brain is composed of billions of neurons

Neurons are cells with branching extensions which reach out and connect to other neurons



# Neurons are electrically active

(sound on)

An electrode near/on a neuron *in vivo* will periodically see transient ( $\sim 1\text{ms}$ ) spikes in electrical potential

The rate of spikes usually depends on what the animal is doing, or seeing, or hearing, thinking, etc.

→  
Electrode attached to speaker

**Pattern drifts down/left:** few spikes

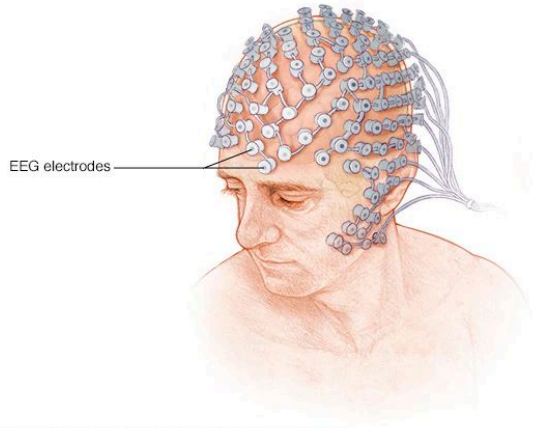
**Pattern drifts up/left:** lots of spikes



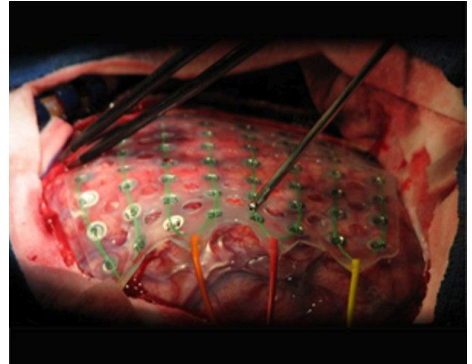
<https://www.youtube.com/watch?v=Qz40mdaDYTU>



# Many studies record from large populations of neurons



Electroencephalography (EEG)



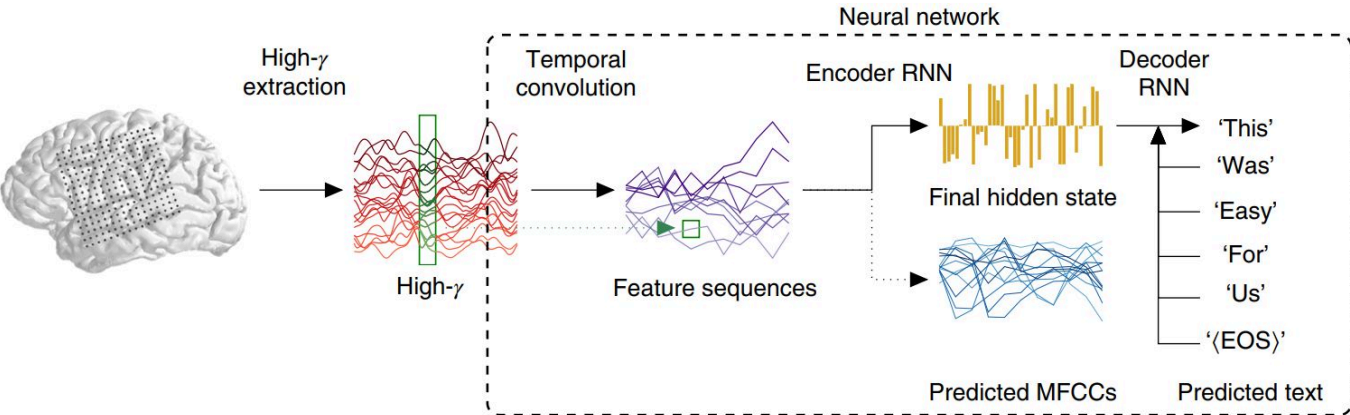
Electrocorticography (ECoG)

EEG and ECoG recordings reflect a superposition of the activity of  $\sim 10^{5 \pm 1}$  neurons  
*Population-level recordings sacrifice **resolution** in favor of **coverage***

# Population-level data contains detailed information

Example: Inferring speech from neural activity

Joseph G. Makin, David A. Moses, & Edward F. Chang (2020)  
<https://doi.org/10.1038/s41593-020-0608-8>



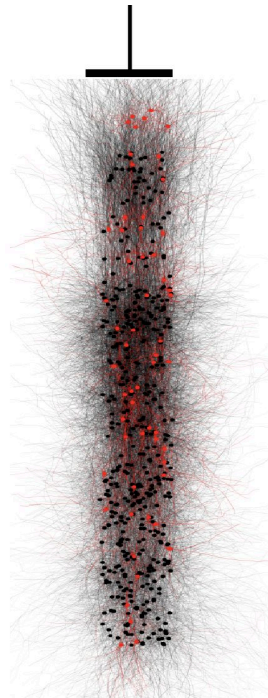
In order to understand exactly **how** this information is represented in the brain, we need to “invert” the population-level signal to reconstruct the activity of the underlying neuronal sources. **A detailed forward model may help.**

EEG and ECoG recordings reflect a superposition of signals from all nearby neurons

In order to understand exactly how information is represented in the brain, we need to “invert” the population-level signal to reconstruct the activity of the underlying neuronal sources.

**This inverse problem is ill-posed:** there may be different distributions of source activity which give rise to the same observed signal

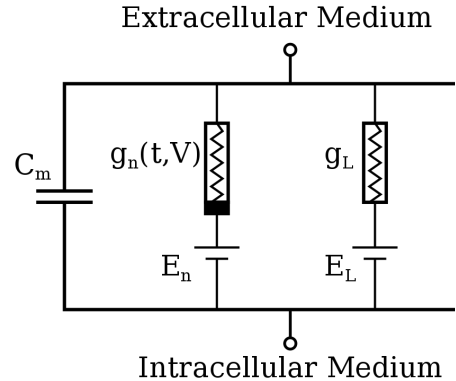
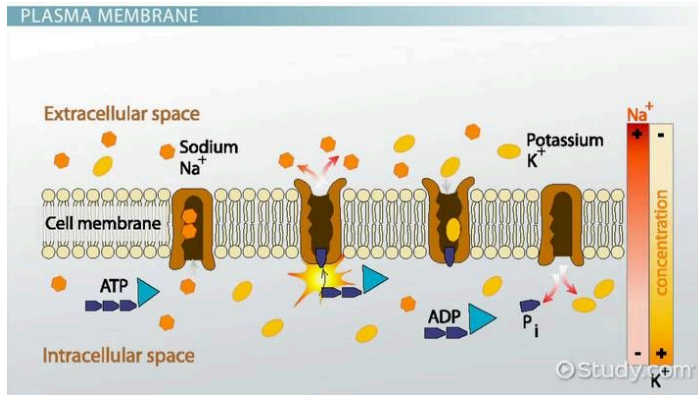
A detailed *forward model* may provide insight into which of these distributions is consistent with biology.



Simulating the brain

# Models of neural activity are precise, accurate

1. Hodgkin & Huxley (1952) show that cell membranes behave like electrical circuits



Lipid bilayer = capacitor

Ion channels = resistors\*

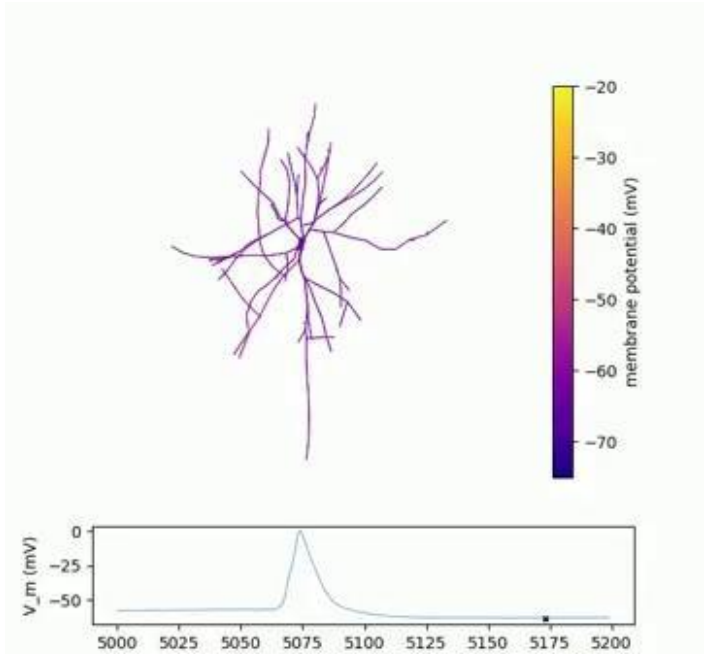
Electrochemical gradients = batteries

2. Cable equation describes spread of electrical potentials through neurons: 
$$\frac{\partial V}{\partial T} = \frac{\partial^2 V}{\partial X^2} - V$$

# Models of neural activity are precise, accurate

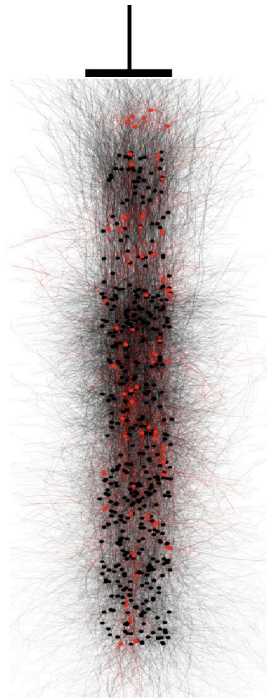
Computers can accurately simulate  
neural activity:

Color indicates difference in electrical potential  
between the inside and outside of the cell



We can simulate the  
*activity of neurons* in a  
chunk of brain.

Next: **what would an  
electrode near these  
neurons read?**



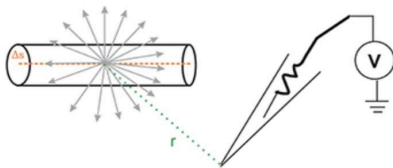
# Models of extracellular potential are drastically simplified

Given an electrical current  $I(t)$  through one segment of a neuronal membrane, what signal  $V(\mathbf{r}, t)$  would an external electrode located at point  $\mathbf{r}$  read?

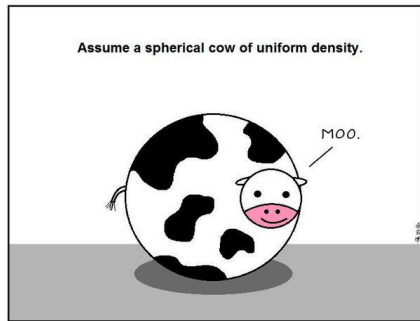
Assume extracellular space is:

- Homogeneous
- Isotropic
- Purely Ohmic (no capacitance)

Point Source Approximation



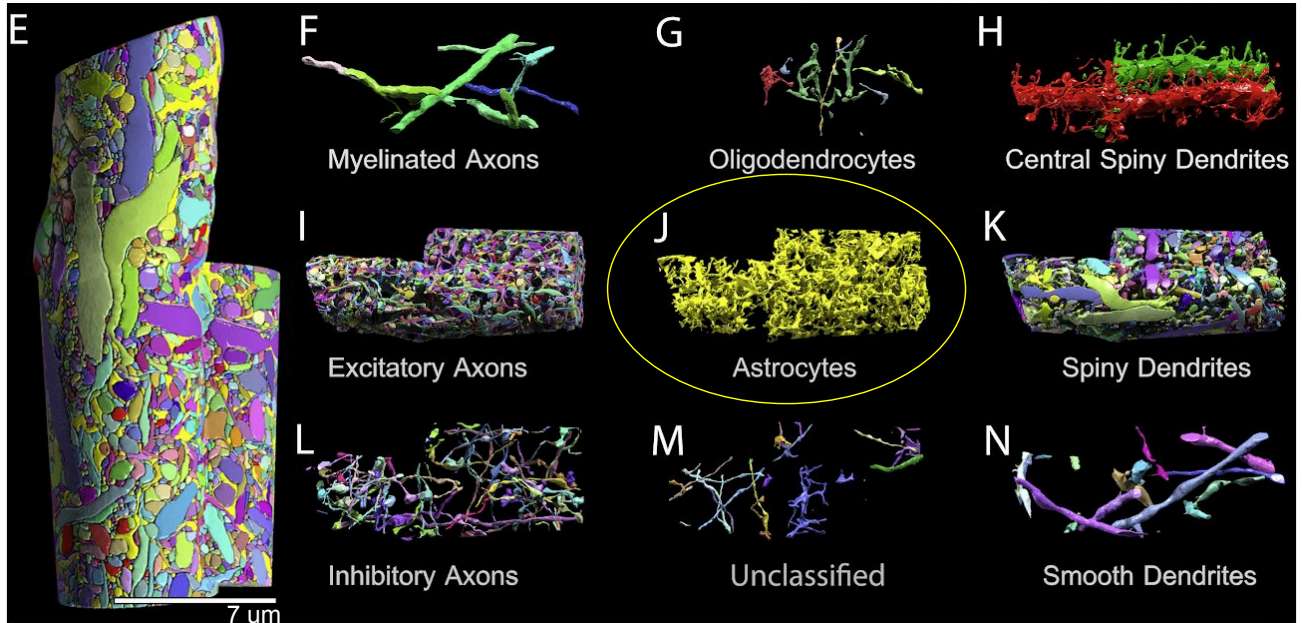
$$V(\mathbf{r}, t) = \frac{I(t)}{4\pi\sigma r}$$





# The extracellular medium is inhomogeneous

Kasthuri, Narayanan et al.  
Cell, Volume 162, Issue 3, 648 - 661



# FEM handles full complexity of extracellular space

Alessio Paolo Buccino et al 2019  
J. Neural Eng. 16 026030

$$\nabla \cdot \sigma_i \nabla u_i = 0 \quad \text{in } \Omega_i, \quad (1)$$

$$\nabla \cdot \sigma_e \nabla u_e = 0 \quad \text{in } \Omega_e, \quad (2)$$

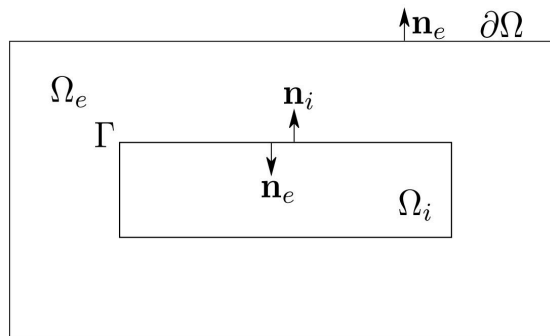
$$u_e = 0 \quad \text{at } \partial\Omega_e, \quad (3)$$

$$\sigma_e \nabla u_e \cdot n_e = 0 \quad \text{at } \partial\Omega_p, \quad (4)$$

$$n_e \cdot \sigma_e \nabla u_e = -n_i \cdot \sigma_i \nabla u_i \stackrel{\text{def}}{=} I_m \quad \text{at } \Gamma, \quad (5)$$

$$u_i - u_e = v \quad \text{at } \Gamma, \quad (6)$$

$$\frac{\partial v}{\partial t} = \frac{1}{C_m} (I_m - I_{\text{ion}}) \quad \text{at } \Gamma. \quad (7)$$



Interior and exterior of cells are separate, but coupled by the membrane (6)

(7) gives the time evolution of the membrane potential

# Overview

(Population recordings)

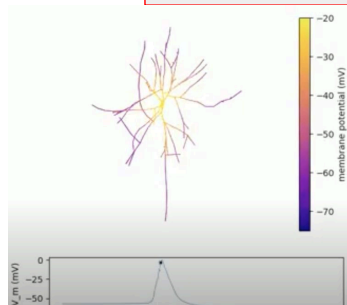
Electrical input  $\rightarrow$  Neural activity  $\rightarrow$  Extracellular potential  $\rightarrow$  “Useful” information

Hodgkin-Huxley  
membrane model,  
cable equation  
simulations

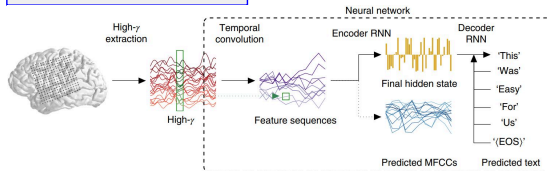
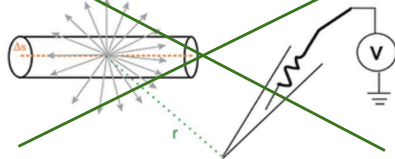
~~Point source~~  
~~approximation~~

FEM

Principal  
Components  
Analysis, Neural  
Networks, etc.



~~Point Source Approximation~~



# Astrophysical tests of gravity using FEniCS

Andrius Tamosiunas, University of Nottingham, United Kingdom

22 March 2021

Various theories of gravity are described by non-linear differential equations, which are difficult to solve analytically in all but a few simple cases. In this talk I will present the work done by myself and my colleagues at the University of Nottingham towards developing a FEniCS code for solving differential equations describing theories of gravity in laboratory and astrophysical environments.

---

You can cite this talk as:

Andrius Tamosiunas. “Astrophysical tests of gravity using FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 18–32. doi: 10.6084/m9.figshare.14494716.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/tamosiunas.html>.

# Astrophysical Tests of Gravity Using FEnICS

CAPT, Nottingham University

Andrius Tamošiūnas

[andrius.tamosiunas@nottingham.ac.uk](mailto:andrius.tamosiunas@nottingham.ac.uk)

In collaboration with:

Clare Burrage, Chad Briddon, Adam Moss



University of  
**Nottingham**  
UK | CHINA | MALAYSIA

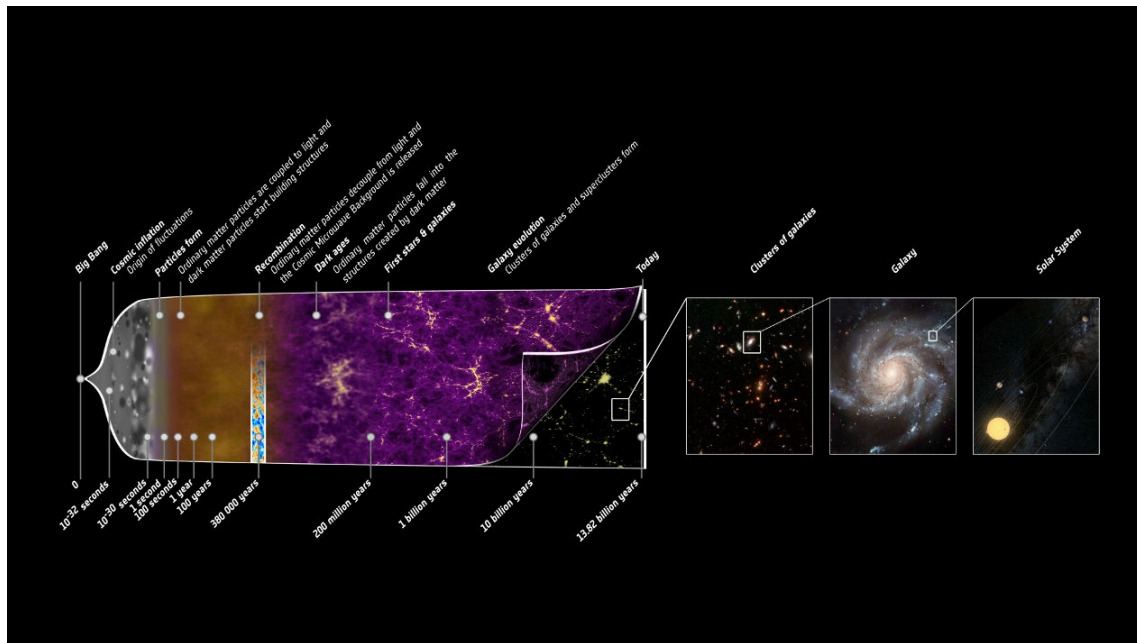


UNIVERSITY OF  
**PORTSMOUTH**

LEVERHULME  
TRUST

- 1 Introduction
  - 1.1 The Standard Model of Cosmology
  - 1.2 Theories of Modified Gravity
  - 1.3 The Chameleon Mechanism
  - 1.4 Fifth Forces in Cosmology and the Laboratory
  
- 2 Numerical Tests of Modified Gravity
  - 2.1 Numerical Methods using FEniCS
  - 2.2 FEniCS in Astrophysics and Cosmology
  - 2.3 Current Work: Realistic Galaxy Clusters
  
- 3 Conclusions and Future Plans

# The Context: The $\Lambda$ CDM Model



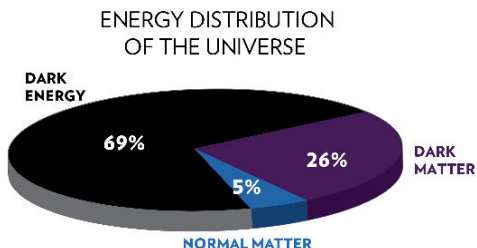
**Fig.:** The Universe according to the standard model of cosmology ([NASA](#)).

# The $\Lambda$ CDM Model

- 1 Gravity is described by the theory of **general relativity** (GR):

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu}. \quad (1)$$

- 2 The Universe is dominated by **dark matter** and **dark energy**.



## Key Questions:

- 1 Dark matter?
- 2 Dark energy?
- 3 Galaxy-scale problems?
- 4 The Hubble constant problem?
- 5 The theory of quantum gravity?

**Fig.:** The mass-energy content of the Universe  
([Chandra](#)).



# Possible Solution: Modified Gravity

- It is possible that some of these problems could be tackled by modifying Einstein's general relativity.
- General relativity can be modified in a variety ways:
  - ① Some of the assumptions can be relaxed ( $f(R)$  gravity);
  - ② Extra spacetime dimensions (Kaluza Klein);
  - ③ Non-local theories (Infinite derivative gravity);
  - ④ Extra (scalar, vector, tensor) fields.
- Modifying GR results in an extra fundamental force ("fifth" force).

# An Example: Chameleon Gravity

- An example of a scalar-tensor theory: Chameleon gravity;
- An extra scalar field interacting with matter in a special way is introduced;
- A "fifth" force arises from the interactions between the field and matter in a **density-dependent** way.

Key Equation to Solve:

$$\nabla^2 \phi = -\frac{n\Lambda^{n+4}}{\phi^{n+1}} + \frac{\beta\rho}{M_{\text{pl}}}, \quad (2)$$

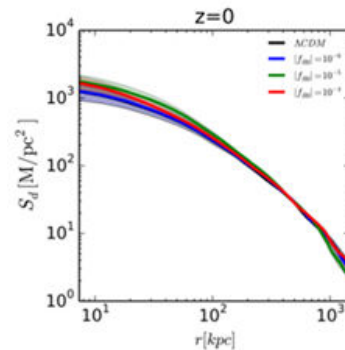
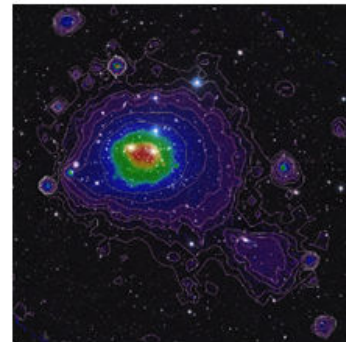
where:  $\phi$ - Chameleon field,  $\rho$ -density,  $n, \Lambda, \beta, M_{\text{pl}}$ -constants.

# Fifth Force in Cosmology

- A fifth force could potentially explain dark energy;
- A fifth would not be relevant in the Solar System;
- Modifying gravity would change the properties of galaxy clusters:



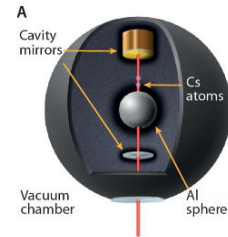
**Fig.:** Galaxy cluster Abel 2744 ([NASA](#)).



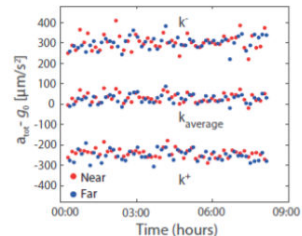
**Fig.:** Clusters in X-ray ([ESA](#)).

## Atom Interferometry Tests:

- Fifth forces could be potentially detected in laboratory tests;
- Recent tests have not found evidence for a fifth force;
- Such tests allow putting strong constraints on the Chameleon model parameters.



**Fig.:** A vacuum chamber experiment ([Hamilton et al. 2015](#)).



**Fig.:** Key results from the experiment ([Hamilton et al. 2015](#)).

# Solving the Chameleon Eq. in FEniCS

- Normalizing the eq.:  $\hat{\phi} = \phi/\phi_\infty$ ,  $\hat{\rho} = \rho/\rho_\infty$ ,

$$\nabla^2 \phi = -\frac{n\Lambda^{n+4}}{\phi^{n+1}} + \frac{\beta\rho}{M_{pl}} \rightarrow \alpha \hat{\nabla}^2 \hat{\phi} = -\hat{\phi}^{-(n+1)} + \hat{\rho}, \quad (3)$$

- Rewriting the eq. in the variational form:

$$\alpha \int_{\Omega} \hat{\nabla} \hat{\phi} \cdot \hat{\nabla} v_j dx = \int_{\Omega} \left( \hat{\phi}^{-(n+1)} - \hat{\rho} \right) v_j dx, \quad (4)$$

- Rewriting using Taylor expansion:

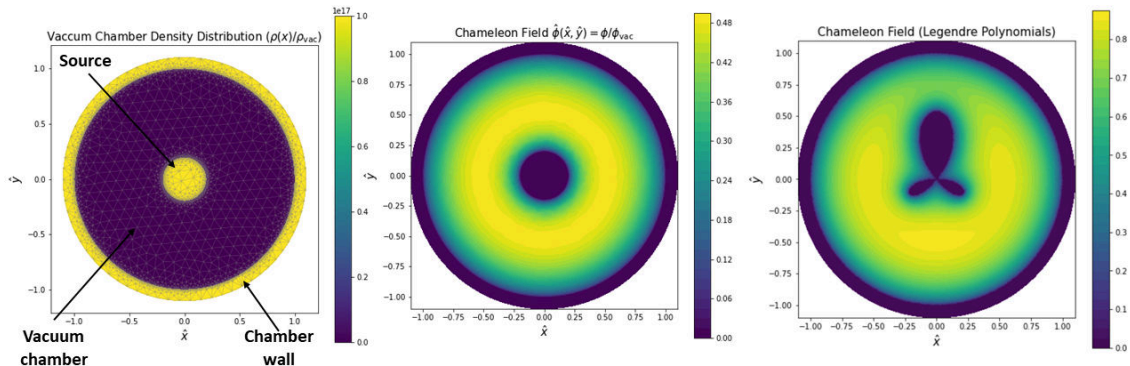
$$\hat{\phi}^{-(n+1)} \approx (n+2)\hat{\phi}_k^{-(n+1)} - (n+1)\hat{\phi}_k^{-(n+2)}\hat{\phi} + O\left(\hat{\phi} - \hat{\phi}_k\right)^2 \quad (5)$$

- The eq. is solved using the Newton/Picard methods:

$$\alpha \int_{\Omega} \hat{\nabla} \hat{\phi} \cdot \hat{\nabla} v_j dx + \int_{\Omega} (n+1)\hat{\phi}_k^{-(n+2)}\hat{\phi} v_j dx = \int_{\Omega} (n+2)\hat{\phi}_k^{-(n+1)} v_j dx - \int_{\Omega} \hat{\rho} v_j dx \quad (6)$$

# Numerical Solutions Using FEniCS

- Using FEniCS allows to simulate the mentioned vacuum chamber experiment;
- The mesh is generated and refined using Gmsh;
- Complex source shapes can be easily handled:

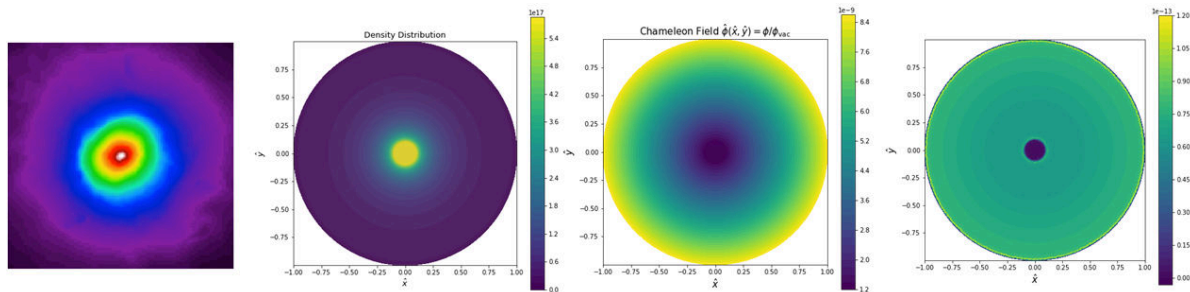


**Fig.:** A vacuum chamber simulation using FEniCS (Chad Briddon).

- Galaxy and galaxy cluster density distribution:

$$\rho_{\text{NFW}}(r) = \frac{\rho_0}{\frac{r}{R_s} \left(1 + \frac{r}{R_s}\right)^2}; \quad (7)$$

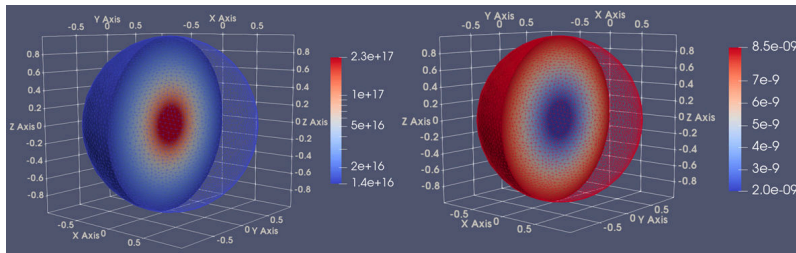
- Galaxy cluster Chameleon field simulation:



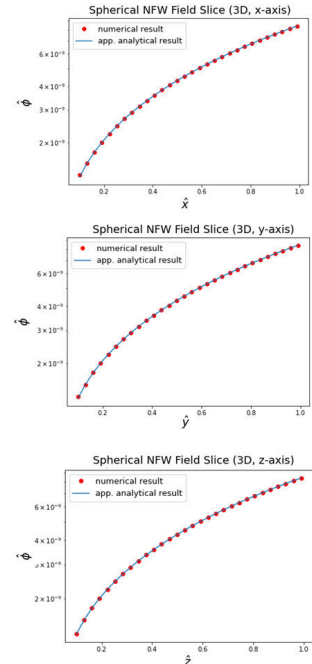
**Fig.:** **Left:** a typical galaxy cluster (X-ray); **center** the density distribution and the Chameleon field **right:** the field gradient.

# 3D Solutions

- An analogous method allows finding the 3D solutions;
- The typical size of the residuals:  
 $\sim 0.1 - 1 \%$ ;
- Finding the solution typically takes 10-40 min depending on the model parameters.



**Fig.:** Left 3D NFW distribution; right: Chameleon field solution.

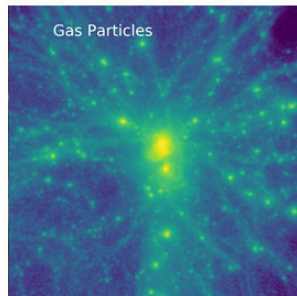
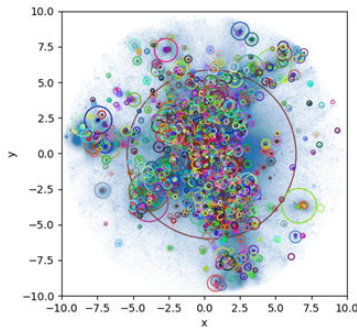


**Fig.:** Comparing the numerical and analytic solutions.



# Realistic Galaxy Clusters

- Realistic galaxy clusters based on simulations:



**Fig.:** Cluster density distribution ([Project 300 simulation](#)).

## Key Questions

- 1 Optimal cluster form?
- 2 Optimal galaxy form?
- 3 Time-dependent density distributions?

# Summary and Future Plans

## The long-term goal:

An open-source FEniCS code for modelling experimental and observational tests for a variety of gravity models.

## Key problems:

- Other gravity theories: Symmetron Gravity,  $f(R)$ , Vainshtein screening models;
- Other cosmology/astrophysics questions: stars, galaxies, cosmic voids;
- Other laboratory setups: spinning sources, multiple sources, more complicated vacuum chambers.

# A study of the time independent grade two model in a 2D contraction rheometer

Måns Andersson, KTH Royal Institute of Technology, Sweden

Ridgway Scott, University of Chicago, United States

Johan Jansson, KTH Royal Institute of Technology, Sweden

22 March 2021

Non-Newtonian fluids are found in many fields of science: food, medicine, engineering, etc, properties of these fluids are studied with rheometers. Not all rheometers can properly describe each model and we aim to evaluate if a contraction rheometer can distinguish the model parameters of the Grade two model.

This is first done in 2D with a simplified well studied algorithm and then with a general method proposed in [1]. Both methods are based on transforming the problem into a coupling between a Stokes-like system and a transport equation.

Incompressibility is enforced by penalty iteration and Scott–Vogelius elements since it is hypothesised that divergence free elements are advantageous when solving the coupled transport equation. The transport is solved with linear polynomials with different stabilizations investigated. We discuss preliminary results and limitations.

This non-Newtonian modeling is part of a larger predictive Real Unified Continuum framework for modeling full systems of eg swallowing (part of the Swallow project), plant-based food production, blood flow, gastrointestinal modeling, etc.

## References

- [1] Nadir Arada, Paulo Correia, and Adélia Sequeira. “Analysis and finite element simulations of a second-order fluid model in a bounded domain”. In: *Numerical Methods for Partial Differential Equations: An International Journal* 23.6 (2007), 1468–1500. DOI: 10.1002/num.20236.

---

You can cite this talk as:

Måns Andersson, Ridgway Scott, and Johan Jansson. “A study of the time independent grade two model in a 2D contraction rheometer”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 33. DOI: 10.6084/m9.figshare.14494791.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/andersson.html>.

# Multiscale-in-time modeling of myocardial growth & disease progression

Marc Hirschvogel (<http://www.deepambit.com>), King's College London, United Kingdom

22 March 2021

A multiscale-in-time framework for simulation of maladaptive growth and remodeling (G&R) in the heart is presented. G&R is assumed to be driven by a deviation of mechanical stress or strain with respect to a homeostatic baseline state. Since ventricular loads vary on a much shorter time scale than processes of G&R occur, a staggered solution scheme discriminating between “small scale” heart beat dynamics and “large scale” G&R is chosen.

On the small scale, a coupled monolithic problem of 3D finite strain elasticity for the heart and 0D lumped-parameter flow is solved, using a closed-loop systemic and pulmonary circulation model to account for physiologic loading conditions on the myocardium. After computing a homeostatic reference state, the system is perturbed by introducing a cardiovascular disease (ie regurgitation of the mitral valve or aortic stenosis), eventually leading to a state of chronic volume or pressure overload for the ventricle.

On the large scale, the spatial field of fiber overstretch or tissue overstress is then imposed, and a pure solid mechanics problem of strain- or stress-mediated volumetric growth is solved together with a remodeling law that allows for change in elastic material parameters depending on the amount of growth.

Small and large time scales are mutually revisited until no further volume change occurs. Physiologically meaningful changes in ventricular pressure-volume relationship are obtained for ventricular volume and pressure overload and comply with general observations.

Nonlinear deformation-dependent growth requires local Newton updates at integration point level and is implemented in FEniCS by expressing growth residual and increments as forms at quadrature points. Inner virtual work is expressed explicitly with help of the fourth-order material tangent operator to account for all tangent contributions arising from the nonlinear G&R model.

This talk was awarded a prize: Best talk by a postdoc (runner up).

---

You can cite this talk as:

Marc Hirschvogel. “Multiscale-in-time modeling of myocardial growth & disease progression”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 34–50. DOI: 10.6084/m9.figshare.14494809.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/hirschvogel.html>.

# Multiscale-in-Time Modeling of Myocardial Growth & Disease Progression

FEniCS conference

**Marc Hirschvogel**  
King's College London

22 Mar 2021

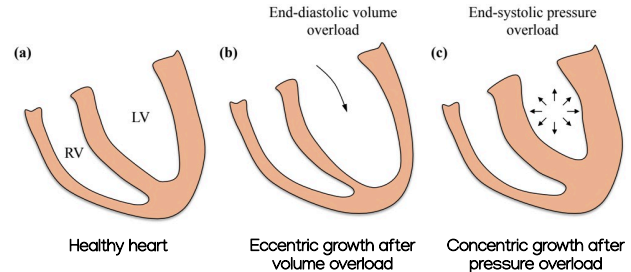
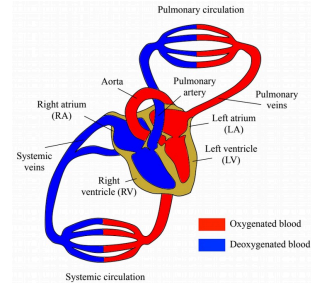
# Outline

---

- 1 Introduction & Motivation
- 2 Modeling the Heart and Disease Mechanisms
- 3 Numerical Implementation using FEniCS-X
- 4 Results
- 5 Summary & Outlook

## 1.1 Heart Models and the Cardiac Cycle

- **Cardiovascular disease entities** most prevalent in industrialized world [Dimmeler 2011, Luepker 2011]
- Diseases of the myocardium (heart muscle) are multifactorial and yet to be fully understood
  - Altered **mechanical loads**
  - **Neurohormonal changes**
- Heart may undergo **adaptations in structure and shape** if loading conditions are chronically above a certain physiological level, referred to as **Growth and Remodeling** (G & R) [Rossi et al. 1991]
- **Volume overload** (Fig. (b)):
  - Heart adapts by **eccentric growth** (systolic heart failure)
- **Pressure overload** (Fig. (c)):
  - Heart adapts by **concentric growth** (diastolic heart failure)



# Outline

---

- 1 Introduction & Motivation
- 2 Modeling the Heart and Disease Mechanisms
- 3 Numerical Implementation using FEniCS-X
- 4 Results
- 5 Summary & Outlook



## 2.1 Patient-specific Geometric 3D-0D Heart Model

- Heart muscle:** Nonlinear nearly-incompressible hyperelastic, anisotropic solid [Guccione et al. 1991]

$$S = 2 \frac{\partial \Psi}{\partial C} + \tau_a(t) f_0 \otimes f_0 \quad \Psi = \frac{C_0}{2} e^Q + \frac{\kappa}{2} (J - 1)^2 \quad Q = b_f E_{ff}^2 + b_t (E_{ss}^2 + E_{nn}^2 + 2E_{sn}^2) + b_{fs} (2E_{fs}^2 + 2E_{fn}^2)$$

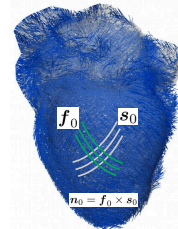
- Contraction:** Time- and fiber stretch-dependent active stress law [Bestel et al. 2001]

$$\dot{\tau}_a = -|u| \tau_a + a \sigma_0 |u|_+$$

$$\dot{a}(\lambda_{myo}) = \dot{g}(\lambda_{myo}) \mathbb{I}_{|u|_- > 0}$$

$$u = \hat{f}(t) \cdot \alpha_{\max} + (1 - \hat{f}(t)) \cdot \alpha_{\min}$$

Frank-Starling mechanism [Solaro 2007]



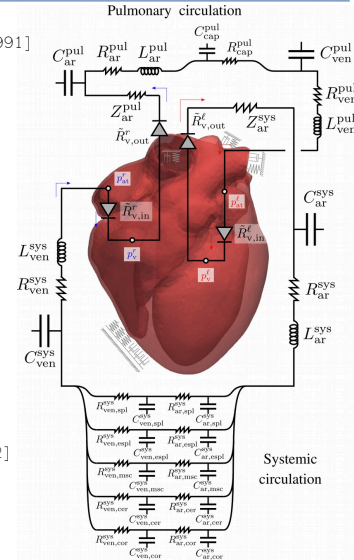
Rule-based fiber directions

Transmural variation ( $-60^\circ, 60^\circ$ )

[Doste et al. 2019, Bayer et al. 2012]

$$\lambda_{myo} = \sqrt{f_0 \cdot C f_0}$$

- Circulatory system** is modeled with a lumped-parameter 0D flow model (compliances, resistances inertances) [Hirschvogel et al. 2017, Trenhago et al. 2016, Ursino and Magosso 2000a,b]



Free heart STL geometry from <https://www.icmm.ru/tomogram-to-fem>

## 2.2 Continuum Mechanical Modeling of G&R

- G&R computed in a **kinematic growth framework** with multiplicative split of deformation gradient into elastic and inelastic (growth) part  
[Lee et al. 1969, Rodriguez et al. 1994]

$$\mathbf{F} = \mathbf{F}^e \mathbf{F}^g$$

- Growth deformation gradient is function of growth stretch  $\vartheta$  and possibly of preferred directions  $\mathbf{f}_0$  :

$$\mathbf{F}^g = f(\vartheta, \mathbf{f}_0, \dots)$$

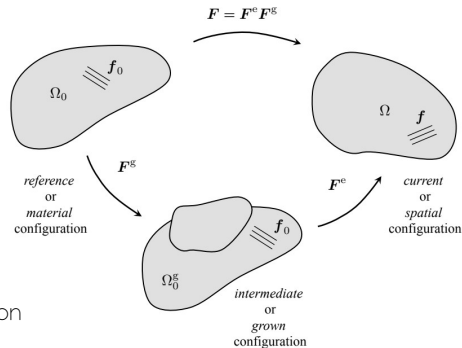
- Growth stretch usually is governed by an evolution equation and can depend on mechanical or other stimuli:

$$\dot{\vartheta} = f(\vartheta, \mathbf{C}, \mathbf{S}, \mathbf{f}_0, \dots)$$

- Remodeling** is taken into account by additively decomposing the stress response into a part governing the reference and one describing the remodeled material (similar to [Thon et al. 2018]):

$$\mathbf{S} = \phi(\vartheta) \mathbf{S}_{(\text{remod})} + (1 - \phi(\vartheta)) \mathbf{S}_{(\text{base})}$$

$\phi(\vartheta)$  : Fraction of grown material



# Outline

---

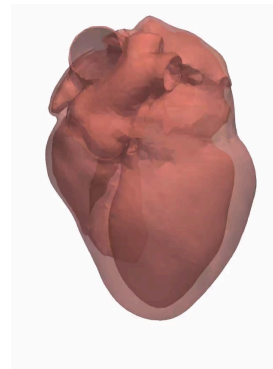
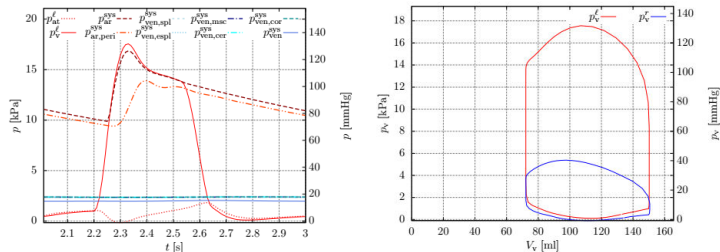
- 1 Introduction & Motivation
- 2 Modeling the Heart and Disease Mechanisms
- 3 Numerical Implementation using FEniCS-X
- 4 Results
- 5 Summary & Outlook

## 3.1 3D-0D Coupled Elastodynamics

- **Nonlinear elastodynamics** using Generalized-alpha time integration [Chung and Hulbert 1993]
- Strongly coupled 3D-0D monolithic solution of **solid mechanics** and **lumped flow models** [Hirschvogel et al. 2017]
- Use of direct solver (SuperLU) or block-pre-conditioned GMRES [Elman et al. 2008]
- ~90'000 linear displacement-based tetrahedral elements, ~60'000 unknowns

$$\begin{bmatrix} \mathbf{K}^{uu} & \mathbf{K}^{us} \\ \mathbf{K}^{su} & \mathbf{K}^{ss} \end{bmatrix}_{n+1} \begin{bmatrix} \Delta \mathbf{d} \\ \Delta \mathbf{s} \end{bmatrix}_{n+1}^{i+1} = - \begin{bmatrix} \mathbf{r}^u \\ \mathbf{r}^s \end{bmatrix}_{n+1}^i$$

➤ Example healthy heart cycle simulation:



Open-source Python FEniCS-based solver for cardiac mechanics  
<https://github.com/marchirschvogel/ambit>

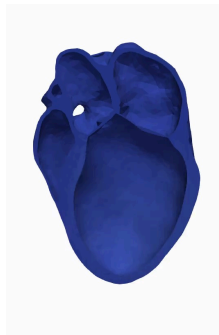
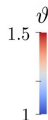
## 3.2 Inelastic Deformation-Dependent Growth & Remodeling

- Fiber stretch-driven **anisotropic growth** in fiber direction

$$\mathbf{F}^g = \mathbf{1} + (\vartheta - 1)\mathbf{f}_0 \otimes \mathbf{f}_0 \quad \dot{\vartheta} = k(\vartheta) \left( \lambda_{\text{myo}}^e - \hat{\lambda}_{\text{myo}}^{\text{crit}} \right)$$

$$k(\vartheta) = \begin{cases} \frac{1}{\tau} \left( \frac{\vartheta_{\text{max}} - \vartheta}{\vartheta_{\text{max}} - \vartheta_{\text{min}}} \right)^\gamma, & \lambda_{\text{myo}}^e \geq \hat{\lambda}_{\text{myo}}^{\text{crit}}, \\ \frac{1}{\tau_{\text{rev}}} \left( \frac{\vartheta - \vartheta_{\text{min}}}{\vartheta_{\text{max}} - \vartheta_{\text{min}}} \right)^{\gamma_{\text{rev}}}, & \lambda_{\text{myo}}^e < \hat{\lambda}_{\text{myo}}^{\text{crit}}, \end{cases}$$

$$\lambda_{\text{myo}}^e = \frac{1}{\vartheta} \lambda_{\text{myo}} = \frac{1}{\vartheta} \sqrt{\mathbf{f}_0 \cdot \mathbf{C} \mathbf{f}_0}$$



- Stress in inner virtual work depending on deformation and internal variable  $\vartheta$ , which is deformation-dependent itself in a nonlinear way (needs local Newton to solve)

$$\delta \mathcal{W}_{\text{int}} = \int_{\Omega_0} \mathbf{S}(\mathbf{C}(\mathbf{u}), \vartheta(\mathbf{C}(\mathbf{u}))) : \frac{1}{2} \delta \mathbf{C} \, dV$$

- Full material tangent operator reads:  $\mathbb{C} = 2 \frac{\partial \mathbf{S}}{\partial \mathbf{C}} + 2 \left( \frac{\partial \mathbf{S}}{\partial \mathbf{F}^g} : \frac{\partial \mathbf{F}^g}{\partial \vartheta} \right) \otimes \frac{\partial \vartheta}{\partial \mathbf{C}}$

➤ FEniCS UFL can only take care of first term, since no analytic expression  $\vartheta(\mathbf{C}) = \dots$  possible

➤ Express virtual work linearization directly as form without using “derivative” and add second term manually to  $\mathbb{C}$

$$D_{\Delta \mathbf{u}} \delta \mathcal{W}_{\text{int}} = \int_{\Omega_0} \left( \text{Grad} \delta \mathbf{u} : \text{Grad} \Delta \mathbf{u} \, \mathbf{S} + \mathbf{F}^T \text{Grad} \delta \mathbf{u} : \mathbb{C} : \mathbf{F}^T \text{Grad} \Delta \mathbf{u} \right) dV$$

$$\frac{\partial \mathbf{S}}{\partial \mathbf{F}^g} = - \left( \mathbf{F}^{g^{-1}} \underline{\otimes} \mathbf{S} + \mathbf{S} \underline{\otimes} \mathbf{F}^{g^{-1}} \right) - \left( \mathbf{F}^{g^{-1}} \underline{\otimes} \mathbf{F}^{g^{-1}} \right) : \frac{1}{2} \check{\mathbb{C}} : \left( \mathbf{F}^{g^{-T}} \underline{\otimes} \mathbf{C}^e + \mathbf{C}^e \underline{\otimes} \mathbf{F}^{g^{-T}} \right)$$

Elastic part of  $2 \frac{\partial \mathbf{S}}{\partial \mathbf{C}}$

➤ Depending on growth law, can render excessive FFC-X compilation times! (between 5 and 30 minutes!)

### 3.3 Multiscale-in-Time Analysis: Volume Overload and Eccentric Growth in the Heart

- Homeostatic healthy heart beat computation
- Acute disease state (e.g. mitral valve regurgitation) computation, evaluation of **end-diastolic volume overload**

- Set state “large time scale”:

$$\mathbf{u}^{(\mathcal{L})} \leftarrow \mathbf{u}(t_{\text{ed}})^{(S)} \quad \hat{p}_c^{i,(\mathcal{L})} \leftarrow p_c^{i,(S)}(t_{\text{ed}}) \quad \hat{\tau}_a^{(\mathcal{L})} \leftarrow \tau_a(t_{\text{ed}})^{(S)}$$

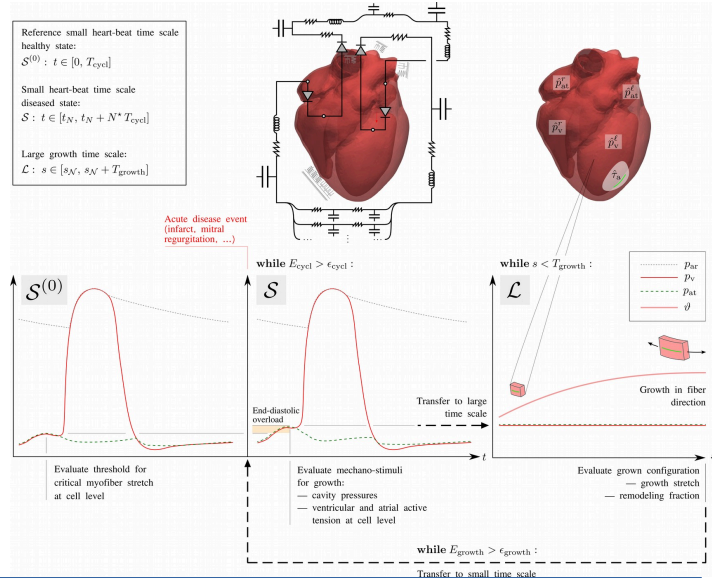
➤ Quasi-static growth computation

- Set state “small time scale”:

$$\hat{v}^{(S)} \leftarrow \hat{v}^{(\mathcal{L})} \quad \mathbf{u}^{(S)} \leftarrow \mathbf{u}^{(\mathcal{L})} - \mathbf{u}(t_{\text{ed}})^{(S)}$$

➤ Compute new homeostatic heart beat state

- Mutually revisit small and large scale until growth falls below a certain tolerance



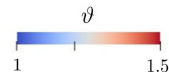
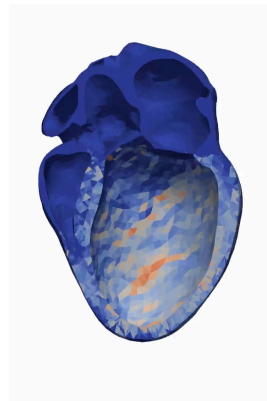
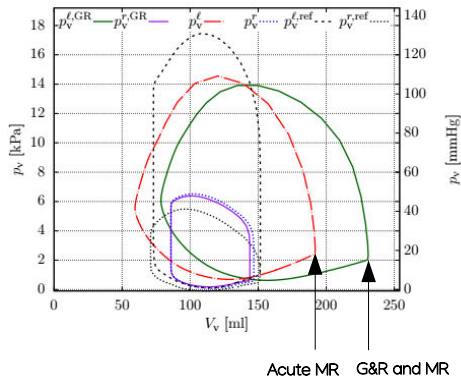
# Outline

---

- 1 Introduction & Motivation
- 2 Modeling the Heart and Disease Mechanisms
- 3 Numerical Implementation using FEniCS-X
- 4 Results
- 5 Summary & Outlook

## 4.1 Eccentric Growth in the Heart: Results for Mitral Regurgitation (MR)

- G&R after mitral valve regurgitation
  - Loss of isovolumetric contraction phases
  - **Right-shift** of pressure-volume relationship
  - LV wall thinning
- “Heart failure with reduced ejection fraction”



- **Remodeling:** Assumption that only active material is reduced with growth (cardiomyocytes are elongated, degradation and disruption of fibrillar collagen, impaired contractility [Aurigemma et al. 2006]):

$$S_{(\text{remod})} = 2 \frac{\partial \Psi}{\partial C}$$



# Outline

---

- 1 Introduction & Motivation
- 2 Modeling the Heart and Disease Mechanisms
- 3 Numerical Implementation using FEniCS-X
- 4 Results
- 5 Summary & Outlook

## 5 Summary & Outlook

---

- **Multiphysics and multiscale approach** to cardiac growth and remodeling using FeniCS-X
  - 3D-0D coupled nonlinear **elastodynamics** and **reduced-dimensional flow**
  - **Inelastic deformation-dependent growth** solved at integration point level
- **Physiological results and growth patterns**, but ...
  - Need of fine-tuning to match experimental data
- Need for **higher-order spatial approximation** to avoid spurious effects of low-order finite elements, but ...
  - **Missing Quadrature function spaces in FEniCS-Xi** For linear elements with one integration point (CG1), growth material is specified as discontinuous DG0 function space
  - No quadratic convergence for growth material living on DG1 space for higher-order mesh (CG2)
- Need for strategies of **reducing FFC-X compiler times** for complex constitutive UFL expressions

Thank you for your attention!

## References

- J. D. Bayer, R. C. Blake, et al. In: Ann Biomed Eng. Vol. 40. 10. 2012, pp. 2243–2254.
- J. Bestel, F. Clément, et al. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI'01). Vol. 2208. Berlin: Springer-Verlag, 2001, pp. 1159–1161.
- J. Chung and G. M. Hulbert. In: Journal of Applied Mechanics 60.2 (1993), pp. 371–375.
- S. Dimmeler. In: EMBO Mol Med 3.12 (2011), p. 697.
- R. Doste, D. Soto-Iglesias, et al. In: Int J Numer Method Biomed Eng 35.4 (2019), e3185.
- H. Elman, V. E. Howle, et al. In: Journal of Computational Physics 227 (2008), pp. 1790–1808.
- J. M. Guccione, A. D. McCulloch, et al. In: J Biomech Eng 113.1 (1991), pp. 42–55.
- M. Hirschvogel, M. Bassilious, et al. In: Int J Numer Method Biomed Eng 33.8 (2017), e2842.
- E. H. Lee. In: J Appl Mech 36.1 (1969), pp. 1–6.
- R. V. Luepker. In: Annual Review of Public Health 32.12 (2011), pp. 1–3.
- E. K. Rodriguez, A. Hoger, et al. In: J Biomech 27.4 (1994), pp. 455–467.
- M. A. Rossi and S. V. Carillo. In: Int J Cardiol 31.2 (1991), pp. 133–141.
- R. J. Solaro. In: Biophys J 93.12 (2007), pp. 4095–4096.
- M. P. Thon, A. Hemmler, et al. In: Biomech Model Mechanobiol 17.3 (2018), pp. 617–644.
- P. R. Trenhago, L. G. Fernandes, et al. In: Int J Numer Method Biomed Eng 32.1 (2016).
- M. Ursino and E. Magosso. In: Am J Physiol Heart Circ Physiol 279.1 (2000), pp. 149–165.
- M. Ursino and E. Magosso. In: Am J Physiol Heart Circ Physiol 279.1 (2000), pp. 166–175.

# External operators in UFL and Firedrake

**Nacime Bouziani** (<https://www.imperial.ac.uk/people/n.bouziani18>), Imperial College London, United Kingdom

**David Ham** (<https://www.imperial.ac.uk/people/david.ham>), Imperial College London, United Kingdom

**22 March 2021**

High level domain specific languages based on the Unified Form Language (UFL) such as FEniCS or Firedrake enable one to write down PDE-based problems in a very productive way. UFL equips FEniCS and Firedrake with a highly expressive interface to specify the variational forms and discrete function spaces, providing the abstractions needed for code generation. However, one of the limitations of UFL is that it does not take into account operators that are not directly expressible in the vector calculus sense. In a nutshell, the UFL abstraction is not rich enough to encompass these operators. We refer to these operators as external operators.

This limitation is critical in many applications where PDEs are not enough to accurately describe the physical problem of interest. These applications include nonlinear implicit constitutive laws such as the Glen's flow law for glacier flow, the use of neural networks to include features not represented in the differential equations, or closures for unresolved spatiotemporal scales. Example applications of neural networks include regularization of inverse problems such as in seismic inversion and subgrid parameterization of atmospheric or oceanographic processes like clouds or turbulence.

We present extensions to the Unified Form Language (UFL) and Firedrake that enable the inclusion of arbitrary external operators. This external operator feature composes seamlessly with the automatic differentiation capabilities of Firedrake.

---

You can cite this talk as:

Nacime Bouziani and David Ham. "External operators in UFL and Firedrake". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 51–67. DOI: 10.6084/m9.figshare.14495187.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/bouziani.html>.

## External operators in UFL and Firedrake

---

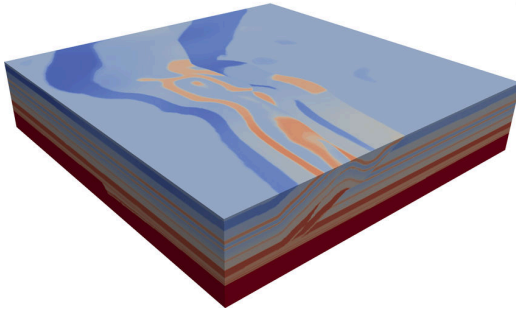
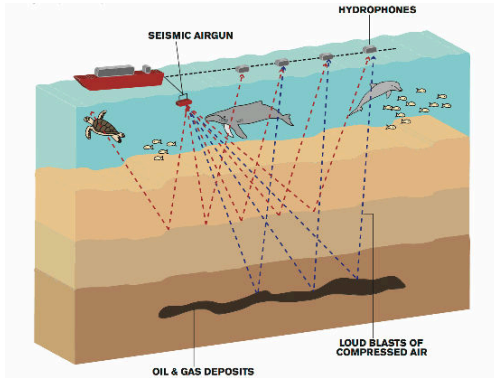
Nacime Bouziani    David A. Ham

March 22-26, 2021

**FEniCS 2021**

Department of Mathematics, Imperial College London

# PDEs are not enough in many cases !



We often need terms not directly expressible in the vector calculus sense !



Let's consider the following standard model for isothermal flow where we have to find  $(u, p, \tau) \in V \times Q \times X$  with appropriate function spaces such that  $\forall (w, \phi) \in V \times Q$  we have :

$$\left\{ \begin{array}{ll} \int \phi \nabla \cdot u = 0 & \text{incompressibility} \\ \int -w \cdot \nabla p + (\nabla \cdot \tau_{i,j}) \cdot w - f \cdot w = 0 & \text{stress balance} \\ \frac{1}{2} (\nabla u + \nabla u^T) = A |\tau|^2 \tau_{i,j} & \text{Glen flow law} \end{array} \right. \quad (1)$$

where  $f = \begin{pmatrix} 0 \\ -\rho g \end{pmatrix}$  refers to the gravity force and  $A \in \mathbb{R}$ .

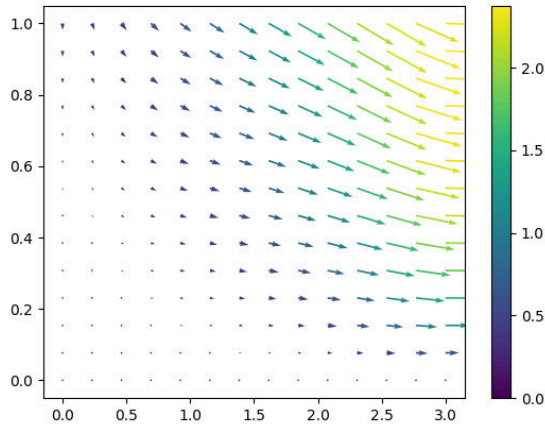


# A Firedrake example: Pointwise solve operator

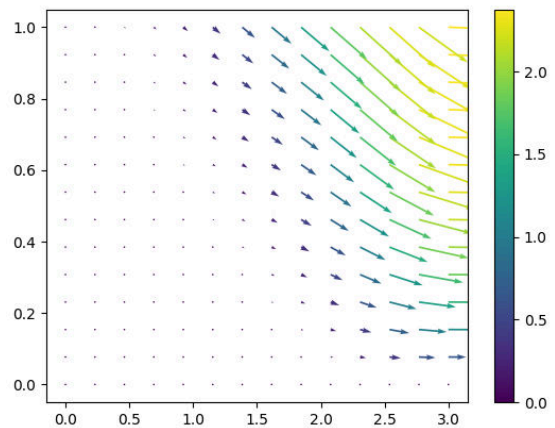


```
1 import sympy as sp
2 ...
3 # Define the function spaces
4 V1 = VectorFunctionSpace(mesh, "CG", 2)
5 V2 = FunctionSpace(mesh, "CG", 1)
6 V3 = TensorFunctionSpace(mesh, "DG", 2)
7 ...
8 # Mixed function space
9 W = MixedFunctionSpace((V1, V2))
10 w, phi = TestFunctions(W)
11 soln = Function(W)
12 u, p = split(soln)
13 ...
14 A = Constant(1.)
15 f = Function(V1).interpolate(as_vector([0, -rho*g]))
16 ...
17
18 ps = point_solve(lambda tau, eps, A: A*sp.Matrix(tau)*sp.Matrix(tau).norm()**2 - eps,
19                 function_space=V3,
20                 solver_params={'x0':u0, 'maxiter':25, 'tol':1.e-7})
21 tau = ps(sym(grad(u)), A)
22 F = div(w)*p*dx - inner(grad(w), tau)*dx - phi*div(u)*dx - inner(f,w)*dx
23 ...
24 # Solve
25 solve(F==0, soln, bcs=...)
```

# Glen's flow law: velocity field



$$\tau = \mathcal{E}(u)$$



$$A|\tau|^2 \tau_{ij} = \mathcal{E}(u)$$

$$\text{with } \mathcal{E}(u) = \frac{1}{2} \left( \nabla u + \nabla u^T \right)$$



Let's consider the function space  $V$  and the parameter space  $M$ , which can be a function space or a subspace of  $\mathbb{R}^m$  for  $m \in \mathbb{N}$  depending on the applications. We introduce the so-called *external operator*

$$N : V \times M \rightarrow X \quad (2)$$

where  $X$  is the *external operator space*, it is a function space.  $N$  is external in the sense that **it can be defined externally with respect to Firedrake**.



Let  $N$  be an ExternalOperator,

$$F(u, m, N(u, m); v) = 0 \quad \forall v \in V'$$

## Assembly steps

```
...  
 $u_h^X, m_h^X = \text{interpolate}(u_h, X), \text{interpolate}(m_h, X)$   
...  
 $\hat{N} = N(u_h^X, m_h^X). \text{assemble}()$   
...  
 $\text{assemble}(F(u_h, m_h, \hat{N}; v_h))$ 
```



Let  $N$  be an ExternalOperator,

$$F(u, m, N(u, m); v) = 0 \quad \forall v \in V'$$

## Assembly steps

```
...  
 $u_h^X, m_h^X = \text{interpolate}(u_h, X), \text{interpolate}(m_h, X)$   
...  
 $\hat{N} = N(u_h^X, m_h^X). \text{assemble}()$   
...  
 $\text{assemble}(F(u_h, m_h, \hat{N}; v_h))$ 
```

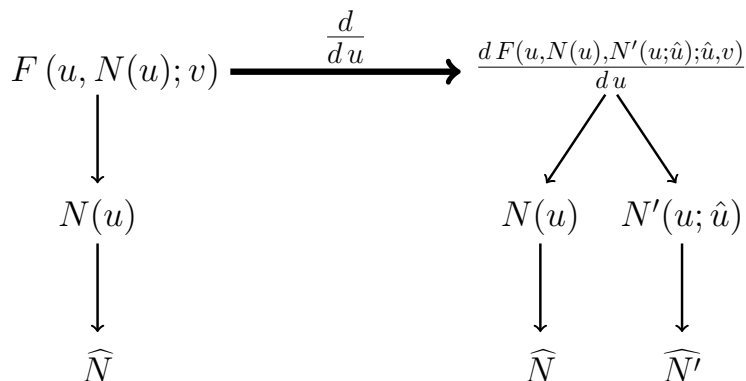
$\hat{N}$  gets evaluated inside the operation of evaluating  $F(u_h, m_h, \hat{N}; v_h)$  !



- Need to compute  $\frac{dF}{du}$  , we have:  $\frac{dF}{du} = \frac{\partial F}{\partial u} + \frac{\partial F}{\partial N} \frac{\partial N}{\partial u}$
- Need to extend UFL to handle :  $\frac{\partial F}{\partial N}$  and  $\frac{\partial N}{\partial u}$
- The external operator subclass is responsible for computing  $\frac{\partial N}{\partial u}$ :  
SymPy, UFL, PyTorch...



- Need to compute  $\frac{dF}{du}$ , we have:  $\frac{dF}{du} = \frac{\partial F}{\partial u} + \frac{\partial F}{\partial N} \frac{\partial N}{\partial u}$
- Need to extend UFL to handle :  $\frac{\partial F}{\partial N}$  and  $\frac{\partial N}{\partial u}$
- The external operator subclass is responsible for computing  $\frac{\partial N}{\partial u}$ :  
SymPy, UFL, PyTorch...



# New Firedrake subclasses of ExternalOperator :



1. *Pointwise solve operator* : An operator that handles pointwise nonlinear relationships. More precisely, the pointwise solve operator is applied on a given UFL expression and provides the root of the function(al) defined by this expression.
2. *Neural Network* : The neural network operator takes an input and returns the output of the associated neural network model.
3. *Layer potentials* : This single (resp. double) layer potential operator computes the single (resp. double) layer potential (see Nonlocal UFL's talk (B. Sepanski) Thursday - 15:00–16:30 GMT ).





Let  $N_1$  and  $N_2$  be two external operators,

$$\min_{m \in M} J(u, m, N_1(u, m)) \quad (3)$$

$$\text{subject to } F(u, m, N_2(u, m); v) = 0 \quad \forall v \in V' \quad (4)$$

where  $J : V \times M \rightarrow \mathbb{R}$  is the objective function,  $m \in M$  the control variable, and  $u \in V$  is the weak solution of the parametrised PDE.

$\Rightarrow$  **Key objective** : Compute  $\frac{dJ}{dm}$



Using chain rule we get

$$\frac{dJ}{dm} = -\lambda^* \left( \frac{\partial F}{\partial m} + \mu_1^* \right) + \mu_2^* + \frac{\partial J}{\partial m} \quad (5)$$

$\lambda^*$ ,  $\mu_1^*$  and  $\mu_2^*$  are the **adjoint variables**, they are obtained by the following relations:

$$\begin{cases} \left( \frac{\partial F}{\partial u} + \frac{\partial F}{\partial N_2} \frac{\partial N_2}{\partial u} \right)^* \lambda = \frac{\partial J^*}{\partial u} + \frac{\partial N_1^*}{\partial u} \frac{\partial J^*}{\partial N_1} \\ \mu_1 = \frac{\partial N_2^*}{\partial m} \frac{\partial F^*}{\partial N_2}, \mu_2 = \frac{\partial N_1^*}{\partial m} \frac{\partial J^*}{\partial N_1} \end{cases} \quad (6)$$

$\Rightarrow$  Adjoint computation depends on  $\frac{\partial N_i^*}{\partial u}$  and  $\frac{\partial N_i^*}{\partial m}$  for  $i = 1, 2$



In a nutshell:

1. The ExternalOperator project enables you to include any operators provided that you can define how the operator and its derivatives are evaluated. That can be anything that can be evaluated (e.g. Gaussian process, FFT, external libraries...)
2. Some classes of operator have already been implemented: **PointsolveOperator**, **PytorchOperator**, **SingleLayerPotential** and **DoubleLayerPotential**.
3. External operators play well with Pyadjoint, i.e. you can add in these operators in a PDE or PDE-constrained optimisation problem.
4. For neural networks, coupling with PyTorch to get derivative with respect to inputs/weights. Extensions to Tensorflow are straightforward.
5. External operators play well with matrix free methods.



In a nutshell:

1. The ExternalOperator project enables you to include any operators provided that you can define how the operator and its derivatives are evaluated. That can be anything that can be evaluated (e.g. Gaussian process, FFT, external libraries...)
2. Some classes of operator have already been implemented: **PointsolveOperator**, **PytorchOperator**, **SingleLayerPotential** and **DoubleLayerPotential**.
3. External operators play well with Pyadjoint, i.e. you can add in these operators in a PDE or PDE-constrained optimisation problem.
4. For neural networks, coupling with PyTorch to get derivative with respect to inputs/weights. Extensions to Tensorflow are straightforward.
5. External operators play well with matrix free methods.

What are the **practical takeaways** ?




In a nutshell:


1. The ExternalOperator project enables you to include any operators provided that you can define how the operator and its derivatives are evaluated. That can be anything that can be evaluated (e.g. Gaussian process, FFT, external libraries...)
2. Some classes of operator have already been implemented: **PointsolveOperator**, **PytorchOperator**, **SingleLayerPotential** and **DoubleLayerPotential**.
3. External operators play well with Pyadjoint, i.e. you can add in these operators in a PDE or PDE-constrained optimisation problem.
4. For neural networks, coupling with PyTorch to get derivative with respect to inputs/weights. Extensions to Tensorflow are straightforward.
5. External operators play well with matrix free methods.

What are the **practical takeaways** ?

- To build your own external operator: subclass the **AbstractExternalOperator** class in firedrake and equip your operator with an evaluate method (i.e. how your operator and its derivatives are evaluated).
- Code accessible via the **pointwise-adjoint-operator** firedrake branch
- Related talks:
  - External operators depend on dual spaces (see I. Marsden talk: Tuesday - 13:00–14:40 GMT).
  - LayerPotential operators (see B. Sepanski talk: Thursday - 15:00–16:30 GMT).

# Linear multipoint constraints in FEniCSx

Jørgen Schartum Dokken (<https://jsdokken.com>,  jorgensd), University of Cambridge, United Kingdom

Garth Wells ( garth-wells), University of Cambridge, United Kingdom

Chris Richardson ( chrisrichardson), University of Cambridge, United Kingdom

22 March 2021

In the finite element method, it is common to encounter boundary conditions such as the Dirichlet, Neumann and Robin boundary condition. However, the zero-slip condition,  $\mathbf{u} \cdot \mathbf{n} = 0$  does not fall under either of these categories when the domain boundary is not aligned with the coordinate axes. For this problem the boundary condition can be written as a linear combination of the degrees of freedom collocated at a boundary coordinate. In this talk, we present a method for enforcing linear multipoint constraints in the FEniCSx framework using master-minion matrix reduction. The multipoint constraint framework is written as a separate module that can be used alongside FEniCSx, supporting all variational forms written in the unified form language. The main components of the module is the multipoint constraint class, which handles communication of non-local degrees of freedom for parallel execution, and custom assemblers for the element-wise matrix reduction operation. Additionally, the module includes several specialized constructors for boundary conditions such as the periodic and zero-slip boundary condition. We illustrate the applicability of the module to non-trivial constraints by solving a contact problem on a non-conforming mesh.

## References

- [1] “dolfinx\_mpc”. [https://github.com/jorgensd/dolfinx\\_mpc](https://github.com/jorgensd/dolfinx_mpc).

---

You can cite this talk as:

Jørgen Schartum Dokken, Garth Wells, and Chris Richardson. “Linear multipoint constraints in FEniCSx”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 68–85. doi: 10.6084/m9.figshare.14495199.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/dokken.html>.

# Linear multipoint constraints in FEniCSx

## FEniCS 2021

**Jørgen S. Dokken**   Garth N. Wells   Chris Richardson



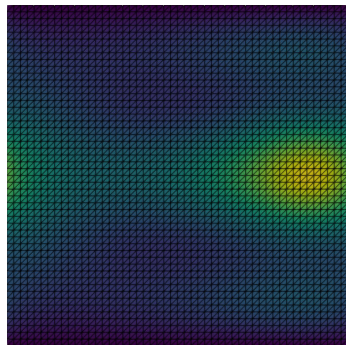
UNIVERSITY OF  
CAMBRIDGE

March 22, 2021

# What is a linear multipoint constraint (MPC)?

A linear combination of degrees of freedom:

- Periodic conditions:  $u(0, y, z) = u(L, y, z)$

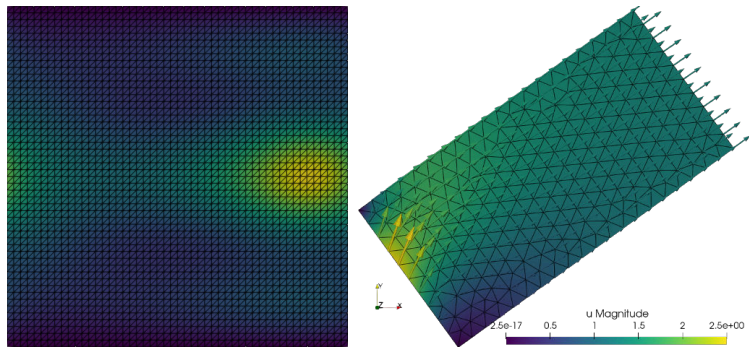




# What is a linear multipoint constraint (MPC)?

A linear combination of degrees of freedom:

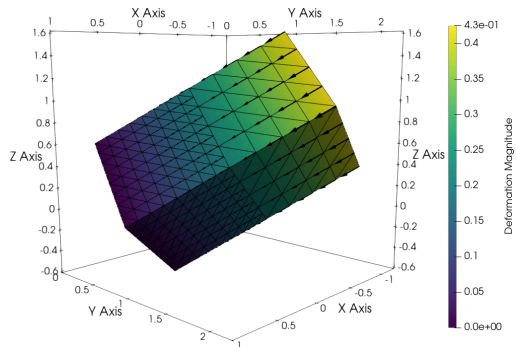
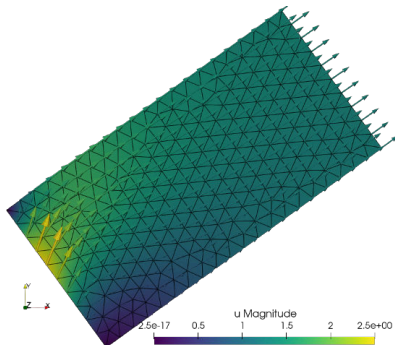
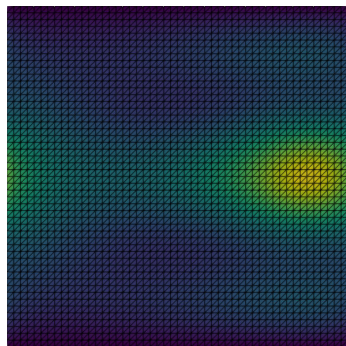
- Periodic conditions:  $u(0, y, z) = u(L, y, z)$
- Slip boundary conditions:  $u \cdot n = 0$



# What is a linear multipoint constraint (MPC)?

A linear combination of degrees of freedom:

- Periodic conditions:  $u(0, y, z) = u(L, y, z)$
- Slip boundary conditions:  $u \cdot n = 0$
- Frictionless contact:  $u_1 \cdot n_1 = u_2 \cdot n_1, \quad u_i \in \Omega_i$



To solve a system of linear equations, we eliminate degrees of freedom by using the additional constraints

Find  $u = (u_0, \dots, u_3)^T$  such that

$$Au = b$$

$$u_3 = \zeta u_0$$

To solve a system of linear equations, we eliminate degrees of freedom by using the additional constraints

Find  $u = (u_0, \dots, u_3)^T$  such that

$$Au = b$$

$$u_3 = \zeta u_0$$

Define the prolongation matrix  $P$

$$P\hat{u} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \zeta & 0 & 0 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix} = u$$

To solve a system of linear equations, we eliminate degrees of freedom by using the additional constraints

Find  $u = (u_0, \dots, u_3)^T$  such that

$$Au = b$$

$$u_3 = \zeta u_0$$

Define the prolongation matrix  $P$

$$P\hat{u} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \zeta & 0 & 0 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix} = u$$

We solve the reduced system

$$(P^T A P)\hat{u} = P^T b$$

where

$$(P^T A P) = \begin{pmatrix} \zeta^2 a_{3,3} + \zeta a_{0,3} + \zeta a_{3,0} + a_{0,0} & \zeta a_{3,1} + a_{0,1} & \zeta a_{3,2} + a_{0,2} \\ \zeta a_{1,3} + a_{1,0} & a_{1,1} & a_{1,2} \\ \zeta a_{2,3} + a_{2,0} & a_{2,1} & a_{2,2} \end{pmatrix}$$

A linear combination gives rise to mixed terms between the master nodes

$$Au = b$$

$$u_1 = \alpha u_0 + \beta u_2$$

A linear combination gives rise to mixed terms between the master nodes

$$Au = b$$

$$u_1 = \alpha u_0 + \beta u_2$$

$$(P^T AP)_{0,0} = \alpha^2 a_{1,1} + \alpha a_{0,1} + \alpha a_{1,0} + a_{0,0}$$

$$(P^T AP)_{0,1} = \alpha\beta a_{1,1} + \alpha a_{1,2} + \beta a_{0,1} + a_{0,2}$$

$$(P^T AP)_{0,2} = \alpha a_{1,3} + a_{0,3}$$

$$(P^T AP)_{1,0} = \alpha\beta a_{1,1} + \alpha a_{2,1} + \beta a_{1,0} + a_{2,0}$$

$$(P^T AP)_{1,1} = \beta^2 a_{1,1} + \beta a_{1,2} + \beta a_{2,1} + a_{2,2}$$

$$(P^T AP)_{1,2} = \beta a_{1,3} + a_{2,3}$$

$$(P^T AP)_{2,0} = \alpha a_{3,1} + a_{3,0}$$

$$(P^T AP)_{2,1} = \beta a_{3,1} + a_{3,2}$$

$$(P^T AP)_{2,2} = a_{3,3}$$

We apply both the conditions we have considered so far

$$Au = b$$

$$u_1 = \alpha u_0 + \beta u_2$$

$$u_3 = \zeta u_0$$

With prolongation matrix  $P$

$$P\hat{u} = \begin{pmatrix} 1 & 0 \\ \alpha & \beta \\ 0 & 1 \\ \zeta & 0 \end{pmatrix} \begin{pmatrix} u_0 \\ u_2 \end{pmatrix} = u$$



We obtain cross terms between the two constraints

$$Au = b$$

$$u_1 = \alpha u_0 + \beta u_2$$

$$u_3 = \zeta u_0$$

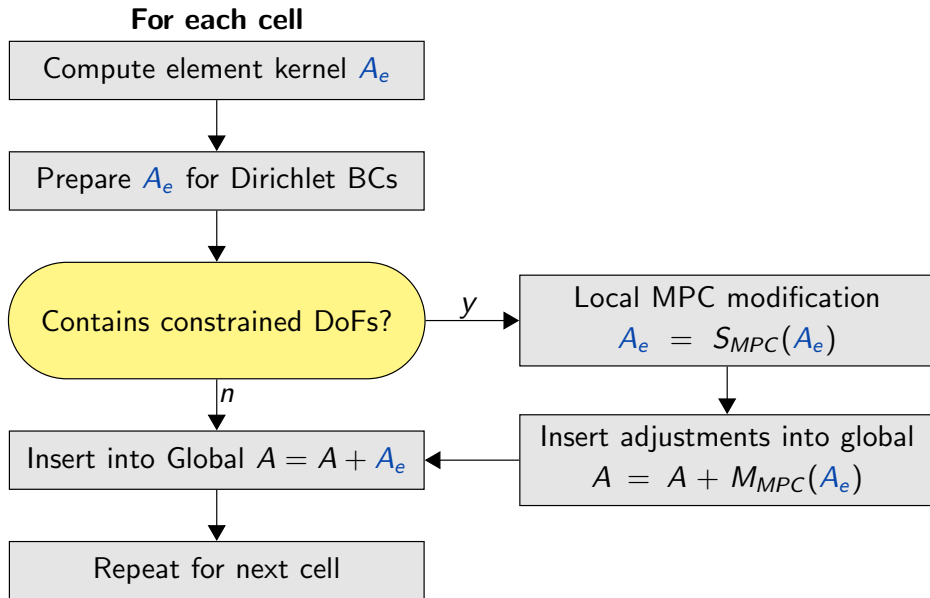
$$\begin{aligned} (P^T A P)_{0,0} = & \alpha^2 a_{1,1} + \alpha a_{0,1} + \alpha a_{1,0} \\ & + \zeta^2 a_{3,3} + \zeta a_{0,3} + \zeta a_{3,0} + \alpha \zeta a_{1,3} + \alpha \zeta a_{3,1} + a_{0,0} \end{aligned}$$

$$(P^T A P)_{0,1} = \alpha \beta a_{1,1} + \alpha a_{1,2} + \beta \zeta a_{3,1} + \beta a_{0,1} + \zeta a_{3,2} + a_{0,2}$$

$$(P^T A P)_{1,0} = \alpha \beta a_{1,1} + \alpha a_{2,1} + \beta \zeta a_{1,3} + \beta a_{1,0} + \zeta a_{2,3} + a_{2,0}$$

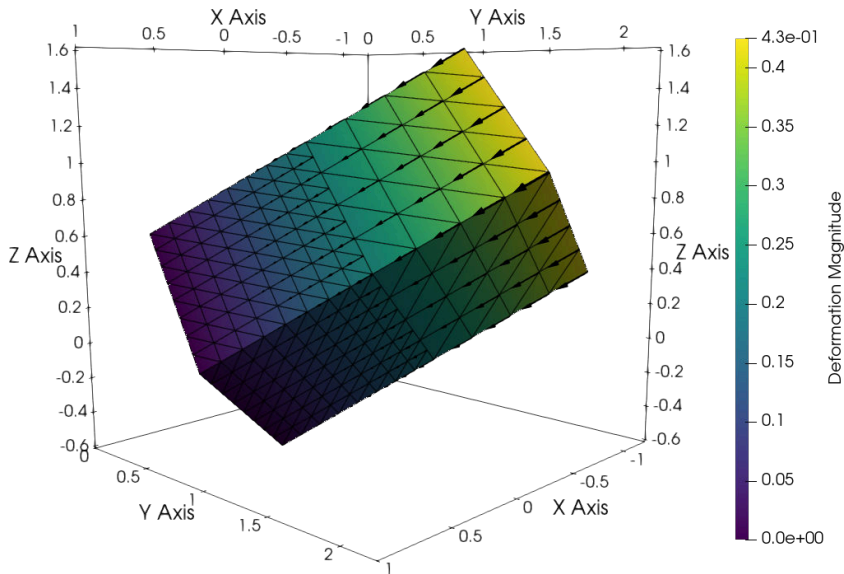
$$(P^T A P)_{1,1} = \beta^2 a_{1,1} + \beta a_{1,2} + \beta a_{2,1} + a_{2,2}$$

To make the assembly feasible for large systems, we compute the product  $P^T A_e P$

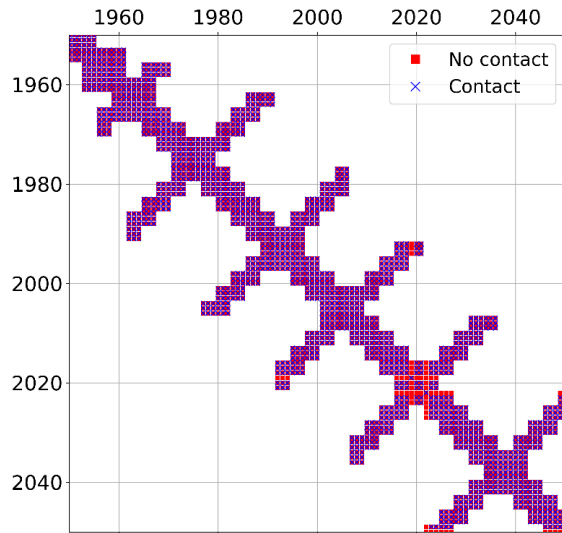
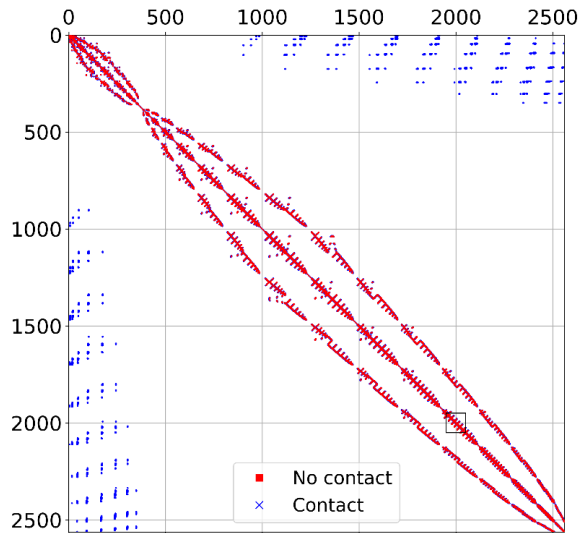


## Contact constraint between non-matching meshes

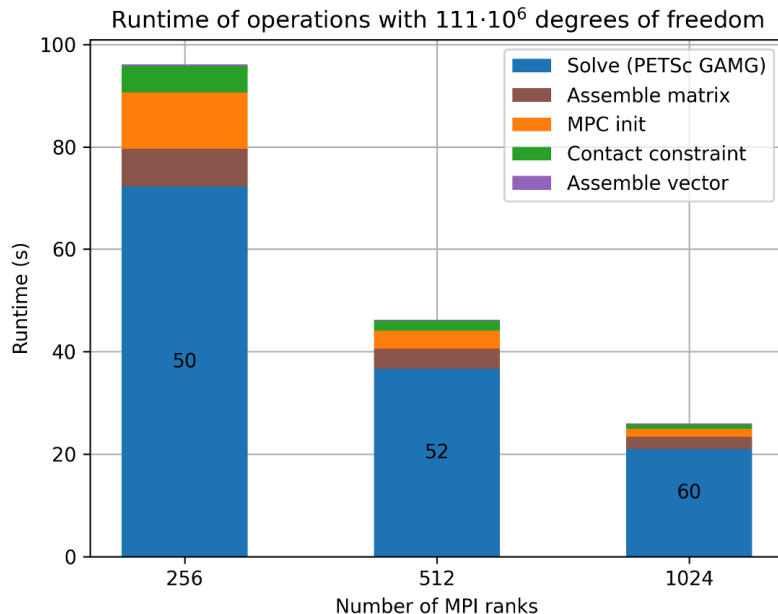
Linear elasticity where a displacement is described on the top (coarse) cube, and the bottom (fine) cube has a slip condition.



The matrix reduction operation introduces new non-diagonal entries to the sparsity pattern



## Strong scaling with 221 million cells



## The implementation is written as an add-on to DOLFINx

```
# Slip constraint on space W using facet-markers
mpc = dolfinx_mpc.MultiPointConstraint(W)
mpc.create_slip_constraint((mt, 1), n, ...)
mpc.finalize()

# Define variational problem using UFL
# ...

# Assemble matrix and vector
A = dolfinx_mpc.assemble_matrix(a, mpc, bcs)
b = dolfinx_mpc.assemble_vector(L, mpc)
A.assemble()

# Solve system using PETSc
# ...

# Backsubstitute from master to puppet dofs
mpc.backsubstitution(uh)
```

**Thank you for your attention**

`https://github.com/jorgensd/dolphinx\_mpc`

# Implementation of $p$ -multigrid and approximate fast diagonalization methods in Firedrake

Pablo Brubeck, University of Oxford, United Kingdom

Patrick Farrell (<https://pefarrell.org>), University of Oxford, United Kingdom

22 March 2021

For problems with smooth solutions, high-order methods have very good convergence properties, and in some cases they do not exhibit locking phenomena found in low-order methods. Moreover, due to data-locality and high arithmetic intensity, they are better suited to make efficient use of modern parallel hardware architectures. Unfortunately, the conditioning of the Galerkin matrices is severely affected by  $p$ , the polynomial degree of the approximation. In order to obtain practical iterative solvers, we require good preconditioners.

In the  $p$ -variant of multigrid, the problem is often coarsened by rediscretizing on the same mesh with a lower  $p$ . We implement a general  $p$ -multigrid ( $p$ -MG) method that can deal with general finite elements and custom coarsening schedules in Firedrake using PETSc. As relaxation, we employ a novel combination of an approximate fast diagonalization method and subspace correction. The scheme is essentially point-block Jacobi in the space of eigenfunctions of a separable approximation to the local stiffness matrix of each cell. The relaxation depends on the tensor-product structure of quadrilateral and hexahedral elements, in a similar manner to sum factorisation.

We employ this relaxation method in two algorithms: a  $p$ -MG preconditioner and a full approximation scheme nonlinear solver. We demonstrate how to combine these two efficiently in a nested iteration with a cascading outer cycle and inner V-cycles. All available solvers, including geometric and algebraic multigrid, may be employed for the  $p$ -coarse level. The associated computational costs are  $O(p^d)$  to approximate the local stiffness matrix in  $d$  dimensions, and  $O(p^{d+1})$  to apply or update the relaxation, while memory requirements are kept at  $O(p^d)$ . We present nonlinear examples such as the  $p$ -Laplacian and incompressible hyperelasticity.

---

You can cite this talk as:

Pablo Brubeck and Patrick Farrell. "Implementation of  $p$ -multigrid and approximate fast diagonalization methods in Firedrake". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 86. doi: 10.6084/m9.figshare.14495202.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/brubeck.html>.



# Dynamic composition of solvers for coupled problems in DOLFINx

Martin Řehoř, Rafinex S.à r.l., Luxembourg

Jack S. Hale, University of Luxembourg, Luxembourg

22 March 2021

Recent developments in DOLFINx allow for the block assembly of linear algebraic systems arising from discretisations of coupled partial differential equations. Each algebraic block represents a subproblem associated with a coupling of the unknown fields. Designing and implementing robust and scalable solution and preconditioning strategies for block-structured linear systems is an active area of research.

In this contribution we show how DOLFINx can now exploit one of the most significant features of PETSc; the dynamic composition of the hierarchical solver and preconditioner options at runtime, see Brown *et al* [1]. The idea is inspired by the work of Kirby and Mitchell [2] that was originally implemented in the Firedrake Project.

One of the most significant benefits of the approach is the possibility to construct advanced preconditioners that require structure beyond a purely algebraic problem description, eg the pressure-convection-diffusion (PCD) approximation of the Schur complement for the Navier–Stokes equations, see Silvester *et al* [3].

We illustrate the capabilities of our implementation on examples ranging from incompressible flow of a viscous fluid, through temperature-driven convection, to flows described by rate-type viscoelastic fluid models.

The slides for this talk are available at <https://mscroggs.github.io/fenics2021/talks/rehor.html> under a CC BY-NC-ND 4.0 license.

## References

- [1] J. Brown, M. G. Knepley, D. A. May, L. C. McInnes, and B. Smith. “Composable Linear Solvers for Multiphysics”. In: *Proceedings of the 2012 11th International Symposium on Parallel and Distributed Computing (ISPD ‘12), IEEE Computer Society, USA* (2012), 55–62. DOI: 10.1109/ISPD.2012.16.
- [2] R. Kirby and L. Mitchell. “Solver Composition Across the PDE/Linear Algebra Barrier”. In: *SIAM Journal on Scientific Computing* 40 (2017). DOI: 10.1137/17M1133208.
- [3] D. Silvester, H. Elman, and A. Wathen. “Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics”. 2014.

---

You can cite this talk as:

Martin Řehoř and Jack S. Hale. “Dynamic composition of solvers for coupled problems in DOLFINx”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 87. DOI: 10.6084/m9.figshare.14495211.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/rehor.html>.

# Additive Schwarz methods for serendipity elements

Jorge Marchena Menendez, Baylor University, United States

Robert Kirby, Baylor University, United States

22 March 2021

While solving partial differential equations with finite element method, serendipity elements allow us to obtain the same order of accuracy as rectangular tensor-product elements with many fewer degrees of freedom. To realize these savings in practice, we utilize p-version Additive Schwarz methods that solve local patch problems together with a low-order global system. For symmetric coercive problems, we obtain condition numbers independent of the mesh size and degree of serendipity space. Numerical experiments using Firedrake and PETSc confirm this theory and demonstrate efficiency relative to standard elements for model problems.

---

You can cite this talk as:

Jorge Marchena Menendez and Robert Kirby. “Additive Schwarz methods for serendipity elements”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 88. DOI: 10.6084/m9.figshare.14495217.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/marchena-menendez.html>.

# Domain decomposition of stochastic PDEs using FEniCS

**Abhijit Sarkar** (<http://abhijitsarkar.net>), Carleton University, Canada

Sudhi Sharma, Carleton University, Canada

Ajit Desai, Carleton University, Canada

Mohammad Khalil, Sandia National Laboratories, United States

Chris Pettit, United States Naval Academy, United States

Dominique Poirel, Royal Military College of Canada, Canada

**22 March 2021**

The intrusive spectral stochastic finite element method (SSFEM) based domain decomposition (DD) solvers for stochastic PDEs are implemented using FEniCS. For efficient uncertainty quantification (UQ) for large-scale computational models, these algorithms demonstrate scalabilities for high resolution spatial discretization and high dimensional random parameter space. However the implementation of these algorithms is intrusive to finite element codes demanding additional programming efforts. In the intrusive SSFEM based DD formalism, the stochastic PDE is converted into a very large set (depending on the number of random parameters) of deterministic coupled PDE system. It leads to a large-scale linear system being solved iteratively using two-level domain decomposition preconditioners. The submatrices for each subdomain are constructed using FEniCS. For each scale of random fluctuation, the associated subdomain level deterministic submatrices required for DD algorithm are extracted through a modified variational form of the PDE. Both three-dimensional stochastic Poisson and linear elasticity problems are tackled through this generic software leveraging FEniCS.

---

You can cite this talk as:

Abhijit Sarkar, Sudhi Sharma, Ajit Desai, Mohammad Khalil, Chris Pettit, and Dominique Poirel. "Domain decomposition of stochastic PDEs using FEniCS". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 89. DOI: 10.6084/m9.figshare.14495220.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/sarkar.html>.

# Towards IEM-FEM coupling for simulation of photoacoustic trace gas sensors

**Xiaoyu Wei**, University of Illinois at Urbana-Champaign, United States

Andreas Kloeckner, University of Illinois at Urbana-Champaign, United States

Robert Kirby, Baylor University, United States

Peter Coogan, Baylor University, United States

22 March 2021

Nonlocal operators such as layer and volume potential operators are prevalent in physical sciences. We are looking towards making such evaluators accessible from UFL. As an example application, we consider a wave-structure interaction model for photoacoustic trace gas sensors. The sensors of interest are based on quartz-enhanced photoacoustic spectroscopy or resonant optoacoustic detection. The model consists of thermoacoustic waves in the exterior fluid domain and thermoelastic waves in the interior solid domain. We plan to use integral equation method (IEM) for the exterior and FEM for the interior. In this contribution, we propose a novel second-kind integral equation formulation for the exterior solution that solves the Morse–Ingard equation subject to Neumann boundary conditions on the solid surface. The resulting boundary integral equation is solved with GMRES where quadrature-by-expansion with fast-multipole acceleration is used to evaluate the nonlocal integral operators. Since the solution representation naturally satisfies the far-field conditions, domain truncation and perfectly matched layer are not needed. We demonstrate with examples that our solver has  $O(n)$  complexity, is high-order, and can handle complex geometries. For the next steps, we are working towards integrating our Morse–Ingard solver with FEM, and enabling more general IEM-FEM coupling in the process.

---

You can cite this talk as:

Xiaoyu Wei, Andreas Kloeckner, Robert Kirby, and Peter Coogan. “Towards IEM-FEM coupling for simulation of photoacoustic trace gas sensors”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 90. DOI: 10.6084/m9.figshare.14495226.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/wei.html>.

# On the reproducibility of numerical experiments with FEniCS

**Paul Garlick** (<http://www.tourbillion-technology.com>), Tourbillion Technology Ltd, United Kingdom

22 March 2021

A key tenet of the scientific method is that an experiment conducted in one laboratory may be reproduced in another. For numerical experiments the process should be straightforward; simply a case of supplying the same inputs to the same software application. However, for software applications with complex hierarchies the essential step of re-creating the same software environment can represent a significant challenge. Difficulties can emerge in installing library dependencies on computer systems with different architectures. Even on the same system changes to low-level libraries over time can prevent compilation or execution of higher-level code. To address these issues methods from functional programming have been developed to guarantee bit-identical software installations. Package definitions have been created for FEniCS and its dependencies within the GNU Guix framework. Used as a package manager, GNU Guix provides the tools to manage a dedicated file structure that sits alongside the host operating system. This presentation shows how the underlying mechanism works, how FEniCS fits in and how to create software environments that can be reproduced identically in space and time.

---

You can cite this talk as:

Paul Garlick. “On the reproducibility of numerical experiments with FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 91–99. DOI: 10.6084/m9.figshare.14495229.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/garlick.html>.

# On the reproducibility of numerical experiments with FEniCS

Paul Garlick

Tourbillion Technology Ltd

22nd March 2021

<https://www.tourbillion-technology.com>

# Reproducibility

- Significant factor when assessing the trustworthiness of results
- Problems can be encountered when:
  - moving a simulation environment from one system to another
  - restoring the simulation environment from a previous project
- Solutions can be categorized as:
  - approximate: *docker*, *singularity*, *spack* . . .
  - exact (bit-identical): *nix*, **guix**



# Guix introduction

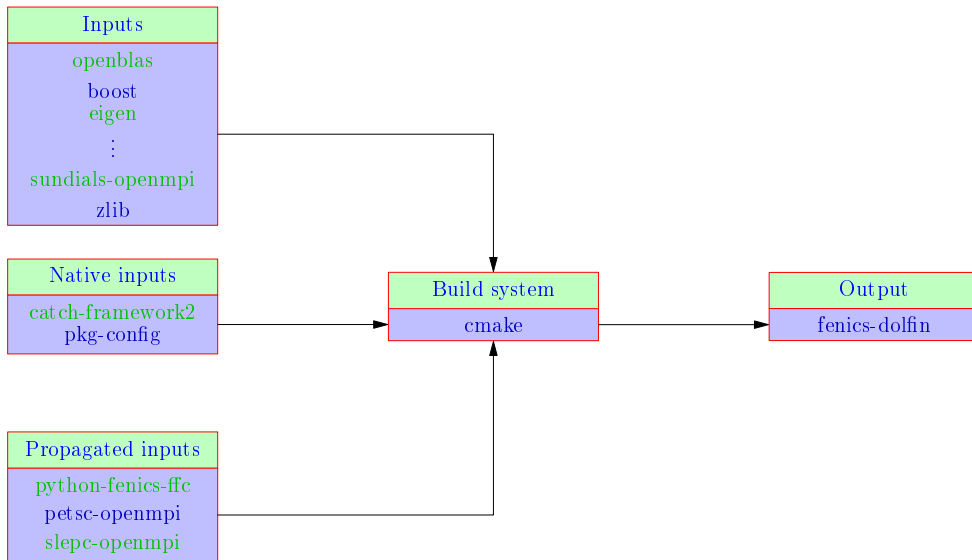
- GNU/Linux distribution and package manager
  - x86\_64, aarch64, powerpc64le architectures
- 16,000+ packages
  - FEniCS is part of the *simulation* module
- 60+ contributors/month
- Guile Scheme API; embedded DSL for defining packages
- File structure allows a separation of concerns. The operating system and *user profiles* can be managed independently:

```
/ (root)
├── OS
└── /gnu
    └── /store
```



# How it works

- Each package defines a list of inputs and a build function
- The build function returns the installed package with no side effects



Example build process.

- Build process can be expressed in mathematical form, relating inputs to output objects via the build function:

$$f_{b,h} : \{i_{0,h}, i_{1,h}, \dots, i_{m,h}\} \mapsto \{o_0, o_1, \dots, o_n\}$$

- Directory name for a package contains a hash of all of the inputs plus the package name and version:

```
/gnu/store/ywcdaz69y36...jbvw1igj-fenics-2019.1.0.post0
```

- The standard tools provide an automated process for managing complexity in package hierarchies
  - a *rolling release* update process
- To guarantee the reproducibility of a complete software environment two advanced features are needed:
  - a means to specify the packages to include
  - the ability to *pin* the distribution

## Small list of lists

- User profile is defined by a *manifest* file:

```
(specifications->manifest
  '("emacs"
    "fenics"
    "gmsh"
    "python"
  ))
```

- Source code location is defined by a *channel specification* file:

```
(list (channel
      (name 'guix)
      (url
        "https://git.savannah.gnu.org/git/guix.git")
      (commit
        "a002e8a4f58a45034075cad27bf8eb65679bcc14")))
```

- Use version control to record changes to the lists:
  - move forward and back through history as needed

# Status

- Package definitions completed:
  - FEniCS 2018.1.0.post1
  - FEniCS 2019.1.0.post0
- In progress:
  - pyadjoint
  - pygmsh, meshio
- Todo:
  - FEniCSX
  - Firedrake



# Links

- Getting started:
  - <https://guix.gnu.org>
- Moving on:
  - <https://guix.gnu.org/manual>
  - <https://guix.gnu.org/cookbook>
- Joining in:
  - Guix simulation channel (site under construction)
- User support:
  - email: [pgarlick@tourbillion-technology.com](mailto:pgarlick@tourbillion-technology.com)

# mgis.fenics Part I: Coupling MFront and FEniCS for complex solid mechanics simulations

**Thomas Helfer** (<http://tfel.sourceforge.net>), CEA, DEN/DEC/SESC, Département d'Études des Combustibles, France

**Jérémy Bleier** (<https://comet-fenics.readthedocs.io/en/latest/index.html>), Laboratoire Navier UMR 8205 (École des Ponts ParisTech-IFSTTAR-CNRS), France

**Raffaele Russo**, University of the Basque Country, Spain

**Tamara Dancheva**, BCAM - Basque Center for Applied Mathematics, Spain

**22 March 2021**

Constitutive equations describe how the internal state variables of a material evolve with changing external conditions or due to gradients of thermodynamic variables. Those state variables can describe many microstructural aspects of the material (grain size, dislocation density, hardening state, etc) or be phenomenological in nature (equivalent plastic strain). The knowledge of those internal state variables allows the computation of local thermodynamic forces which affect the material equilibrium at the structural scale.

MFront is an open-source code generator for complex constitutive laws which aims at ease of use, numerical efficiency and portability (see [5][7]). MFront has been developed under very stringent quality requirements in the context of nuclear fuel element simulation under the PLEIADES platform (see [6]), which is co-developed by CEA, EDF and Framatome. MFront provides several domain specific languages (DSL) built on top of the C++ language and associated with specific integration schemes that allows to readily implement the constitutive equations in source code close to their mathematical expressions. Numerical details are hidden by default allowing the user to focus on the physics. The underlying mathematical library, called TFEL/Math, provides optimised tensor objects and makes heavily use of template metaprogramming to generate optimised code.

Those DSLs are translated into C++ sources adapted to the targeted solver. Interfaces are provided for Cast3M, Code Aster, Europlexus, Cyranos, Abaqus/Implicit, Abaqus/Explicit, Ansys, CalculiX, AMITEX FFTP, etc. A so-called generic interface has recently been introduced and is meant to be used through the MFrontGenericInterfaceSupport project (MGIS) (See [3][4]).

The mgis.fenics python module aims at leveraging the power of the FEniCS platform, used for the discretization of the balance equations, the assembly of residuals and stiffness matrices and the parallel distribution of the resolution, combined with MFront, used for the local integration of the constitutive equations, to build complex mechanical simulations. Several examples, illustrating the use of the new module, will be presented (see [1][2]), including:

- Finite strain plasticity in the logarithmic space.
- Phase-field approach to brittle fracture.
- Finite strain polycrystal computations based on the Méric-Cailletaud behaviour.

---

You can cite this talk as:

Thomas Helfer, Jérémy Bleier, Raffaele Russo, and Tamara Dancheva. “mgis.fenics Part I: Coupling MFront and FEniCS for complex solid mechanics simulations”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 100–120. doi: 10.6084/m9.figshare.14495232.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/helfer.html>.

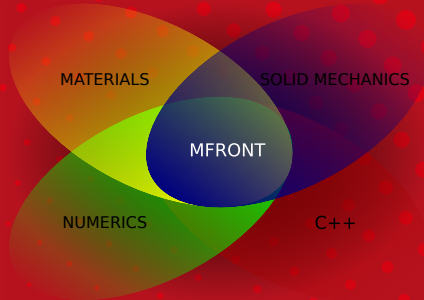
## References

- [1] J  r  my Bleier and Thomas Helfer. “Elasto-plastic analysis implemented using the MFront code generator”. In: *Numerical tours of continuum mechanics using FEniCS* (2019).
- [2] J  r  my Bleier and Thomas Helfer. “FEniCS and MFront for complex non linear solid mechanics simulation”. 2019.
- [3] Thomas Helfer. “The MFrontGenericInterfaceSupport project”. <https://thelfer.github.io/mgis/web/index.html>.
- [4] Thomas Helfer, Jeremy Bleier, Tero Frondelius, Ivan Yashchuk, Thomas Nagel, and Dmitri Naumov. “The MFrontGenericInterfaceSupport project”. In: *Journal of Open Source Software* 5:2003.48 (). DOI: 10.21105/joss.02003.
- [5] Thomas Helfer, Bruno Michel, Jean-Michel Proix, Maxime Salvo, J  r  me Sercombe, and Michel Casella. “Introducing the open-source mfront code generator: Application to mechanical behaviours and material knowledge management within the PLEIADES fuel element modelling platform”. In: *Computers & Mathematics with Applications* 70.5 (2015), 994–1023. DOI: 10.1016/j.camwa.2015.06.027.
- [6] V. Marelle, P. Goldbronn, S. Bernaud, E. Castelier, J. Julien, K. Nkonga, L.Noivot, and I. Ramiere. “New developments in ALCYONE 2.0 fuel performance code”. In: *Top Fuel* (2016).
- [7] “MFront website”. <http://www.tfel.sourceforge.net>.

# mgis.fenics: coupling MFfront and FEniCS for complex solid mechanics simulations



DE LA RECHERCHE À L'INDUSTRIE



FEniCS conference 2021 - 22-26 March 2021

T. Helfer<sup>(1)</sup>, J. Bleyer<sup>(2)</sup>, R. Russo<sup>(3)</sup>, T. Dancheva<sup>(4)</sup>

<sup>(1)</sup> CEA, DES, IRESNE, DEC, SESC, LSC, Cadarache, France

<sup>(2)</sup> Laboratoire Navier UMR 8205 (École des Ponts ParisTech-IFSTTAR-CNRS), France

<sup>(3)</sup> University of the Basque Country

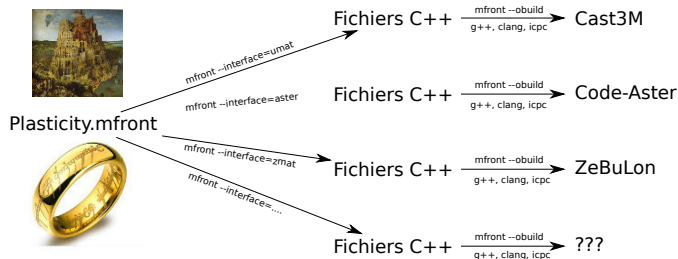
<sup>(4)</sup> BCAM - Basque Center for Applied Mathematics



- `mgis.fenics` aims at combining the power of :
  - FEniCS for automatic generation of assembly of user defined weak forms in the UFL syntax, the HPC performances, etc...

$$R(\mathbf{v}) = \sum_{i=1}^p \int_{\Omega} \boldsymbol{\sigma}^i(\mathbf{u}) \cdot \delta \mathbf{g}^i(\mathbf{v}) \, dx - L(\mathbf{v}) = 0 \quad \forall \mathbf{v} \in V$$

- MFront for the local description of the material behaviour :
  - Complex kernels at quadrature points to compute  $\boldsymbol{\sigma}^i(\mathbf{u})$ .
  - See also the talks about the AceGEN and Matériaux projects.
- The MGIS project that provides classes to :
  - Retrieve **metadata** from an MFront behaviour
  - Allocate memory associated with the **state variables** handled by the behaviour
  - Call the behaviour integration over a time step.
  - <https://github.com/thelfer/MFrontGenericInterfaceSupport>



- ▶ MFront is a code generation tool dedicated to material knowledge (material properties, mechanical behaviours, point-wise models) :
  - Support for small and finite strain behaviours, cohesive zone models, **generalised behaviours** (non local and or multiphysics).
- ▶ Main goals :
  - Numerical efficiency (see various benchmarks on the website).
  - Portability (Cast3M, Cyrano, code\_aster, Europlexus, TMFTT, AMITEX\_FFTP, Abaqus, CalculiX, MTest).
  - **Ease of use** : *Longum iter est per praecepta, breve et efficax per exempla* (It's a long way by the rules, but short and efficient with examples).

```
@DSL Implicit;
@Behaviour Norton;
@Brick StandardElasticity;

@MaterialProperty stress E;
E.setGlossaryName("YoungModulus");
@MaterialProperty real ν, A, nn;
ν.setGlossaryName("PoissonRatio");
A.setEntryName("NortonCoefficient");
nn.setEntryName("NortonExponent");

@StateVariable real p;
p.setGlossaryName("EquivalentViscoplasticStrain");
```

```
@Integrator{
  constexpr const auto Me = Stensor4::M();
  const auto μ = computeMu(E, ν);
  const auto σe = sigmaeq(σ);
  const auto iσe = 1 / (max(σe, real(1.e-12) · E));
  const auto vp = A · pow(σe, nn);
  const auto ∂vp/∂σe = nn · vp · iσe;
  const auto n = 3 · deviator(σ) · (iσe / 2);
  // Implicit system
  fεel += Δp · n;
  fp -= vp · Δt;
  // jacobian
  ∂fεel/∂Δεel += 2 · μ · θ · dp · iσe · (Me - (n ⊗ n));
  ∂fεel/∂Δp = n;
  ∂fp/∂Δεel = -2 · μ · θ · ∂vp/∂σe · Δt · n;
} // end of @Integrator
```

- Implicit integration.
- Implicit system :

$$\begin{cases} f_{\epsilon^{el}} = \Delta \epsilon^{el} - \Delta \epsilon^{to} + \Delta p \underline{n} \\ f_p = \Delta p - A \sigma_{eq}^n \end{cases}$$

- Jacobian :

$$\begin{cases} \frac{\partial f_{\epsilon^{el}}}{\partial \Delta \epsilon^{el}} = \underline{\underline{I}} + \frac{2 \mu \theta \Delta p}{\sigma_{eq}} (\underline{\underline{M}} - \underline{n} \otimes \underline{n}) \\ \frac{\partial f_{\epsilon^{el}}}{\partial \Delta p} = \underline{n} \\ \frac{\partial f_p}{\partial \Delta \epsilon^{el}} = -2 \mu \theta A n \sigma_{eq}^{n-1} \Delta t \underline{n} \end{cases}$$

- All programming and numerical details are hidden (by default).

- Inside a custom `NonlinearProblem`, define  $\sigma^i$  on a Quadrature space and the generalized residual :

$$R(\mathbf{v}) = \sum_{i=1}^p \int_{\Omega} \sigma^i(\mathbf{u}) \cdot \delta \mathbf{g}^i(\mathbf{v}) \, dx - L(\mathbf{v}) = 0 \quad \forall \mathbf{v} \in V$$

- MGIS gives metadata to know on which blocks  $\mathcal{B}(i)$  of gradients each flux  $\sigma_i$  depends :

$$a_{\text{tangent}}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^p \sum_{j \in \mathcal{B}(i)} \int_{\Omega} \delta \mathbf{g}^i(\mathbf{v}) \cdot \mathbb{T}_{\mathbf{g}^j}^{\sigma^i} \cdot \delta \mathbf{g}^j(\mathbf{u}) \, dx$$

- **This default variational problem can be overloaded by the user using UFL.**
- **`mgis.fenics` almost automatically make the links between `FEniCS` and `MFront`.**

- ▶ Documented demos have been designed to progressively illustrate the use of the interface and the versatility of the approach when implementing complex generalized behaviours, both on the MFront and FEniCS sides.
- ▶ We recommend browsing the demos in the following order :
  - Stationnary non-linear heat transfer
  - Stationnary non-linear heat transfer : 3D problem and performance comparisons
  - Transient heat equation with phase change
  - Monolithic transient thermoelasticity
  - Small-strain von Mises elastoplasticity
  - **Finite-strain elastoplasticity within the logarithmic strain framework**
  - Multiphase model for fiber-reinforced materials
  - **Phase-field approach to brittle fracture**
- ▶ Repository : a repository containing the demos sources files is available <https://gitlab.enpc.fr/navier-fenics/mgis-fenics-demos>

```

1 @DSL Implicit;
2
3 @Behaviour LogarithmicStrainPlasticity;
4
5 @StrainMeasure Hencky;
6
7 @Brick StandardElastoViscoPlasticity{
8   stress_potential : "Hooke" {
9     young_modulus : 210e9,
10    poisson_ratio : 0.3
11  },
12  inelastic_flow : "Plastic" {
13    criterion : "Mises",
14    isotropic_hardening : "Linear" {H : 500e6,
15                                   R0 : 250e6}
16  }
17 };

```

```

material = mf.MFrontNonlinearMaterial("./src/libBehaviour.so", "
    LogarithmicStrainPlasticity")
problem = mf.MFrontNonlinearProblem(u, material, bcs=bc)
problem.set_loading(dot(selfweight, u)*dx)

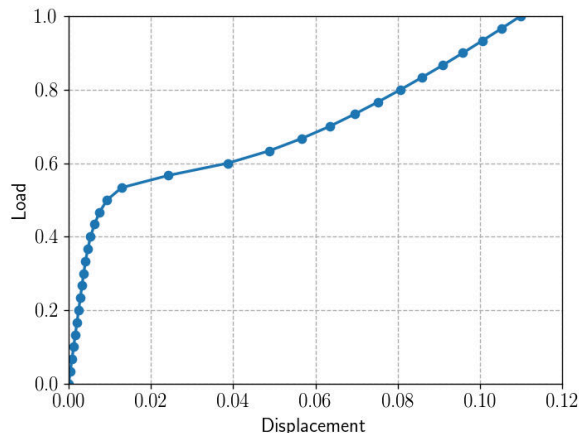
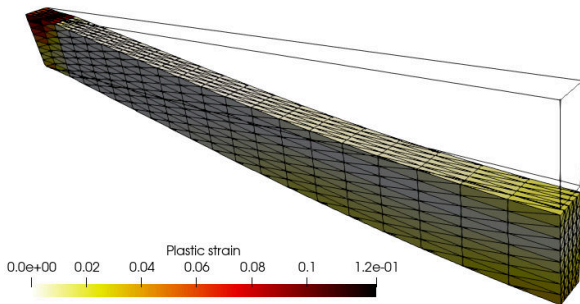
prm = problem.solver.parameters
prm["absolute_tolerance"] = 1e-6
prm["relative_tolerance"] = 1e-6
prm["linear_solver"] = "mumps"

for (i, t) in enumerate(load_steps[1:]):
    selfweight.t = t
    problem.solve(u.vector())

```

Automatic registration of  
as  $I + (\text{grad}(\text{Displacement}))$

- ▶ Residual is :  $R(\mathbf{v}) = \int_{\Omega} \mathbf{P} : \delta \mathbf{R}(\mathbf{v}) \, dx - W_{\text{ext}}(\mathbf{v})$
  - ▶ Consistent tangent bilinear form is :  $a_{\text{tangent}}(\mathbf{u}, \mathbf{v}) = \int_{\Omega} \nabla \mathbf{u} : \frac{\partial \mathbf{P}}{\partial \mathbf{F}} : \nabla \mathbf{v} \, dx$
  - ▶ Logarithmic strain plasticity (Miehe, 2002) :
    - Hencky strain measure  $\mathbf{H} = \frac{1}{2} \log(\mathbf{F}^T \cdot \mathbf{F})$  ✗
    - Use a small strain constitutive relation on  $\mathbf{H} \Rightarrow \mathbf{H} = \mathbf{H}^e + \mathbf{H}^p$
- $$\mathbf{F} \longrightarrow \mathbf{H} \longrightarrow \boxed{\text{small strain law}} \longrightarrow \mathbf{T} \longrightarrow \boldsymbol{\sigma}$$



- ▶ [https://thelfer.github.io/mgis/web/mgis\\_fenics\\_finite\\_strain\\_elastoplasticity.html](https://thelfer.github.io/mgis/web/mgis_fenics_finite_strain_elastoplasticity.html)
- ▶ [https://gitlab.enpc.fr/navier-fenics/mgis-fenics-demos/-/tree/master/demos/finite\\_strain\\_elastoplasticity](https://gitlab.enpc.fr/navier-fenics/mgis-fenics-demos/-/tree/master/demos/finite_strain_elastoplasticity)

- Bourdin/Francfort/Marigo **variational phase-field approach** :

$$\mathbf{u}(t), d(t) = \arg \min_{\mathbf{u}, d} \int_{\Omega} (1 - d)^2 \psi^+(\boldsymbol{\varepsilon}) + \psi^-(\boldsymbol{\varepsilon}) \, dx - W_{\text{ext}}(\mathbf{u}) + \frac{G_c}{c_w} \int_{\Omega} \left( \frac{w(d)}{\ell_0} + \ell_0 \|\nabla d\|^2 \right) \, dx$$

- Example of Tension/compression splitting (Miehe et al.) :

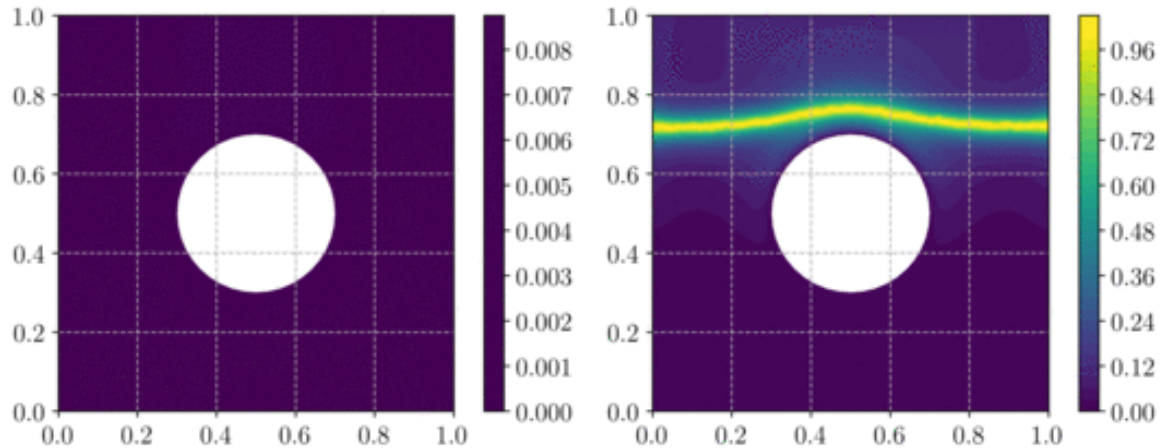
$$\psi^+(\boldsymbol{\varepsilon}) = \frac{1}{2} \lambda \langle \text{tr}(\boldsymbol{\varepsilon}) \rangle_+^2 + \mu \sum_I \langle \varepsilon_I \rangle_+^2$$

- Implementation of the alternate minimisation algorithm :

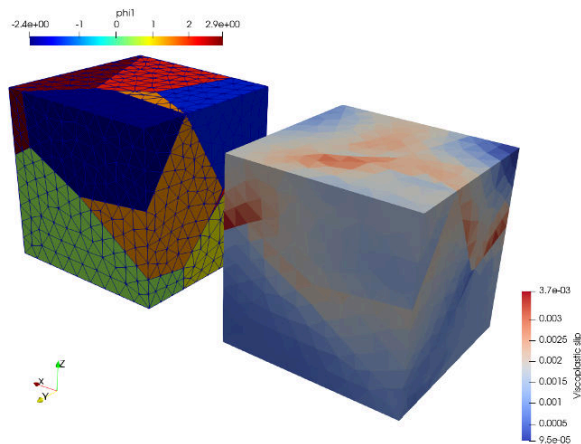
```
problem_u = mf.MFrontNonlinearProblem(u, material_u, bcs=bcu)
problem_u.register_external_state_variable("Damage", d)
psi = problem_u.get_state_variable("PositiveEnergyDensity")
```

```
for (i, t) in enumerate(loading[1:]):
    Uimp.t = t
    while res > tol and j < Nitermax:
        problem_u.solve(u.vector()) # Solve displacement u—problem
        problem_d.solve(d.vector()) # Solve damage d—problem
```





- Classical example of crack propagation (Bourdin et al.)



- Finite strain implementation of the Méric-Cailletaud single crystal behaviour :
  - <http://tfel.sourceforge.net/MericCailletaudSingleCrystalPlasticity.html>
- Example of orthotropic behaviour support.
- Périodic boundary conditions.

## Conclusions and perspectives

- ▶ MFront is an ever improving code generation tool dedicated to material knowledge with one foot in the industrial world and one foot in the academic world.
- ▶ The development of TFEL-3.4 (this year version) has been geared around three main axes :
  - Generalised behaviours
  - Porous plasticity
  - Extension and implementation of the Madnex specifications for the storage of MFront files.
- ▶ The development of TFEL-4.0 (next year) will be driven by :
  - The port to the C++-17 standard. MFront files will be backward-compatible.
  - Homogeneisation
  - Data driven simulation?
  - Support of GPUs?

- ▶ MGIS is a young project with many interesting perspectives :
  - Support of GPU? Support of Eigen?
- ▶ What's next in `mgis.fenics`?
  - Tests!
  - Integration in **dolfin-x** and performances improvements.
  - Multi-materials, plates/shells
  - Other examples :
    - Cosserat elastoplasticity (see part II).
    - Micromorphic crystal plasticity.
- ▶ New users and contributions are welcomed!



**Thank you for your attention.**

**Time for discussion!**

<https://tfel.sourceforge.net>

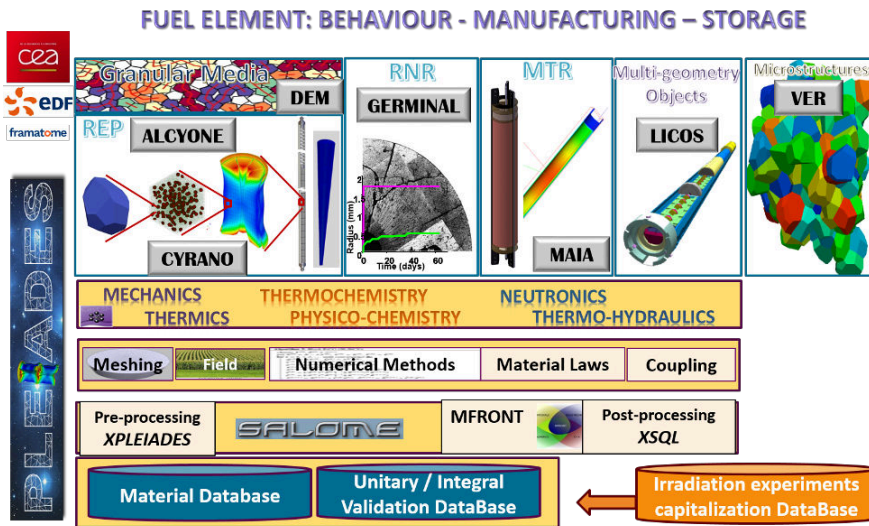
<https://www.researchgate.net/project/TFEL-MFront>

[https://twitter.com/TFEL\\_MFront](https://twitter.com/TFEL_MFront)

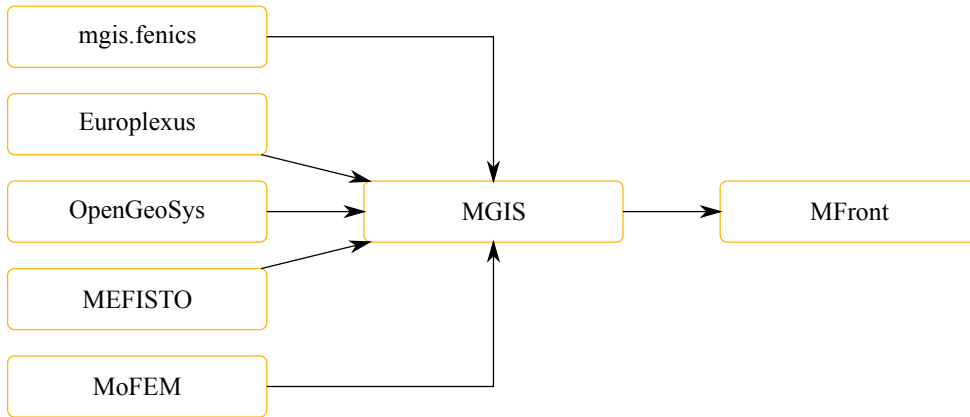
<https://github.com/thelfer/>

**[tfel-contact@cea.fr](mailto:tfel-contact@cea.fr)**

**The development of MFront is supported  
financially by CEA, EDF and Framatome  
in the framework of the PLEIADES project.**

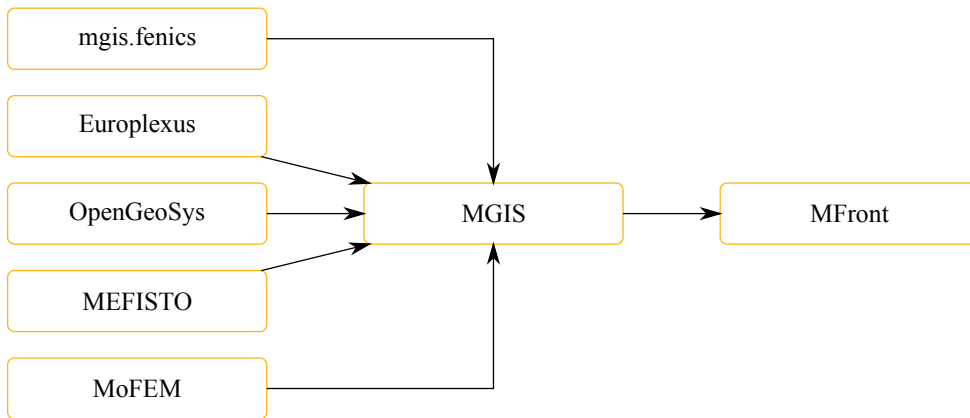


- ▶ A wide range of materials (ceramics, metals, composites).
- ▶ A wide range of mechanical phenomena and behaviours.
  - Creep, swelling, irradiation effects, phase transitions, etc..
- ▶ A wide range of mechanical loadings.

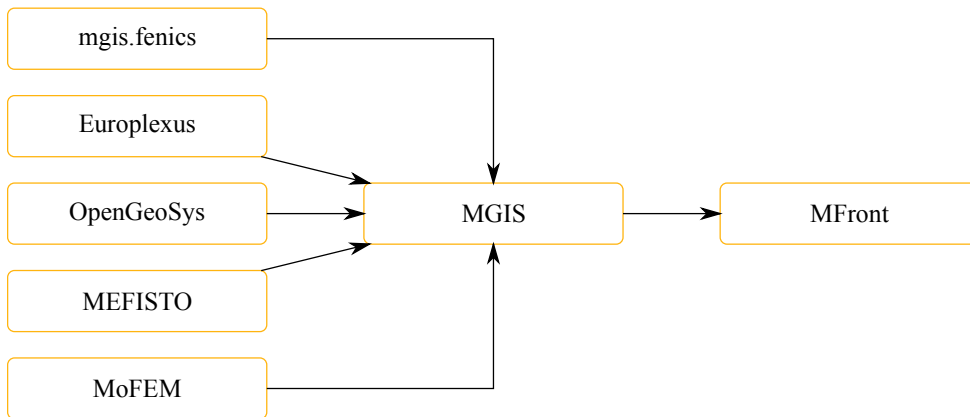


- The MGIS project provides classes on the solver side to retrieve **metadata** from an MFront behaviour and call the behaviour integration over a time step.





- The MGIS project provides classes on the solver side to retrieve **metadata** from an MFront behaviour and call the behaviour integration over a time step.



- The MGIS project provides classes on the solver side to retrieve **metadata** from an MFront behaviour and call the behaviour integration over a time step.
- Written in C++. Bindings exists for C, Fortran2003, python, Julia. And also used/tested in XPer, Kratos Multiphysics, JuliaFEM, NairmMPM, esys.escript, DUNE, HELIX (based on MFEM).

# mgis.fenics Part II: Cosserat media in small deformation with mgis.fenics

**Tamara Dancheva** (<http://www.bcamath.org/en/>), BCAM - Basque Center for Applied Mathematics, Spain

Unai Alonso, University of the Basque Country, Spain

Michael Barton, BCAM - Basque Center for Applied Mathematics, Spain

Jérémy Bleyer, Ecole des Ponts ParisTech, France

Thomas Hefler, CEA (French Alternative Energies and Atomic Energy Commission), France

Raffaele Russo, University of the Basque Country, Spain

22 March 2021

When exposed to high strain rates of deformation, metal alloys with low thermal conductivity are typically liable to large quantities of plastic strain. This phenomenon of localization of high temperatures in a narrow band, known as an adiabatic shear band, occurs in many manufacturing processes such as machining or metal forming. We aim to simulate it based on the research of Forest *et al* [2] and Russo *et al* [4] that propose a thermodynamically consistent framework for both elastoplasticity and elastoviscoplasticity. Their approach builds upon a bulk of work stemming from that of the Cosserat brothers in 1909 [1] on generalized continua by introducing independent translational and rotational degrees of freedom.

Our contribution is the implementation of the elastoplasticity framework using FEniCS, the MFront library [3], and the corresponding interface between them, the mgis.fenics module. These tools automate the implementation of material behaviours and allow for complex models. We present the results for the glide and bending/torsion test, and their validation against another finite element solver (Zset) and the analytical solution. Moreover, we run the performance analysis on the Atlas-EDR cluster, at the Donostia International Physics Center (DIPC). We break down the execution time, identify the most computationally intensive components, and analyze the results from the point of view of parallel scaling.

## References

- [1] Eugene Cosserat and François Cosserat. “Theorie des corps déformables”. 1909.
- [2] Samuel Forest and Rainer Sievert. “Elastoviscoplastic constitutive frameworks for generalized continua”. In: *Acta Mechanica* 160.1 (2003), 71–111. DOI: 10.1007/s00707-002-0975-0.
- [3] Thomas Helfer, Bruno Michel, Jean-Michel Proix, Maxime Salvo, Jérôme Sercombe, and Michel Casella. “Introducing the open-source mfront code generator: Application to mechanical behaviours and material knowledge management within the pleiades fuel element modelling platform”. In: *Computers & Mathematics with Applications* 70.5 (2015), 994–1023. DOI: 10.1016/j.camwa.2015.06.027.
- [4] Raffaele Russo, Samuel Forest, and Franck Andrés Giroto Mata. “Thermomechanics of cosserat medium: modeling adiabatic shear bands in metals”. In: *Continuum Mechanics and Thermodynamics* (2020), 1–20. DOI: 10.1007/s00161-020-00930-z.

---

You can cite this talk as:

Tamara Dancheva, Unai Alonso, Michael Barton, Jérémy Bleyer, Thomas Hefler, and Raffaele Russo. “mgis.fenics Part II: Cosserat media in small deformation with mgis.fenics”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 121–136. DOI: 10.6084/m9.figshare.14495241.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/dancheva.html>.

# enable

## mgis.fenics Part II: Cosserat media in small deformation with mgis.fenics

T. Dancheva<sup>(1)</sup>, U. Alonso<sup>(2)</sup>, M. Barton<sup>(1)</sup>, J. Bleyer<sup>(3)</sup>, T. Heffler<sup>(4)</sup>, R. Russo<sup>(2)</sup>

<sup>(1)</sup> BCAM - Basque Center for Applied Mathematics

<sup>(2)</sup> University of the Basque Country

<sup>(3)</sup> Laboratoire Navier UMR 8205 (École des Ponts ParisTech-IFSTTAR-CNRS), France

<sup>(4)</sup> CEA, DES, IRESNE, DEC, SESC, LSC, Cadarache, France



This project received funding from the European Union's Marie Skłodowska-Curie Actions (MSCA) Innovative Training Networks (ITN) H2020-MSCA-ITN-2017 under the grant agreement N°764979.

# AGENDA

- ❖ Introduction
- ❖ Cosserat media:
  - Motivation
  - Model
  - Implementation
- ❖ Performance
- ❖ Conclusion
- ❖ Next steps

# Introduction

## ENABLE H2020 project

This European Training Network actively involves academics and industrial partners in training a new generation of young researchers for the future of **manufacturing**. By developing new solutions for **metallic alloys**, ENABLE proposes a complete rethink of the usual process simulation methods. Innovative **multiscale** (from microscopic to macroscopic scales), and **multi-physics** (strong thermomechanical and microstructural couplings) are addressed.



## Cosserat Media - Motivation

- Development of an **Adiabatic Shear Band**
- Localization phenomena & prediction of characteristic length and size effect
- aim to regularize the model and avoid mesh dependency

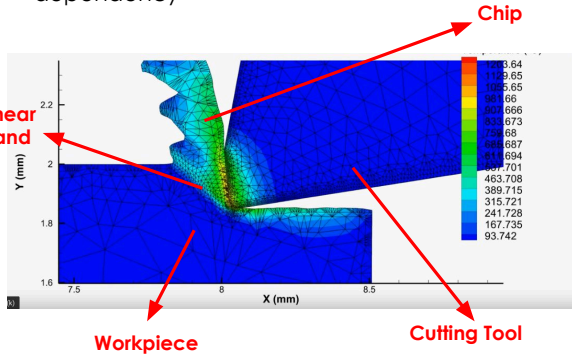


Fig.1 2D machining of Ti-6Al-4V - using Third Wave Systems AdvantEdge - Temperature

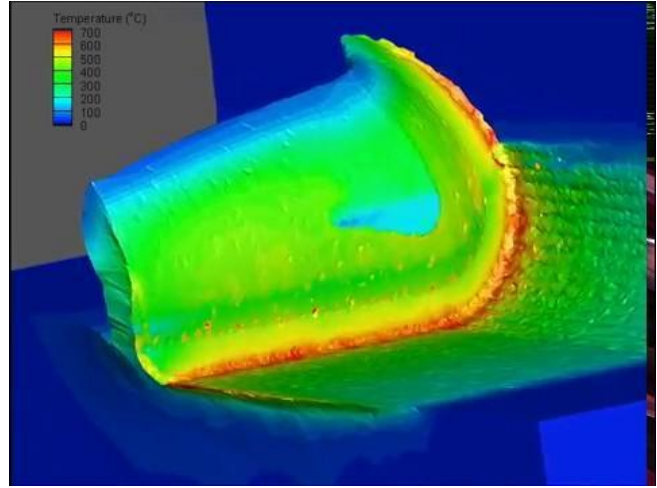


Fig.2 Chip formation during the machining of grade 316L stainless steel - using Third Wave Systems AdvantEdge - Temperature, courtesy of Sandvik Coromant

# Cosserat Media in small deformation - model

- ❖ the model was initially introduced in 1909 by the Cosserat brothers [Cosserat 1909]
- ❖ Raffaele Russo has been working on formulating a thermodynamically consistent model for small deformation and large deformation [Russo et al. 2020]

Displacement  $\leftarrow$   $\{u_i, \theta_i\}, i = 1, 2, 3$   $\rightarrow$  Extra degrees of freedom - the rotation of the microstructure

Deformation measures, where  $\epsilon_{ijk} = \begin{cases} 1, & \text{if } (i,j,k) = (1,2,3), (2,3,1) \text{ or } (3,1,2); \\ -1, & \text{if } (i,j,k) = (3,2,1), (2,1,3) \text{ or } (1,3,2); \\ 0, & \text{otherwise.} \end{cases}$

$\underline{\mathbf{e}} = \underline{\mathbf{u}} \otimes \nabla + \underline{\underline{\epsilon}} \cdot \underline{\underline{\theta}} \rightarrow$  Cosserat deformation tensor

$\underline{\underline{\mathbf{k}}} = \underline{\underline{\theta}} \otimes \nabla \rightarrow$  Cosserat wryness tensor

Balance/equilibrium equation:

$$\int_{\Omega} \left( \underline{\underline{\sigma}} : \dot{\underline{\underline{e}}} + \underline{\underline{\mu}} : \dot{\underline{\underline{k}}} \right) dV = \int_{\Omega} \left( \underline{\underline{\mathbf{f}}} \cdot \dot{\underline{\underline{u}}} + \underline{\underline{\mathbf{c}}} \cdot \dot{\underline{\underline{\theta}}} \right) dV + \int_{\partial\Omega} \left( \underline{\underline{\mathbf{t}}} \cdot \dot{\underline{\underline{u}}} + \underline{\underline{\mathbf{m}}} \cdot \dot{\underline{\underline{\theta}}} \right) dS;$$

$\underline{\underline{\sigma}}$   $\rightarrow$  classical stress     
  $\underline{\underline{\mu}}$   $\rightarrow$  couple stress     
  $\underline{\underline{\mathbf{f}}}$   $\rightarrow$  body force / couple     
  $\underline{\underline{\mathbf{c}}}$   $\rightarrow$  couple     
  $\underline{\underline{\mathbf{t}}}$   $\rightarrow$  external surface / couple traction     
  $\underline{\underline{\mathbf{m}}}$   $\rightarrow$  couple traction



# Cosserat Media in small deformation - model

## ❖ Material model for elasto-plasticity

From the Helmholtz free energy and the Clausius-Duhem inequality (2<sup>nd</sup> thermodynamic law) we can verify the compatibility and derive the following:

- assuming single plastic multiplier we calculate using the consistency condition and the

normality rule: 
$$\dot{p} = \frac{\underline{\mathbf{n}} : \underline{\underline{\Lambda}} : \dot{\underline{\mathbf{e}}} + \underline{\mathbf{n}}_c : \underline{\underline{\mathbf{C}}} : \dot{\underline{\mathbf{k}}}}{\frac{\partial A}{\partial p} + \underline{\mathbf{n}}_c : \underline{\underline{\Lambda}} : \underline{\mathbf{n}} + \underline{\mathbf{n}}_c : \underline{\underline{\mathbf{C}}} : \underline{\mathbf{n}}};$$

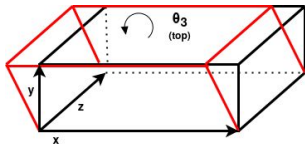
$$\frac{\partial f}{\partial \underline{\underline{\boldsymbol{\sigma}}}} = \underline{\mathbf{n}} = \frac{3}{2} \frac{a_1 \underline{\underline{\boldsymbol{\sigma}}}' + a_2 \underline{\underline{\boldsymbol{\sigma}}}'^T}{\sigma_{eq}};$$

- Normals to the yield surface in the stress and couple stress spaces 
$$\frac{\partial f}{\partial \underline{\underline{\boldsymbol{\mu}}}} = \underline{\mathbf{n}}_c = \frac{3}{2} \frac{b_1 \underline{\underline{\boldsymbol{\mu}}} + b_2 \underline{\underline{\boldsymbol{\mu}}}^T}{\sigma_{eq}};$$
- Equivalent Stress as in [Borst 1991; Lippmann 1969; Mühlhaus and Vardoulakis 1987]

$$\sigma_{eq} = \sqrt{\frac{3}{2} (a_1 \underline{\underline{\boldsymbol{\sigma}}}' : \underline{\underline{\boldsymbol{\sigma}}}' + a_2 \underline{\underline{\boldsymbol{\sigma}}}' : \underline{\underline{\boldsymbol{\sigma}}}'^T + b_1 \underline{\underline{\boldsymbol{\mu}}} : \underline{\underline{\boldsymbol{\mu}}} + b_2 \underline{\underline{\boldsymbol{\mu}}} : \underline{\underline{\boldsymbol{\mu}}}^T)};$$

Characteristic length: 
$$l_p = \sqrt{\frac{a}{b}};$$

# Cosserat Media Implementation - Glide test



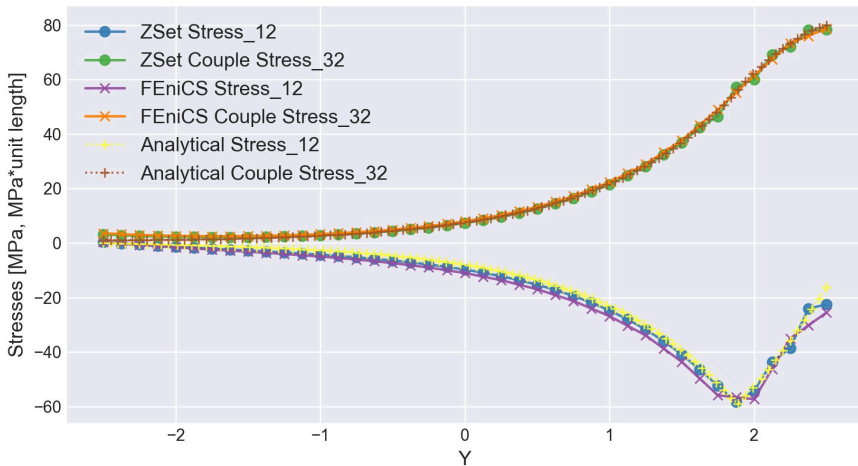
## Boundary Conditions

$u_3 = 0$  (whole)

$\theta_1 = 0$  (whole)

$\theta_2 = 0$  (whole)

$\theta_3 = 0.001$  (top)



Iterations to converge - equilibrium:

- FEniCS:

[1,1,1,1,1,3,7,8,8,9,10,10,11,12,13,15,16,18]

- Zset:[1,1,1,1,1,4,5,6,6,6,7,7,8,9,10,10,9]

Fig.3 Comparison MFront+FEniCS with ZSet and the analytical solution [S.Forest et al.]

# Cosserat Media Implementation

Fig. 4 Explicit Implementation MFront

```
// Note @InitLocalVariables is called before @PredictionOperator and
// @TangentOperator and @Derivative, so we can define the elastic operator here
// to factorize the code a little bit
@InitLocalVariables {
    ∂σ/∂Δε = lambda * t2tot2<N, real>::IxI() +
              (mu + mu_c) * t2tot2<N, real>::Id() +
              (mu - mu_c) * t2tot2<N, real>::transpose_derivative();
    ∂μ<sub>κ</sub>/∂Δκ = alpha * t2tot2<N, real>::IxI() +
                  (beta + gamma) * t2tot2<N, real>::Id() +
                  (beta - gamma) * t2tot2<N, real>::transpose_derivative();
}

@ComputeThermodynamicForces {
    σ = lambda * trace(ε<sup>e</sup>) * Tensor::Id() + 2 * mu * sym(ε<sup>e</sup>) +
        mu_c * (ε<sup>e</sup> - transpose(ε<sup>e</sup>));
    μ<sub>κ</sub> = alpha * trace(κ<sup>e</sup>) * Tensor::Id() + 2 * beta * sym(κ<sup>e</sup>) +
            gamma * (κ<sup>e</sup> - transpose(κ<sup>e</sup>));
}
```

$$\underline{\underline{\Lambda}} = \begin{bmatrix} (\lambda + 2\mu) & \lambda & \lambda & & & \\ \lambda & (\lambda + 2\mu) & \lambda & & & \\ \lambda & \lambda & (\lambda + 2\mu) & & & \\ & & & (\mu + \mu_c) & (\mu - \mu_c) & \\ & & & (\mu + \mu_c) & (\mu - \mu_c) & \\ & & & & (\mu + \mu_c) & (\mu - \mu_c) \\ & & & & & (\mu + \mu_c) & (\mu - \mu_c) \\ & & & & & & (\mu + \mu_c) & (\mu - \mu_c) \end{bmatrix}$$

$$\underline{\underline{\mathbb{C}}} = \begin{bmatrix} (\alpha + 2\beta) & \alpha & \alpha & & & \\ \alpha & (\alpha + 2\beta) & \alpha & & & \\ \alpha & \alpha & (\alpha + 2\beta) & & & \\ & & & (\beta + \gamma) & (\beta - \gamma) & \\ & & & (\beta + \gamma) & (\beta - \gamma) & \\ & & & & (\beta + \gamma) & (\beta - \gamma) \\ & & & & & (\beta + \gamma) & (\beta - \gamma) \\ & & & & & & (\beta + \gamma) & (\beta - \gamma) \end{bmatrix}$$

$$\underline{\underline{\sigma}} = \lambda \text{trace}(\underline{\underline{\mathbf{e}}}^e) \underline{\underline{\mathbf{I}}} + 2\mu (\underline{\underline{\mathbf{e}}}^e)^{sym} + 2\mu_c (\underline{\underline{\mathbf{e}}}^e)^{skew}$$

$$\underline{\underline{\mu}} = \alpha \text{trace}(\underline{\underline{\mathbf{k}}}^e) \underline{\underline{\mathbf{I}}} + 2\beta (\underline{\underline{\mathbf{k}}}^e)^{sym} + 2\gamma (\underline{\underline{\mathbf{k}}}^e)^{skew}$$

Fig.4 .mfront file for the Cosserat glide test

## Cosserat Media Implementation

Fig. 5 Explicit Implementation MFront

```
@Derivative {
  const auto se = 2 * mu * deviator(syme(εe1)) + mu_c * (εe1 - transpose(εe1));
  const auto seq = sqrt(3 * (a_1 * (se | se) + a_2 * (se | transpose(se)) +
                             b_1 * (μk | μk) + b_2 * (μk | transpose(μk))) /
                        2);
  d_tεe1 = d_teto;
  d_tκe1 = d_tκ;
  if (seq - R0 - H * p > 0) {
    const auto n = eval(3 * (a_1 * se + a_2 * transpose(se)) / (2 * seq));
    const auto n_c = eval(3 * (b_1 * μk + b_2 * transpose(μk)) / (2 * seq));
    const auto cste = 1 / (H + (n | ∂σ/∂Δeto | n) + (n_c | ∂μk/∂Δκ | n_c));
    d_t p = ((n | ∂σ/∂Δeto | d_teto) + (n_c | ∂μk/∂Δκ | d_tκ)) * cste;
    d_tεe1 -= d_t p * n;
    d_tκe1 -= d_t p * n_c;
  }
}
```

$$\sigma_{eq} = \sqrt{\frac{3}{2} (a_1 \boldsymbol{\sigma}' : \boldsymbol{\sigma}' + a_2 \boldsymbol{\sigma}' : \boldsymbol{\sigma}'^T + b_1 \boldsymbol{\mu} : \boldsymbol{\mu} + b_2 \boldsymbol{\mu} : \boldsymbol{\mu}^T)};$$

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \mathbf{n} = \frac{3}{2} \frac{a_1 \boldsymbol{\sigma}' + a_2 \boldsymbol{\sigma}'^T}{\sigma_{eq}};$$

$$\frac{\partial f}{\partial \boldsymbol{\mu}} = \mathbf{n}_c = \frac{3}{2} \frac{b_1 \boldsymbol{\mu} + b_2 \boldsymbol{\mu}^T}{\sigma_{eq}};$$

$$\dot{p} = \frac{\mathbf{n} : \boldsymbol{\Lambda} : \dot{\boldsymbol{\varepsilon}} + \mathbf{n}_c : \mathbf{C} : \dot{\boldsymbol{\kappa}}}{\frac{\partial A}{\partial p} + \mathbf{n}_c : \boldsymbol{\Lambda} : \mathbf{n} + \mathbf{n}_c : \mathbf{C} : \mathbf{n}};$$

Fig.5 .mfront file for the Cosserat glide test - continuation  
Optimization: converting to an implicit implementation

# Cosserat Media Implementation

FEniCS + MFront



Zset

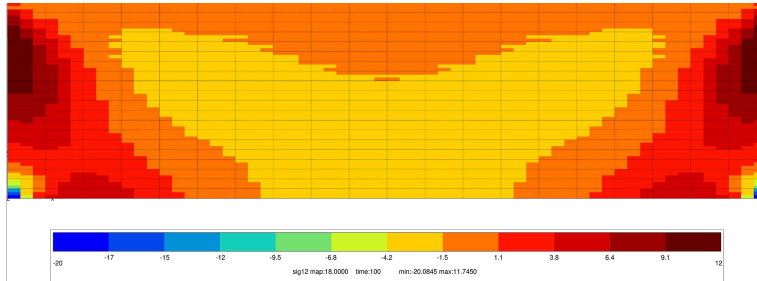
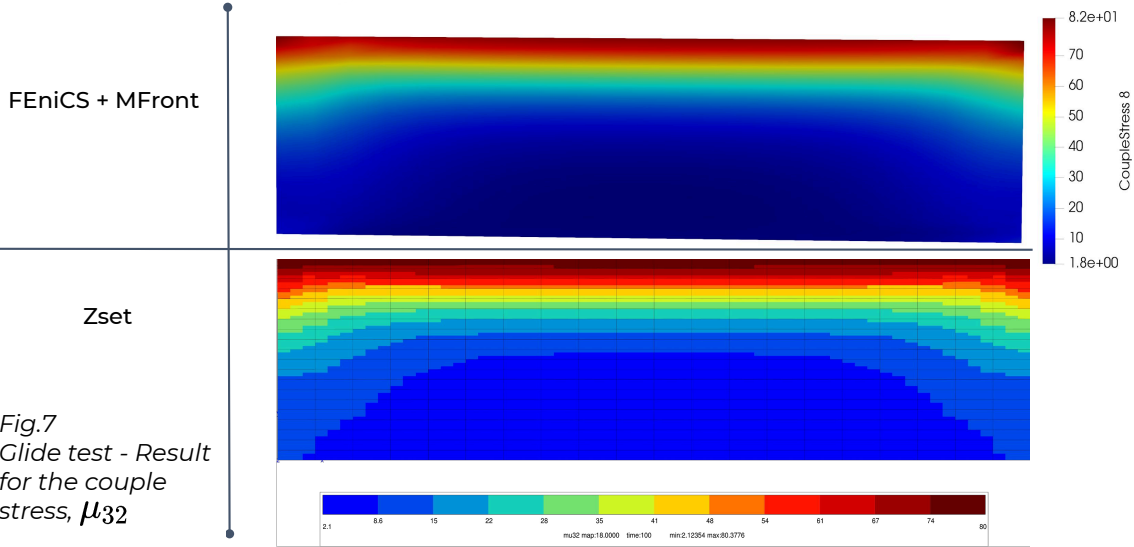
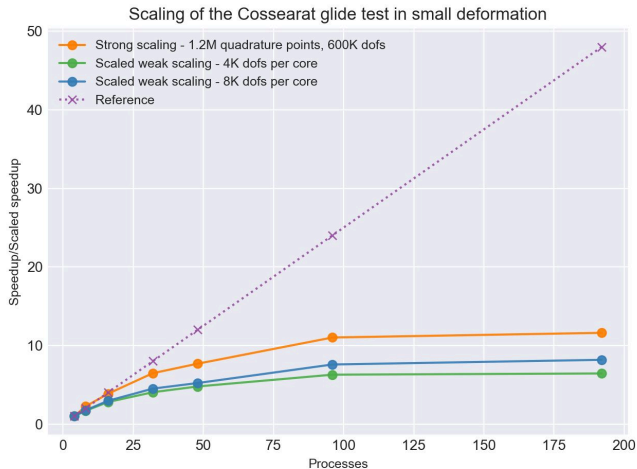


Fig.6  
Glide test - Result  
for the stress,  $\sigma_{12}$

# Cosserat Media Implementation



# Performance



## ATLAS EDR @ Donostia International Physics Center

- Infiniband EDR network
- 37 nodes with Intel Xeon Platinum 8168 (24 cores per node x 2 threads)
- 8 nodes with Intel Xeon Platinum 8280 (28 cores per node x 2 threads)
- 2x NVIDIA Tesla P40, 1x NVIDIA Tesla P40

## Current setup:

- using Singularity container
- using MPICH using the UCX network framework

Fig.8 Strong and weak scaling plot for the glide test

## Conclusions

- From the profiling and scaling results we can conclude that the major bottleneck is the resolution of the system of nonlinear equations (quasi Newton line search) using MUMPS as a linear solver

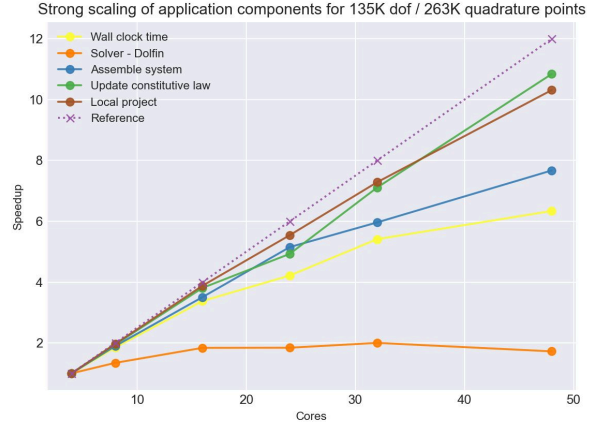
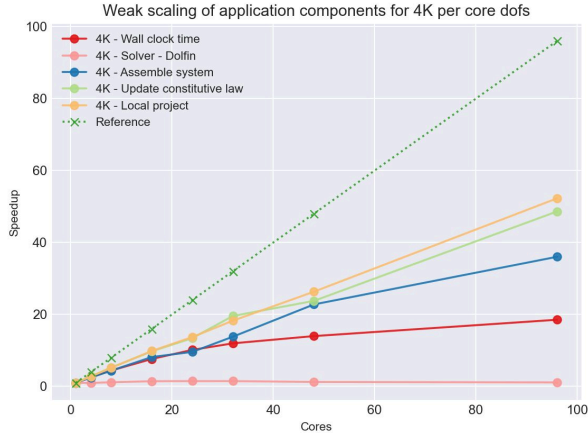


Fig.9 Strong and weak scaling plot for various routines part of the simulation



## Next steps

- ❖ Increase problem size
- ❖ Native installation of the software stack on ATLAS-EDR
- ❖ Profiling with EXTRAE for MPI statistics, DCRAB for node statistics
- ❖ Exploring other linear solvers (and nonlinear)
- ❖ Implicit scheme implementation
- ❖ Porting to dolfin-x
- ❖ Further HPC analysis and code optimizations
- ❖ Implementation of the full thermodynamically consistent Cosserat model in Large deformation - elasto viscoplasticity

## Questions



Thank you for your  
attention!

# FFCx code generation for expressions

Michal Habera, University of Luxembourg, Luxembourg

Andreas Zilian, University of Luxembourg, Luxembourg

23 March 2021

The FEniCSx Form Compiler (FFCx) compiles the symbolic weak formulation of PDEs into C code.

Historically, its primary focus was on integral operators originating from weak forms. Using a quadrature rule it generated code to evaluate integrals. However, a number of applications appeared where symbolic UFL expressions not involving integrals needed to be compiled into C code—including interpolation into point-evaluation function spaces.

In this contribution a code generation for UFL symbolic expressions is described. Code generation and optimisation techniques of FFCx are briefly reviewed and improvements needed to compile C code for non-integral, tensor-valued expressions are described.

New functionality is demonstrated with several examples: efficient interpolation into Lagrange point-evaluation spaces and assembly of interpolation and other non-integral operators (discrete gradient, divergence, curl, etc.). Examples are prepared using the development version of FEniCSx.

It is believed that this extension to FFCx will be used in common practice by FEniCS users, improve their code performance and widen the applicability of the entire FEniCS package environment.

---

You can cite this talk as:

Michal Habera and Andreas Zilian. “FFCx code generation for expressions”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 137. doi: 10.6084/m9.figshare.14495247.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/habera.html>.

# Explicit dual space representation in UFL

India Marsden, Department of Computing, Imperial College London, United Kingdom

David A Ham, Department of Mathematics, Imperial College London, United Kingdom

Reuben Nixon-Hill, Department of Mathematics, Imperial College London, United Kingdom

23 March 2021

This talk will discuss proposed changes to the Unified Form Language to include symbolic types representing dual spaces along with associated objects and functions. UFL represents forms over finite element spaces, and operations on these forms naturally results in objects in the dual space, or operators mapping to or from dual spaces. Since UFL currently does not have a representation of these objects, the language is not closed, meaning these operations result in objects outside of the language, which these changes aim to solve.

We will discuss the changes being made and their structure, the mathematical background and the potential benefits, applications and simplifications that this work enables.

This talk was awarded a prize: Best talk by a PhD student or undergraduate (runner up).

---

You can cite this talk as:

India Marsden, David A Ham, and Reuben Nixon-Hill. “Explicit dual space representation in UFL”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 138–150. DOI: 10.6084/m9.figshare.14495250.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/marsden.html>.

# Explicit Dual Space Representation in UFL

---

India Marsden<sup>1</sup>, David A. Ham<sup>2</sup> and Reuben Nixon-Hill<sup>2,3</sup>

March 2021

<sup>1</sup>Department of Computing, Imperial College London

<sup>2</sup>Department of Mathematics, Imperial College London

<sup>3</sup>Science and Solutions for a Changing Planet DTP, Grantham Institute for Climate Change and the Environment, Imperial College London



UFL provides an intuitive way to represent mathematical forms in code.

In particular, it is able to represent function spaces, finite elements within function spaces and functions on these spaces, among other things.



Typically, operations such as assemble are applied to the defined forms in UFL. Doing this results in objects that are not within UFL.

This means that the language is not *closed*.

```
element = FiniteElement("Lagrange", triangle, 1)
u = TrialFunction(element)
v = TestFunction(element)
f = Coefficient(element)

a = (u*v - inner(grad(u), grad(v))) * dx
L = f * v * dx
res = assemble(a)
res2 = assemble(L)
```



**Operator Composition** Where  $\tau(u)$  is an external operator:

$$\text{grad}(u) \cdot \tau(u) \cdot \text{grad}(v) * dx$$

**Interpolation** Interpolation is not first class

$$\text{interp}(e, u) * v * dx$$

**Adjoint Forward Operations**

$$\text{action}(\text{interp}^*(\hat{e}, u), \text{adjoint}(u * v * dx))$$

**Composing Assembled forms**

$$\text{assemble}(v * dx + \text{assemble}(e * dx))$$





These operations depend on objects in the *dual* to the function space, the space of bounded linear functionals on  $V$ :

$$V^* = V \rightarrow \mathbb{R}$$

An example of an operation on a dual space is the Dirac Delta functional ( $V^* \rightarrow \mathbb{R}$ ), ie point evaluation:

$$\delta_x(v) = v(x)$$



A function space can be represented by its (primal) basis. A function in the space is then a set of coefficients of that basis:

$$v = v_i \phi_i \in V$$

A dual space can be similarly represented by *dual basis* functions,  $\phi^* \in V \rightarrow \mathbb{R}$ . Call the set of coefficients of a dual space a *cofunction*:

$$u = u_i \phi_i^* \in V^*$$

Writing  $u(v)$  would be evaluation of the dual basis and result in a scalar.



In UFL, a 1-form represents a mathematical object with one unknown, such as below, which we can write in terms of the basis:

$$\begin{aligned} h(v) &= \int_{\Omega} v \, dx = \int_{\Omega} \phi_i \, dx \, v_i \\ &= \int_{\Omega} \phi_i \, dx \, l_{ij} v_j = \int_{\Omega} \phi_i \, dx \, \phi_i^*(\phi_j) v_j \\ &= \int_{\Omega} \phi_i \, dx \, \phi_i^*(v_j \phi_j) \\ &= \int_{\Omega} \phi_i \, dx \, \phi_i^*(v) \end{aligned}$$

Using the property  $\phi_i^*(\phi_j) = \delta_{ij}$  and the linearity of the dual basis.



Therefore, we can see that 1-forms can be represented as cofunctions with coefficients:

$$h_i = \int_{\Omega} \phi_i dx$$
$$h = h_i \phi^*$$

This is a cofunction, an object in the dual space of  $V$ .  
Computationally, we write:

$$L = v \star dx$$
$$\text{obj} = \text{assemble}(L)$$

Obviously, **obj** is not a current UFL object.



Define interpolation from a space  $U$  to a space  $V$  as the operator:

$$\text{interp}(u, v^*) : U \rightarrow V$$

We can write this as a form:

$$U \times V^* \rightarrow \mathbb{R}$$

As  $V = V^{**} = V^* \rightarrow \mathbb{R}$ . Then, taking the adjoint of this form we get:

$$V^* \times U \rightarrow \mathbb{R} = V^* \rightarrow U^*$$

which matches the expectation of linear operators.



Seeing interpolation as a function, we have the first argument as  $u \in U$  and the second  $v^* \in V$ . Interpolation is dual evaluation of  $v^*$ :

$$\text{interp}(u, v^*) = v^*(u)$$

$v^*$  is termed a *coargument*, and in code would be:

```
v_star = TestFunction(V.dual())
```

Introducing Cofunctions makes the adjoint behave correctly.



With these draft additions, users will be able to write code such as:

```
V = FunctionSpace(domain, element)
v = TestFunction(V)
V_dual = V.dual()
L = v * dx
obj = assemble(L)
a = Cofunction(V)
res = a + obj
```

where `res` would be a valid operation and `V_dual` is the function space that is dual to `V`.



This change will need to be propagated into the implementation of *assemble* and other similar operations. This includes attaching data to these objects and adapting the implementations to take into account pre-assembled sections.



# Turning FEniCS inside out

Chris Richardson, BP Institute, University of Cambridge, United Kingdom

23 March 2021

Previous versions of FEniCS have focused on adding more features and functionality by adding interfaces to an increasing list of third party libraries, for linear algebra, plotting, mesh partitioning, etc. In the latest version of FEniCSx we are moving away from this trend, on the assumption that users can interface third-party libraries themselves, given the right interface to FEniCS.

This can be viewed as “turning FEniCS inside-out” as the previous internal workings have become public interfaces, and FEniCS itself is simplified and reduced.

I will show some examples using the Trilinos solvers, custom mesh partitioning and how to access data directly for I/O and visualisation.

---

You can cite this talk as:

Chris Richardson. “Turning FEniCS inside out”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 151–163. doi: 10.6084/m9.figshare.14495253.

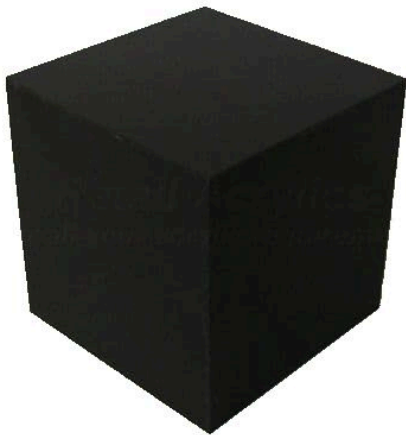
BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/richardson.html>.

# Turning FEniCS inside-out

Chris Richardson  
chris@bpi.cam.ac.uk

# DOLFIN 2017

```
GenericMatrix::set_local(...);
```



PETSc



Eigen



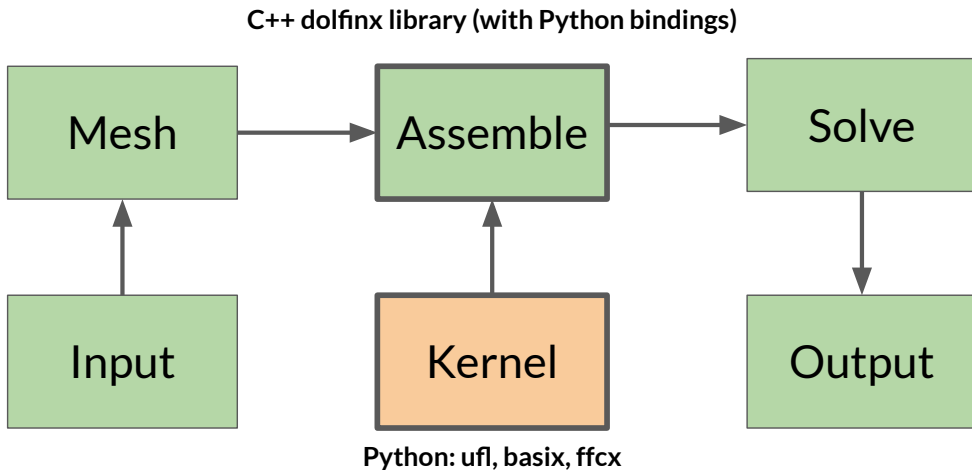
Trilinos

```
GenericTransport::move(src, dest);
```



# FEniCS-X: A modular Finite Element code

“Keep it simple and give the user freedom to do things their own way”



# Matrix assembly plugins/callbacks

```
dolfinx::fem::assemble_matrix(...)
```

```
for cells in mesh:  
    geom = geometry(cell)  
    ...  
    Ae = tabulate_tensor(geom, ...)  
    global_mat_insert(Ae, ...)
```

mat\_insert()



A black arrow points from the cyan box containing 'mat\_insert()' to the 'global\_mat\_insert(Ae, ...)' line in the code block above.

fem::Form

tabulate\_tensor()



A black arrow points from the orange box containing 'fem::Form' and 'tabulate\_tensor()' to the 'Ae = tabulate\_tensor(geom, ...)' line in the code block above.

## Turning things inside out: C++ lambda functions

```
auto mat_insert = [&A](int nrows, int* rows,  
                     int ncols, int* cols, double *Ael)  
{  
    for (int i = 0; i < nrows; ++i)  
        for (int j = 0; j < ncols; ++j)  
            A[rows[i], cols[j]] += Ael[i*ncols + j];  
}  
  
dolfinx::fem::assemble_matrix(mat_insert, form, bcs);
```

## 1. Using other linear algebra backends: Trilinos

```
Tpetra::CrsMatrix<double> A;

auto mat_insert = [&A](int nrows, int* rows,
                      int ncols, int* cols, double *Ael)
{
    for (int i = 0; i < nrows; ++i)
    {
        ArrayView<const int> col_view(cols, ncols);
        ArrayView<double> col_view(Ael + i*ncols, nc);
        for (int j = 0; j < ncols; ++j)
            A.sumIntoLocalValues(rows[i], col_view, val_view);
    }
}
```



## 2. Lumping of mass matrix into diagonal

```
std::vector<double> diag;

auto vec_insert = [&diag](int nrows, int* rows,
                        int ncols, int* cols, double *Ael)
{
    for (int i = 0; i < nrows; ++i)
        for (int j = 0; j < ncols; ++j)
            diag[rows[i]] += Ael[i*ncols + j];
}

fem::assemble_matrix(vec_insert, form, bcs);
```

### 3. Action of a form (matrix-free)

```
auto mat_apply = [&uvec, &wvec](int nr, const int* rows,
                                int nc, const int* cols, const double* Ae)
{
    for (int i = 0; i < nr; ++i)
        for (int j = 0; j < nc; ++j)
            wvec[rows[i]] += Ae[i * nc + j] * uvec[cols[j]];
};

fem::assemble_matrix(mat_apply, form, bcs);
```

## 4. Assemble diagonal, and action of $L+U \rightarrow$ Jacobi

```
std::vector<double> diag, w, u;

auto insert_diag = [&diag] (int nrows, int* rows,
                           int ncols, int* cols, double *Ael)
{
    for (int i = 0; i < nrows; ++i)
        diag[rows[i]] += Ael[i*ncols + i];
}

auto apply_lu = [&w, &u] (int nrows, int* rows,
                          int ncols, int* cols, double *Ael)
{
    for (int i = 0; i < nrows; ++i)
        for (int j = 0; j < ncols; ++j)
            if (j != i)
                w[rows[i]] += Ael[i*ncols + j] * u[cols[j]];
}
```



Python version? But don't do this without an adult present...

```
a = inner(grad(u), grad(v))*dx
a_form = fem.Form(a)
rdata = []
cdata = []
vdata = []

def mat_insert(rows, cols, vals):
    vdata.append(vals)
    rdata.append(numpy.repeat(rows, len(cols)))
    cdata.append(numpy.tile(cols, len(rows)))
    return 0

# Using Python callback is SLOW...
dolfinx.cpp.fem.assemble_matrix(mat_insert, a_form._cpp_object, [])
scipy.sparse.coo_matrix((vdata, (rdata, cdata)))
```

# Summary

- FEniCS-X is more open to experimentation at the low level
- Plugin functionality via `std::function` in C++ for matrix insertion
  - FEniCS-X need know nothing about your LA backend
- Can also prototype in Python (but don't do this for a real application)
  - Better to use e.g. *cppimport* to wrap a snippet
- There are a number of things you can do with `fem::assemble_matrix`
- Mesh partitioning has some similar plugins with `std::function`

# Making point sets first class in UFL and Firedrake

**Reuben W. Nixon-Hill** (<https://www.imperial.ac.uk/people/reuben.nixon-hill10>),  
Science and Solutions for a Changing Planet DTP & Department of Mathematics, Imperial  
College London, United Kingdom

Daniel Shapero, Polar Science Center, Applied Physics Laboratory, University of Washington,  
United States

Colin J. Cotter, Department of Mathematics, Imperial College London, United Kingdom

David A. Ham, Department of Mathematics, Imperial College London, United Kingdom

**23 March 2021**

Point data turn up all the time in scientific computing problems—perhaps you want to know the value of a field at a particular point or set of points, or you have point data that you want to assimilate into a model of some phenomenon. At the moment points are typically expressed as lists of coordinates, thereby bypassing the Unified Form Language’s (UFL’s) type system of function spaces and meshes.

In this talk I will discuss how new concepts such as a “Mesh of Disconnected Vertices” allow point sets to be treated as vectors in a finite element vector space with appropriate UFL arguments and coefficients. I will show how an interpolation operator, onto a function space on such a mesh, can be constructed to represent point evaluations. Since such an operator can be differentiated it fits neatly into the pyadjoint/dolfin-adjoint ecosystem allowing PDE constrained optimisation problems to be solved when, for example, assimilating point data. I will also discuss how this could impact future work to turn UFL into a domain specific language (DSL) for expressing and automating diagnostics on big field datasets produced by climate models.

---

You can cite this talk as:

Reuben W. Nixon-Hill, Daniel Shapero, Colin J. Cotter, and David A. Ham. “Making point sets first class in UFL and Firedrake”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 164–182. DOI: 10.6084/m9.figshare.14495256.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/nixon-hill.html>.

# Making point sets first class in UFL and Firedrake

**Reuben W. Nixon-Hill**<sup>1,2</sup>, Daniel Shapero<sup>3</sup>, Colin J. Cotter<sup>2</sup>, David A. Ham<sup>2</sup>

<sup>1</sup> Science and Solutions for a Changing Planet DTP, Grantham Institute for  
Climate Change and the Environment, Imperial College London

<sup>2</sup> Department of Mathematics, Imperial College London

<sup>3</sup> Polar Science Center, Applied Physics Laboratory, University of Washington

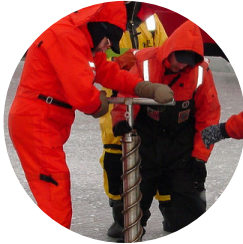
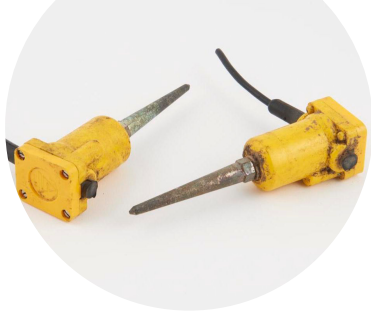
**Grantham Institute**  
Climate Change and the Environment  
An Institute of Imperial College London

Science and  
Solutions for a  
Changing Planet  
DTP



Natural  
Environment  
Research Council





## Why care about point data?

- Geoscience Examples
  - Ice Sheets: elevation from **satellite altimetry** and **ice cores**
  - Ocean: salinity and temperature from **drifting buoys**
  - Atmosphere Climate: conditions at **weather stations**
  - **And more!**
- Typical uses
  - **Point evaluation** of PDE solutions
  - PDE Constrained Optimisation
    - Variational **data assimilation**
    - Goal based error estimation
    - **And more!**

"File:Weather Buoy MDS.jpg" by MDS is licensed with CC BY-SA 4.0. To view a copy of this license, visit <https://creativecommons.org/licenses/by-sa/4.0/>

"Garden Wall Weather Station, MT" by U.S. Geological Survey is marked under CC0 1.0. To view the terms, visit <https://creativecommons.org/licenses/by-sa/4.0/>

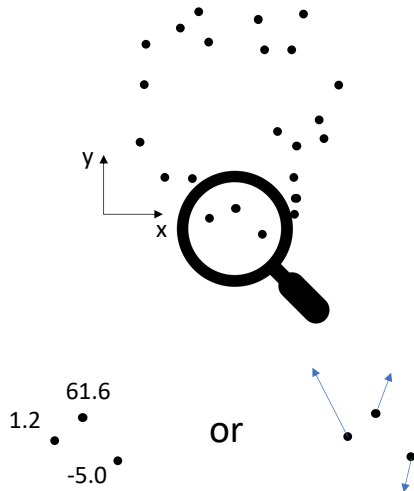
"Ice core sampling in Green Bay, Lake Michigan" by NOAA Great Lakes Environmental Research Laboratory is licensed with CC BY-SA 2.0. To view a copy of this license, visit <https://creativecommons.org/licenses/by-sa/2.0/>

"2 Geospace PE3 geophones, for near-field studies," is licensed with CC BY-NC-SA 4.0. To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-sa/4.0/>



# Definition: Point Data

1. A “point cloud”  $\{X_i\}$  – a set of spatial coordinates and
2. Values  $y_i$  (scalar, vector or tensor valued) at those coordinates.



# How we deal with point data at the moment


```
# UFL  
# evaluate coefficient f at two  
# 2-dimensional points  
vals[0] = f([0.2, 0.4])  
vals[1] = f([1.2, 0.5])
```

```
# Firedrake (dolfin similar)  
# evaluate Function f at two  
# 2-dimensional points  
from firedrake import *  
...  
vals = f.at([0.2, 0.4], [1.2, 0.5])
```

- UFL expressions of coefficients can be evaluated at given point coordinates
- Can get values from point clouds

# How we deal with point data at the moment

```
# UFL
# evaluate coefficient f at two
# 2-dimensional points
vals[0] = f([0.2, 0.4])
vals[1] = f([1.2, 0.5])
```

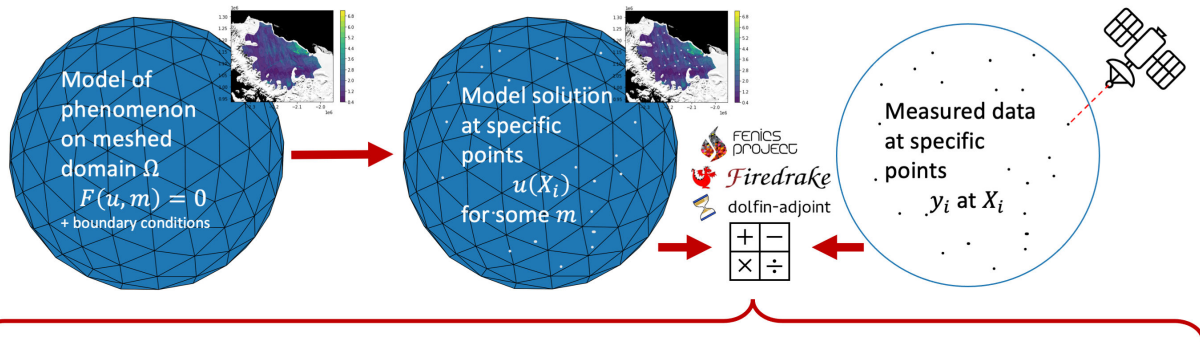
- **outside of the UFL type system of fields in function spaces on meshes**
- Value driven: no symbolic point evaluation.
- ~~ dolfin adjoint~~

```
# Firedrake
# evaluate Function f at two
# 2-dimensional points
from firedrake import *
...
vals = f.at([0.2, 0.4], [1.2, 0.5])
```

- end up special-casing point evaluation code pathways
- Often slow!

# Example Use Case

## Point Data Assimilation: A PDE Constrained Optimisation Problem



Minimization Problem:

$$\min_{u, m} J[u, m] \quad \text{where } J = \underbrace{\|u(X_i) - y_i\|^2}_{\text{misfit}} + \underbrace{c\|m\|^2}_{\text{regularization}}$$

subject to

$$F(u, m) = 0$$

+ boundary conditions

misfit  
(Fit of model  
to data)

regularization  
(e.g. to ensure  
smooth fit)



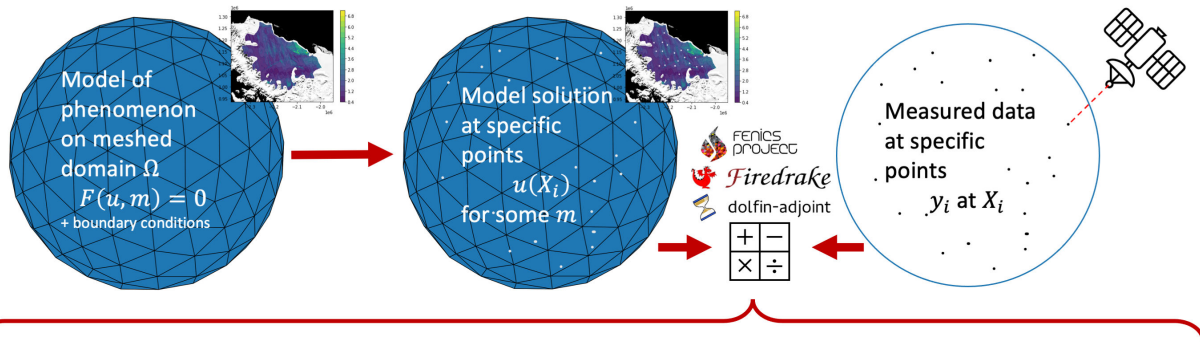
Gradient Based Optimisation Needs

$$\frac{\partial J}{\partial u} \frac{du}{dm}$$

Find via solution to the adjoint  
 problem using dolfin-adjoint

# Example Use Case

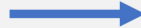
## Point Data Assimilation: A PDE Constrained Optimisation Problem



Minimization Problem:

$$\min_{u, m} J[u, m] \quad \text{where } J = \underbrace{\|u(X_i) - y_i\|^2}_{\text{data fit}} + \underbrace{c\|m\|^2}_{\text{regularization}}$$

**Need to express this in UFL and annotate it with dolfin-adjoint/pyadjoint**



Gradient Based Optimisation Needs

$$\frac{\partial J}{\partial u} \frac{du}{dm}$$

Find via solution to the adjoint problem using  dolfin-adjoint

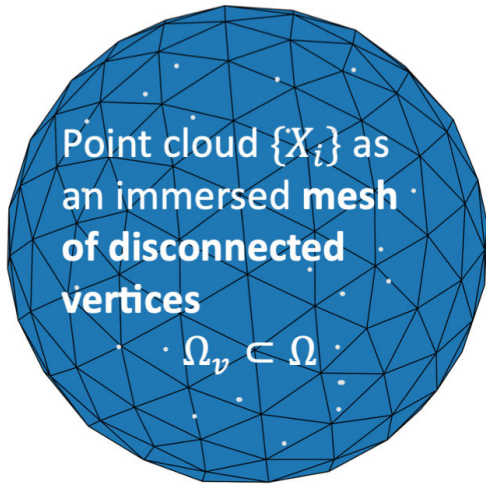
# Solution Part 1: A point cloud is a mesh $\Omega_v$

## UFL

- Vertex Cells
- Topological dimension = 0
- Geometric dimension =  $\dim(X_i)$  (point cloud coordinate dimension)

## Firedrake

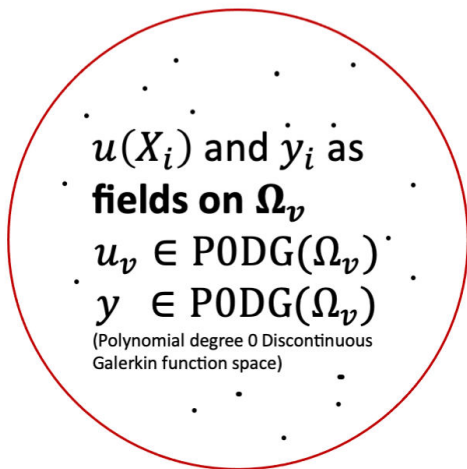
- Vertices at point cloud coordinates  $\{X_i\}$
- Immersed (for now)
  - makes implementation simpler
  - care about the point data with respect to field on “parent” mesh  $\Omega$  (e.g. a PDE solution)



## Solution Part 2: Point data are functions in a PODG function space on $\Omega_v$

- Scalar, vector or tensor valued
- Each function in this function space represents a **complete set of point data**
  - Vertex coordinates  $\{X_i\}$  are fixed\*
  - Can **declare UFL arguments** to **solve for values** at points  $\{X_i\}$
- Changes: UFL ✓ FIAT ✓ Firedrake ✓

\*unless we represent moving points – something for the future!



# How do we use this?

## Interpolate into the point data space

Point evaluation is now  
interpolation into  $W = \text{P0DG}(\Omega_v)$   
 $u_v = I(u, W)$



Rewrite  $J = \|u(X_i) - y_i\|^2 + c\|m\|^2$

$$J = \int_{\Omega_v} (I(u, W) - y)^2 dx + c\|m\|^2$$
$$\frac{\partial J}{\partial u} = \int_{\Omega_v} (I(u, W) - y) \underbrace{\frac{\partial I(u, W)}{\partial u}}_{\text{newly added to}} dx$$

Newly added to

 *Firedrake* and  dolfin-adjoint

```
# UFL+Firedrake Pseudocode
# blue is proposed new UFL

# u is a solution from a function space on parent_mesh
# m is source term from some function space

vm = VertexOnlyMesh(parent_mesh, point_cloud_coords)
W = FunctionSpace(vm, "DG", 0)

# interpolate
v = TestFunction(W).dual()
u_v = interp(u, v) # or u_v = v(u)

... # create y in W from observation data

# Functional for minimisation
J = assemble( (u_v - y)**2 * dx + c*inner(m, m) * dx )

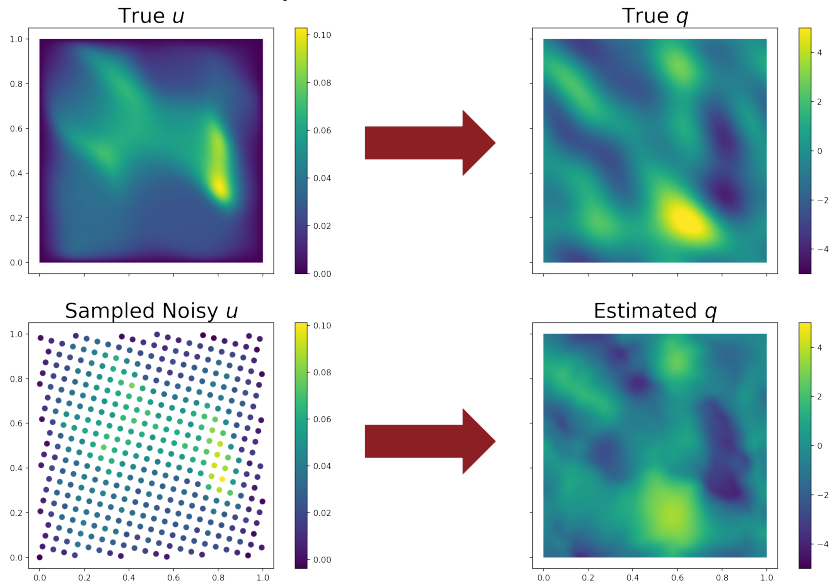
# Reduce using control from parameter space
m_hat = firedrake_adjoint.Control(m)
J_hat = firedrake_adjoint.ReducedFunctional(J, m_hat)

# Find optimal control
m_min = firedrake_adjoint.minimize(J_hat, method='Newton-CG')
```



# It really works!

Estimating Log-Conductivity  $q$   
where  $k = k_0 e^q$  and  $-\nabla \cdot k \nabla u = f$  for known  $f$



# Future: UFL for Automated Diagnostics

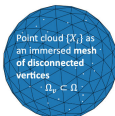
- Aim to develop UFL into a Domain Specific Language (DSL) for specifying model diagnostics
  - A user will specify, as some high level integration (e.g. over points or planes), the diagnostic then code will be generated to calculate it
  - Example: Ocean current from multiple climate models e.g. Atlantic Meridional Overturning Circulation in Medhaug and Furevik 2011 [4]
- Form compiler can then generate code for calculating the diagnostic
- **Scalable** and able to run on **big datasets** (e.g. climate) on the **HPCs closest to the data**.

<sup>1</sup> Medhaug, I., & Furevik, T. (2011). North Atlantic 20th century multidecadal variability in coupled climate models: Sea surface temperature and ocean overturning circulation. *Ocean Science*, 7(3), 389-404.

# Conclusion

- Point data are everywhere and can be treated more rigorously
- We can represent point data as a function in a PODG function space on a point cloud mesh
- Point evaluations are interpolations into this function space
- `interp` proposed as new UFL operator for interpolation
- Can use point data in PDE constrained optimisation problems by annotating interpolation with `pyadjoint/dolfin-adjoint`
- First step towards turning UFL into a DSL for automated diagnostics

Get in touch! [reuben.nixon-hill10@imperial.ac.uk](mailto:reuben.nixon-hill10@imperial.ac.uk)



# Possible work-around existing limitations

Minimization Problem:

$$\min_{u,m} J[u, m] \quad \text{where } J = \|u(X_i) - y_i\|^2 + c\|m\|^2$$

subject to

$$F(u, m) = 0$$

+ boundary conditions

- Interpolate the point data into some function space on my mesh then calculate

$$J = \|u - y_{approx}\|^2 + c'\|m\|^2$$

- Have to make difficult-to-test decisions about appropriate interpolation hyperparameters to get  $y_{approx}$ ,
- Particularly a problem if the point data is sparse

## Solution Part 2: Point data are functions in a PODG function space on $\Omega_v$

Some Properties:

- Integrating sums values at points

$$\int_{\Omega_v} u_v dx = \sum_i u_v(X_i)$$

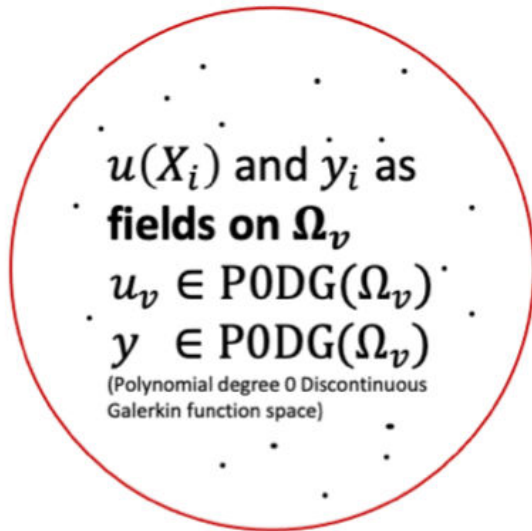
- $L^2$  inner product equivalent to  $l_2$  inner product of each component (if vector or tensor valued)

$$\langle u_v, y \rangle_{L^2} = \int_{\Omega_v} u_v y dx = \dots = \sum_i u_v(X_i) y(X_i) = \langle u_v, y \rangle_{l_2}$$

- Mass matrix is identity matrix
- Non-differentiable
- Reisz map is  $L^2$  inner product

$$\langle u_v, \cdot \rangle_{L^2} = u_v'(\cdot)$$

- Nodal interpolation is  $L^2$  Galerkin projection



## Why is this helpful in this example?

- No extra approximations necessary
- Can **be rigorous about the statistical interpretation** of data assimilation results via misfit functional (check if errors are normally distributed for example) and directly investigate different regularisations.
- **Feed back from modeller to experimenter**: can quickly model say, the impact of more measurements vs better SNR with simulated data.
- Dan and I working on a paper right now to make this point!

# Future: UFL for Automated Diagnostics

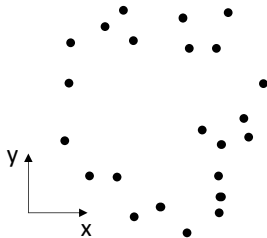
- Possible approach:
  1. Read-in gridded field data as equivalent finite element field
  2. Create mesh to represent region to integrate:
    - Points - “Vertex Only Mesh”
    - lines “mesh of disconnected lines”
    - Planes “mesh of disconnected planes” etc.
  3. Calculate the diagnostic by interpolating onto the region and performing desired integration
    - Point evaluations
    - Fluxes etc.
- **Note:** Requires UFL to be extended to include interpolation operations in the language e.g. via the `interp` form.



# Next Steps

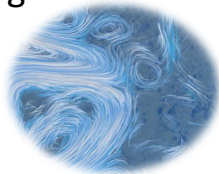
## Point Data

- Demonstration on real models showing advantages
- Moving points (optional)



## Automated Diagnostics

- Higher dimension disconnected mesh abstractions (lines, planes and polyhedra) for interpolation onto
- Define the interpolation operations e.g. via supermeshing
- Improve dataset parsing tools
- Integration with existing tool-chains e.g. Pangeo





# dolfiny: Convenience wrappers for DOLFINx

Andreas Zilian, University of Luxembourg, Luxembourg

Michal Habera, University of Luxembourg, Luxembourg

23 March 2021

With the increased flexibility of DOLFINx and its reduction to core functionality, the responsibility for even some basic components of computational analysis is shifted to the user.

This presentation provides an overview of the open-source package dolfiny, which provides end-user API interfaces to mesh/meshtags generation and processing, expression list handling, function interpolation and projection as well as the restriction of function spaces to parts of the computational domain. This functionality is consistently considered in interfaces to PETSc/SNES as nonlinear solver and SLEPc as eigensolver backend, both allowing the operation on block and nested operators. In addition, the package provides a convenient approach to incorporate time integration into the UFL formulation of the problem, which is exemplified for the generalised alpha method.

The capability of dolfiny is demonstrated in a number of examples, ranging between finite strain structural analysis, plasticity and fluid-structure interaction.

---

You can cite this talk as:

Andreas Zilian and Michal Habera. “dolfiny: Convenience wrappers for DOLFINx”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 183–192.  
DOI: 10.6084/m9.figshare.14495262.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/zilian.html>.

# do1f1ny: Convenience wrappers for DOLFINx

Andreas Zilian, Michal Habera

Department of Engineering | University of Luxembourg

23 March 2021 | FEniCS'21 conference



# Contents

Why dolfiny?

Mesh and MeshTags

Restrictions

Interface to SNES (with restrictions)

Interpolation

Time-dependent forms

# Motivation | why `dolfiny`?

## DOLFINx advantages

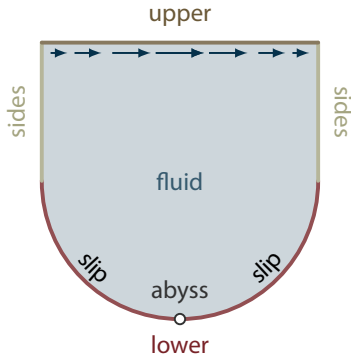
- ▶ access to low level interfaces, light core
- ▶ more flexible, more explicit
- ▶ demands bottleneck-awareness

## DOLFINx shortcomings (today)

- ▶ user code often quite verbose
- ▶ increased complexity to end-user
- ▶ consistent approach to extensions?

`dolfiny`: Python, collection of wrappers, extensions + new functionality

# Mesh and MeshTags | challenge: problem setup



- Gmsh to DOLFINx
- merge named non-overlapping MeshTags
- sub-classed XDMFFile

```
import mesh_cavity_gmshapi as mg
```

```
# Create the geometry/mesh of the cavity using Gmsh Python API  
gmsh, tdim = mg.mesh_cavity_gmshapi() # see demos in dolfiny repo
```

```
import dolfiny.mesh
```

```
# Get mesh and meshtags  
mesh, mts = dolfiny.mesh.gmsh_to_dolfin(gmsh, tdim, prune_z=True)
```

```
# Get merged MeshTags for each codimension  
subdomains, subdomains_keys = dolfiny.mesh.merge_meshtags(mts, tdim - 0)  
interfaces, interfaces_keys = dolfiny.mesh.merge_meshtags(mts, tdim - 1)  
markpoints, markpoints_keys = dolfiny.mesh.merge_meshtags(mts, tdim - 2)
```

```
# Define shorthands for labelled tags  
fluid = subdomains_keys["fluid"]  
upper = interfaces_keys["upper"]  
lower = interfaces_keys["lower"]  
sides = interfaces_keys["sides"]  
abyss = markpoints_keys["abyss"]
```

```
import dolfiny.io
```

```
# Create output XDMF file  
ofile = dolfiny.io.XDMFFile(comm, f"{name}.xdmf", "w")
```

```
# Write mesh, meshtags  
ofile.write_mesh_meshtags(mesh, mts)
```

# Restriction | challenge: weak interface constraints

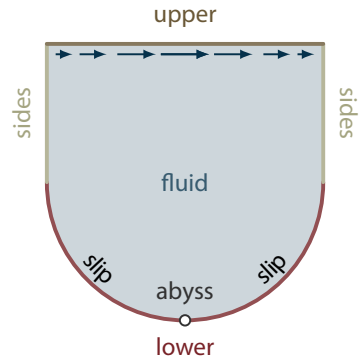
```
import dolfinx, ufl

# Function spaces, functions and test functions
V, P, L = # ... e.g. dolfinx.VectorFunctionSpace(mesh, ("CG", 2))
v, p, λ = # ... e.g. dolfinx.Function(V, name="v")
δv, δp, δλ = # ... e.g. ufl.TestFunction(V)

# Integration measures
dx = ufl.Measure("dx", domain=mesh, subdomain_data=subdomains)
ds = ufl.Measure("ds", domain=mesh, subdomain_data=interfaces)

# Non-Newtonian fluid, with  $\mu = \mu(D)$ 
D = ufl.sym(ufl.grad(v))
T = 2 * mu(D) * D - p * ufl.Identity(2)

# Weak form (as one-form), with  $n = ufl.FacetNormal(mesh)$ 
f = ufl.inner(ufl.grad(δv), T) * dx + δp * ufl.div(v) * dx \
    + ufl.dot(δv, n) * λ * ds(lower) + δλ * ufl.dot(v, n) * ds(lower)
```



```
import dolfiny.function

# Locate dofs: restriction
rdofsV = dolfiny.mesh.locate_dofs_topological(V, subdomains, fluid)
rdofsV = dolfiny.function.unroll_dofs(rdofsV, V.dofmap.bs)
rdofsP = dolfiny.mesh.locate_dofs_topological(P, subdomains, fluid)
rdofsL = dolfiny.mesh.locate_dofs_topological(L, interfaces, lower)

import dolfiny.restriction

# Set up restriction
r_fspaces, r_dofs = [V, P, L], [rdofsV, rdofsP, rdofsL]
restriction = dolfiny.restriction.Restriction(r_fspaces, r_dofs)
```

- restrict full function space to subset of dofs
- discrete/algebraic approach

# SNES interface | challenge: nonlinear (restricted) problem

- ▶ SNESBlockProblem interfaces to PETSc SNES
- ▶ custom block-wise convergence monitors
- ▶ allows MatNest and AIJ/BAIJ as matrix storage layout
- ▶ supports restrictions

```
# Define state as (ordered) list of functions
m, δm = [v, p, λ], [δv, δp, δλ]

# Overall form (as list of forms)
F = dolfiny.function.extract_blocks(f, δm)

import dolfiny.snesblockproblem

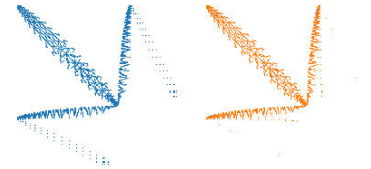
# Create nonlinear problem: SNES
problem = dolfiny.snesblockproblem.SNESBlockProblem(F, m, restriction)

# Set/update boundary conditions
problem.bcs = # ...

# Solve nonlinear problem
problem.solve()
```

```
### SNES iteration 4
# sub 0 |x|=1.052e+01 |dx|=2.515e-02 |r|=1.903e-03 (v)
# sub 1 |x|=1.359e+02 |dx|=7.764e-01 |r|=2.139e-17 (p)
# sub 2 |x|=4.723e+02 |dx|=2.210e+00 |r|=2.895e-18 (λ)
# all |x|=4.913e+02 |dx|=2.342e+00 |r|=1.993e-03

### SNES iteration 5
# sub 0 |x|=1.051e+01 |dx|=9.610e-04 |r|=3.658e-06 (v)
# sub 1 |x|=1.359e+02 |dx|=1.336e-02 |r|=1.890e-17 (p)
# sub 2 |x|=4.723e+02 |dx|=6.682e-03 |r|=1.713e-18 (λ)
# all |x|=4.913e+02 |dx|=1.497e-02 |r|=3.658e-06
...
```



- ▶ Restriction uses PETSc.Mat.createSubMatrix()

# Interpolation | challenge: expression evaluation

- ▶ interpolate UFL expressions into functions
- ▶ supports arbitrary cell-wise UFL expressions: into CG/DG (FFCx/numba)
- ▶ supports linear combination of functions for arbitrary function spaces

## Cell-wise

```
import dolfiny.interpolation

# Function space for interpolated stress tensor
S = dolfinx.TensorFunctionSpace(mesh, ("DG", 1), \
                                   symmetry=True)

# Function
s = dolfinx.Function(S)

# Interpolate UFL expression T
dolfiny.interpolation.interpolate(T, s)
```

## Facet-wise (soon)

```
import dolfiny.interpolation

# Function space for interpolated stress vector
S = dolfinx.VectorFunctionSpace(mesh, ("DG", 1))

# Function
s = dolfinx.Function(S)

# Interpolate UFL expression T * n
dolfiny.interpolation.interpolate(T * n, s)
```



# Time-dependent forms | challenge: ease-of-use

```
# Global time, Time step size
time, dt = dolfinx.Constant(mesh, 0.0), dolfinx.Constant(mesh, 0.1)

# Define functions representing 1st time derivative
vt, pt,  $\lambda$ t = # ... e.g. dolfinx.Function(V, name="vt"), ...

# Define state as (ordered) list of functions
m, mt,  $\delta$ m = [v, p,  $\lambda$ ], [vt, pt,  $\lambda$ t], [ $\delta$ v,  $\delta$ p,  $\delta$  $\lambda$ ]

import dolfiny.odeint

# Time integrator
odeint = dolfiny.odeint.ODEInt(t=time, dt=dt, x=m, xt=mt)

# Weak form (as one-form), with time-dependent terms
f = # ... \
    + ufl.inner( $\delta$ v, rho * vt + rho * ufl.grad(v) * v) * dx

# Overall form (as one-form)
f = odeint.discretise_in_time(f)

# Overall form (as list of forms)
F = dolfiny.function.extract_blocks(f,  $\delta$ m)

# Create nonlinear problem: SNES
problem = dolfiny.snesblockproblem.SNESBlockProblem(F, m, restriction)

# Time steps
for step in timesteps:
    odeint.stage()
    problem.solve()
    odeint.update()
```

- ▶ readability of forms
- ▶ single step methods
- ▶ stencil as expression
- ▶ expression-based state updates (interpolation)
- ▶ ODEInt, ODEInt2 for 1st/2nd order in time ODEs
- ▶ based on modified generalised- $\alpha$  method

# Summary and outlook

<https://github.com/michalhabera/dolfiny>



## Now


- ▶ transfer Gmsh/DOLFINx
- ▶ Restriction
- ▶ interpolation, projection
- ▶ SNESBlockProblem, SLEPcBlockProblem
- ▶ ODEInt, ODEInt2
- ▶ supports MPI parallelism
- ▶ various demos


## Future

- ▶ interpolation on facets
- ▶ flexible API for static condensation
- ▶ ...

# Back to Basix: Construction of arbitrary order finite element DOF maps on polygonal and polyhedral cell meshes

**Matthew Scroggs** (<https://www.mscroggs.co.uk>,  mscroggs,  @mscroggs), University of Cambridge, United Kingdom

Jørgen S. Dokken (<https://jsdokken.com/>,  jorgensd), University of Cambridge, United Kingdom

Chris Richardson ( chrisrichardson), BP Institute, United Kingdom

Garth N. Wells ( garth-wells), University of Cambridge, United Kingdom

23 March 2021

We develop an approach to generating degree-of-freedom (DOF) maps for arbitrary order finite element spaces for any cell shape. The approach is based on the composition of permutations and transformations by cell sub-entity. Current approaches to generating degree-of-freedom maps for arbitrary order problems typically rely on a consistent orientation of cell entities that permits the definition of a common local coordinate system on shared edges and faces. However, while orientation of a mesh is straightforward for simplex cells and is local operation, it is not a strictly local operation for quadrilateral cells and in the case of hexahedral cells not all meshes are orientable. The permutation and transformation approach is developed for a range of element types, including Lagrange, divergence-conforming Raviart–Thomas and curl-conforming Nédélec elements, and for a range of cell shapes. The approach is local and can be applied to cells of any shape, including general polytopes and meshes with mixed cell types. This method has been implemented in Basix, FEniCSx’s element tabulator.

## References

- [1] Matthew W Scroggs, Jørgen S Dokken, Chris N Richardson, and Garth N Wells. “Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes”. <https://arxiv.org/abs/2102.11901>. 2021.

---

You can cite this talk as:

Matthew Scroggs, Jørgen S. Dokken, Chris Richardson, and Garth N. Wells. “Back to Basix: Construction of arbitrary order finite element DOF maps on polygonal and polyhedral cell meshes”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 193–220. DOI: 10.6084/m9.figshare.14495268.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/scroggs.html>.

# Back to Basix

## Construction of arbitrary order elements on polygons and polyhedrons in FEniCSx


**Matthew Scroggs**

(University of Cambridge)

 [mascroggs.co.uk](https://mascroggs.co.uk)

 [mws48@cam.ac.uk](mailto:mws48@cam.ac.uk)

 [mascroggs](https://github.com/mascroggs)

 [@mascroggs](https://twitter.com/mascroggs)

**Jørgen Dokken**

(University of Cambridge)

**Chris Richardson**

(University of Cambridge)

**Garth Wells**

(University of Cambridge)

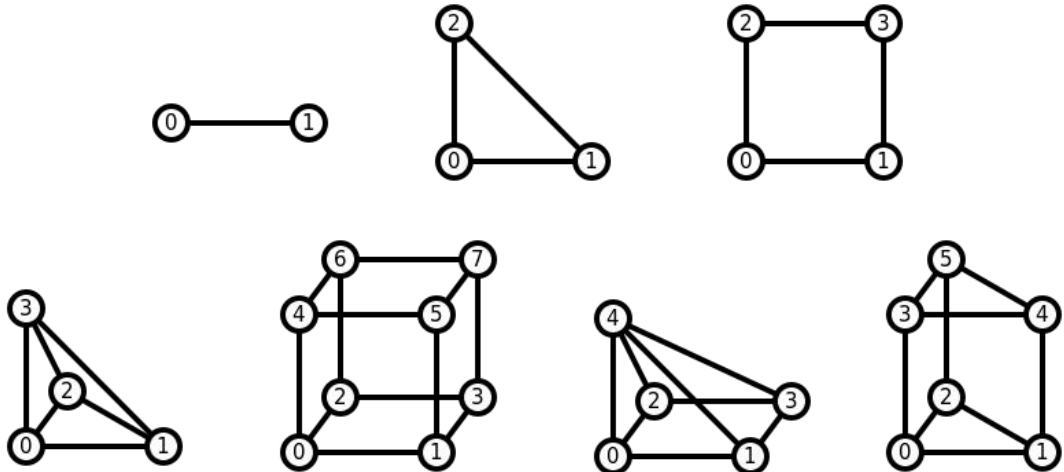
# Tabulation in FEniCSx

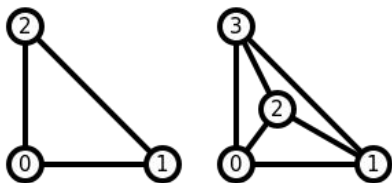
- FEniCS uses FIAT.
- We want:
  - Tabulation at runtime.
  - C++.

# Basix

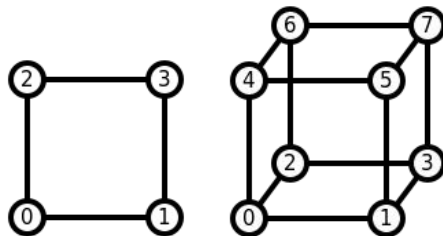
[github.com/FEniCS/basix](https://github.com/FEniCS/basix)

- C++ with Python interface



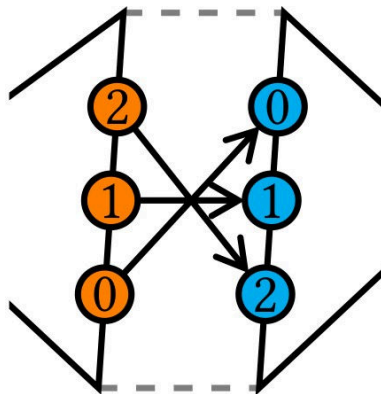
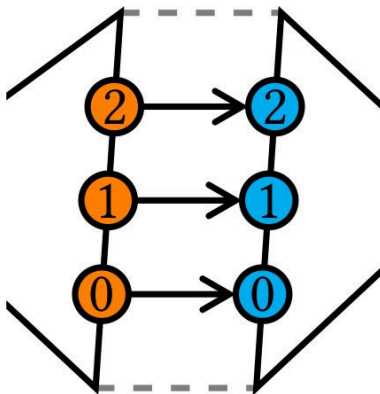


Lagrange  
 Nédélec (first kind)  
 Nédélec (second kind)  
 Raviart–Thomas  
 Brezzi–Douglas–Marini  
 Bubble  
 Crouzeix–Raviart  
 Regge



Lagrange (Q)  
 Nédélec  
 Raviart–Thomas  
 Bubble  
 DPC  
 Serendipity

# Higher order spaces





# DOF transformations

- Solution?: order cells in the mesh

Input cells

[0, 5, 1]

[1, 2, 3]

[5, 3, 0]

[5, 1, 2]



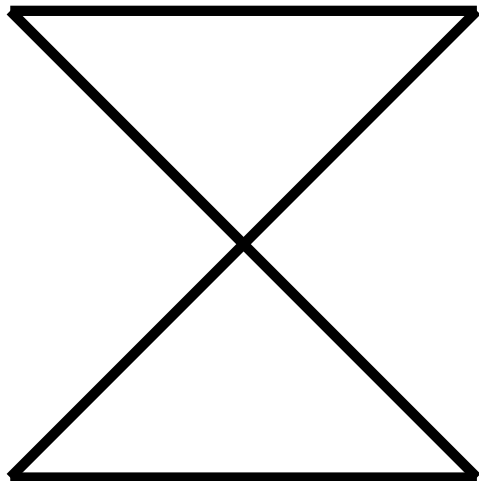
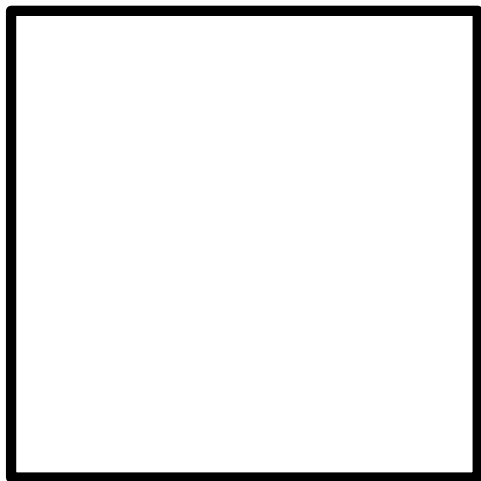
Cells

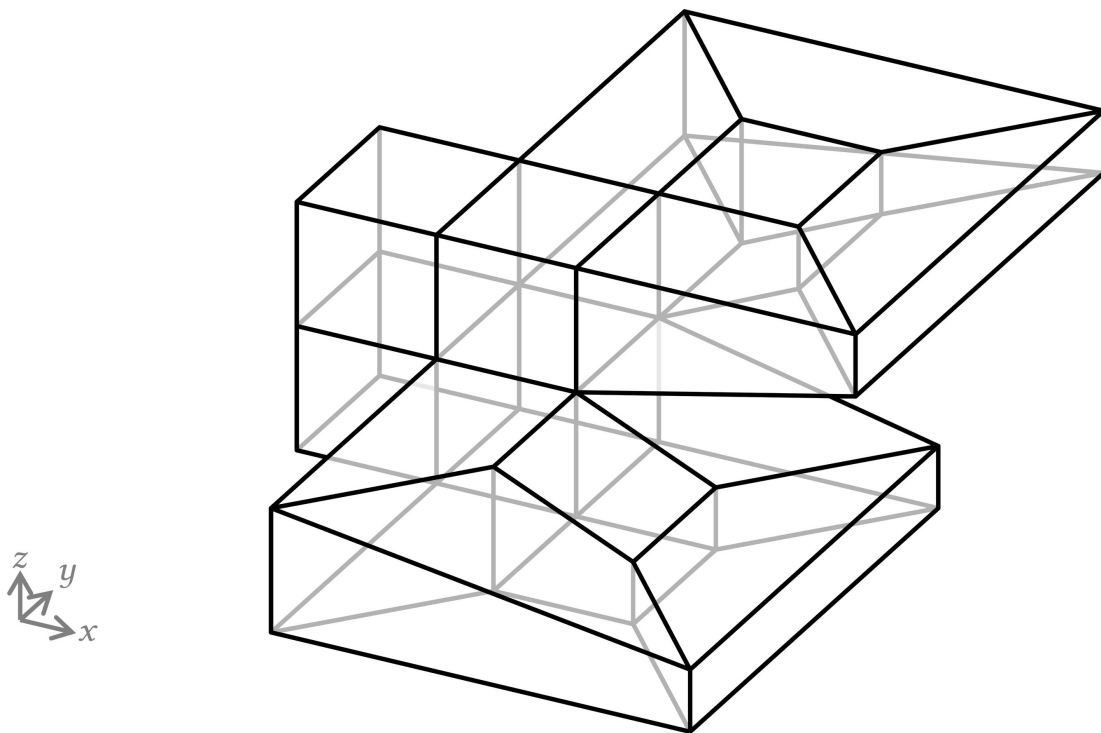
[0, 1, 5]

[1, 2, 3]

[0, 3, 5]

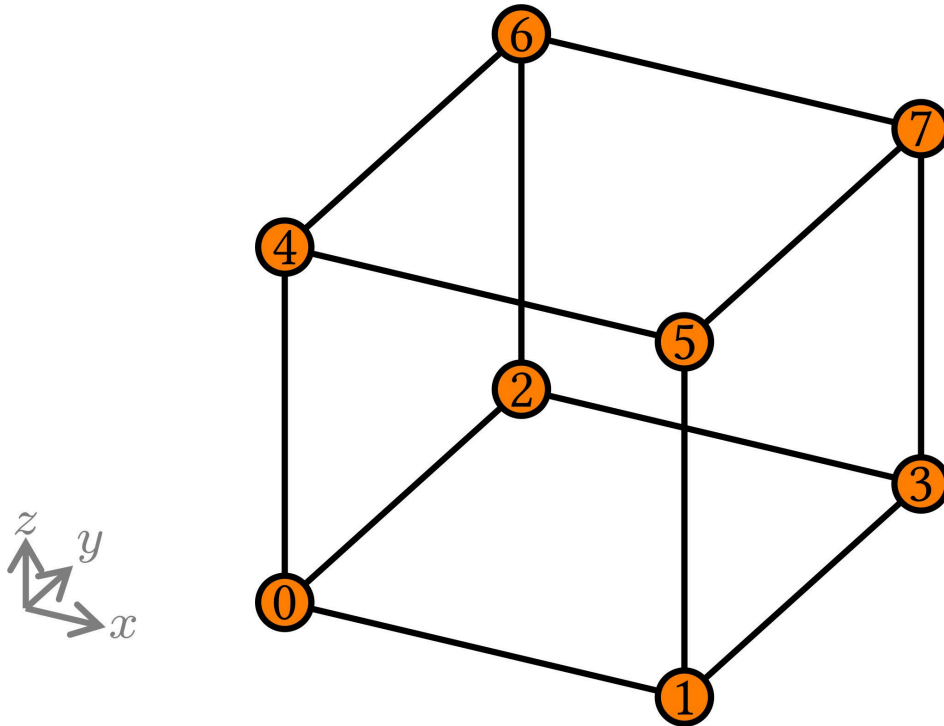
[1, 2, 5]



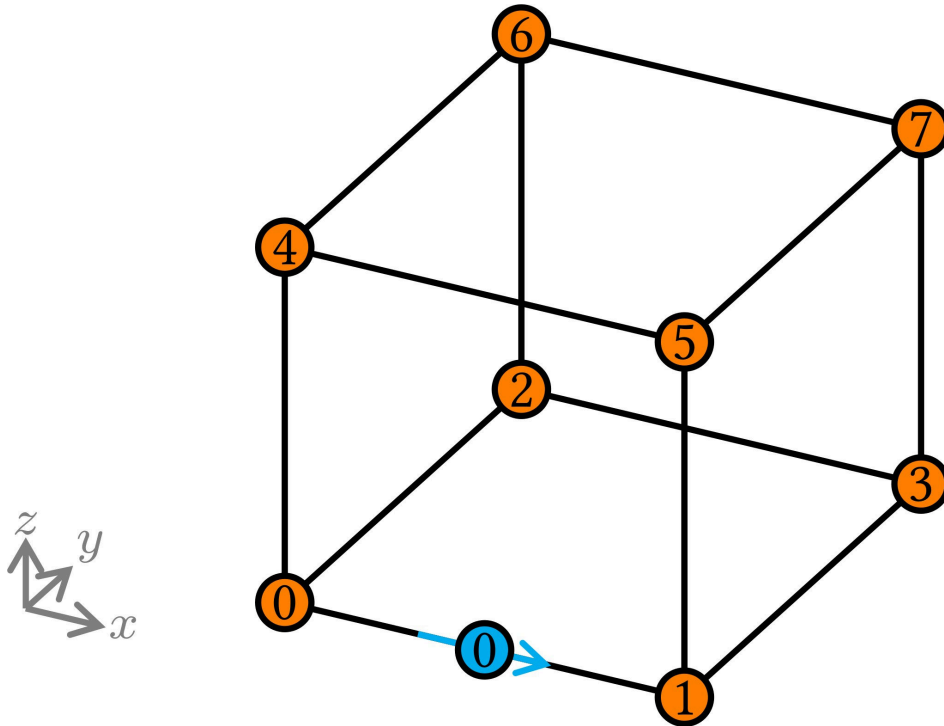


Rainer Agelek, Michael Anderson, Wolfgang Bangerth, and William L. Barth.  
On orienting edges of unstructured two- and three-dimensional meshes. (2017)  
ACM Transactions on Mathematical Software 44, 1, 5:1–5:22. DOI: [10.1145/3061708](https://doi.org/10.1145/3061708)

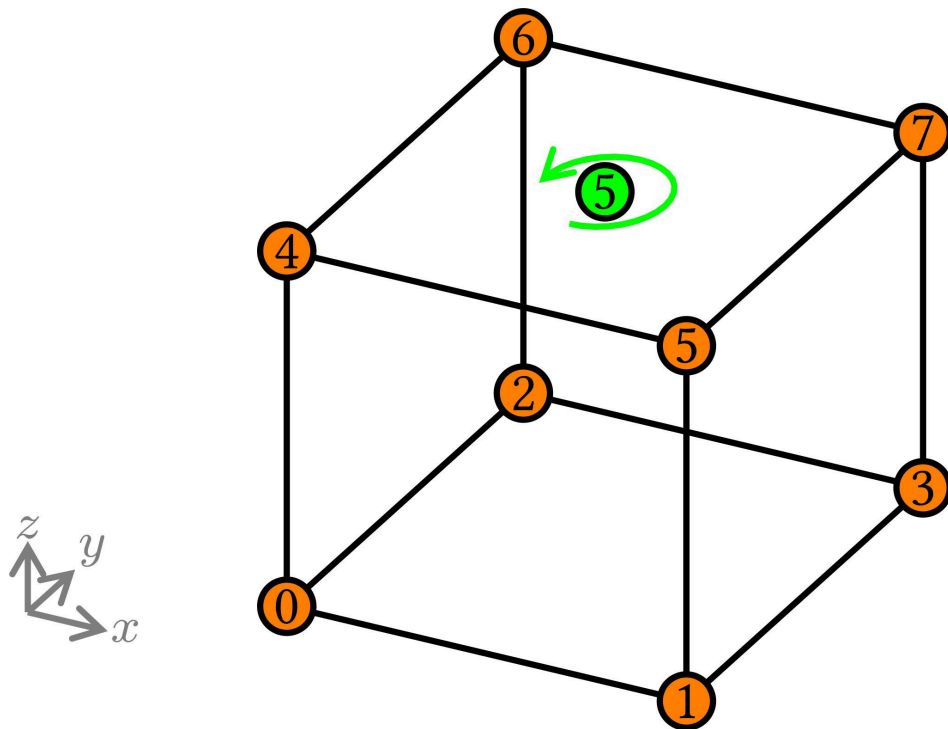
# Example: Hexahedron



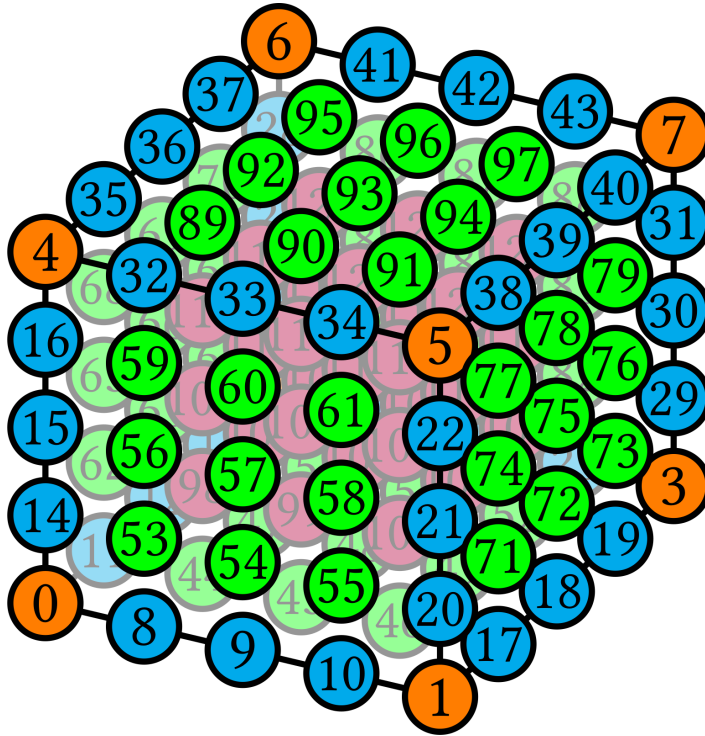
# Example: Hexahedron



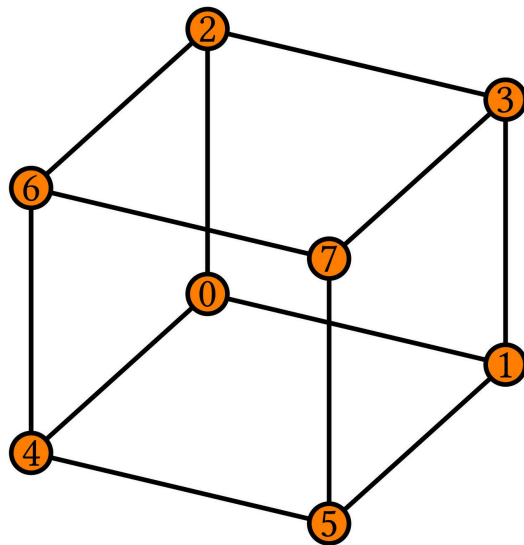
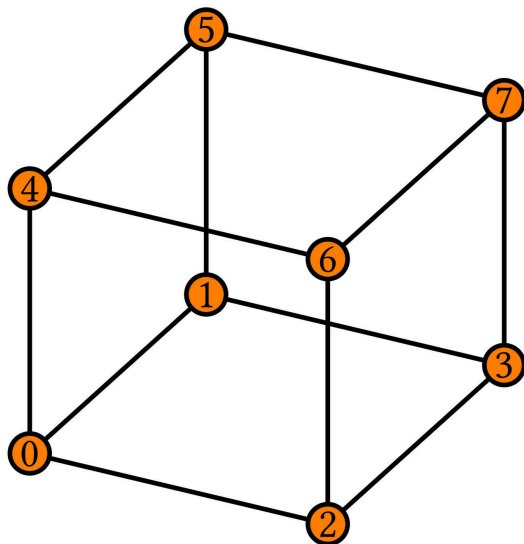
# Example: Hexahedron



# Example: Order 4 Lagrange

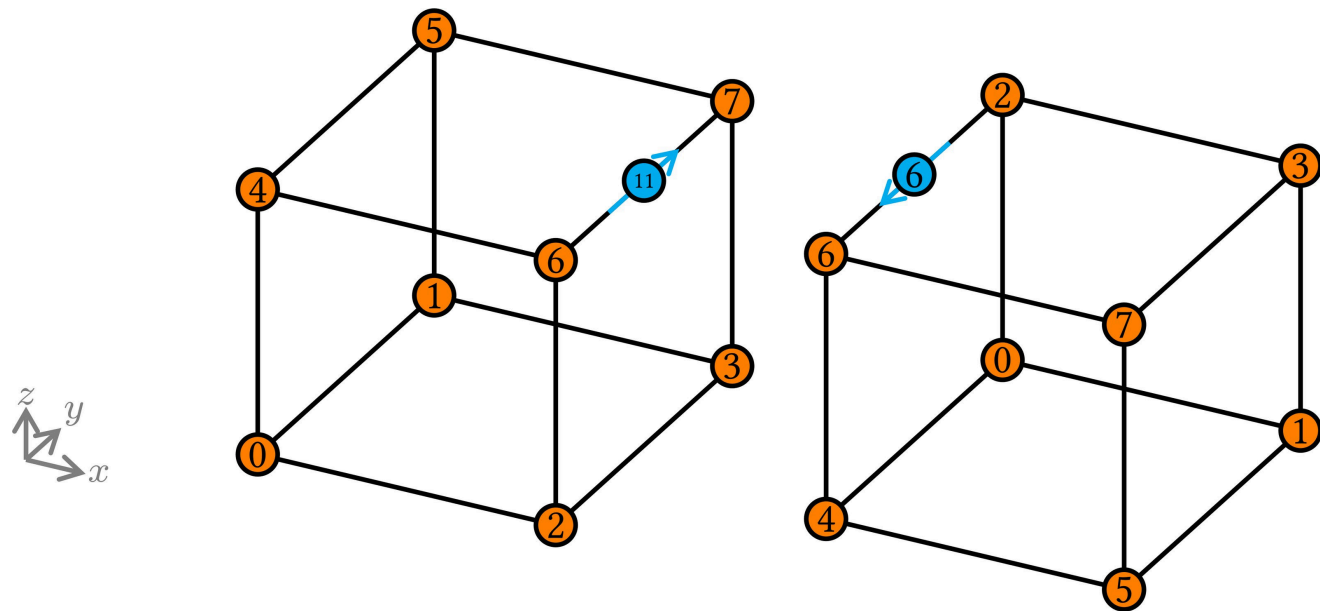


# Example: Order 4 Lagrange

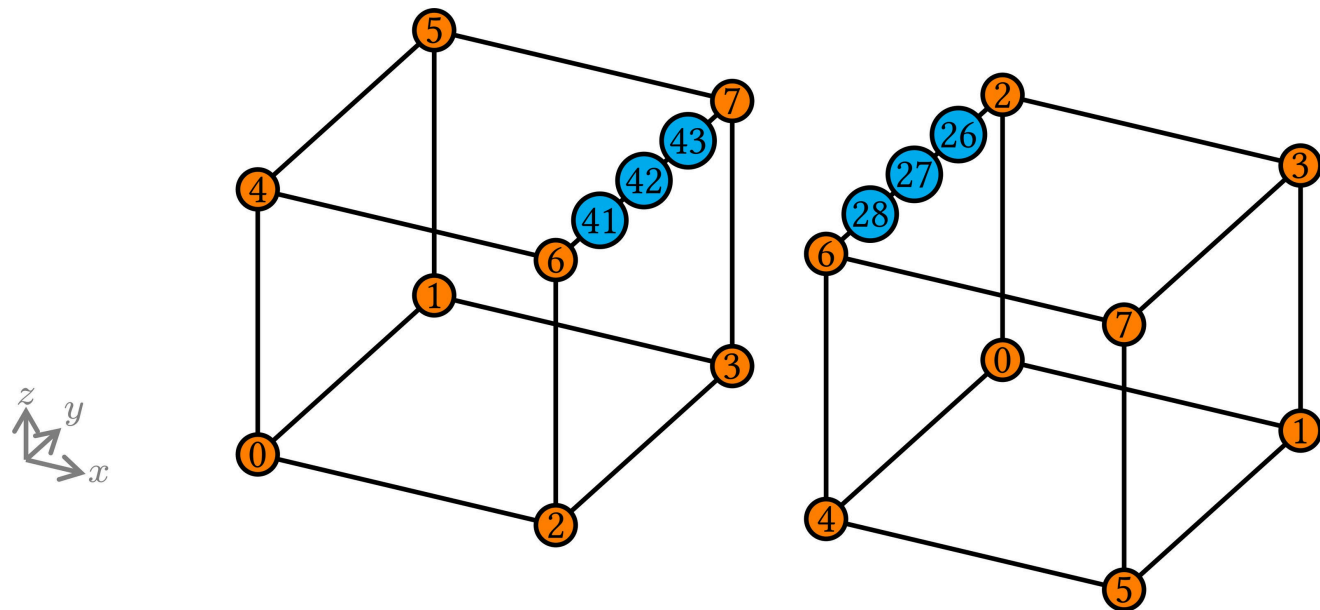




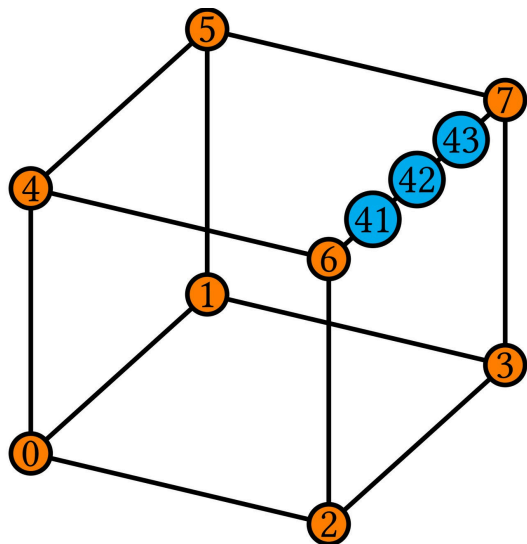
# Example: Order 4 Lagrange



# Example: Order 4 Lagrange

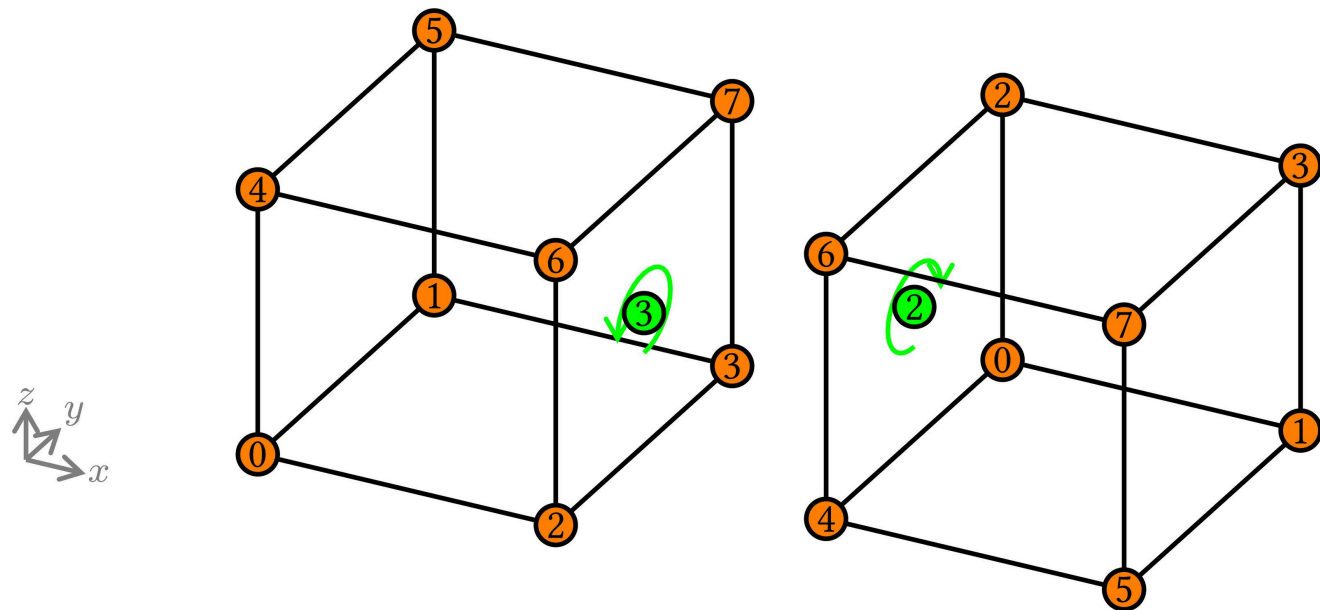


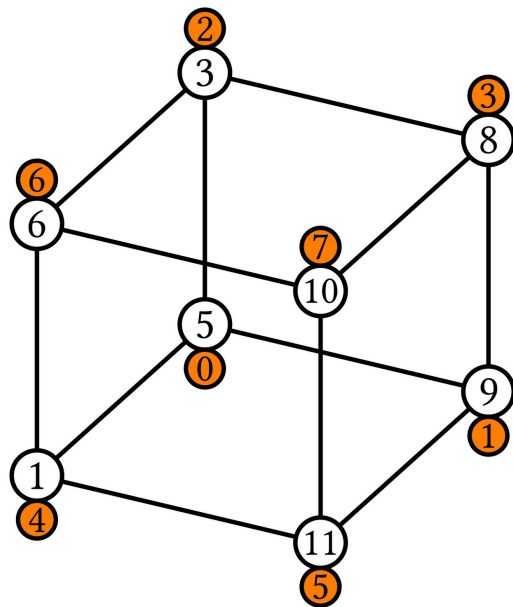
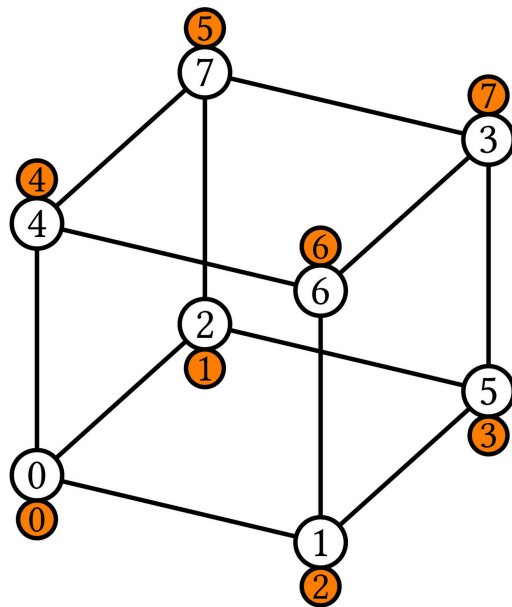
# Example: Order 4 Lagrange

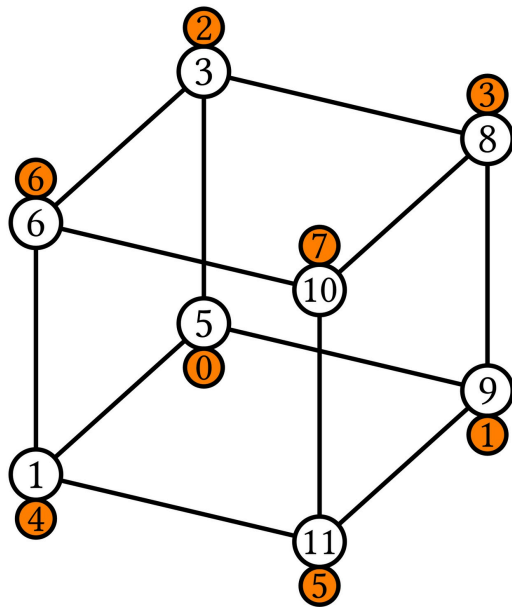
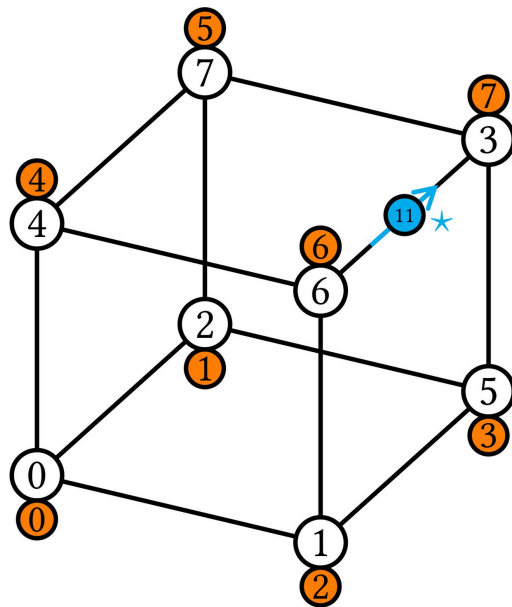


$$\begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & 0 & 0 & 1 & & \\ & & & 0 & 1 & 0 & & \\ & & & 1 & 0 & 0 & & \\ & & & & & & 1 & \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix}$$

# Example: Order 4 Lagrange





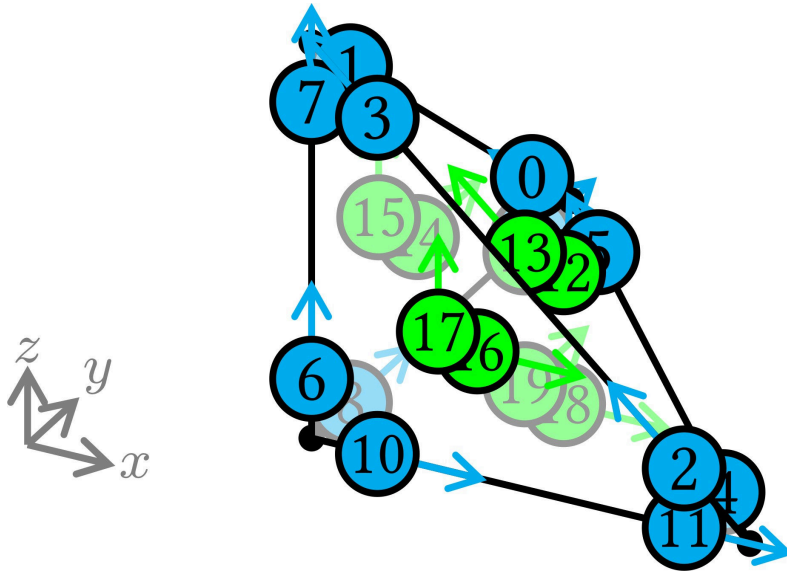


# In general

- Precompute:
  - 1 matrix for each edge
  - 2 matrices for each face
- Compare local and global numbers to decide which matrices to apply

# Example: Order 2 Nédélec

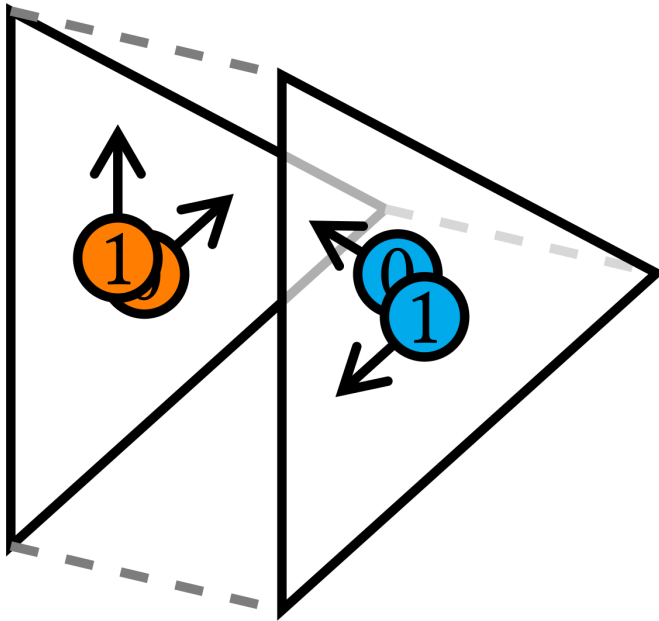
(first kind)





# Example: Order 2 Nédélec

(first kind)

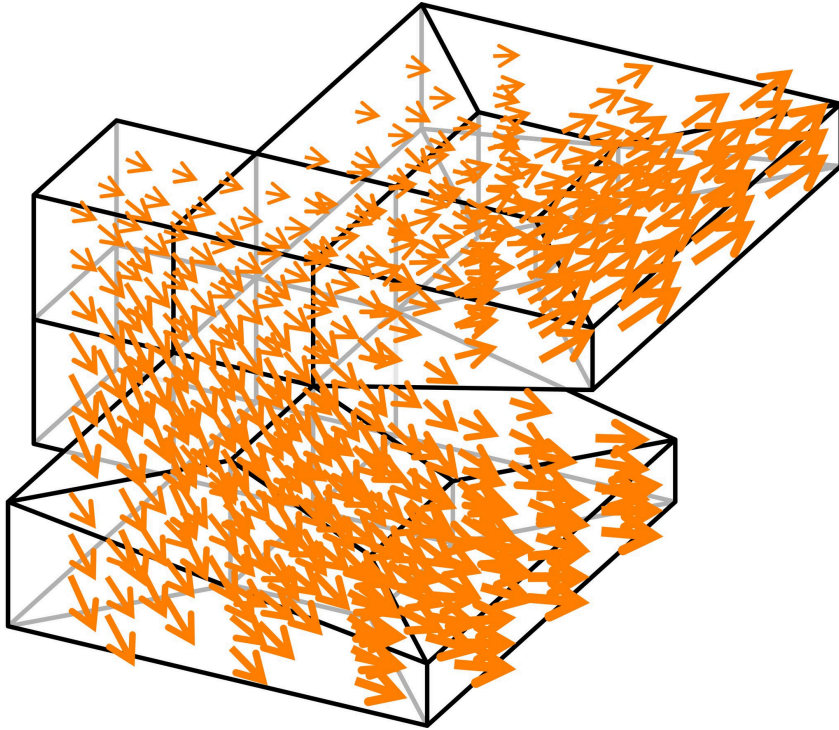


$$\begin{pmatrix} \ddots & & & \\ & -1 & -1 & \\ & 1 & 0 & \\ & & & \ddots \end{pmatrix}$$

# Basix

```
import basix
element = basix.create_element(
    "Nedelec 1st kind H(curl)", "tetrahedron", 2)
print(element.base_transformations)
```

# Example: Order 3 Nédélec



# arxiv.org/abs/2102.11901

## Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes

MATTHEW W. SCROGGS, Department of Engineering, University of Cambridge, United Kingdom

JØRGEN S. DOKKEN, Department of Engineering, University of Cambridge, United Kingdom

CHRIS N. RICHARDSON, BP Institute, University of Cambridge, United Kingdom

GARTH N. WELLS, Department of Engineering, University of Cambridge, United Kingdom

We develop an approach to generating degree-of-freedom maps for arbitrary order Ciarlet-type finite element spaces for any cell shape. The approach is based on the composition of permutations and transformations by cell sub-entity. Current approaches to generating degree-of-freedom maps for arbitrary order problems typically rely on a consistent orientation of cell entities that permits the definition of a common local coordinate system on shared edges and faces. However, while orientation of a mesh is straightforward for simplex cells and is a local operation, it is not a strictly local operation for quadrilateral cells and in the case of hexahedral cells not all meshes are orientable. The permutation and transformation approach is developed for a range of element types, including arbitrary degree Lagrange, serendipity, and divergence- and curl-conforming elements, and for a range of cell shapes. The approach is local and can be applied to cells of any shape, including general polytopes and meshes with mixed cell types. A number of examples are presented and the developed approach has been implemented in open-source libraries.

CCS Concepts: • **Mathematics of computing** → **Discretization**; *Partial differential equations*.

Additional Key Words and Phrases: finite element methods, degrees-of-freedom, polyhedral cells

# defelement.com

## DefElement an encyclopedia of finite element definitions

Welcome to DefElement: an encyclopedia of finite element definitions.

This website contains a collection of definitions of finite elements, including commonly used elements such as [Lagrange](#), [Raviart-Thomas](#), [Nédélec \(first kind\)](#) and [Nédélec \(second kind\)](#) elements, and more exotic elements such as [serendipity H\(div\)](#), [serendipity H\(curl\)](#) and [Regge](#) elements.

You can:

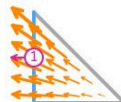
- [view the full alphabetical list of elements](#)
- [view the elements by category](#)
- [view the elements by reference element](#)
- [view the elements by family](#)

### The finite element method

The finite element method is a numerical method that involves discretising a problem using a finite dimensional function space. These function spaces are commonly defined using a finite element on a reference element to derive basis functions for the space. This website contains a collection of finite elements, and examples of the basis functions they define.

Following the [Ciarlet definition](#) of a finite element, the elements on this website are defined using a reference element, a polynomial space, and a set of functionals. Each element's page describes how these are defined for that element, and gives examples of these and the basis functions they lead to for a selection of low-order spaces.

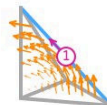
You can read a detailed description of how the finite element definitions can be understood [here](#).



A basis function of an order 1 [Raviart-Thomas space](#) on a triangle



A basis function of an order 2 [space](#) on a quadrilateral



A basis function of an order 1 [Nédélec \(first kind\) space](#) on a tetrahedron

# Thanks for listening!

[arxiv.org/abs/2102.11901](https://arxiv.org/abs/2102.11901)

[defelement.com](https://defelement.com)

**Matthew Scroggs**

(University of Cambridge)

 [mascroggs.co.uk](https://mascroggs.co.uk)

 [mws48@cam.ac.uk](mailto:mws48@cam.ac.uk)

 [mascroggs](https://github.com/mascroggs)

 [@mascroggs](https://twitter.com/mascroggs)

**Jørgen Dokken**

(University of Cambridge)

**Chris Richardson**

(University of Cambridge)

**Garth Wells**

(University of Cambridge)

# hIPPYlib-MUQ: Scalable Markov chain Monte Carlo sampling methods for large-scale Bayesian inverse problems governed by PDEs

Ki-Tae Kim, University of California, Merced, United States

Umberto Villa, Washington University in St. Louis, United States

Matthew Parno, The United States Army Corps of Engineers, United States

Noemi Petra, University of California, Merced, United States

Youssef Marzouk, Massachusetts Institute of Technology, United States

Omar Ghattas, The University of Texas at Austin, United States

23 March 2021

With a massive explosion of datasets across all areas of science and engineering, the central questions are: How do we optimally learn from data through the lens of models? And how do we account for uncertainties in both data and models? These questions can be mathematically framed as Bayesian inverse problems. While powerful and sophisticated approaches have been developed to tackle these problems, such methods are often challenging to implement and there is no available software that easily facilitates the analysis of Bayesian inverse problems. In this talk, we present an extensible FEniCS-based software framework hIPPYlib-MUQ that overcomes these challenges by providing access to state-of-the-art algorithms that offer the capability to solve complex large-scale Bayesian inverse problems across a broad spectrum of scientific and engineering areas.

---

You can cite this talk as:

Ki-Tae Kim, Umberto Villa, Matthew Parno, Noemi Petra, Youssef Marzouk, and Omar Ghattas. “hIPPYlib-MUQ: Scalable Markov chain Monte Carlo sampling methods for large-scale Bayesian inverse problems governed by PDEs”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 221. doi: 10.6084/m9.figshare.14495274.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/kim.html>.

# Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs

Federico Pichi (<https://people.sissa.it/~fpichi/>), SISSA, International School for Advanced Studies, Italy

Francesco Ballarin (<https://www.francescoballarin.it>), Catholic University of the Sacred Heart, Italy

Gianluigi Rozza (<https://people.sissa.it/~grozza/>), SISSA, International School for Advanced Studies, Italy

Jan S. Hesthaven, EPFL, École Polytechnique Fédérale de Lausanne, Switzerland

23 March 2021

This work aims to develop and investigate a computational framework to study parametrized partial differential equations (PDEs) which model nonlinear systems undergoing bifurcations. Bifurcation analysis, ie following the coexisting branches due to the non-uniqueness of the solution, as well as determining the bifurcation points themselves, are complex computational tasks [3] [4] [5]. The combination of reduced basis (RB) model reduction and artificial neural network (ANN) can potentially reduce the computational burden by several orders of magnitude and shed light on new strategies. Following the POD-NN approach [1], we analyzed two CFD applications where both physical and geometrical parameters were considered. We studied the Navier–Stokes equations for a viscous, steady, and incompressible flow: (i) in a planar straight channel with a narrow inlet of varying width, and (ii) in a triangular parametrized cavity [2]. All the simulations were performed within the open source software FEniCS and RBniCS [6] for the RB framework, integrated with PyTorch to construct the neural network.

## References

- [1] J. S. Hesthaven and S. Ubbiali. “Non-intrusive reduced order modeling of nonlinear problems using neural networks”. In: *Journal of Computational Physics* 363 (2018), 55–78. DOI: 10.1016/j.jcp.2018.02.037.
- [2] F. Pichi, F. Ballarin, G. Rozza, and J. S. Hesthaven. “Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs”. In: *in preparation* (2020).
- [3] F. Pichi, A. Quaini, and G. Rozza. “A reduced order modeling technique to study bifurcating phenomena: Application to the Gross–Pitaevskii equation”. In: *SIAM Journal on Scientific Computing* 42.5 (2020), B1115–B1135. DOI: 10.1137/20M1313106.
- [4] F. Pichi and G. Rozza. “Reduced basis approaches for parametrized bifurcation problems held by non-linear Von Kármán equations”. In: *Journal of Scientific Computing* 81.1 (2019), 112–135. DOI: 10.1007/s10915-019-01003-3.
- [5] F. Pichi, M. Strazzullo, F. Ballarin, and G. Rozza. “Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier–Stokes equations with model order reduction”. <https://arxiv.org/abs/2010.13506>. 2020.
- [6] “RBniCS - reduced order modelling in FEniCS”. <https://www.rbnicsproject.org>.

---

You can cite this talk as:

Federico Pichi, Francesco Ballarin, Gianluigi Rozza, and Jan S. Hesthaven. “Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 222–240. DOI: 10.6084/m9.figshare.14495280.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/pichi.html>.



# Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs



UNIVERSITÀ  
CATTOLICA  
del Sacro Cuore

**F. Pichi**<sup>1,2</sup>

F. Ballarin<sup>3</sup>, G. Rozza<sup>1</sup>

J. S. Hesthaven<sup>2</sup>

<sup>1</sup> mathLab, Mathematics Area, SISSA

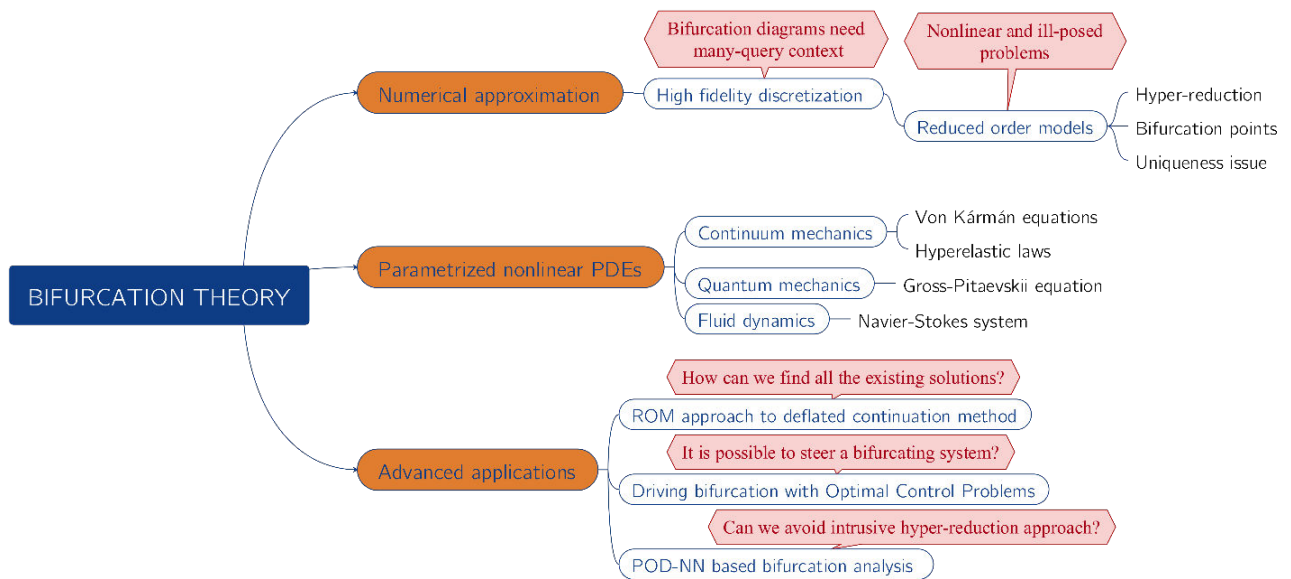
<sup>2</sup>EPFL Lausanne, MCSS

<sup>3</sup>Catholic University of the Sacred Heart

FE<sub>n</sub>iCS 2021

22-26 March

# Outline map



All the simulations were performed within **FEniCS**<sup>1</sup>, **RBniCS**<sup>2</sup> and **PyTorch**.

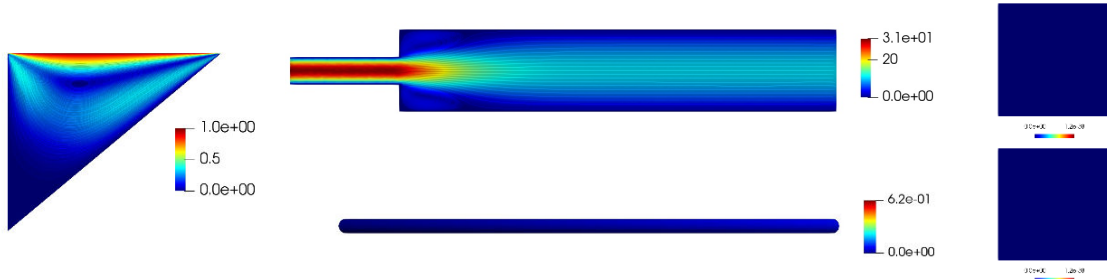
<sup>1</sup>A. Logg et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

<sup>2</sup>RBniCS - reduced order modelling in FEniCS. <https://www.rbnicsproject.org>

# Preliminary examples

## Question:

What have in common complex models coming from different physical contexts?  
⇒ Sudden changes linked to qualitatively different behaviour of the solutions.



## Example:

The situation for a compressed beam can change abruptly when the load is increased beyond a certain critical level at which the beam buckles.



## Bifurcation theory and its numerical approximation

We represent a nonlinear PDE with the parametrized mapping  $G : \mathbb{X} \times \mathcal{P} \rightarrow \mathbb{X}'$ .  
Given  $\mu \in \mathcal{P} \subset \mathbb{R}^P$ , seek  $X(\mu) \in \mathbb{X}$  such that:

Strong form

$$G(X(\mu); \mu) = 0, \quad \text{in } \mathbb{X}'. \quad (1)$$

Weak form

$$g(X(\mu), Y; \mu) = 0, \quad \forall Y \in \mathbb{X}. \quad (2)$$

Consider the finite dimensional space  $\mathbb{X}_{\mathcal{N}} \subset \mathbb{X}$ , with dimension  $\mathcal{N}$ .

---

**Algorithm 1** A pseudo-code for the reconstruction of a branch

---

- |   |                              |
|---|------------------------------|
| 1: $X_0 = X_{\text{guess}}$   | ▷ Initial guess              |
| 2: <b>for</b> $\mu_j \in \mathcal{P}_K$ <b>do</b>   | ▷ <b>Continuation loop</b>   |
| 3: $X_j^{(0)} = X_{j-1}$  | ▷ Continuation guess         |
| 4: <b>while</b> $\ G_{\mathcal{N}}(X_j^{(k)}; \mu_j)\ _{\mathbb{X}_{\mathcal{N}}} > \epsilon$ <b>do</b> | ▷ <b>Newton method</b>       |
| 5: $J_{\mathcal{N}}(X_j^{(k)}; \mu_j) \delta X = G_{\mathcal{N}}(X_j^{(k)}; \mu_j)$                     | ▷ <b>Galerkin-FE method</b>  |
| 6: $X_j^{(k+1)} = X_j^{(k)} - \delta X$   |                              |
| 7: <b>end while</b>   |                              |
| 8: $J_{\mathcal{N}}(X_j; \mu_j) X_e = \sigma_{\mu_j} M_{\mathcal{N}} X_e$                               | ▷ Eigenproblem for stability |
| 9: <b>end for</b>   |                              |

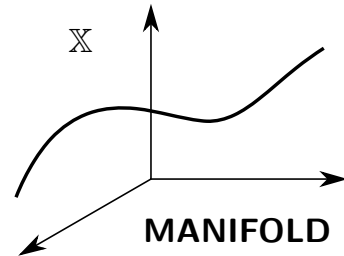
## Reduced Order Models (ROMs)<sup>3,4</sup>

We focus on **Reduced Basis** (RB) method based on **POD** strategy, defining

$$\mathbb{X}_N = \text{span}\{\Sigma^m, m = 1, \dots, N\} \subset \mathbb{X}_N, \quad \text{where } N \ll \mathcal{N},$$

where  $\{\Sigma^m\}_{m=1}^N$  are the basis functions obtained from the **snapshots**  $\{X_N(\mu^n)\}_{n=1}^{N_{train}}$ .

$$\underbrace{G(X(\mu); \mu) = 0}_{\text{Partial differential equation}}$$



<sup>3</sup>J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2015.

<sup>4</sup>A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2015.

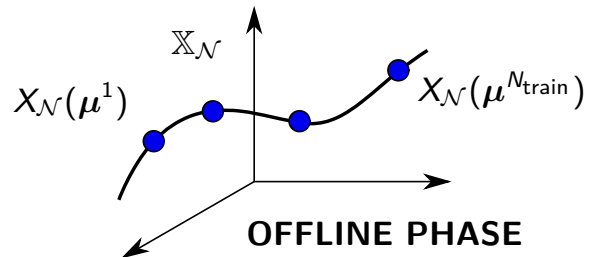
## Reduced Order Models (ROMs)<sup>3,4</sup>

We focus on **Reduced Basis** (RB) method based on **POD** strategy, defining

$$\mathbb{X}_N = \text{span}\{\Sigma^m, m = 1, \dots, N\} \subset \mathbb{X}_{\mathcal{N}}, \quad \text{where } N \ll \mathcal{N},$$

where  $\{\Sigma^m\}_{m=1}^N$  are the basis functions obtained from the **snapshots**  $\{X_{\mathcal{N}}(\mu^n)\}_{n=1}^{N_{\text{train}}}$ .

$$\underbrace{G(X(\mu); \mu) = 0}_{\text{Partial differential equation}} \rightsquigarrow \underbrace{G_{\mathcal{N}}(X_{\mathcal{N}}(\mu); \mu) = 0}_{\text{High Fidelity approximation}}$$



<sup>3</sup>J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2015.

<sup>4</sup>A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2015.

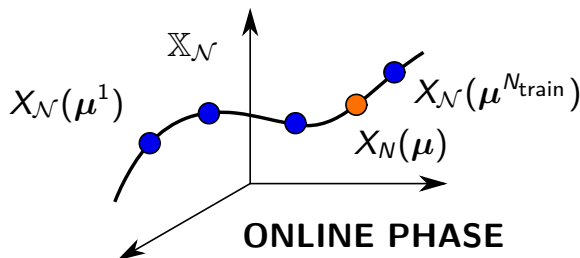
## Reduced Order Models (ROMs)<sup>3,4</sup>

We focus on **Reduced Basis** (RB) method based on **POD** strategy, defining

$$\mathbb{X}_N = \text{span}\{\Sigma^m, m = 1, \dots, N\} \subset \mathbb{X}_{\mathcal{N}}, \quad \text{where } N \ll \mathcal{N},$$

where  $\{\Sigma^m\}_{m=1}^N$  are the basis functions obtained from the **snapshots**  $\{X_{\mathcal{N}}(\mu^n)\}_{n=1}^{N_{\text{train}}}$ .

$$\underbrace{G(X(\mu); \mu) = 0}_{\text{Partial differential equation}} \rightsquigarrow \underbrace{G_{\mathcal{N}}(X_{\mathcal{N}}(\mu); \mu) = 0}_{\text{High Fidelity approximation}} \rightsquigarrow \underbrace{G_N(X_N(\mu); \mu) = 0}_{\text{Reduced Basis approximation}}$$



<sup>3</sup>J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2015.

<sup>4</sup>A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2015.

## Reduced Order Models (ROMs)<sup>3,4</sup>

We focus on **Reduced Basis** (RB) method based on **POD** strategy, defining

$$\mathbb{X}_N = \text{span}\{\Sigma^m, m = 1, \dots, N\} \subset \mathbb{X}_{\mathcal{N}}, \quad \text{where } N \ll \mathcal{N},$$

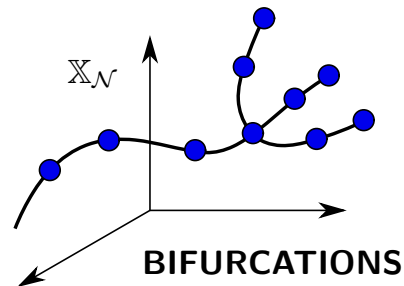
where  $\{\Sigma^m\}_{m=1}^N$  are the basis functions obtained from the **snapshots**  $\{X_{\mathcal{N}}(\mu^n)\}_{n=1}^{N_{train}}$ .

$$\underbrace{G(X(\mu); \mu) = 0}_{\text{Partial differential equation}} \rightsquigarrow \underbrace{G_{\mathcal{N}}(X_{\mathcal{N}}(\mu); \mu) = 0}_{\text{High Fidelity approximation}} \rightsquigarrow \underbrace{G_N(X_N(\mu); \mu) = 0}_{\text{Reduced Basis approximation}}$$

- **Global approach:**

**Pros:** single space encoding all branches

**Cons:** larger  $N$  and higher errors.



<sup>3</sup> J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2015.

<sup>4</sup> A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2015.



# Reduced Order Models (ROMs)<sup>3,4</sup>

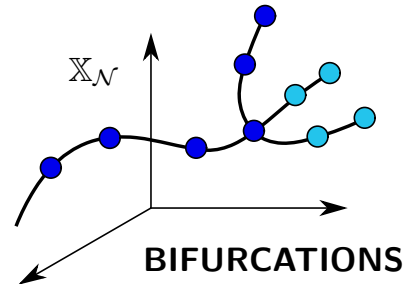
We focus on **Reduced Basis** (RB) method based on **POD** strategy, defining

$$\mathbb{X}_N = \text{span}\{\Sigma^m, m = 1, \dots, N\} \subset \mathbb{X}_{\mathcal{N}}, \quad \text{where } N \ll \mathcal{N},$$

where  $\{\Sigma^m\}_{m=1}^N$  are the basis functions obtained from the **snapshots**  $\{X_{\mathcal{N}}(\mu^n)\}_{n=1}^{N_{train}}$ .

$$\underbrace{G(X(\mu); \mu) = 0}_{\text{Partial differential equation}} \rightsquigarrow \underbrace{G_{\mathcal{N}}(X_{\mathcal{N}}(\mu); \mu) = 0}_{\text{High Fidelity approximation}} \rightsquigarrow \underbrace{G_N(X_N(\mu); \mu) = 0}_{\text{Reduced Basis approximation}}$$

- **Global approach:**  
**Pros:** single space encoding all branches  
**Cons:** larger  $N$  and higher errors.
- **Branch-wise approach:**  
**Pros:** low dimensional space  
**Cons:** hidden unsampled branch.



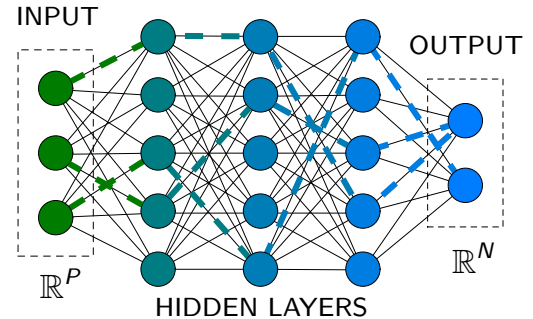
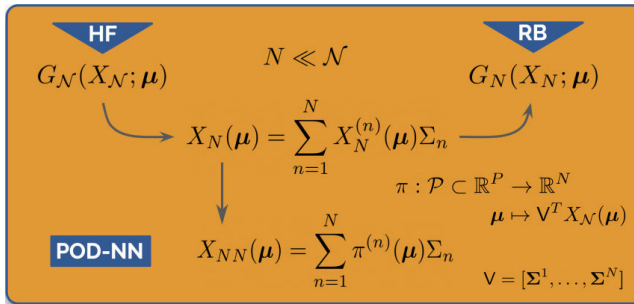
<sup>3</sup>J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2015.

<sup>4</sup>A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2015.

# Motivations for non-intrusive approach

<sup>5</sup>**Goal:** Investigate efficiently complex **bifurcating** behaviour in a **real-time** context.

<sup>6</sup>**How:** **POD-NN** approach combining ROMs and learning of reduced coefficients.



## POD-NN approach:

approximate  $\pi : \mathcal{P} \subset \mathbb{R}^P \rightarrow \mathbb{R}^N$  such that  $\mu \mapsto V^T X_N(\mu)$  from a training set given by the pairs  $\{(\mu^i, V^T X_N(\mu^i))\}_{i=1}^{N_{train}}$  obtained from the offline POD procedure.

<sup>5</sup>F. Pichi, F. Ballarin, G. Rozza, and J. S. Hesthaven. *Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs*. Preprint, 2020.

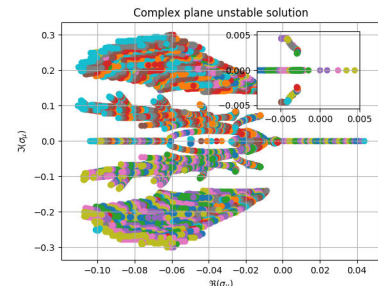
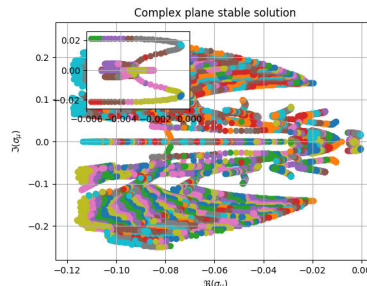
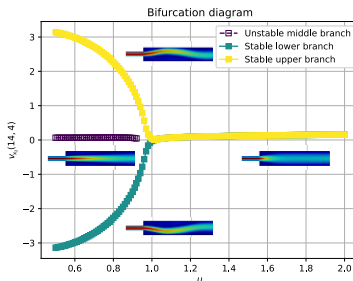
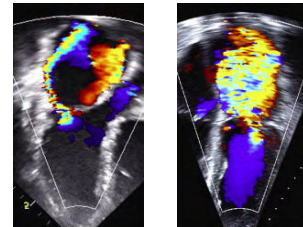
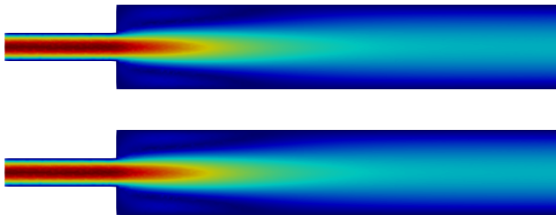
<sup>6</sup>J. S. Hesthaven and S. Ubbiali. *Non-intrusive reduced order modeling of nonlinear problems using neural networks*. Journal of Computational Physics, 363:55–78, 2018.

# Navier-Stokes application: the Coanda effect in a channel

## NS system for viscous, steady and incompressible flow

$$\begin{cases} -\mu\Delta v + v \cdot \nabla v + \nabla p = 0 & \text{in } \Omega, \\ \nabla \cdot v = 0 & \text{in } \Omega, \end{cases} \quad \text{with} \quad \begin{cases} v = v_{\text{in}} & \text{on } \Gamma_{\text{in}}, \\ v = 0 & \text{on } \Gamma_0, \\ -pn + (\mu\nabla v)n = 0 & \text{on } \Gamma_{\text{out}}, \end{cases}$$

**ISSUE:** For viscosities  $\mu \leq \mu^* \approx 0.96$  **wall-hugging** (stable) phenomena occur.

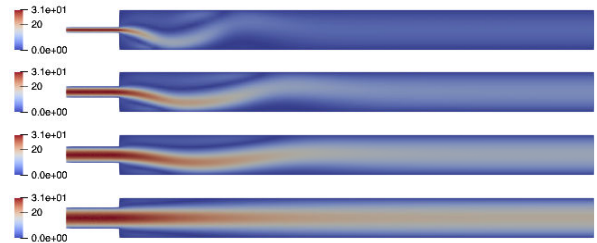
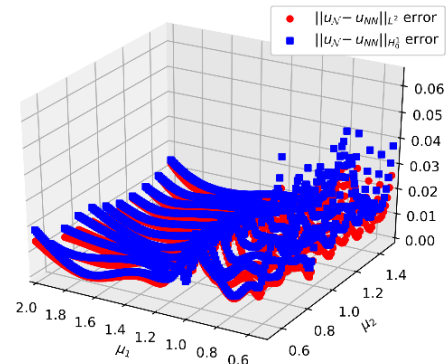
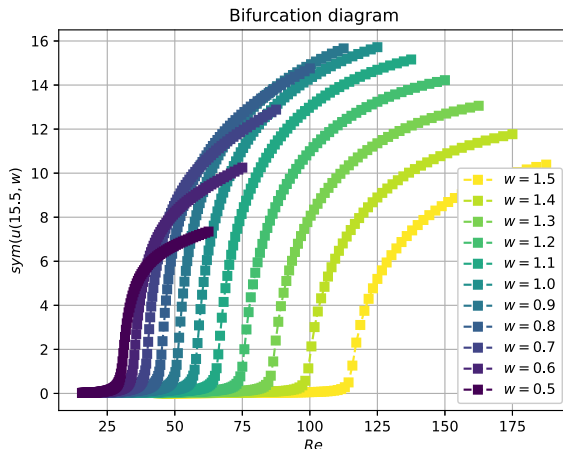


# Navier-Stokes application: the Coanda effect in a channel

## SETTING

**Parameter space**  $P = 2$ ,  $(\mu, w) \in \mathcal{P} = [0.5, 2] \times [0.5, 1.5]$ , viscosity and ch. width.  
**RB dimension**  $N_u = 50, N_p = 24$ . **Network** 2 layers, 15 neurons,  $N_{train} = 200 \cdot 6$ .

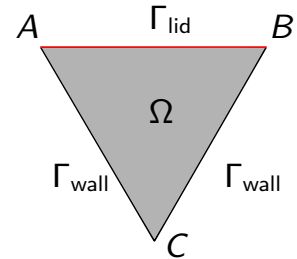
**POD-NN** speed-up =  $10^6$   
 $\epsilon_{NN}^{\max} = 0.0625$ ,  $\bar{\epsilon}_{NN} = 0.0118$ .  
**RB** speed-up = 1.5  
 $\epsilon_{RB}^{\max} = 0.7553$ ,  $\bar{\epsilon}_{RB} = 0.0129$ .



# Navier-Stokes application: triangular cavity flow

## Towards multiple bifurcating regimes:

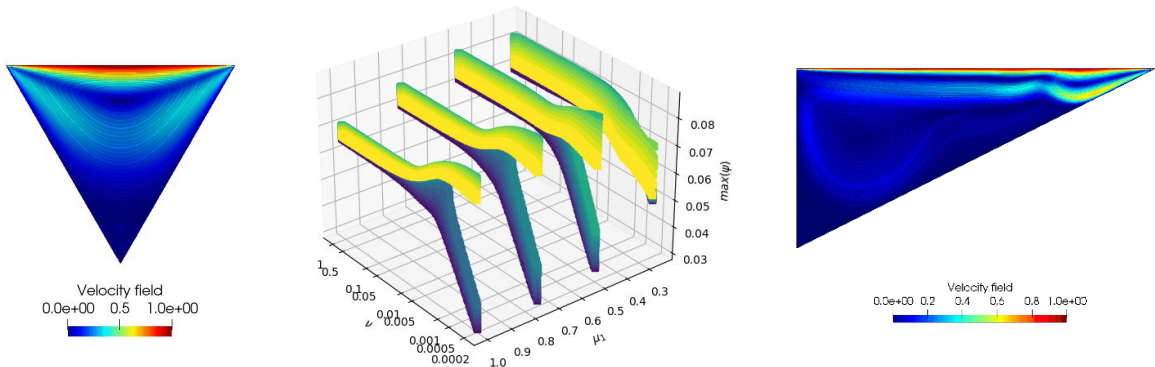
existence of a critical angle for the parametrized geometry causing a vortex attaching to vertex B increasing the Reynolds number.



## SETTING

**Parameter space**  $P = 3$ ,  $(\nu, \mu_1, \mu_2) \in \mathcal{P} = [2 \cdot 10^{-4}, 1] \times [-0.5, 0.5] \times [-.25, -1]$ , viscosity and bottom vertex position. **RB dimension**  $N_u = 100$ ,  $N_p = 44$ .

**Network** 3 layers, 20 neurons, log-equispaced sampling, tanh, epochs, Adam opt.

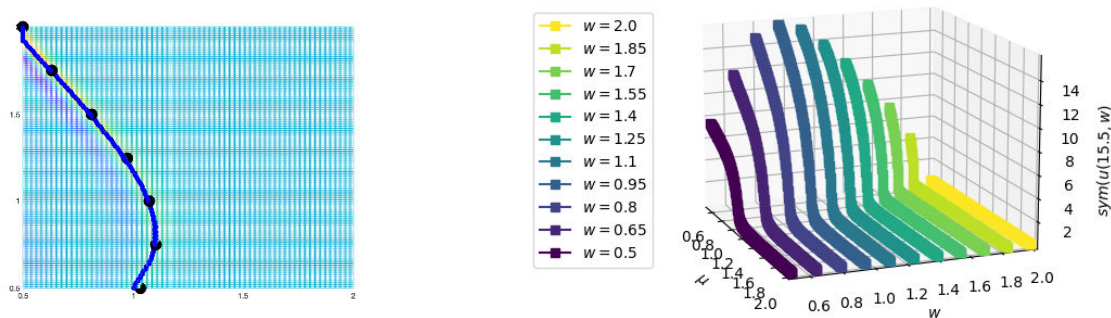


## A reduced manifold based bifurcation diagram

**Aim:** efficiently reconstruct a bifurcation diagram, where the output is entirely based on the **reduced coefficients** appearing in the RB expansion.

**Idea:** take advantage of the **non-smoothness** of the manifold, constructing a **detection tool** that is able to track the critical points employing its **curvature**.

**Result:**  $L^2$  relative error for the vector of the **critical points** is of the order  $10^{-2}$ .



**Figure:** Multi-parameter Coanda test case: (Left) Reduced manifold based bifurcation diagram reconstruction. (Right) RB/POD-NN based 3D bifurcation diagram.

## Conclusions and Perspectives

---

- △ We described the general framework for the approximation of **bifurcating nonlinear parametrized PDEs**.
- △ We investigated the intrusive **Reduced Basis** method to obtain an efficient evaluation of the bifurcation diagrams.
- △ We applied the non-intrusive **POD-NN** technique to recover the decoupling between offline and online phases.
- △ We presented an application of the methodology to the multi-parameter test cases: the **Coanda effect** in a channel and the **triangular cavity** flow.
- △ We developed a new empirical strategy employing the reduced coefficients to recover the bifurcation diagram from the manifold's curvature.

## Conclusions and Perspectives

---

- △ We described the general framework for the approximation of **bifurcating nonlinear parametrized PDEs**.
- △ We investigated the intrusive **Reduced Basis** method to obtain an efficient evaluation of the bifurcation diagrams.
- △ We applied the non-intrusive **POD-NN** technique to recover the decoupling between offline and online phases.
- △ We presented an application of the methodology to the multi-parameter test cases: the **Coanda effect** in a channel and the **triangular cavity** flow.
- △ We developed a new empirical strategy employing the reduced coefficients to recover the bifurcation diagram from the manifold's curvature.
- ▽ Reduce the number of training points needed incorporating physics with **Physics Informed Neural Networks** (PINNs).
- ▽ Embed **Automatic Machine Learning** (AutoML) to select best configuration for the hyper-parameters of the neural network.
- ▽ **Improve the decay** of the POD-NN technique w.r.t. the number of RB modes by developing new algorithmic procedure.



# List of publications

---

- [1] F. Pichi and G. Rozza. [Reduced basis approaches for parametrized bifurcation problems held by non-linear Von Kármán equations](#). *Journal of Scientific Computing*, 81(1):112–135, 2019.
- [2] D. B. P. Huynh, F. Pichi, and G. Rozza. [Reduced Basis Approximation and A Posteriori Error Estimation: Applications to Elasticity Problems in Several Parametric Settings](#), pages 203–247. Springer International Publishing, Cham, 2018.
- [3] M. Pintore, F. Pichi, M. Hess, G. Rozza, and C. Canuto. [Efficient computation of bifurcation diagrams with a deflated approach to reduced basis spectral element method](#). *Advances in Computational Mathematics*, 47(1), 2020.
- [4] F. Pichi, A. Quaini, and G. Rozza. [A reduced order modeling technique to study bifurcating phenomena: Application to the Gross–Pitaevskii equation](#). *SIAM Journal on Scientific Computing*, 42(5):B1115–B1135, 2020.
- [5] F. Pichi, M. Strazzullo, F. Ballarin, and G. Rozza. [Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier-Stokes equations and model reduction](#). ArXiv preprint, arXiv:2010.13506, 2020.
- [6] F. Pichi, J. Eftang, G. Rozza, and A. T. Patera. [Reduced order models for the buckling of hyperelastic beams](#). MIT-FVG “ROM2S” report, 2020.
- [7] F. Pichi, F. Ballarin, G. Rozza, and J. S. Hesthaven. [Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs](#). Preprint, 2020.
- [8] F. Pichi. [Reduced order models for parametric bifurcation problems in nonlinear PDEs](#). Ph.D. Thesis, 2020.

# List of publications

---

- [1] F. Pichi and G. Rozza. [Reduced basis approaches for parametrized bifurcation problems held by non-linear Von Kármán equations](#). *Journal of Scientific Computing*, 81(1):112–135, 2019.
- [2] D. B. P. Huynh, F. Pichi, and G. Rozza. [Reduced Basis Approximation and A Posteriori Error Estimation: Applications to Elasticity Problems in Several Parametric Settings](#), pages 203–247. Springer International Publishing, Cham, 2018.
- [3] M. Pintore, F. Pichi, M. Hess, G. Rozza, and C. Canuto. [Efficient computation of bifurcation diagrams with a deflated approach to reduced basis spectral element method](#). *Advances in Computational Mathematics*, 47(1), 2020.
- [4] F. Pichi, A. Quaini, and G. Rozza. [A reduced order modeling technique to study bifurcating phenomena: Application to the Gross–Pitaevskii equation](#). *SIAM Journal on Scientific Computing*, 42(5):B1115–B1135, 2020.
- [5] F. Pichi, M. Strazzullo, F. Ballarin, and G. Rozza. [Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier-Stokes equations and model reduction](#). ArXiv preprint, arXiv:2010.13506, 2020.
- [6] F. Pichi, J. Eftang, G. Rozza, and A. T. Patera. [Reduced order models for the buckling of hyperelastic beams](#). MIT-FVG “ROM2S” report, 2020.
- [7] F. Pichi, F. Ballarin, G. Rozza, and J. S. Hesthaven. [Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs](#). Preprint, 2020.
- [8] F. Pichi. [Reduced order models for parametric bifurcation problems in nonlinear PDEs](#). Ph.D. Thesis, 2020.

*Thank for your attention*

# Run-time from 300 years to 300 min: Lessons learned in large-scale modeling in FEniCS

**Abhinav Gupta** ([https://computationalmechanics.in/rajob\\_teams/abhinav-gupta](https://computationalmechanics.in/rajob_teams/abhinav-gupta)),  
Indian Institute of Technology Roorkee, India

U Meenu Krishnan, Indian Institute of Technology Roorkee, India

Rajib Chowdhury, Indian Institute of Technology Roorkee, India

Anupam Chakrabarti, Indian Institute of Technology Roorkee, India

**23 March 2021**

Topology optimization and phase-field fracture simulations necessitate the use of high mesh density in the model. Thus, practical problems could range from a million to more than a billion degrees of freedom. It is impossible to solve such systems without proper knowledge of good programming practices, efficient memory management, and parallel computation. Fortunately, FEniCS works out of the box for parallel computation. However, a researcher working with FEniCS should also understand the algorithm's computational complexity and optimize it for the best performance.

This talk aims to provide beginners with proper guidelines for handling large-scale systems in FEniCS. We investigate the work cost and the storage cost related to different components of the program. We will also comment on good programming practices that allow for a drastic reduction in program run-time. Furthermore, we present our study on a topology optimization problem and discuss our mistakes while working with small mesh that resulted in an estimated run-time of 300 years for a large-scale system. We finally present the redesigned implementation that brought the run-time close to 300 mins.

---

You can cite this talk as:

Abhinav Gupta, U Meenu Krishnan, Rajib Chowdhury, and Anupam Chakrabarti. "Run-time from 300 years to 300 min: Lessons learned in large-scale modeling in FEniCS". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 241–257. doi: 10.6084/m9.figshare.14495289.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/gupta.html>.



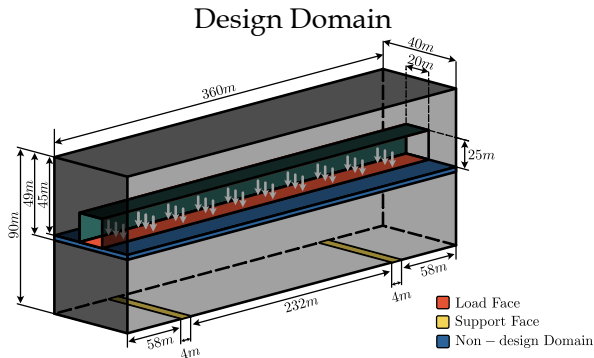
# RUN-TIME FROM 300 YEARS TO 300 MIN: LESSONS LEARNED IN LARGE-SCALE MODELING IN FENICS.

Abhinav Gupta, U Meenu Krishnan, Rajib Chowdhury, Anupam Chakrabarti

Department of Civil Engineering  
Indian Institute of Technology Roorkee, India

23 March 2021

# Motivation



## Topologically optimised design



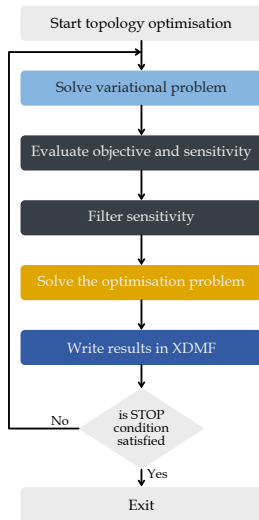
- (I) Solve the topology optimization problem for a medium to large scale engineering structure.
- (II) The problem could contain degrees of freedom ranging from a million to over a billion.

# Coding topology optimization in FEniCS

$$\underset{\theta}{\text{minimize}} \quad f(\theta) = \theta^p \psi(\epsilon)$$

$$\text{subject to} \quad g(\theta) = \int_{\Omega} \theta d\Omega - kV_0 \leq 0$$

$$\psi(\epsilon) = E_0 \left[ \frac{\lambda_0}{2} (tr[\epsilon])^2 + \mu_0 tr[\epsilon^2] \right]$$



MATLAB  
99-Line code

```

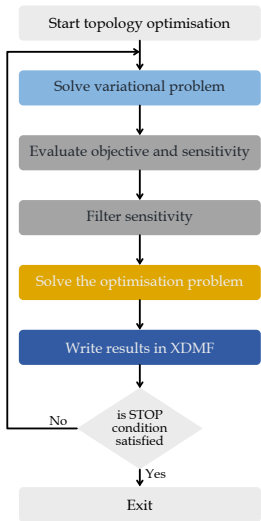
function top(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 1;
change = 1.;
% START ITERATION
while change > 0.01
    loop = loop + 1;
    xold = x;
% FE-ANALYSIS
[U]=FE(nelx,nely,x,penal);
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
[V0] = 1x;
C = 0.;
for ely = 1:nely
    for elx = 1:nelx
        nl = (nely+1)*(elx-1)+ely;
        n2 = (nely-1)*elx + ely;
        de = U([nl-1];nl); de2 = [2*nl; 2*n2]; 2*n2+2; 2*n1+1; 2*n1+2; 1);
        c = C + x(ely,elx)*penal*de'*K*de;
        dc(ely,elx) = -penal*x(ely,elx)*(penal-1)*de'*K*de;
    end
end
% FILTERING OF SENSITIVITIES
[dc1] = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
[x] = OC(nelx,nely,x,volfrac,dc);
% PRINT RESULTS
change = max(max(abs(x-xold)));
disp(['It.: ', sprintf('%d',loop) ' Obj.: ', sprintf('%8.4f',c) ...
    ' Vol.: ', sprintf('%6.3f',sum(sum(x)))/(nelx*nely) ...
    ' Ch.: ', sprintf('%6.3f',change) ]);
% PLOT DENSITIES
figure;imagesc(x); imagesc(xold); axis equal; axis tight; axis off; pause(1e-6);
end
    
```

FEniCS  
55-Line code

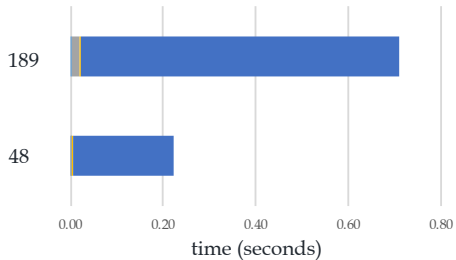
```

while change > 0.01 and loop < 100:
    loop = loop + 1
    density_old.assign(density)
    # FE-ANALYSIS
    solver.solve()
    # OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    objective = density * penal * ps(u_sol)
    sensitivity = project(-diff(objective, density), D).vector()[1:]
    # FILTERING/MODIFICATION OF SENSITIVITIES
    sensitivity = np.divide(distance_mat @ np.multiply(density,vector()[1:]),
        1); sensitivity, np.multiply(density,vector()[1:], distance_sum))
    # DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
    while l2 - l1 > 1e-4:
        Lmid = 0.5 * (l2 + l1)
        density_new.vector()[1:] = np.maximum(0.001, np.minimum(density,
            vector()[1:] - move, np.minimum(1.0, np.minimum(density,vector()[1:] + move,
            density,vector()[1:] * np.sqrt(-sensitivity / V0 / Lmid))))
        current_vol = assemble(density_new * dx)
        l1, l2 = (Lmid, l2) if current_vol > volfrac * V0.sum() else (l1, Lmid)
    # PRINT RESULTS
    change = max(density_new.vector()[1:] - density_old.vector()[1:])
    print("It.: %d, Obj.: %1.3f Vol.: %2.3f, Ch.: %3.3f" % (loop, project(objective, D).vector().sum(), current_vol / V0.sum(), change))
    density.assign(density_new)
    xdf.write(density, loop)
    
```

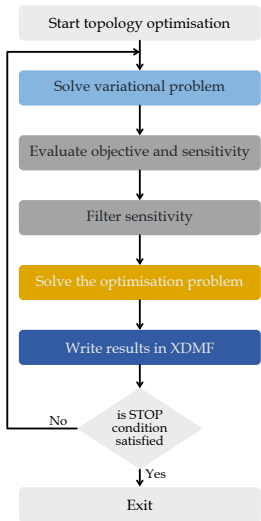
# The problem



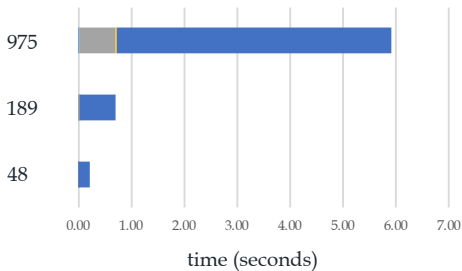
degrees of freedom



# The problem

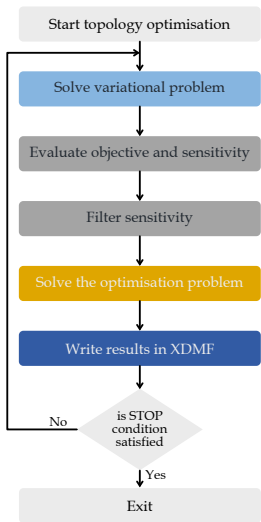


degrees of  
freedom

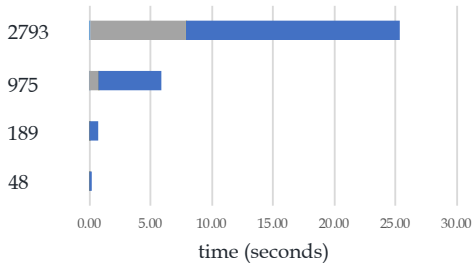




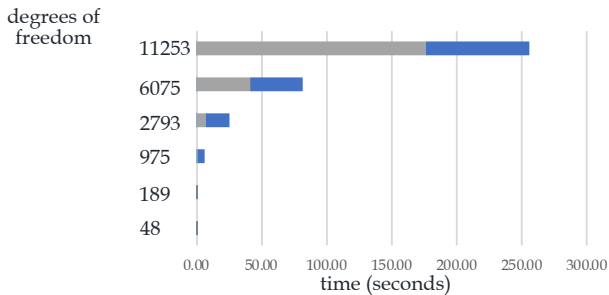
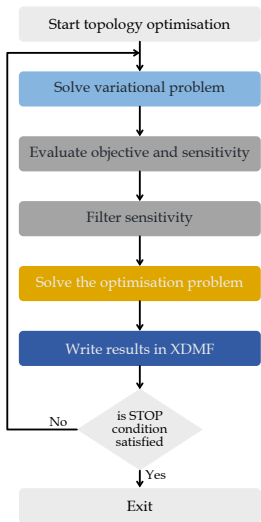
# The problem



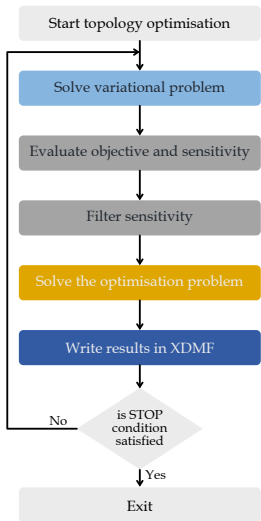
degrees of freedom



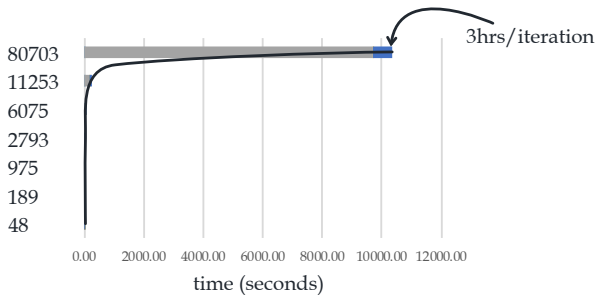
# The problem



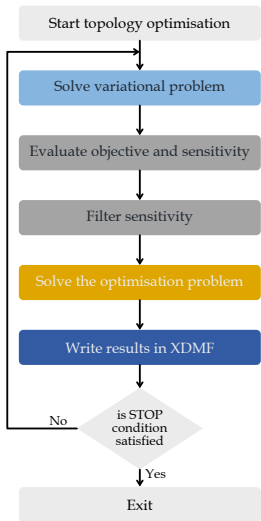
# The problem



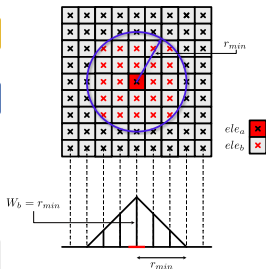
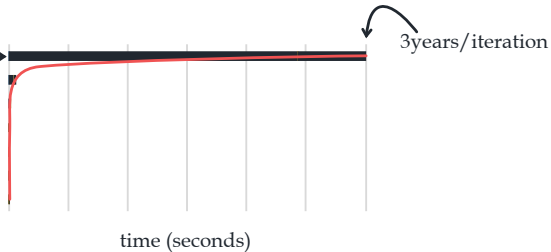
degrees of freedom



# Loops are excruciatingly slow.



1 million degrees of freedom



```
# PREPARE DISTANCE MATRICES FOR FILTER
midpoint = [cell.midpoint() for cell in cells(mesh)]
row_ind, col_ind, data = [], [], []
for row, ele_a in enumerate(midpoint):
    for col, ele_b in enumerate(midpoint):
        distance = ele_a.distance(ele_b)
        if distance < rmin + pow(10.0, -12.0):
            row_ind.append(row)
            col_ind.append(col)
            data.append(distance)
distance_mat = sp.csr_matrix((data, (row_ind, col_ind)), shape = (len(
    midpoint), len(midpoint)))
distance_sum = np.array(distance_mat.sum(1))[:, 0]
```

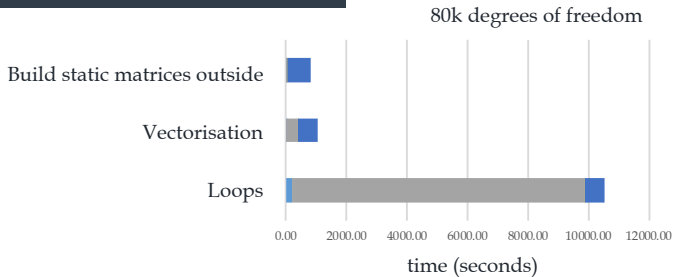
# Vectorization and static matrices

## Loops

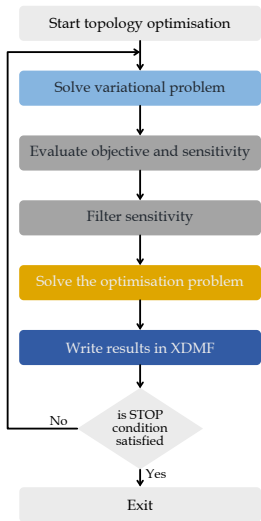
```
# PREPARE DISTANCE MATRICES FOR FILTER -----
midpoint = [cell.midpoint() for cell in cells(mesh)]
row_ind, col_ind, data = [], [], []
for row, ele_a in enumerate(midpoint):
    for col, ele_b in enumerate(midpoint):
        distance = ele_a.distance(ele_b)
        if distance < rmin + pow(10.0,-12.0):
            row_ind.append(row)
            col_ind.append(col)
            data.append(distance)
distance_mat = sp.csr_matrix((data, (row_ind, col_ind)), shape = (len(
    midpoint), len(midpoint)))
distance_sum = np.array(distance_mat.sum(1))[:, 0]
```

## Vectorisation

```
# PREPARE DISTANCE MATRICES FOR FILTER -----
midpoint = [cell.midpoint().array()[:] for cell in cells(mesh)]
distance_mat = np.maximum(rmin - sp.euclidean_distances(midpoint,
    midpoint), 0)
distance_sum = distance_mat.sum(1)
```



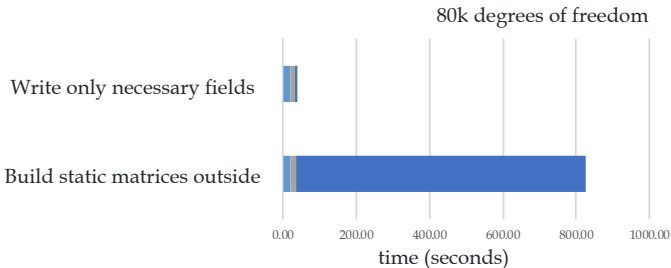
# Controlled post processing



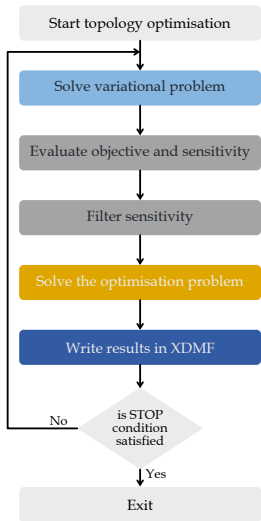
```
xdmf.write(project(psi(u_sol),D), loop)
```

Each call to project results in a call to solve for approximating the field by finite element method.

$$w = E_0 \left[ \frac{\lambda_0}{2} (tr[\epsilon])^2 + \mu_0 tr[\epsilon^2] \right]$$

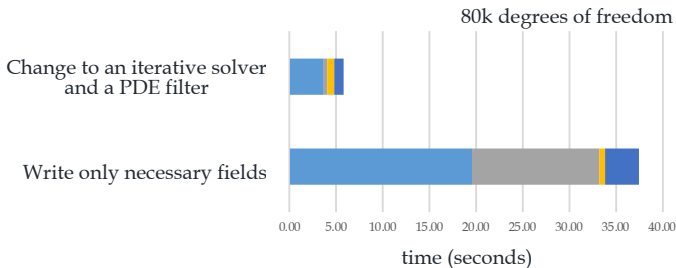


# Properly select/configure the solver and preconditioner.

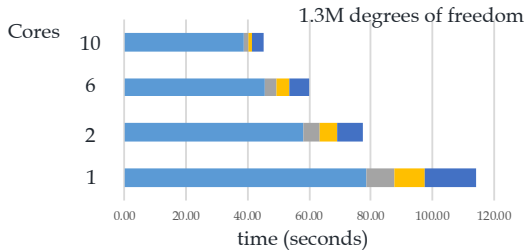
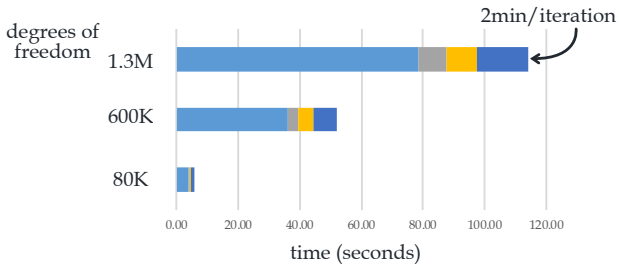
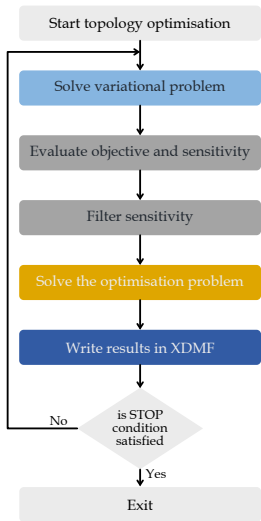


```
problem = LinearVariationalProblem(K, F, u_sol, bcs)
solver = LinearVariationalSolver(problem)
solver.parameters["linear_solver"] = "gmres"
solver.parameters["preconditioner"] = "hypr_euclid"

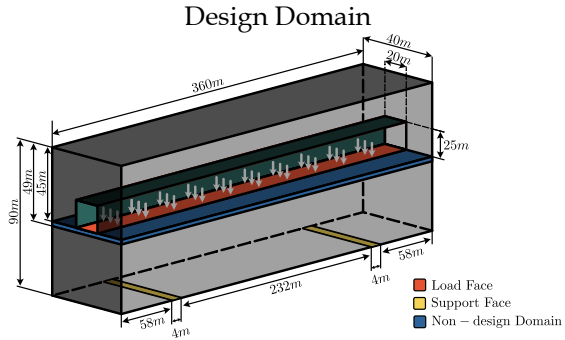
project(psi(u_sol), D, solver_type="gmres", preconditioner_type="hypr_euclid")
```



# Parallelization.







Topologically optimised design



- (I) General guidelines for handling medium to large-scale systems in FEniCS
  - (i) Always profile the code and look for bottlenecks.
  - (ii) Avoid use of loops in python. Look for efficient alternatives.
  - (iii) Avoid re-evaluation of matrices that do not change.
  - (iv) Evaluate and write only necessary simulation outputs.
  - (v) In an iterative process evaluate output at every  $n^{th}$  step to further speed up the simulation.
  - (vi) Properly select/configure the solver and preconditioner based on the problem.
- (II) Stepping into the realm of large scale simulations require knowledge of good programming practices, parallelization, and a deep understanding of the working principles of the tools/libraries.

Thanks

---

iitrabhi@gmail.com  
computationalmechanics.in

# Piola-mapped finite elements in Firedrake for linear elasticity and Stokes flow

**Francis Aznaran** (<https://www.maths.ox.ac.uk/people/francis.aznaran>), University of Oxford, United Kingdom

**Patrick Farrell** (<https://www.maths.ox.ac.uk/people/patrick.farrell>), University of Oxford, United Kingdom

**Robert Kirby** ([https://sites.baylor.edu/robert\\_kirby/](https://sites.baylor.edu/robert_kirby/)), Baylor University, United States

23 March 2021

Many finite element spaces are not preserved by the standard pullback to the reference cell. Robust implementation therefore requires studying the relation between degrees of freedom under pushforward, in order to obtain the correct bases on a generic physical triangle [1]. In this work, we extend this transformation theory to vector- and tensor-valued elements mapped by the contravariant Piola transform. We apply this theory, and describe its efficient implementation in Firedrake, for the the Mardal–Tai–Winther elements discretizing  $H(\text{div})$  for Stokes–Darcy flow, and the Arnold–Winther elements discretizing  $H(\text{div}; \mathbb{S})$  for the stress-displacement formulation of linear elasticity.

In particular, the Arnold–Winther elements were the first to stably enforce exact symmetry of the Cauchy stress tensor; we demonstrate how they may be efficiently mapped, while the few prior implementations are either custom-made for specific numerical experiments, or require the explicit element-by-element construction of the basis. Our novel implementation of these exotic elements composes inexpensively and automatically with the rest of the Firedrake code stack; numerical results are presented.

## References

- [1] R. C. Kirby. “A general approach to transforming finite elements”. In: *The SMAI journal of computational mathematics* 4 (2018), 197–224. DOI: 10.5802/smai-jcm.33.

---

You can cite this talk as:

Francis Aznaran, Patrick Farrell, and Robert Kirby. “Piola-mapped finite elements in Firedrake for linear elasticity and Stokes flow”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 258–301. DOI: 10.6084/m9.figshare.14495295.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/aznaran.html>.



Mathematical  
Institute

# Piola-mapped finite elements for linear elasticity and Stokes flow

FRANCIS AZNARAN <sup>\*</sup>, PATRICK FARRELL <sup>\*</sup>, ROBERT KIRBY <sup>†</sup>

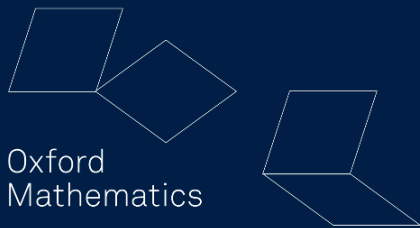
<sup>\*</sup> *University of Oxford, UK*

<sup>†</sup> *Baylor University, TX*

FEniCS 2021

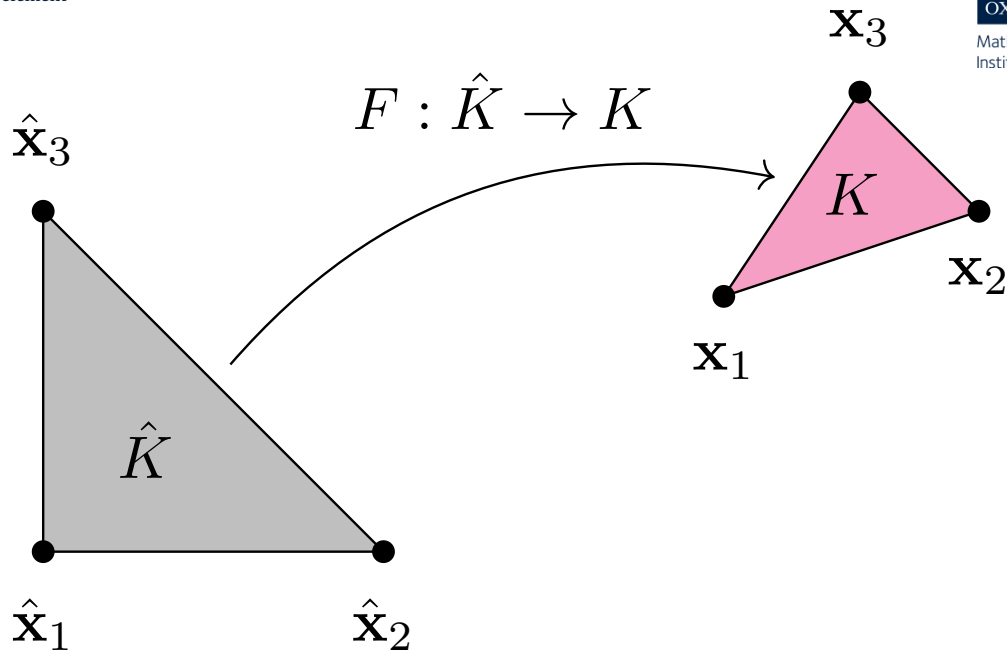
23rd March 2021

Oxford  
Mathematics



# Motivation

The reference element



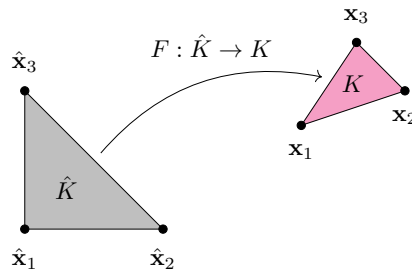
## Motivation

The reference-to-physical  
map  $F : K \rightarrow \hat{K}$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

induces a *pullback operator* on  
reference functions  $\hat{\phi}$ :

$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$



## Motivation

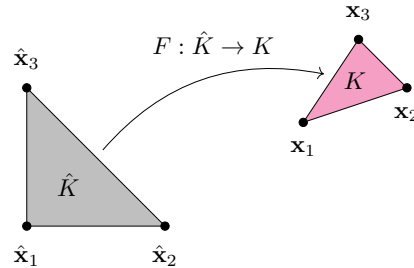
The reference-to-physical  
map  $F : K \rightarrow \hat{K}$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

induces a *pullback operator* on  
reference functions  $\hat{\phi}$ :

$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$

- Standard pullback preserves bases of **affine equivalent** elements:  
Lagrange, Crouzeix–Raviart, ...





## Motivation

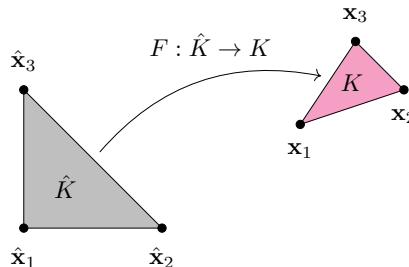
The reference-to-physical  
map  $F : K \rightarrow \hat{K}$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

induces a *pullback operator* on  
reference functions  $\hat{\phi}$ :

$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$

- Standard pullback preserves bases of **affine equivalent** elements:  
Lagrange, Crouzeix–Raviart, ...
- Many elements are **not preserved**: Hermite, Argyris, Morley, Bell, ...



## Motivation

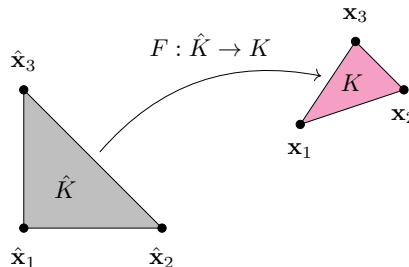
The reference-to-physical  
map  $F : K \rightarrow \hat{K}$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

induces a *pullback operator* on  
reference functions  $\hat{\phi}$ :

$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$

- Standard pullback preserves bases of **affine equivalent** elements: Lagrange, Crouzeix–Raviart, ...
- Many elements are **not preserved**: Hermite, Argyris, Morley, Bell, ...
- Transformation theory of Kirby [2018] showed how to obtain the correct bases on a generic physical cell.



## Motivation

The reference-to-physical map  $F : \hat{K} \rightarrow K$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

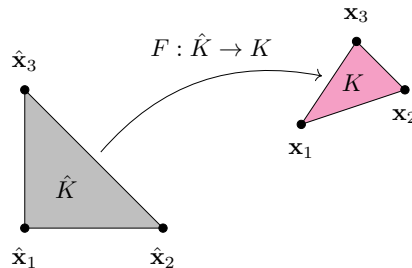
induces a *pullback operator* on reference functions  $\hat{\phi}$ :

$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$

- Standard pullback preserves bases of **affine equivalent** elements: Lagrange, Crouzeix–Raviart, ...
- Many elements are **not preserved**: Hermite, Argyris, Morley, Bell, ...
- Transformation theory of Kirby [2018] showed how to obtain the correct bases on a generic physical cell.

### Goal of this work:

Extend this theory to  $H(\text{div})$  elements.



## Motivation

The reference-to-physical  
map  $F : K \rightarrow \hat{K}$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

induces a *pullback operator* on  
reference functions  $\hat{\phi}$ :

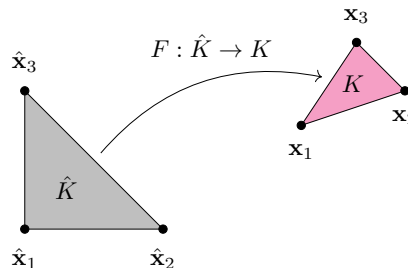
$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$

- Standard pullback preserves bases of **affine equivalent** elements: Lagrange, Crouzeix–Raviart, ...
- Many elements are **not preserved**: Hermite, Argyris, Morley, Bell, ...
- Transformation theory of Kirby [2018] showed how to obtain the correct bases on a generic physical cell.

### Goal of this work:

Extend this theory to  $H(\text{div})$  elements.

*Representative elements:*  $\begin{cases} H(\text{div}) : \text{Mardal–Tai–Winther [2002]}. \\ H(\text{div}; \mathbb{S}) : \text{Arnold–Winther [2002, 2003]}. \end{cases}$



## Motivation

The reference-to-physical  
map  $F : K \rightarrow \hat{K}$  between cells

$$F(\hat{\mathbf{x}}) = J\hat{\mathbf{x}} + \mathbf{b}$$

induces a *pullback operator* on  
reference functions  $\hat{\phi}$ :

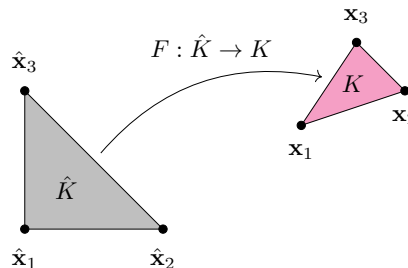
$$\left( \hat{\mathbf{x}} \mapsto \hat{\phi}(\hat{\mathbf{x}}) \right) \mapsto \left( \mathbf{x} \mapsto \phi(\mathbf{x}) := \hat{\phi}(F^{-1}(\mathbf{x})) \right).$$

- Standard pullback preserves bases of **affine equivalent** elements: Lagrange, Crouzeix–Raviart, ...
- Many elements are **not preserved**: Hermite, Argyris, Morley, Bell, ...
- Transformation theory of Kirby [2018] showed how to obtain the correct bases on a generic physical cell.

### Goal of this work:

Extend this theory to  $H(\text{div})$  elements.

*Representative elements:*  $\begin{cases} H(\text{div}) : \text{Mardal–Tai–Winther [2002]}. \\ H(\text{div}; \mathbb{S}) : \text{Arnold–Winther [2002, 2003]}. \end{cases}$



## Definition

Let  $F : \hat{K} \rightarrow K$ ,  $J(\hat{\mathbf{x}}) = \hat{\nabla} F(\hat{\mathbf{x}})$  its Jacobian. The *contravariant Piola transform* takes

$$\left( \hat{\Phi} : \hat{K} \rightarrow \mathbb{R}^d \right) \mapsto \left( \mathcal{F}^{\text{div}}(\hat{\Phi}) = \Phi : K \rightarrow \mathbb{R}^d \right),$$

$$\mathcal{F}^{\text{div}}(\hat{\Phi}) := \frac{1}{\det J} J \hat{\Phi} \circ F^{-1}.$$

## Definition

Let  $F : \hat{K} \rightarrow K$ ,  $J(\hat{\mathbf{x}}) = \hat{\nabla} F(\hat{\mathbf{x}})$  its Jacobian. The *contravariant Piola transform* takes

$$\begin{aligned} \left( \hat{\Phi} : \hat{K} \rightarrow \mathbb{R}^d \right) &\mapsto \left( \mathcal{F}^{\text{div}}(\hat{\Phi}) = \Phi : K \rightarrow \mathbb{R}^d \right), \\ \mathcal{F}^{\text{div}}(\hat{\Phi}) &:= \frac{1}{\det J} J \hat{\Phi} \circ F^{-1}. \end{aligned}$$

The *double contravariant Piola transform* is

$$\begin{aligned} \left( \hat{\tau} : \hat{K} \rightarrow \mathbb{S} \right) &\mapsto \left( \mathcal{F}^{\text{div}, \text{div}}(\hat{\tau}) = \tau : K \rightarrow \mathbb{S} \right), \\ \mathcal{F}^{\text{div}, \text{div}}(\hat{\tau}) &:= \frac{1}{(\det J)^2} J(\hat{\tau} \circ F^{-1}) J^{\top}. \end{aligned}$$

## Definition

Let  $F : \hat{K} \rightarrow K$ ,  $J(\hat{\mathbf{x}}) = \hat{\nabla} F(\hat{\mathbf{x}})$  its Jacobian. The *contravariant Piola transform* takes

$$\begin{aligned} \left( \hat{\Phi} : \hat{K} \rightarrow \mathbb{R}^d \right) &\mapsto \left( \mathcal{F}^{\text{div}}(\hat{\Phi}) = \Phi : K \rightarrow \mathbb{R}^d \right), \\ \mathcal{F}^{\text{div}}(\hat{\Phi}) &:= \frac{1}{\det J} J \hat{\Phi} \circ F^{-1}. \end{aligned}$$

The *double contravariant Piola transform* is

$$\begin{aligned} \left( \hat{\tau} : \hat{K} \rightarrow \mathbb{S} \right) &\mapsto \left( \mathcal{F}^{\text{div}, \text{div}}(\hat{\tau}) = \tau : K \rightarrow \mathbb{S} \right), \\ \mathcal{F}^{\text{div}, \text{div}}(\hat{\tau}) &:= \frac{1}{(\det J)^2} J(\hat{\tau} \circ F^{-1}) J^{\top}. \end{aligned}$$

## Fact

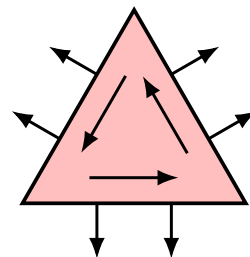
These are isomorphisms

$$H(\text{div}, \hat{K}) \xrightarrow{\sim} H(\text{div}, K), \quad H(\text{div}, \hat{K}; \mathbb{S}) \xrightarrow{\sim} H(\text{div}, K; \mathbb{S}).$$



## The Mardal–Tai–Winther element

$$MTW(K) = \{ \Phi \in \mathcal{P}_3(K; \mathbb{R}^2) \mid \operatorname{div} \Phi \in \mathcal{P}_0(K), (\Phi \cdot \mathbf{n})|_e \in \mathcal{P}_1(e) \forall \text{ edges } e \}$$

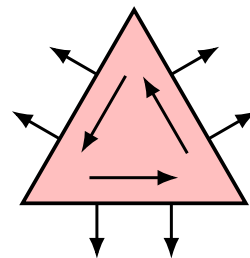


## The Mardal–Tai–Winther element

$$MTW(K) = \{ \Phi \in \mathcal{P}_3(K; \mathbb{R}^2) \mid \operatorname{div} \Phi \in \mathcal{P}_0(K), (\Phi \cdot \mathbf{n})|_e \in \mathcal{P}_1(e) \forall \text{ edges } e \}$$

Novelties:

- Discretises  $H(\operatorname{div})$  and (nonconforming)  $\mathbf{H}^1(\Omega)$

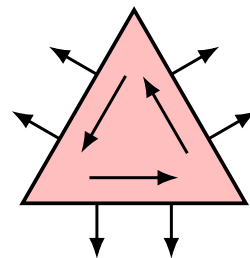


## The Mardal–Tai–Winther element

$$MTW(K) = \{ \Phi \in \mathcal{P}_3(K; \mathbb{R}^2) \mid \operatorname{div} \Phi \in \mathcal{P}_0(K), (\Phi \cdot \mathbf{n})|_e \in \mathcal{P}_1(e) \forall \text{ edges } e \}$$

Novelties:

- Discretises  $H(\operatorname{div})$  and (nonconforming)  $\mathbf{H}^1(\Omega)$
- Stable for both Stokes and Darcy flow

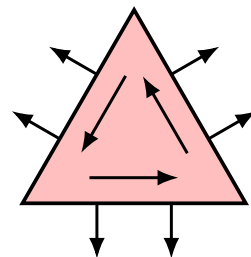


## The Mardal–Tai–Winther element

$$MTW(K) = \{ \Phi \in \mathcal{P}_3(K; \mathbb{R}^2) \mid \operatorname{div} \Phi \in \mathcal{P}_0(K), (\Phi \cdot \mathbf{n})|_e \in \mathcal{P}_1(e) \forall \text{ edges } e \}$$

Novelties:

- Discretises  $H(\operatorname{div})$  and (nonconforming)  $\mathbf{H}^1(\Omega)$
- Stable for both Stokes and Darcy flow
- Locking-free element for (poro)elasticity; discrete Korn inequality

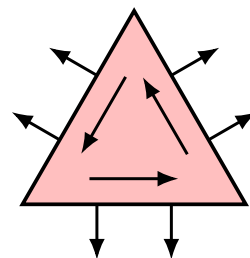


## The Mardal–Tai–Winther element

$$MTW(K) = \{ \Phi \in \mathcal{P}_3(K; \mathbb{R}^2) \mid \operatorname{div} \Phi \in \mathcal{P}_0(K), (\Phi \cdot \mathbf{n})|_e \in \mathcal{P}_1(e) \forall \text{ edges } e \}$$

Novelties:

- Discretises  $H(\operatorname{div})$  and (nonconforming)  $\mathbf{H}^1(\Omega)$
- Stable for both Stokes and Darcy flow
- Locking-free element for (poro)elasticity; discrete Korn inequality
- Divergence-free for Stokes when paired with  $DG(0)$



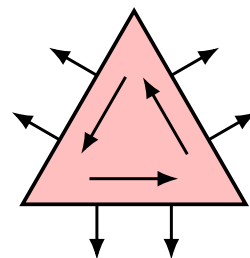
## The Mardal–Tai–Winther element

$$MTW(K) = \{ \Phi \in \mathcal{P}_3(K; \mathbb{R}^2) \mid \operatorname{div} \Phi \in \mathcal{P}_0(K), (\Phi \cdot \mathbf{n})|_e \in \mathcal{P}_1(e) \forall \text{ edges } e \}$$

Novelties:

- Discretises  $H(\operatorname{div})$  and (nonconforming)  $\mathbf{H}^1(\Omega)$
- Stable for both Stokes and Darcy flow
- Locking-free element for (poro)elasticity; discrete Korn inequality
- Divergence-free for Stokes when paired with  $DG(0)$
- Discretises the 2D Stokes complex:

$$\mathbb{R} \xrightarrow{\subset} H^2(\Omega) \xrightarrow{\operatorname{curl}} \mathbf{H}^1(\Omega) \xrightarrow{\operatorname{div}} L^2(\Omega) \longrightarrow 0.$$





### The Arnold–Winther stress elements

Conforming:  $AW^c(K) = \{\tau \in \mathcal{P}_3(K; \mathbb{S}) \mid \operatorname{div} \tau \in \mathcal{P}_1(K; \mathbb{R}^2)\}$

Nonconforming:  $AW^{nc}(K) = \{\tau \in \mathcal{P}_2(K; \mathbb{S}) \mid \mathbf{n} \cdot \tau \mathbf{n} \in \mathcal{P}_1(\mathbf{e}) \forall \text{ edges } \mathbf{e}\}$

paired with  $\mathbf{DG}(1)$  for the displacement.

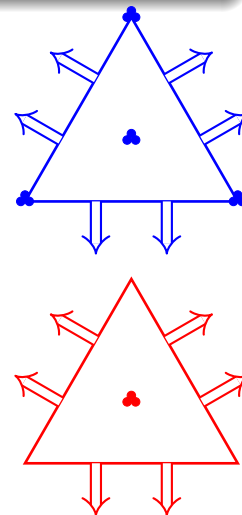
## The Arnold–Winther stress elements

Conforming:  $AW^c(K) = \{\tau \in \mathcal{P}_3(K; \mathbb{S}) \mid \operatorname{div} \tau \in \mathcal{P}_1(K; \mathbb{R}^2)\}$

Nonconforming:  $AW^{nc}(K) = \{\tau \in \mathcal{P}_2(K; \mathbb{S}) \mid \mathbf{n} \cdot \tau \mathbf{n} \in \mathcal{P}_1(\mathbf{e}) \forall \text{ edges } \mathbf{e}\}$

paired with  $\mathbf{DG}(1)$  for the displacement.

- Exact enforcement of the symmetry of the Cauchy stress.





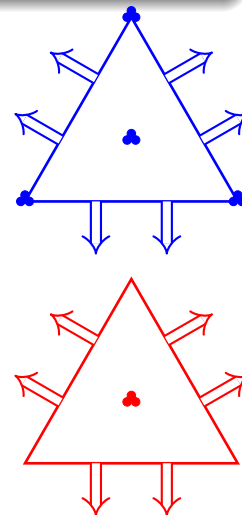
## The Arnold–Winther stress elements

Conforming:  $AW^c(K) = \{\tau \in \mathcal{P}_3(K; \mathbb{S}) \mid \operatorname{div} \tau \in \mathcal{P}_1(K; \mathbb{R}^2)\}$

Nonconforming:  $AW^{nc}(K) = \{\tau \in \mathcal{P}_2(K; \mathbb{S}) \mid \mathbf{n} \cdot \tau \mathbf{n} \in \mathcal{P}_1(\mathbf{e}) \forall \text{ edges } \mathbf{e}\}$

paired with  $\mathbf{DG}(1)$  for the displacement.

- Exact enforcement of the symmetry of the Cauchy stress.
- Stable and convergent for stress-displacement linear elasticity.



## The Arnold–Winther stress elements

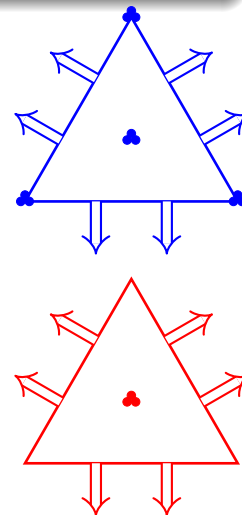
Conforming:  $AW^c(K) = \{\tau \in \mathcal{P}_3(K; \mathbb{S}) \mid \operatorname{div} \tau \in \mathcal{P}_1(K; \mathbb{R}^2)\}$

Nonconforming:  $AW^{nc}(K) = \{\tau \in \mathcal{P}_2(K; \mathbb{S}) \mid \mathbf{n} \cdot \tau \mathbf{n} \in \mathcal{P}_1(\mathbf{e}) \forall \text{ edges } \mathbf{e}\}$

paired with  $\mathbf{DG}(1)$  for the displacement.

- Exact enforcement of the symmetry of the Cauchy stress.
- Stable and convergent for stress-displacement linear elasticity.
- Discretise the 2D stress complex:

$$\begin{array}{ccccccccccc}
 0 & \longrightarrow & \mathcal{P}_1(\Omega) & \xrightarrow{\subset} & H^2(\Omega) & \xrightarrow{\text{airy}} & H(\operatorname{div}; \mathbb{S}) & \xrightarrow{\operatorname{div}} & \mathbf{L}^2(\Omega) & \longrightarrow & 0 \\
 & & \downarrow \operatorname{id} & & \downarrow I_h & & \downarrow \Pi_h & & \downarrow P_h & & \\
 0 & \longrightarrow & \mathcal{P}_1(\Omega) & \xrightarrow{\subset} & Q_h & \xrightarrow{\text{airy}} & \Sigma_h & \xrightarrow{\operatorname{div}} & V_h & \longrightarrow & 0.
 \end{array}$$



## The Arnold–Winther stress elements

Conforming:  $AW^c(K) = \{\tau \in \mathcal{P}_3(K; \mathbb{S}) \mid \operatorname{div} \tau \in \mathcal{P}_1(K; \mathbb{R}^2)\}$

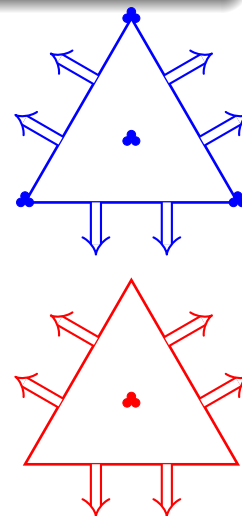
Nonconforming:  $AW^{nc}(K) = \{\tau \in \mathcal{P}_2(K; \mathbb{S}) \mid \mathbf{n} \cdot \tau \mathbf{n} \in \mathcal{P}_1(\mathbf{e}) \forall \text{ edges } \mathbf{e}\}$

paired with  $\mathbf{DG}(1)$  for the displacement.

- Exact enforcement of the symmetry of the Cauchy stress.
- Stable and convergent for stress-displacement linear elasticity.
- Discretise the 2D stress complex:

$$\begin{array}{ccccccccccc}
 0 & \longrightarrow & \mathcal{P}_1(\Omega) & \xrightarrow{\subset} & H^2(\Omega) & \xrightarrow{\text{airy}} & H(\operatorname{div}; \mathbb{S}) & \xrightarrow{\operatorname{div}} & \mathbf{L}^2(\Omega) & \longrightarrow & 0 \\
 & & \downarrow \operatorname{id} & & \downarrow I_h & & \downarrow \Pi_h & & \downarrow P_h & & \\
 0 & \longrightarrow & \mathcal{P}_1(\Omega) & \xrightarrow{\subset} & Q_h & \xrightarrow{\text{airy}} & \Sigma_h & \xrightarrow{\operatorname{div}} & V_h & \longrightarrow & 0.
 \end{array}$$

- Almost never systematically implemented.**



# Piola-inequivalent spaces

Denote:

- $\mathcal{F}^* : \hat{V} \rightarrow V$  a reference-to-physical Piola pullback
- $\hat{\Psi}, \Psi$  nodal bases for the *MTW* or *AW* spaces.

Then unfortunately,

$$\mathcal{F}^*(\hat{\Psi}) \neq \Psi, \quad \text{but} \quad \Psi = M\mathcal{F}^*(\hat{\Psi}) \text{ for some invertible } M.$$

# Piola-inequivalent spaces

Denote:

- $\mathcal{F}^* : \hat{V} \rightarrow V$  a reference-to-physical Piola pullback
- $\hat{\Psi}, \Psi$  nodal bases for the *MTW* or *AW* spaces.

Then unfortunately,

$$\mathcal{F}^*(\hat{\Psi}) \neq \Psi, \quad \text{but} \quad \Psi = M\mathcal{F}^*(\hat{\Psi}) \text{ for some invertible } M.$$

Similarly, define:

- pushforward  $\mathcal{F}_* : V^* \rightarrow \hat{V}^*$
- sets of DOFs  $\mathcal{L}, \hat{\mathcal{L}}$

$$\text{then} \quad \hat{\mathcal{L}} = P\mathcal{F}_*(\mathcal{L}) \text{ for some invertible } P.$$

## Piola-inequivalent spaces

Denote:

- $\mathcal{F}^* : \hat{V} \rightarrow V$  a reference-to-physical Piola pullback
- $\hat{\Psi}, \Psi$  nodal bases for the *MTW* or *AW* spaces.

Then unfortunately,

$$\mathcal{F}^*(\hat{\Psi}) \neq \Psi, \quad \text{but} \quad \Psi = M\mathcal{F}^*(\hat{\Psi}) \text{ for some invertible } M.$$

Similarly, define:

- pushforward  $\mathcal{F}_* : V^* \rightarrow \hat{V}^*$
- sets of DOFs  $\mathcal{L}, \hat{\mathcal{L}}$

$$\text{then} \quad \hat{\mathcal{L}} = P\mathcal{F}_*(\mathcal{L}) \text{ for some invertible } P.$$

**Theorem** [Kirby (2018)]

$$M = P^\top.$$

# Piola-inequivalent spaces

Denote:

- $\mathcal{F}^* : \hat{V} \rightarrow V$  a reference-to-physical Piola pullback
- $\hat{\Psi}, \Psi$  nodal bases for the  $MTW$  or  $AW$  spaces.

Then unfortunately,

$$\mathcal{F}^*(\hat{\Psi}) \neq \Psi, \quad \text{but} \quad \Psi = M\mathcal{F}^*(\hat{\Psi}) \text{ for some invertible } M.$$

Similarly, define:

- pushforward  $\mathcal{F}_* : V^* \rightarrow \hat{V}^*$
- sets of DOFs  $\mathcal{L}, \hat{\mathcal{L}}$

$$\text{then} \quad \hat{\mathcal{L}} = P\mathcal{F}_*(\mathcal{L}) \text{ for some invertible } P.$$

**Theorem** [Kirby (2018)]

$$M = P^\top.$$

**Proposition** [A., Farrell, Kirby (2020)]

Explicit construction of the (sparse) dual transformations  $P$  for the  $MTW$ ,  $AW^c$ , and  $AW^{nc}$  spaces.

## Implementation: the $MTW$ element

Implementation was carried out in **Firedrake** .



Implementation was carried out in **Firedrake** 🐉.

### A perturbed saddle point system

Seek  $(u, p) \in (H_0(\text{div}) \cap \epsilon \mathbf{H}_0^1(\Omega)) \times L^2(\Omega)$  such that

$$\begin{aligned}(I - \epsilon^2 \Delta) u - \nabla p &= f && \text{in } \Omega, \\ \text{div } u &= g && \text{in } \Omega, \\ u &= h && \text{on } \Gamma_D, \\ \epsilon^2 \nabla u \mathbf{n} - p \mathbf{n} &= 0 && \text{on } \Gamma_N.\end{aligned}$$

$\epsilon = 1$ : Stokes-like incompressible flow.

$\epsilon \rightarrow 0$ : Darcy flow ( $\sim$  mixed Poisson).

Implementation was carried out in **Firedrake** 🦖.

## A perturbed saddle point system

Seek  $(u, p) \in (H_0(\text{div}) \cap \epsilon \mathbf{H}_0^1(\Omega)) \times L^2(\Omega)$  such that

$$\begin{aligned}(I - \epsilon^2 \Delta) u - \nabla p &= f && \text{in } \Omega, \\ \text{div } u &= g && \text{in } \Omega, \\ u &= h && \text{on } \Gamma_D, \\ \epsilon^2 \nabla u \mathbf{n} - p \mathbf{n} &= 0 && \text{on } \Gamma_N.\end{aligned}$$

$\epsilon = 1$ : Stokes-like incompressible flow.

$\epsilon \rightarrow 0$ : Darcy flow ( $\sim$  mixed Poisson).

We validate robustness with respect to  $\epsilon$  using a smooth MMS on  $\Omega = (0, 1)^2$ .

# The $MTW$ element: $\epsilon$ -independent MMS convergence rates

$\epsilon \setminus N$	$2^0$	$2^1$	$2^2$	$2^3$	$2^4$	$2^5$	EOC
1	0.00456896	0.00128355	0.000341183	8.8577e-05	2.26311e-05	5.72514e-06	1.92807
$2^{-2}$	0.00447605	0.00126505	0.000335227	8.68331e-05	2.21707e-05	5.60825e-06	1.92809
$2^{-4}$	0.00421902	0.00120629	0.000319928	8.22134e-05	2.09019e-05	5.2806e-06	1.9284
$2^{-6}$	0.00405483	0.00113654	0.000305674	7.96031e-05	2.02428e-05	5.10031e-06	1.92697
$2^{-8}$	0.0040504	0.00112421	0.000296652	7.67045e-05	1.97471e-05	5.02906e-06	1.93071
$2^{-10}$	0.00405058	0.00112407	0.000296248	7.60249e-05	1.92762e-05	4.88509e-06	1.9391
0	0.00405059	0.00112407	0.000296246	7.60168e-05	1.92522e-05	4.84429e-06	1.94153

$L^2$  errors and convergence rates of  $MTW$  velocity for a range of  $\epsilon$ .

$\epsilon \setminus N$	$2^0$	$2^1$	$2^2$	$2^3$	$2^4$	$2^5$	EOC
1	0.430242	0.200198	0.102848	0.051765	0.0259237	0.0129668	1.01045
$2^{-2}$	0.420754	0.198049	0.102341	0.051599	0.0258535	0.0129335	1.00476
$2^{-4}$	0.420711	0.19804	0.102339	0.0515984	0.0258533	0.0129334	1.00473
$2^{-6}$	0.420711	0.19804	0.102339	0.0515983	0.0258533	0.0129334	1.00473
$2^{-8}$	0.420711	0.19804	0.102339	0.0515983	0.0258533	0.0129334	1.00473
$2^{-10}$	0.420711	0.19804	0.102339	0.0515983	0.0258533	0.0129334	1.00473
0	0.420711	0.19804	0.102339	0.0515983	0.0258533	0.0129334	1.00473

$L^2$  pressure errors and convergence rates.

## The Hellinger–Reissner principle

Seek a stress-displacement pair  $(\sigma, u) \in H(\operatorname{div}; \mathbb{S}) \times \mathbf{L}^2(\Omega)$  such that

$$\begin{aligned}\mathcal{A}\sigma &= \varepsilon(u) && \text{in } \Omega, \\ \operatorname{div} \sigma &= f && \text{in } \Omega, \\ u &= u_0 && \text{on } \Gamma_D, \\ \sigma \mathbf{n} &= g && \text{on } \Gamma_N,\end{aligned}$$

where  $\mathcal{A} = \mathcal{A}(\mu, \lambda)$  denotes the compliance tensor.

We validate the implementation again using a smooth MMS.

$N$	$u$ error	$u$ EOC	$\sigma$ error	$\sigma$ EOC	$\operatorname{div}_h \sigma$ error	$\operatorname{div}_h \sigma$ EOC
$2^1$	0.10522	–	0.490015	–	0.832183	–
$2^2$	0.0278746	1.91638	0.209729	1.2243	0.267865	1.63539
$2^3$	0.00707398	1.97836	0.0890996	1.23504	0.0714705	1.90609
$2^4$	0.00177271	1.99656	0.0412454	1.11118	0.0181626	1.97638
$2^5$	0.000442977	2.00066	0.0199779	1.04583	0.00455929	1.99409

$L^2$  errors and convergence rates for  $AW^{nc}$ , at Poisson ratio  $\nu = 0.3$ .

$N$	$u$ error	$u$ EOC	$\sigma$ error	$\sigma$ EOC	$\operatorname{div} \sigma$ error	$\operatorname{div} \sigma$ EOC
$2^0$	4.94678	–	0.00816494	–	0.037794	–
$2^1$	1.0283	2.26623	0.0010553	2.95179	0.01005	1.91095
$2^2$	0.0826871	3.63646	0.000129246	3.02946	0.0025538	1.97649
$2^3$	0.00561489	3.88033	1.59231e-05	3.02093	0.000641093	1.99404
$2^4$	0.000365775	3.94023	1.98093e-06	3.00687	0.00016044	1.9985
$2^5$	2.7322e-05	3.74282	2.47482e-07	3.00078	4.01203e-05	1.99963

$AW^c$ , near the incompressible limit  $\nu = 0.49999$ .

To enforce the traction condition

$$\sigma \mathbf{n} = g \quad \text{on } \Gamma_N$$

which is **particularly difficult with  $AW$  elements** [Carstensen et al. (2008)],

To enforce the traction condition

$$\sigma \mathbf{n} = g \quad \text{on } \Gamma_N$$

which is **particularly difficult with AW elements** [Carstensen et al. (2008)], and to aid multigrid preconditioning, seek stationary points of

$$\begin{aligned} \mathcal{H}_{h,\gamma,\omega}(\sigma_h, u_h) := & \int_{\Omega} \frac{1}{2} \mathcal{A} \sigma_h : \sigma_h + (\operatorname{div} \sigma_h - f) \cdot u_h \\ & - \int_{\Gamma_N} (\sigma_h \mathbf{n} - g) \cdot u_h \, ds + \frac{\gamma}{2h} \int_{\Gamma_N} \|\sigma_h \mathbf{n} - g\|^2 \, ds \end{aligned}$$

- Nitsche penalty for the traction condition,

To enforce the traction condition

$$\sigma \mathbf{n} = g \quad \text{on } \Gamma_N$$

which is **particularly difficult with AW elements** [Carstensen et al. (2008)], and to aid multigrid preconditioning, seek stationary points of

$$\mathcal{H}_{h,\gamma,\omega}(\sigma_h, u_h) := \int_{\Omega} \frac{1}{2} \mathcal{A} \sigma_h : \sigma_h + (\operatorname{div} \sigma_h - f) \cdot u_h \\ - \int_{\Gamma_N} (\sigma_h \mathbf{n} - g) \cdot u_h \, ds + \frac{\gamma}{2h} \int_{\Gamma_N} \|\sigma_h \mathbf{n} - g\|^2 \, ds + \frac{\omega}{2} \int_{\Omega} \|\operatorname{div} \sigma_h - f\|^2 \, dx.$$

- **Nitsche penalty** for the traction condition,
- **augmented Lagrangian penalty** to control the Schur complement.



To enforce the traction condition

$$\sigma \mathbf{n} = g \quad \text{on } \Gamma_N$$

which is **particularly difficult with AW elements** [Carstensen et al. (2008)], and to aid multigrid preconditioning, seek stationary points of

$$\mathcal{H}_{h,\gamma,\omega}(\sigma_h, u_h) := \int_{\Omega} \frac{1}{2} \mathcal{A} \sigma_h : \sigma_h + (\operatorname{div} \sigma_h - f) \cdot u_h \\ - \int_{\Gamma_N} (\sigma_h \mathbf{n} - g) \cdot u_h \, ds + \frac{\gamma}{2h} \int_{\Gamma_N} \|\sigma_h \mathbf{n} - g\|^2 \, ds + \frac{\omega}{2} \int_{\Omega} \|\operatorname{div} \sigma_h - f\|^2 \, dx.$$

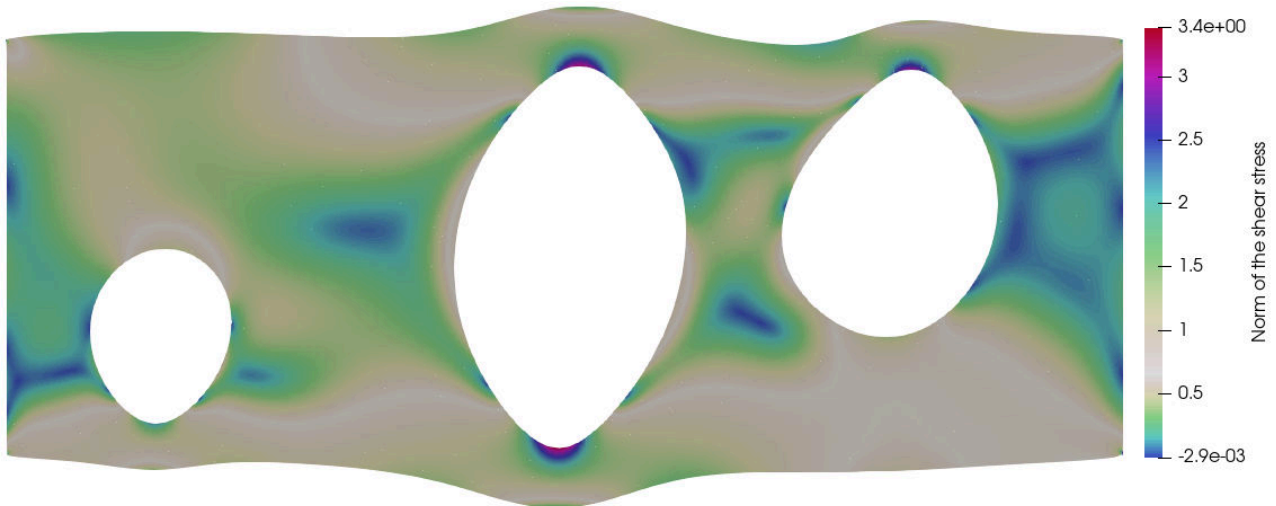
- **Nitsche penalty** for the traction condition,
- **augmented Lagrangian penalty** to control the Schur complement.

**Patch-based additive Schwarz smoother** [e.g. Schöberl (1999)]

We employ the *vertex-star iteration* to precondition the augmented stress block after block factorisation, applied by PCASM.

# Arnold–Winther elements for linear elasticity

Numerical results



A traction-free condition except at both ends, coloured by the shear stress, near the incompressible limit ( $\nu = 0.499999$ );  $1.14 \times 10^6$  DOFs using  $AW^{nc}$ . Considered in [\[Li \(2018\)\]](#).

- We have generalised contravariant Piola transformation theory to Piola-inequivalent elements.

- We have generalised contravariant Piola transformation theory to Piola-inequivalent elements.
- Robust implementation of:
  - ▶ Mardal–Tai–Winther elements for Stokes–Darcy flow.
  - ▶ Arnold–Winther elements for stress-displacement linear elasticity.

- We have generalised contravariant Piola transformation theory to Piola-inequivalent elements.
- Robust implementation of:
  - ▶ Mardal–Tai–Winther elements for Stokes–Darcy flow.
  - ▶ Arnold–Winther elements for stress-displacement linear elasticity.
- Conformity to complexes allows for the deployment of patch-based multigrid smoothers.

- We have generalised contravariant Piola transformation theory to Piola-inequivalent elements.
- Robust implementation of:
  - ▶ Mardal–Tai–Winther elements for Stokes–Darcy flow.
  - ▶ Arnold–Winther elements for stress-displacement linear elasticity.
- Conformity to complexes allows for the deployment of patch-based multigrid smoothers.
- Our approach is inexpensive, composing neatly with the existing software stack.

- We have generalised contravariant Piola transformation theory to Piola-inequivalent elements.
- Robust implementation of:
  - ▶ Mardal–Tai–Winther elements for Stokes–Darcy flow.
  - ▶ Arnold–Winther elements for stress-displacement linear elasticity.
- Conformity to complexes allows for the deployment of patch-based multigrid smoothers.
- Our approach is inexpensive, composing neatly with the existing software stack.

*Thanks.*

# UFL to GPU: Generating near roofline actions kernels

**Kaushik Kulkarni**, University of Illinois at Urbana-Champaign, United States

Andreas Kloeckner (<http://www.cs.illinois.edu/~andreask>), University of Illinois at Urbana-Champaign, United States

23 March 2021

Recent GPUs are capable of providing peak performance up to 6 TFLOps/s (DP), making them an attractive computational target for Finite Element Methods. However, effectively mapping an FEM solver to a GPU remains challenging due to the scattered memory access, large amounts of on-chip state space (eg registers) required for efficient execution, and the inherently large algorithmic variety encountered in local assembly kernels for variational forms.

In this talk, we focus on accelerating FEM action kernels for matrix-free operators on simplices. We describe a parametrized family of transformation strategies targeting these kernels, a heuristic cost model, and an auto-tuning strategy that enables us to achieve near-roofline performance for a wide variety of variational forms across domains such as fluid dynamics, solid mechanics, and wave motion. We propose a new computational offloading interface for Firedrake, and we have realized our UFL-to-GPU compilation pipeline within this interface. The pipeline can process general UFL with very few limitations and reliably produce routines for high-performance matrix-free 1-form assembly.

We close with remarks on further potential performance improvement opportunities.

---

You can cite this talk as:

Kaushik Kulkarni and Andreas Kloeckner. “UFL to GPU: Generating near roofline actions kernels”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 302. doi: 10.6084/m9.figshare.14495301.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/kulkarni.html>.



# A second order scheme to compute geometric interfaces with applications in microfluids

**Stephan Schmidt**, Humboldt University Berlin, Germany

Melanie Gräßer, Paderborn University, Germany

Hans-Joachim Schmid, Paderborn University, Germany

**23 March 2021**

Second order shape calculus is used to solve the Young–Laplace problem, which determines the shape of energy minimal interfaces such as droplets and capillary bridges. Knowledge of the shape and resulting capillary force of droplets in micro fluids has multiple application in granulate flows and lubrication.

To this end, 2nd order shape calculus is combined with a variety of contact and subset constraints to make the problem tractable with FEniCS. In particular, a level-set formulation to described the external geometry is coupled with a curvature-free variational formulation of the shape Hessian on shells, combining the multi and inter mesh capabilities of FEniCS with finite elements on shells.

Alternative, non-smooth interface energies and the resulting geometric flows will also be discussed.

---

You can cite this talk as:

Stephan Schmidt, Melanie Gräßer, and Hans-Joachim Schmid. “A second order scheme to compute geometric interfaces with applications in microfluids”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 303. doi: 10.6084/m9.figshare.14495304.

BibTeX for this citation can be found at <https://mcsroggs.github.io/fenics2021/talks/schmidt.html>.

# Stochastic topology optimisation for robust and manufacturable designs

Johannes Neumann (<https://www.rafinex.com>), Rafinex SARL, Germany

23 March 2021

Modern production methods, such as 3D printing, can manufacture almost constraint free form variations. Topology optimization enables engineers to explore the vastly increased design possibilities. Given the performance requirements, a part is optimized to be as cheap or lightweight as possible. Provided with the greatest allowed extent of the part, the algorithm fully controls the shape and placement of material and the incorporation of holes.

Care needs to be taken as there is a high risk of over-optimization towards the provided load cases. As a result the part might perform well in the simulated environment but fail in the actual application as unforeseen load conditions might occur or load cases might deviate due to some margin of error in manufacturing and application.

Stochastic topology optimization yields reliable optimal forms by extending physics models with risk assessment approaches from financial mathematics using formal uncertainty quantification methods. Loads are allowed to have an error in direction or magnitude, materials might have production faults. Incorporating these defects in the optimization yields unique designs that are not achievable by conventional topology optimization.

The generated designs react much more robustly to changing and unknown conditions in the physical world and have greatly increased reusability potential thanks to a greater performance envelope while maintaining or reducing weight compared to conventional design methods.

Manufacturability can be ensured with additional constraints such as two mold casting or printability for different printing techniques. State of the art adaptive numerical algorithms ensure high resolution, high quality ready to manufacture designs. These can be used to rapidly design and manufacture performance parts or as a blueprint for a more traditional design approach.

The degree of robustness can be controlled on a high level with model parameters when employing the Bayesian approach or on a low level frequentist approach by prescribing probabilities and a desired risk measure for more control and even more optimization potential.

The slides for this talk are available at <https://mscroggs.github.io/fenics2021/talks/neumann.html> under a CC BY-NC-ND 4.0 license.

---

You can cite this talk as:

Johannes Neumann. “Stochastic topology optimisation for robust and manufacturable designs”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 304. doi: 10.6084/m9.figshare.14495307.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/neumann.html>.

# Computing multiple solutions of topology optimisation problems with FEniCS

**Ioannis Papadopoulos** (<http://www.maths.ox.ac.uk/people/ioannis.papadopoulos>),  
University of Oxford, United Kingdom

Patrick Farrell, University of Oxford, United Kingdom

Thomas Surowiec, Philipps-Universität Marburg, Germany

23 March 2021

Topology optimisation finds the optimal material distribution of a fluid or solid in a domain, subject to PDE and volume constraints. The models often result in a PDE, volume and inequality constrained, nonconvex, infinite-dimensional optimisation problem. These problems can exhibit many local minima. In practice, heuristics are used to obtain the global minimum, but these can fail even in the simplest of cases. In this talk, we will introduce the deflated barrier method, an algorithm, implemented in both FEniCS and Firedrake, that solves such problems and can systematically discover many of these local minima. We will present examples which include finding 42 solutions of the topology optimisation of a fluid satisfying the Navier–Stokes equations and more recent work involving the three-dimensional topology optimisation of a fluid in Stokes flow. We also discuss block preconditioners for solving the linear systems arising in three-dimensional problems.

---

You can cite this talk as:

Ioannis Papadopoulos, Patrick Farrell, and Thomas Surowiec. “Computing multiple solutions of topology optimisation problems with FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 305. doi: 10.6084/m9.figshare.14495310.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/papadopoulos.html>.

# Shape optimization in coupled fluid-structure systems using Multiphenics

Harisankar Ramaswamy, University of Southern California, United States

Saikat Dey, U.S. Naval Research Laboratory, United States

Assad Oberai, University of Southern California, United States

23 March 2021

We present a gradient-based shape optimization framework for coupled structural acoustic systems using Multiphenics, a FEniCS-based open source software. In our approach, the primary variables are displacement in the solid domain and pressure in the fluid domain; they are obtained by solving the Navier–Cauchy and the Helmholtz equation respectively. Further, these fields are coupled through explicit interface conditions that ensure the continuity of displacements and tractions across fluid-structure boundary. In our method, the adjoint based formulation is employed to derive gradients that drive the movement of the interface boundary at every iteration. The curvature at the interface boundary, which is a crucial ingredient for obtaining the shape sensitivity, is computed by considering a smooth-continuous extension of the normal vector. Finally, the effectiveness of our approach in solving both interior and exterior optimization problems is demonstrated with simple examples.

---

You can cite this talk as:

Harisankar Ramaswamy, Saikat Dey, and Assad Oberai. “Shape optimization in coupled fluid-structure systems using Multiphenics”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 306. DOI: 10.6084/m9.figshare.14495313.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/ramaswamy.html>.

# ATOMiCS: topology optimization using FEniCS and OpenMDAO

Jiayao Yan, University of California San Diego, United States

Ru Xiang, University of California San Diego, United States

David Kamensky, University of California San Diego, United States

John Hwang, University of California San Diego, United States

23 March 2021

We present a toolbox called ATOMiCS, which uses FEniCS as the partial differential equation (PDE) solver for topology optimization and provides partial derivatives for density-based topology optimization in a modular large-scale optimization framework, OpenMDAO. OpenMDAO is a gradient-based multi-disciplinary design optimization framework developed by NASA. Many existing models from different disciplines have been implemented in OpenMDAO as modular components by various users. Integrating FEniCS with OpenMDAO brings a practical PDE solution approach and symbolic derivative computation while maintaining interoperability with existing OpenMDAO models. In addition, using FEniCS as the PDE solver for topology optimization inside a modular framework enables a general topology optimization toolbox with user-specified governing PDEs. We have applied ATOMiCS to topology optimization problems such as minimizing compliance in linear and nonlinear elastic structures, minimizing compliance and weight of thermoelastic material in battery packs, and shape-matching with liquid crystal elastomer structures.

---

You can cite this talk as:

Jiayao Yan, Ru Xiang, David Kamensky, and John Hwang. "ATOMiCS: topology optimization using FEniCS and OpenMDAO". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 307. DOI: 10.6084/m9.figshare.14495316.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/yan.html>.

# Simple and sharp: Error estimates of Bank–Weiser type in the FEniCS Project

Jack S. Hale, University of Luxembourg, Luxembourg

Raphael Bulle, University of Luxembourg, France

Alexei Lozinski (<http://lmb.univ-fcomte.fr/Lozinski-Alexei>), Université Bourgogne Franche-Comté, France

Stéphane P. A. Bordas, University of Luxembourg, Luxembourg

Franz Chouly, Université de Bourgogne-Franche-Comté, France

24 March 2021

We propose a simple, cheap and parallelisable implementation [3] in DOLFIN and DOLFINx of an implicit *a posteriori* error estimator introduced in [1].

The computation of the implicit estimator requires the solution of local Neumann problems in non-standard finite element spaces on each cell of the mesh. These special spaces are usually not available in modern automated finite element software, including the FEniCS Project.

Our method bypasses this issue by constructing a linear system on each cell corresponding to the problem in an available finite element space. We restrict this linear system to a non-standard space. On affine-equivalent finite elements, this restriction is constant and its application involves only small dense matrix-matrix multiplications.

We show several numerical examples of adaptive mesh refinement driven by this estimator applied to Poisson, Stokes and incompressible linear elasticity, as well as for goal-oriented problems [2].

## References

- [1] R. E. Bank and A. Weiser. “Some a posteriori error estimators for elliptic partial differential equations”. 1985.
- [2] Roland Becker, Elodie Estecahandy, and David Trujillo. “Weighted marking for goal-oriented adaptive finite element methods”. 2011.
- [3] Raphaël Bulle, Jack S. Hale, Alexei Lozinski, Stéphane P. A. Bordas, and Franz Chouly. “Hierarchical a posteriori error estimation of Bank–Weiser type in the FEniCS Project”. In: *submitted* (2020).

---

You can cite this talk as:

Jack S. Hale, Raphael Bulle, Alexei Lozinski, Stéphane P. A. Bordas, and Franz Chouly. “Simple and sharp: Error estimates of Bank–Weiser type in the FEniCS Project”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 308–335. doi: 10.6084/m9.figshare.14495328.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/hale.html>.

# Simple and sharp: Error estimates of Bank–Weiser type in the FEniCS Project

**Jack S. Hale**

Raphaël Bulle, Alexei Lozinski

Stéphane P. A. Bordas, Franz Chouly

University of Luxembourg  
Université de Bourgogne Franche-Comté

March 24, 2021



**DRIVEN**

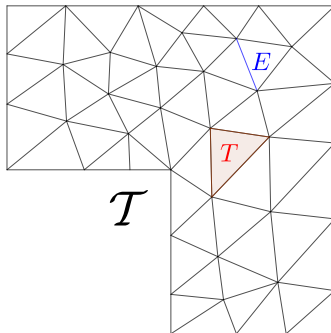


FENICS  
PROJECT

- The problem.
- Estimates of Bank-Weiser type.
- Implementation.
- Results.



# Problem setting



Find  $u_k$  in  $V^k$  such that

$$\int_{\Omega} \nabla u_k \cdot \nabla v_k = \int_{\Omega} f v_k \quad \forall v_k \in V^k. \quad (1)$$

# Error

We quantify the discretization error  $e := u_k - u$  using the energy norm  $\eta_{\text{err}} := \|\nabla e\|_{\Omega} = \|\nabla u_k - \nabla u\|_{\Omega}$ .

# Error

We quantify the discretization error  $e := u_k - u$  using the energy norm  $\eta_{\text{err}} := \|\nabla e\|_{\Omega} = \|\nabla u_k - \nabla u\|_{\Omega}$ .

**Goal:** estimate  $\eta$  i.e. find a computable quantity  $\eta_{\text{bw}}$  such that

$$\eta_{\text{bw}} \approx \eta_{\text{err}}.$$

# Contributions

- A high-level way of expressing Bank–Weiser type error estimators in DOLFIN and DOLFINx [Bank and Weiser, 1985].
- A simple dual-weighted error estimation and marking strategy originally proposed in [Becker et al., 2011].
- A proof of the reliability of the Bank–Weiser estimator in dimension three [Bulle et al., 2020].
- arXiv: <https://arxiv.org/abs/2102.04360>
- Code: <https://github.com/rbulle/fenics-error-estimation>

# The Bank–Weiser Estimator

The restriction  $e_T$  of  $e$  to any cell  $T$  of the mesh satisfies the equation

$$\int_T \nabla e_T \cdot \nabla v_T := \int_T (f - \Delta u_k) v_T + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T \quad \forall v \in H_0^1(T).$$

# The Bank–Weiser Estimator

The restriction  $e_T$  of  $e$  to any cell  $T$  of the mesh satisfies the equation

$$\int_T \nabla e_T \cdot \nabla v_T := \int_T (f - \Delta u_k) v_T + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T \quad \forall v \in H_0^1(T).$$

On a cell  $T$ , the Bank–Weiser problem is given by:  
find  $e_T^{\text{bw}}$  in  $V_T^{\text{bw}}$  such that

$$\int_T \nabla e_T^{\text{bw}} \cdot \nabla v_T^{\text{bw}} = \int_T (f - \Delta u_k) v_T^{\text{bw}} + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^{\text{bw}} \quad \forall v_T^{\text{bw}} \in V_T^{\text{bw}}.$$

# The Bank–Weiser Estimator

The restriction  $e_T$  of  $e$  to any cell  $T$  of the mesh satisfies the equation

$$\int_T \nabla e_T \cdot \nabla v_T := \int_T (f - \Delta u_k) v_T + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T \quad \forall v \in H_0^1(T).$$

On a cell  $T$ , the Bank–Weiser problem is given by:  
find  $e_T^{\text{bw}}$  in  $V_T^{\text{bw}}$  such that

$$\int_T \nabla e_T^{\text{bw}} \cdot \nabla v_T^{\text{bw}} = \int_T (f - \Delta u_k) v_T^{\text{bw}} + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^{\text{bw}} \quad \forall v_T^{\text{bw}} \in V_T^{\text{bw}}.$$

The Bank–Weiser estimator is defined as

$$\eta_{\text{bw}}^2 := \sum_{T \in \mathcal{T}} \eta_{\text{bw},T}^2, \quad \eta_{\text{bw},T} := \|\nabla e_T^{\text{bw}}\|_T.$$

The space  $V_T^{\text{bw}}$



# The space $V_T^{\text{bw}}$

- Different definitions of  $V_T^{\text{bw}}$  lead to different variants of the estimator.

# The space $V_T^{\text{bw}}$

- Different definitions of  $V_T^{\text{bw}}$  lead to different variants of the estimator.
- General principle: let  $V_T^- \subsetneq V_T^+$  be two finite element spaces and

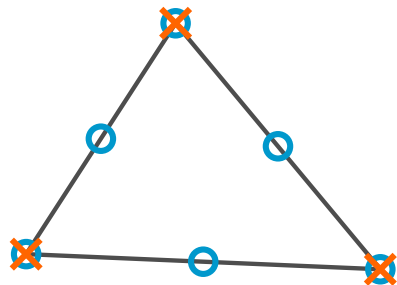
$$\mathcal{L}_T : V_T^+ \longrightarrow V_T^-,$$

be the local Lagrange interpolation operator,

$$V_T^{\text{bw}} := \ker(\mathcal{L}_T) = \{v_T^+ \in V_T^+, \mathcal{L}_T(v_T^+) = 0\}.$$

# Example

For  $V_T^+ = V_T^2$  and  $V_T^- = V_T^1$



# Implementation

We need to compute the matrix  $A_T^{\text{bw}}$  and vector  $b_T^{\text{bw}}$  from

$$\int_T \nabla e_T^{\text{bw}} \cdot \nabla v_T^{\text{bw}} = \int_T (f - \Delta u_k) v_T^{\text{bw}} + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^{\text{bw}} \quad \forall v_T^{\text{bw}} \in V_T^{\text{bw}}.$$

# Implementation

We need to compute the matrix  $A_T^{\text{bw}}$  and vector  $b_T^{\text{bw}}$  from

$$\int_T \nabla e_T^{\text{bw}} \cdot \nabla v_T^{\text{bw}} = \int_T (f - \Delta u_k) v_T^{\text{bw}} + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^{\text{bw}} \quad \forall v_T^{\text{bw}} \in V_T^{\text{bw}}.$$

**Problem:** the space  $V_T^{\text{bw}}$  is not provided by DOLFIN.

# Implementation

We need to compute the matrix  $A_T^{\text{bw}}$  and vector  $b_T^{\text{bw}}$  from

$$\int_T \nabla e_T^{\text{bw}} \cdot \nabla v_T^{\text{bw}} = \int_T (f - \Delta u_k) v_T^{\text{bw}} + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^{\text{bw}} \quad \forall v_T^{\text{bw}} \in V_T^{\text{bw}}.$$

**Problem:** the space  $V_T^{\text{bw}}$  is not provided by DOLFIN.

**Idea:** we rely on the matrix  $A_T^+$  and vector  $b_T^+$  from

$$\int_T \nabla e_T^+ \cdot \nabla v_T^+ = \int_T (f - \Delta u_k) v_T^+ + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^+ \quad \forall v_T^+ \in V_T^+,$$

since  $V_T^+$  is provided by DOLFIN.

# Implementation

We need to compute the matrix  $A_T^{\text{bw}}$  and vector  $b_T^{\text{bw}}$  from

$$\int_T \nabla e_T^{\text{bw}} \cdot \nabla v_T^{\text{bw}} = \int_T (f - \Delta u_k) v_T^{\text{bw}} + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^{\text{bw}} \quad \forall v_T^{\text{bw}} \in V_T^{\text{bw}}.$$

**Problem:** the space  $V_T^{\text{bw}}$  is not provided by DOLFIN.

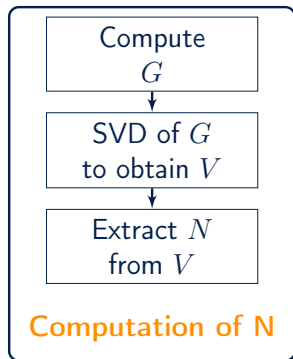
**Idea:** we rely on the matrix  $A_T^+$  and vector  $b_T^+$  from

$$\int_T \nabla e_T^+ \cdot \nabla v_T^+ = \int_T (f - \Delta u_k) v_T^+ + \sum_{E \in \partial T} \frac{1}{2} \int_E \llbracket \partial_n u_k \rrbracket_E v_T^+ \quad \forall v_T^+ \in V_T^+,$$

since  $V_T^+$  is provided by DOLFIN. and we look for a matrix  $N$  such that:

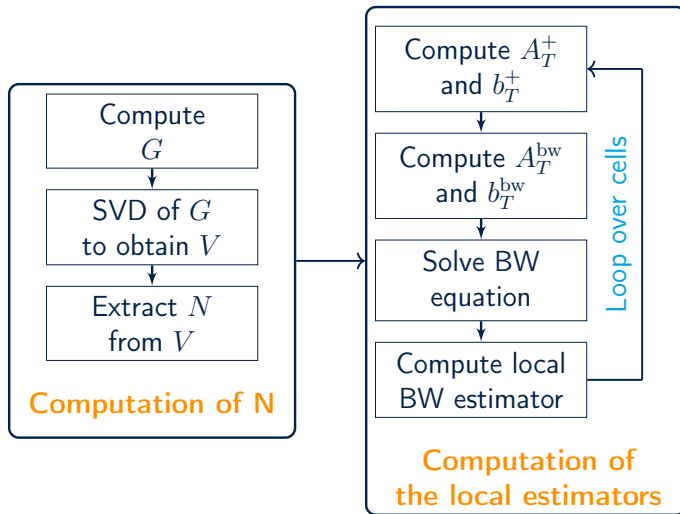
$$A_T^{\text{bw}} = N^t A_T^+ N, \quad \text{and} \quad b_T^{\text{bw}} = N^t b_T^+.$$

# Algorithm

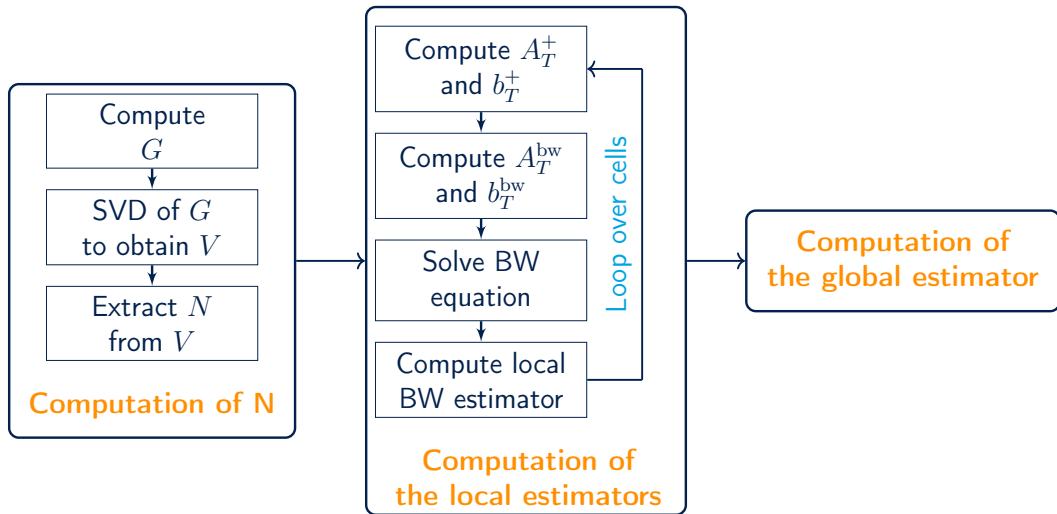




# Algorithm



# Algorithm



# In code

```
def estimate(u_h):
    mesh = u_h.function_space().mesh()
    element_f = FiniteElement("DG", triangle, 2)
    element_g = FiniteElement("DG", triangle, 1)

    N = fenics_error_estimation.create_interpolation(element_f, element_g)

    V_f = FunctionSpace(mesh, element_f)
    e = TrialFunction(V_f)
    v = TestFunction(V_f)
    f = Constant(0.0)
    bcs = DirichletBC(V_f, Constant(0.0), "on_boundary", "geometric")

    n = FacetNormal(mesh)
    a_e = inner(grad(e), grad(v))*dx
    L_e = inner(f + div(grad(u_h)), v)*dx + \
        inner(jump(grad(u_h), -n), avg(v))*dS

    e_h = fenics_error_estimation.estimate(a_e, L_e, N, bcs)
    error = norm(e_h, "H10")

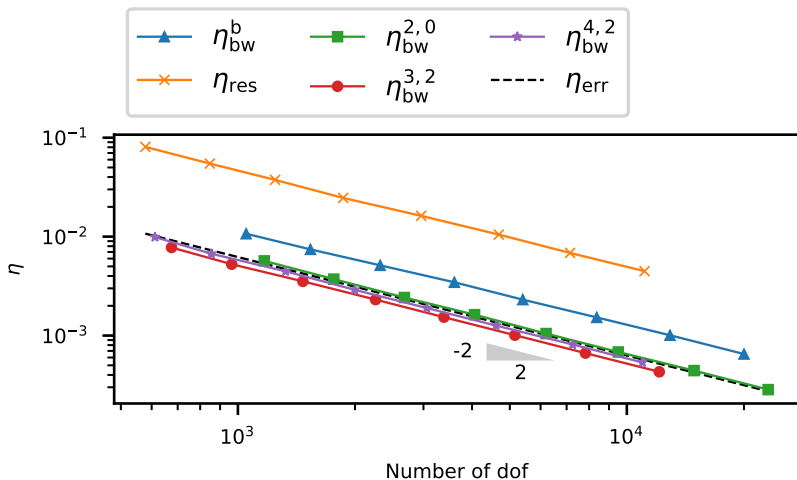
    V_e = FunctionSpace(mesh, "DG", 0)
    v = TestFunction(V_e)
    eta_h = Function(V_e, name="eta_h")
    eta = assemble(inner(inner(grad(e_h), grad(e_h)), v)*dx)
    eta_h.vector()[:] = eta

    return eta_h
```

# Results I

Adaptive finite elements for a Poisson problem:

$-\Delta u = 0$  in  $\Omega$ ,  $u = u_D$  on  $\Gamma$ . Quadratic finite elements.



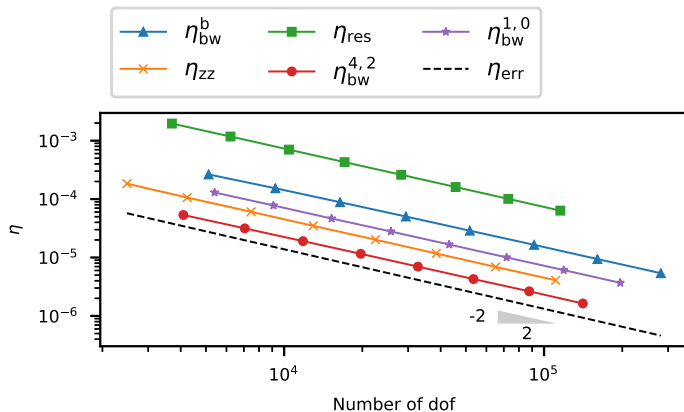
Notation	$V_T^+$	$V_T^-$
$\eta_{bw}^{k_+, k_-}$	$V_T^{k_+}$	$V_T^{k_-}$
$\eta_{bw}^b$	$V_T^2 + \text{bubble}$	$V_T^1$

# Results II

Goal oriented adaptive finite elements for a Poisson problem:

$-\Delta u = 0$  in  $\Omega$ ,  $u = u_D$  on  $\Gamma$ .  $\eta_{\text{err}} := J(u - u_1) = \int_{\Omega} (u - u_h)c$ , where  $c$  is a smooth weight function.

The estimators are computed using the WGO method from [Becker et al., 2011].



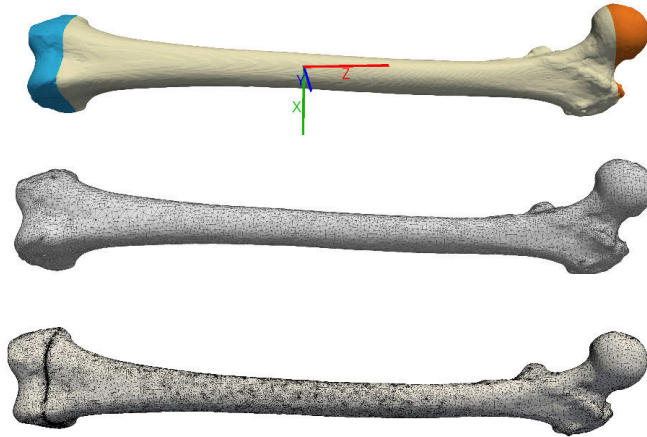
Notation	$V_T^+$	$V_T^-$
$\eta_{\text{bw}}^{k_+, k_-}$	$V_T^{k_+}$	$V_T^{k_-}$
$\eta_{\text{bw}}^b$	$V_T^2 + \text{bubble}$	$V_T^1$

## Results II

GO AFEM for a linear elasticity problem:

we used a technique from [Khan et al., 2019] to compute the estimators. The goal functional is

defined by  $J(\mathbf{u}_2, p_1) := \int_{\Gamma} \mathbf{u}_2 \cdot \mathbf{n} c$ .

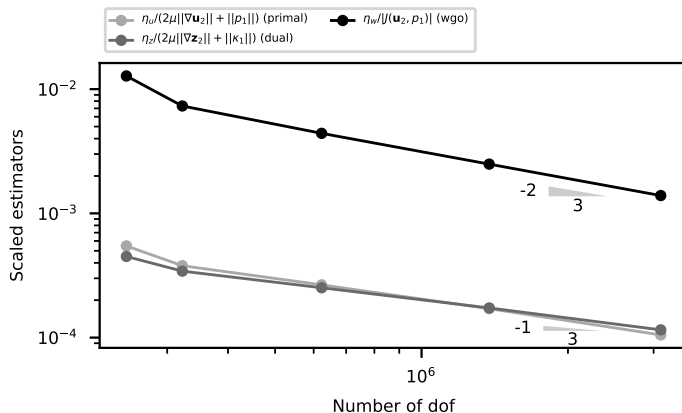


# Results II

GO AFEM for a linear elasticity problem:

we used a technique from [Khan et al., 2019] to compute the estimators. The goal functional is

defined by  $J(\mathbf{u}_2, p_1) := \int_{\Gamma} \mathbf{u}_2 \cdot \mathbf{n} c$ .



# Thank you for your attention!







---

I would like to acknowledge the support of the ASSIST research project of the University of Luxembourg. This presentation has been prepared in the framework of the DRIVEN project funded by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 811099.



# References I

-  Bank, R. E. and Weiser, A. (1985).  
Some A Posteriori Error Estimators for Elliptic Partial Differential Equations.  
*Math. Comput.*, 44(170):283–301.
-  Becker, R., Estecahandy, E., and Trujillo, D. (2011).  
Weighted Marking for Goal-oriented Adaptive Finite Element Methods.  
*SIAM J. Numer. Anal.*, 49(6):2451–2469.
-  Bulle, R., Chouly, F., Hale, J. S., and Lozinski, A. (2020).  
Removing the saturation assumption in Bank–Weiser error estimator analysis in dimension three.  
*Appl. Math. Lett.*, 107(1):106429.
-  Khan, A., Powell, C. E., and Silvester, D. J. (2019).  
Robust a posteriori error estimators for mixed approximation of nearly incompressible elasticity.  
*Int. J. Numer. Methods Eng.*, 119(1):18–37.

# Local *a posteriori* error estimates for the spectral fractional Laplacian

**Raphaël Bulle** (<https://www.researchgate.net/profile/Raphael-Bulle>), University of Luxembourg, Luxembourg

Stéphane P. A. Bordas, University of Luxembourg, Luxembourg

Franz Chouly, Université de Bourgogne-Franche-Comté, France

Jack S. Hale, University of Luxembourg, Luxembourg

Alexei Lozinski (<http://lmb.univ-fcomte.fr/Lozinski-Alexei>), Université Bourgogne Franche-Comté, France

24 March 2021

The fractional Laplacian has a global character and its solutions have strong boundary layers. Its efficient solution is still an open challenge for the community.

In this talk, we will show a novel *a posteriori* error estimation method for the spectral fractional Laplacian.

Our method begins with the work of [2] where a fractional operator is represented by an integral over non-fractional (ie local) parametric operators. The integral and the local operators can then be discretised using a quadrature rule and a standard finite element method, respectively.

We show that the integral representation of [2] can equally be applied to the construction of an error estimator. A key building block of the method is an efficient hierarchical estimator introduced in [1].

The estimator has numerous benefits: it is numerically sharp, it allows the measurement of the error in various norms, and it is defined for one, two and three-dimensional problems. It is also fully local and cheap to compute in parallel.

The estimator leads to optimal convergence rates when used to steer adaptive refinement algorithms.

The implementation is based on FEniCS Error Estimation (FEniCS-EE), a finite element error estimation package for DOLFIN and DOLFINx [3].

## References

- [1] R. E. Bank and A. Weiser. “Some *a posteriori* error estimators for elliptic partial differential equations”. 1985.
- [2] A. Bonito and J. E. Pasciak. “Numerical Approximation of Fractional Powers of Elliptic Operators”. 2015.
- [3] Raphaël Bulle, Jack S. Hale, Alexei Lozinski, Stéphane P. A. Bordas, and Franz Chouly. “Hierarchical *a posteriori* error estimation of Bank–Weiser type in the FEniCS Project”. In: *submitted* (2020).

---

You can cite this talk as:

Raphaël Bulle, Stéphane P. A. Bordas, Franz Chouly, Jack S. Hale, and Alexei Lozinski. “Local *a posteriori* error estimates for the spectral fractional Laplacian”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 336–368. DOI: 10.6084/m9.figshare.14495334.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/bulle.html>.

# Local a posteriori error estimates for the spectral fractional Laplacian

**Raphaël Bulle**

Stéphane P.A. Bordas, Jack S. Hale,

Franz Chouly, Alexei Lozinski

University of Luxembourg

Université de Bourgogne Franche-Comté

March 24, 2021



**DRIVEN**



FENICS  
PROJECT

# The spectral fractional Laplacian

- The spectral fractional Laplacian
- Contribution
- Discretization
- A posteriori error estimation
- Numerical results

# The spectral fractional Laplacian

Fractional operators are used in a wide range of different fields such as statistics, hydrogeology, finance, physics...

# The spectral fractional Laplacian

Fractional operators are used in a wide range of different fields such as statistics, hydrogeology, finance, physics...

- Main advantage: they are nonlocal.

# The spectral fractional Laplacian

Fractional operators are used in a wide range of different fields such as statistics, hydrogeology, finance, physics...

- Main advantage: they are nonlocal.
- Main drawback: they are nonlocal.

# The spectral fractional Laplacian

Let  $\Omega \subset \mathbb{R}^d$ ,  $\alpha \in (0, 2)$  and  $f \in L^2(\Omega)$ .

$$(-\Delta)^{\alpha/2} u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \quad (1)$$



# The spectral fractional Laplacian

Let  $\Omega \subset \mathbb{R}^d$ ,  $\alpha \in (0, 2)$  and  $f \in L^2(\Omega)$ .

$$(-\Delta)^{\alpha/2} u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \quad (1)$$

Let  $\{\psi_i, \lambda_i\}_{i=1}^{+\infty} \subset L^2(\Omega) \times \mathbb{R}^+$  be such that

$$-\Delta \psi_i = \lambda_i \psi_i \quad \text{in } \Omega, \quad \psi_i = 0 \quad \text{on } \partial\Omega, \quad \forall i = 1, \dots, +\infty. \quad (2)$$

# The spectral fractional Laplacian

Let  $\Omega \subset \mathbb{R}^d$ ,  $\alpha \in (0, 2)$  and  $f \in L^2(\Omega)$ .

$$(-\Delta)^{\alpha/2} u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \quad (1)$$

Let  $\{\psi_i, \lambda_i\}_{i=1}^{+\infty} \subset L^2(\Omega) \times \mathbb{R}^+$  be such that

$$-\Delta \psi_i = \lambda_i \psi_i \quad \text{in } \Omega, \quad \psi_i = 0 \quad \text{on } \partial\Omega, \quad \forall i = 1, \dots, +\infty. \quad (2)$$

The solution  $u$  of (1) is defined by

$$u := \sum_{i=1}^{+\infty} \lambda_i^{-\alpha/2} (f, \psi_i)_{L^2}. \quad (3)$$

# Contribution

- The spectral fractional Laplacian
- **Contribution**
- Discretization
- A posteriori error estimation
- Numerical results

# Contribution

We present the first a posteriori error estimator for a numerical method presented in [Bonito and Pasciak, 2015] for solving the spectral fractional Laplacian.

# Discretization

- The spectral fractional Laplacian
- Contribution
- Discretization
- A posteriori error estimation
- Numerical results

# Discretization

How to solve (1) numerically ?

# Discretization

How to solve (1) numerically ?

Using an **integral representation** of the solution

$$u = C_\alpha \int_{-\infty}^{+\infty} e^{\alpha y} u_y \, dy, \quad (4)$$

where  $u_y$  is solution to

$$e^{2y} \int_{\Omega} \nabla u_y \cdot \nabla v + \int_{\Omega} u_y v = \int_{\Omega} f v, \quad \forall v \in H_0^1(\Omega). \quad (5)$$

# Discretization

- Quadrature discretization: given a quadrature rule  $\{\omega_l, y_l\}_{l=-N}^N$ ,

$$u = C_\alpha \int_{-\infty}^{+\infty} e^{\alpha y} u_y \, dy \approx C_\alpha \sum_{l=-N}^N \omega_l e^{\alpha y_l} u_{y_l} =: u^N. \quad (6)$$



# Discretization

- **Quadrature discretization:** given a quadrature rule  $\{\omega_l, y_l\}_{l=-N}^N$ ,

$$u = C_\alpha \int_{-\infty}^{+\infty} e^{\alpha y} u_y \, dy \approx C_\alpha \sum_{l=-N}^N \omega_l e^{\alpha y_l} u_{y_l} =: u^N. \quad (6)$$

- **Finite element discretization:** given a mesh  $\mathcal{T}_h$  on  $\Omega$  and  $V_h$  a FE space,

$$u \approx C_\alpha \sum_{l=-N}^N \omega_l e^{\alpha y_l} u_{h,y_l} =: u_h^N, \quad (7)$$

where  $u_{h,y_l}$  solves

$$e^{2y_l} \int_{\Omega} \nabla u_{h,y_l} \cdot \nabla v_h + \int_{\Omega} u_{h,y_l} v_h = \int_{\Omega} f v_h \quad \forall v_h \in V_h. \quad (8)$$

# A posteriori error estimation

- The spectral fractional Laplacian
- Contribution
- Discretization
- A posteriori error estimation
- Numerical results

# A posteriori error estimation

We neglect the quadrature discretization error and we focus on the FE discretization error

$$\eta \approx \|u - u_h^N\|_{L^2}. \quad (9)$$

# A posteriori error estimation

We consider the **Bank–Weiser a posteriori error estimator** [Bank and Weiser, 1985] on the parametric problem associated to  $u_{y_l}$ .

# A posteriori error estimation

We consider the **Bank–Weiser a posteriori error estimator** [Bank and Weiser, 1985] on the parametric problem associated to  $u_{y_l}$ . For each cell  $T$  of  $\mathcal{T}_h$  we solve

$$e^{2y} \int_T \nabla w_{T,y_l} \cdot \nabla v_T + \int_T w_{T,y_l} v_T = R_T(v_T) \quad \forall v_T \in V^{\text{bw}}(T). \quad (10)$$

# A posteriori error estimation

We consider the **Bank–Weiser a posteriori error estimator** [Bank and Weiser, 1985] on the parametric problem associated to  $u_{y_l}$ . For each cell  $T$  of  $\mathcal{T}_h$  we solve

$$e^{2y} \int_T \nabla w_{T,y_l} \cdot \nabla v_T + \int_T w_{T,y_l} v_T = R_T(v_T) \quad \forall v_T \in V^{\text{bw}}(T). \quad (10)$$

The local fractional Bank–Weiser solution is given by

$$w_T := C_\alpha \sum_{l=-N}^N \omega_l e^{\alpha y_l} w_{T,y_l}. \quad (11)$$

# A posteriori error estimation

We consider the **Bank–Weiser a posteriori error estimator** [Bank and Weiser, 1985] on the parametric problem associated to  $u_{y_l}$ . For each cell  $T$  of  $\mathcal{T}_h$  we solve

$$e^{2y} \int_T \nabla w_{T,y_l} \cdot \nabla v_T + \int_T w_{T,y_l} v_T = R_T(v_T) \quad \forall v_T \in V^{\text{bw}}(T). \quad (10)$$

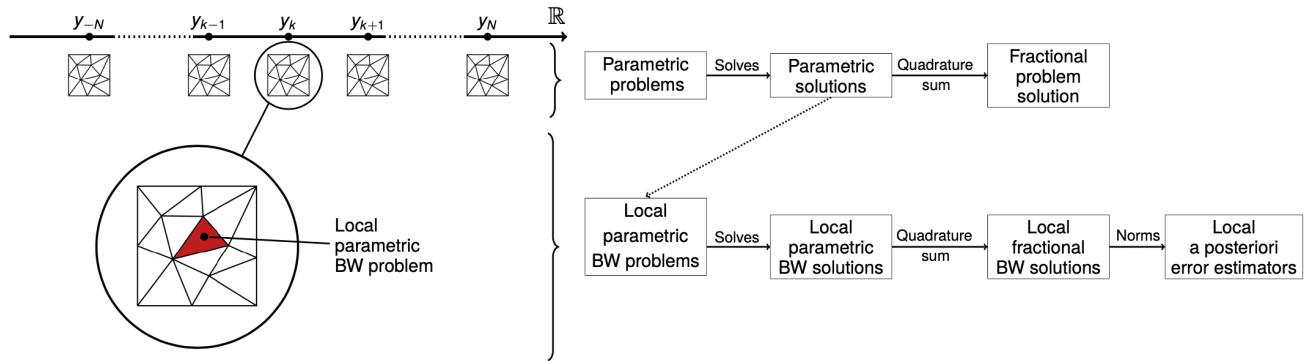
The local fractional Bank–Weiser solution is given by

$$w_T := C_\alpha \sum_{l=-N}^N \omega_l e^{\alpha y_l} w_{T,y_l}. \quad (11)$$

The local and global Bank–Weiser estimators are given by

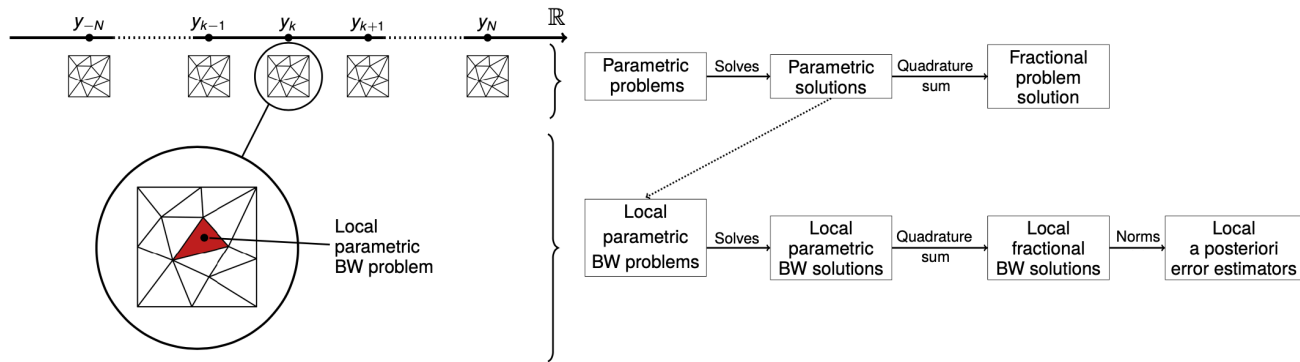
$$\eta_{\text{bw},T} := \|w_T\|_{L^2(T)}, \quad \eta_{\text{bw}}^2 := \sum_{T \in \mathcal{T}_h} \|w_T\|_{L^2(T)}^2. \quad (12)$$

# A posteriori error estimation





# A posteriori error estimation

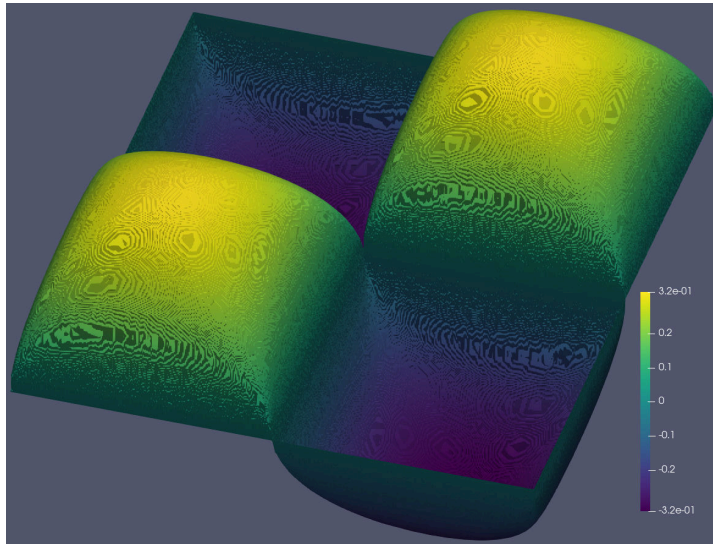


Fully local and fully parallelizable.

# Numerical results

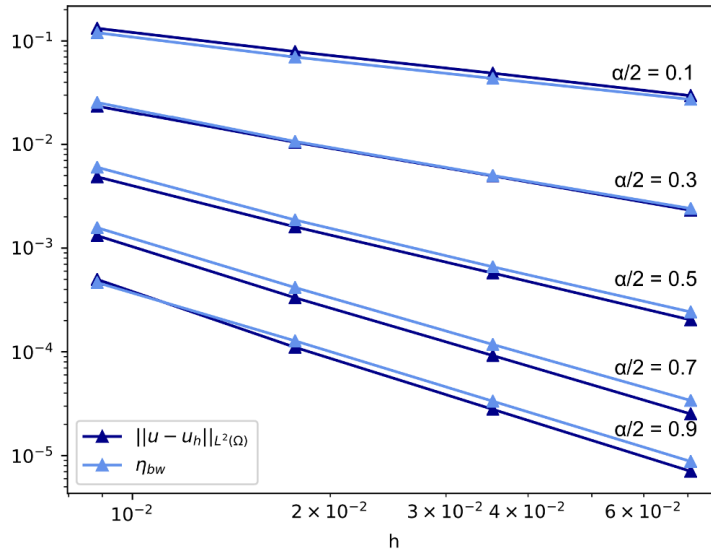
- The spectral fractional Laplacian
- Contribution
- Discretization
- A posteriori error estimation
- Numerical results

# Numerical results



# Numerical results

Uniform mesh refinement.



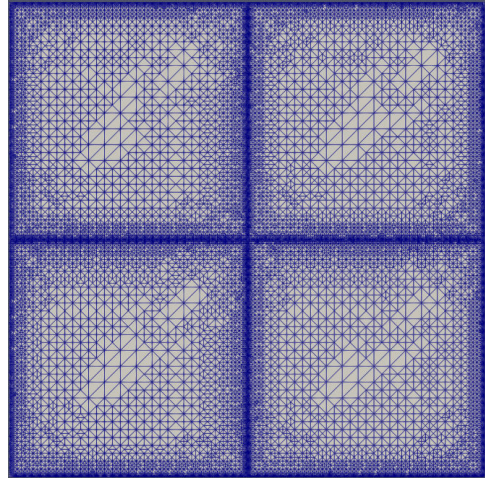
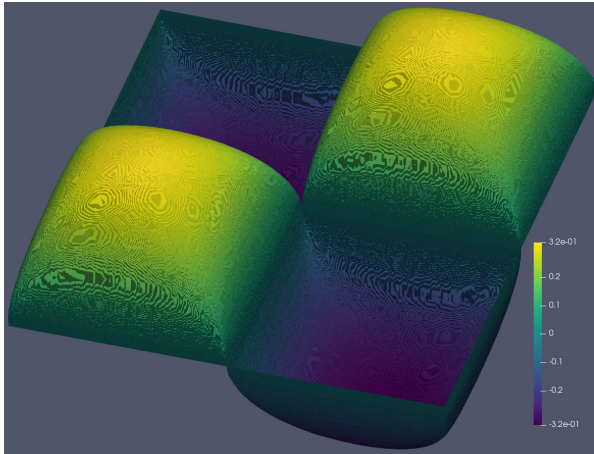
# Numerical results

Uniform mesh refinement[Bonito and Pasciak, 2015].

Frac. pow.	0.1	0.3	0.5	0.7	0.9
Th. slope	0.7	1.1	1.5	1.9	2.0
Err. slope	0.71	1.11	1.52	1.9	2.04
Est. slope	0.71	1.13	1.54	1.84	1.91

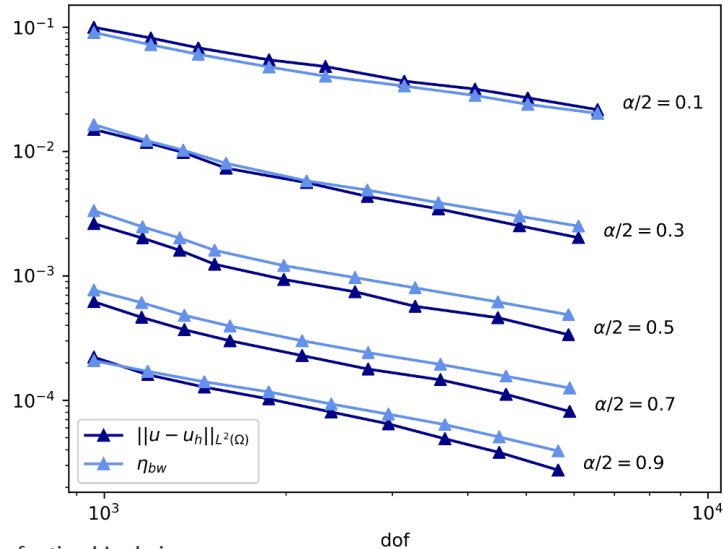
# Numerical results

Adaptive mesh refinement.



# Numerical results

Adaptive mesh refinement.



# Numerical results

Adaptive mesh refinement.

Frac. pow.	0.1	0.3	0.5	0.7	0.9
Th. slope (unif.)	0.35	0.55	0.75	0.95	1.0
Err. slope (adapt.)	0.71	1.13	1.54	1.84	1.91
Est. slope (adapt.)	0.72	1.11	1.52	1.9	2.04



# References I



Bank, R. E. and Weiser, A. (1985).  
Some A Posteriori Error Estimators for Elliptic Partial Differential Equations.  
*Math. Comput.*, 44(170):283–301.



Bonito, A. and Pasciak, J. E. (2015).  
Numerical approximation of fractional powers of elliptic operators.  
*Math. Comput.*, (295).

# Thank you for your attention!



---

I would like to acknowledge the support of the ASSIST research project of the University of Luxembourg. This presentation has been prepared in the framework of the DRIVEN project funded by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 811099.

# Making linearized methods in fluid mechanics available to a broader audience with FEniCS

**Thomas Ludwig Kaiser**, Laboratory for Flow Instabilities and Dynamics, TU Berlin, Germany

Chuhan Wang, LadHyx, École polytechnique, Paris, France

Kilian Oberleithner, Laboratory for Flow Instabilities and Dynamics, TU Berlin, Germany

Lutz Lesshafft, LadHyx, École polytechnique, Paris, France

24 March 2021

In fluid mechanics research, linearizing the governing equations around a base state has yielded significant insight in flow unsteadiness, as for example turbulence. The principles of these linearized methods go back to the 19th century to illustrious names, such as Helmholtz, Rayleigh, Kelvin and Reynolds. Despite over a century of successful application of the linearized Navier–Stokes equations in research to understand countless different forms of flow unsteadiness, its full potential for application in academia and especially industry up to today remains not exhausted by far: Although research groups around the globe successfully develop and apply in-house codes to address various unsteady flow configurations using the linearized equations, no common coding-platform exists that enables these groups to share their expertise and benefit from each other’s developments, both in modelling and coding. As a consequence, the hurdle of applying such tools to a new configuration is high, which makes the method—despite its potential—unattractive to researchers which are new to this field and fluid mechanics engineers. With our solver FELiCS (Finite-Element Linearized Combustion Solver, not to be confused with FEniCS), our goal is to fill this gap, and make the rich toolbox of linearized methods available to a broader audience. Our code uses the FEniCS package in order to discretize the governing equations in space, which together with the python environment, and efficient toolboxes, such as SLEPc and ARPACK is the ideal framework for our purposes. Currently, the code is applied to various configurations, such as investigating turbulence in jet flows, hydrodynamic instabilities, such as eg the cylinder vortex street, and laminar and turbulent combustion dynamics. Our talk will give an overview of linearized methods in fluid mechanics as well as current and potential application fields.

---

You can cite this talk as:

Thomas Ludwig Kaiser, Chuhan Wang, Kilian Oberleithner, and Lutz Lesshafft. “Making linearized methods in fluid mechanics available to a broader audience with FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 369. doi: 10.6084/m9.figshare.14495340.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/kaiser.html>.

# Predicting sound absorption in additively manufactured porous materials using multiscale simulations in FEniCS

**Kamil C. Opiela**, Institute of Fundamental Technological Research, Polish Academy of Sciences, Poland

Tomasz G. Zieliński (<http://bluebox.ippt.pan.pl/~tzielins/index.php>), Institute of Fundamental Technological Research, Polish Academy of Sciences, Poland

24 March 2021

Porous materials are widely used in soundproofing and noise control applications. Their microgeometry, which determines the sound absorption properties, can be developed to fit specific needs and produced employing modern additive manufacturing technologies. However, the design process usually requires running multiscale and multiphysics simulations, which is greatly facilitated by the FEniCS computing platform. The airborne propagation and attenuation of acoustic waves in a 3D printed rigid-frame porous material is considered with the aid of the 2019.1.0 FEniCS release. Two cases are studied: a basic passive configuration, and an adaptable configuration with steel balls introduced to the main pores that effectively modify visco-inertial and thermal dissipation within the system. The stationary Stokes, Laplace, and Poisson analyses are performed on a periodic fluid (air) domain representative for the microgeometry of the medium imposing periodic boundary conditions within a parallelised code. The respective solution fields are averaged over the domain using the built-in DOLFIN algorithms and upscaled to serve as an input to macroscopic calculations. The coupled Helmholtz problem of harmonic linear acoustics is solved to model the distribution of acoustic pressure in both a homogenised fluid equivalent to the porous material, and a layer of air adjacent to it. Suitable continuity boundary condition at the interface is implemented weakly into the finite element formulation of the problem. Finally, the numerical predictions of normal incidence sound absorption coefficient are compared with impedance tube measurements made on a 3D sample. Along with FEniCS, only open-source software is involved to prepare volumetric meshes and visualise the results.

---

You can cite this talk as:

Kamil C. Opiela and Tomasz G. Zieliński. “Predicting sound absorption in additively manufactured porous materials using multiscale simulations in FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 370. doi: 10.6084/m9.figshare.14495349.


BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/opiela.html>.

# Modelling of solidification problems with FEniCS

Florian Arbes, IFE, Norway

Kent-Andre Mardal, UiO, Norway

Øyvind Jensen, IFE, Norway

Jørgen S. Dokken (<https://jsdokken.com/>,  jorgensd), University of Cambridge, United Kingdom

24 March 2021

Modelling the solidification of alloys remains challenging, as there is a multi-physics problem to be solved involving fluid flow, heat transfer as well as phase changes. Shape casting processes are constantly sought to be improved, which can potentially be achieved through accurate and fast modelling of the alloy solidification process. We present in a short overview our approach of an application relying on FEniCS that will be used to model such solidification problems. The model, which is currently under development, follows a volume-averaged two-phase approach, as proposed by Beckermann *et al* [1].

Simplified models have been tested and used to generate snapshots for a non-intrusive reduced order model. As this application is being built in Python, we hope its flexible core will open up new opportunities for fast and accurate reduced order models.

## References

- [1] J. Ni and C. Beckermann. “A volume-averaged two-phase model for transport phenomena during solidification”. In: *Metallurgical Transactions B* 22.349 (1991). DOI: 10.1007/BF02651234.

---

You can cite this talk as:

Florian Arbes, Kent-Andre Mardal, Øyvind Jensen, and Jørgen S. Dokken. “Modelling of solidification problems with FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 371. DOI: 10.6084/m9.figshare.14495358.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/arbes.html>.

# Digital Math: Solving the reproducibility crisis in the digital era

Johan Jansson (<http://digimat.tech>), KTH Royal Institute of Technology, Sweden

Måns Andersson, KTH Royal Institute of Technology, Sweden

Linde Van Beers, KTH Royal Institute of Technology, Sweden

Ridgway Scott, University of Chicago, United States

Claes Johnson, KTH Royal Institute of Technology, Sweden

24 March 2021

We present the Digital Math framework as the foundation for modern science-based on constructive digital mathematical computation. The computed result (coefficient vector, FEM function, plot, etc) is a mathematical theorem, and the mathematical Open Source code, here in the FEniCS framework, and computation is the mathematical proof. Based on the Digital Math framework and the FEniCS realization, we present solutions to some of the grand challenges in science, education and industry.

In this seminar, we specifically focus on solving the reproducibility crisis in research, recently highlighted in our panel debate with the European Commission, Swedish Parliament, Lorena Barba and top Swedish journalist:

<http://digimat.tech/paneldebate-kth/>

We see education as reproducing scientific results of key importance for society. As part of the seminar, you are invited to participate in the online DigiMat course on reproducible research, where anyone in an accessible Ubiquitous High-Performance Computing (UHPC) Digital Math environment can create your own reproducible “Digital Math” publication.

Reports from the education system show that almost all students lose motivation for math and programming somewhere along the way. At the same time computational math is the fundament of modern society, creating an enormous societal problem.

We show results that our DigiMat—Digital Math online educationprogram [1]—from pre-school through university and professional is a solution to this grand challenge in education. DigiMat creates motivation and learning of the key abstract concepts by playing with, editing and building your own Digital Math interactive simulations and virtual worlds. DigiMat Pro has reached 30000+ participants with great feedback, and DigiMat Basic is now reaching schools in Sweden with good results.

We show reproducible predictive Real Flight Simulation as a solution to the grand challenge of aerodynamics connected to results in the High Lift Prediction Workshop. For general continuum modeling, such as biomechanics, we present our Real Unified Continuum framework [2] with results in food tech, heart modeling, etc.

## References

- [1] “Digital Math online educationprogram”. <http://digimat.tech/digimat>.
- [2] Johan Jansson, Massimiliano Leoni, Måns Andersson, Patricia Lopez Sanchez, Mats Stading, Ridgway Scott, and Claes Johnson. “Predictive Real Unified Continuum modeling of multi-physics in the Digital Math framework”. In: *Digital Math: Human workshop* (2020).

---

You can cite this talk as:

Johan Jansson, Måns Andersson, Linde Van Beers, Ridgway Scott, and Claes Johnson. “Digital Math: Solving the reproducibility crisis in the digital era”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 372. DOI: 10.6084/m9.figshare.14495364.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/jansson.html>.

# Coupling of non-matching isogeometric shells, with applications in aerospace structures

Han Zhao, University of California San Diego, United States

Xiangbei Liu, University of California San Diego, United States

John T. Hwang (<https://lsdo.eng.ucsd.edu/>), University of California San Diego, United States

David Kamensky (<https://david-kamensky.eng.ucsd.edu/>), University of California San Diego, United States

24 March 2021

This talk presents a framework for penalty coupling of non-matching Kirchhoff–Love shells using a FEniCS-based implementation of isogeometric analysis (IGA) called tIGAr. Topologically-1D, geometrically-3D mortar meshes are generated to integrate penalty terms in the variational problem, which penalize deviations from displacement and rotational continuity at the geometric intersections between separately-parameterized spline surfaces modeling different structural components. The framework allows one to directly perform shell structure analysis on computer-aided design (CAD) models consisting of multiple surface patches with non-matching parameterizations at their intersections. We verify the method and implementation using benchmark examples like the Scordelis–Lo roof problem with multiple NURBS patches and a T-beam subjected to a point load at one corner. Quantities of interest are computed accurately over a wide range of values of a dimensionless, problem-independent penalty coefficient. Coupling of non-matching shell structures with IGA is attractive for the design and analysis of aircraft, which are usually modeled geometrically by many intersecting parametric surfaces. We use the new framework to perform structural analysis of an aircraft model given in STEP format, which includes 42 B-spline surfaces and 84 non-matching intersections.

---

You can cite this talk as:

Han Zhao, Xiangbei Liu, John T. Hwang, and David Kamensky. “Coupling of non-matching isogeometric shells, with applications in aerospace structures”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 373. doi: 10.6084/m9.figshare.14495373.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/zhao.html>.

# Generating layer-adapted meshes using mesh PDEs

Róisín Hill, NUI Galway, Ireland

Niall Madden (<http://www.maths.nuigalway.ie/~niall>), NUI Galway, Ireland

24 March 2021

We consider the numerical solution, by finite element methods, of singularly-perturbed differential equations (SPDEs) whose solutions exhibit boundary layers. We will discuss our numerical method and the implementation in FEniCS [1], including some technical problems we overcame.

Our interest lies in developing parameter-robust methods, where the quality of the solution is independent of the value of the perturbation parameter. One way of achieving this is to use layer resolving methods based on meshes that concentrate their mesh points in regions of large variations in the solution.

We investigate the use of Mesh PDEs (MPDEs), as first presented in [3], to generate layer resolving meshes that yield parameter robust solutions to SPDEs. Specifically, we present MPDEs whose solutions, in the 1D case, yield the celebrated graded “Bakhvalov” meshes [2].

The true value of the proposed approach comes to the fore when we investigate 2D problems. Whereas the classical Bakhvalov mesh is restricted to generating tensor product grids, the use of MPDEs allows us to generate non-tensor product grids that are still highly anisotropic and layer-adapted grids, and yield robust solutions. We demonstrate this by solving problems on irregular domains, and with space-varying diffusion.

As the MPDEs are non-linear problems, we use a fixed-point iterative method to solve them numerically. We present an approach involving alternating between  $h$ - and  $r$ -refinement which is highly efficient, especially for larger meshes and small values of the perturbation parameter.

The manuscript on which this talk is based, and the code that generated the results, are available at <https://osf.io/dpexh/>

## References

- [1] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. “The FEniCS project version 1.5”. In: *Archive of Numerical Software* 3.100 (2015). DOI: 10.11588/ans.2015.100.20553.
- [2] N. S. Bakhvalov. “The optimization of methods of solving boundary value problems with a boundary layer”. In: *USSR Computational Mathematics and Mathematical Physics* 9.4 (1969), 139–166. DOI: 10.1016/0041-5553(69)90038-X.
- [3] Weizhang Huang, Yuhe Ren, and Robert D. Russell. “Moving mesh partial differential equations (MM-PDES) based on the equidistribution principle”. In: *SIAM Journal on Numerical Analysis* 31.3 (1994), 709–730. DOI: 10.1137/0731038.

---

You can cite this talk as:

Róisín Hill and Niall Madden. “Generating layer-adapted meshes using mesh PDEs”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 374–400. DOI: 10.6084/m9.figshare.14495376.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/hill.html>.



# Generating layer-adapted meshes using mesh PDEs

Róisín Hill and Niall Madden

FEniCS 2021

24 March, 2021



NUI Galway  
OÉ Gaillimh



IRISH RESEARCH COUNCIL  
An Chomhairle um Thaighde in Éirinn

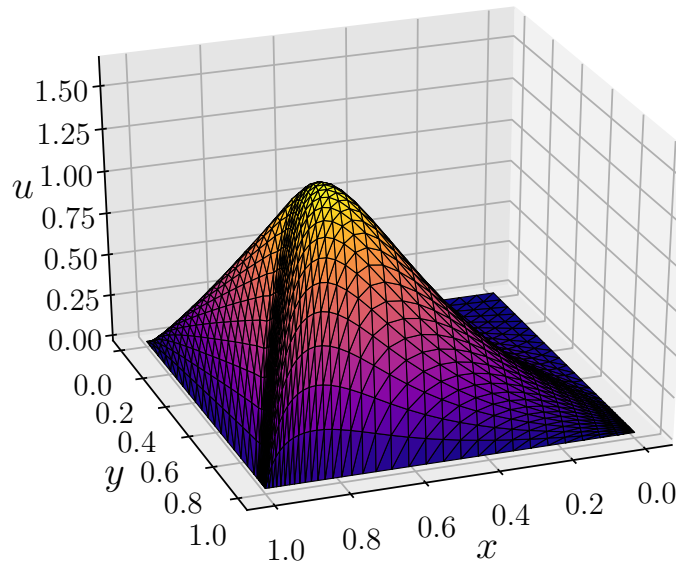


FEniCS  
PROJECT

# Motivation



We want to solve **singularly perturbed differential equations** (SPDEs) whose solutions have boundary layers, so require special **layer-adapted meshes**.



Example with boundary layers near  $x = 1$  and  $y = 1$ .

# Motivation (cont.)



Our goal is to generate layer-adapted meshes for solving SPDEs, with mesh-adaption driven by **Mesh Partial Differential Equations** (MPDEs) [Huang and Russell, 2011].

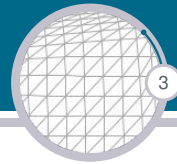
Typically **layer adapted meshes** are formulated using *a priori* information about the SPDE's solution; the most successful (arguably) of these is due to Bakhvalov [Bakhvalov, 1969]. For problems in 2D, they are restricted to tensor product grids.

In this talk we will present:

1. a more general formulation based on MPDEs, and
2. an algorithm for efficiently solving these nonlinear problems.

Results and source code are available as: *Generating layer-adapted meshes using mesh partial differential equations*; [osf.io/dpexh/](https://osf.io/dpexh/) (to appear in Numer. Math. Theor. Meth. Appl.) [Hill and Madden, 2021]

# Reaction-diffusion equation and method



Our SPDEs are reaction-diffusion problems of the form

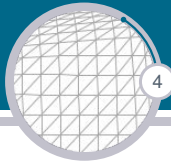
$$-\epsilon^2 \Delta u + ru = f \text{ in } \Omega \subseteq \mathbb{R}^d, \text{ with } u|_{\partial\Omega} = 0, \quad (1)$$

where  $d = 1, 2$ ;  $\epsilon > 0$ ,  $r$  and  $f$  are given (smooth) functions and  $r \geq \beta^2$  on  $\overline{\Omega}$ , with  $\beta > 0$ .

When  $\epsilon$  is small the solutions exhibit boundary layers.

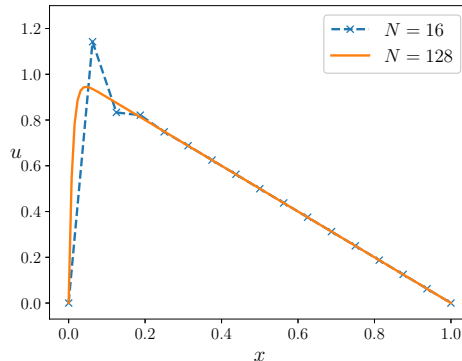
We use a standard Galerkin **finite element method** (FEM), with linear elements, to compute our numerical solutions to (1), and implement the method in **FEniCS** [Alnæs et al., 2015].

# Solutions to (2) on **uniform** meshes

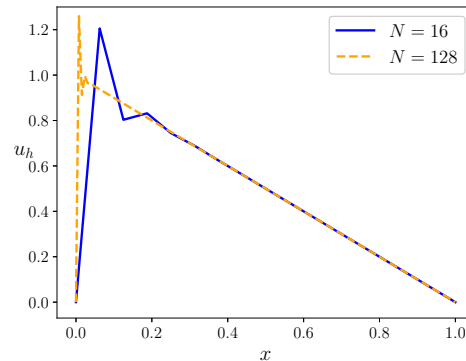


## Model 1D scalar reaction-diffusion equation

$$-\varepsilon^2 u'' + u = 1 - x, \quad \text{on } (0, 1), \quad u(0) = u(1) = 0. \quad (2)$$



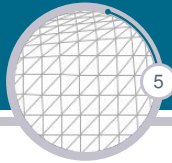
(a)  $\varepsilon = 10^{-2}$



(b)  $\varepsilon = 10^{-4}$

The **oscillations** occur in the solutions when  $\varepsilon < C/N$  due to the **lack of stability** in the discrete problem.

# Layer-resolving meshes



We are interested in using **layer-resolving** meshes, which **concentrate** mesh points in regions where large variations occur in the solution. In 1D we consider these meshes in terms of this *“mesh generating function”*.

## Definition

A mesh generating function (on  $[0,1]$ ) is a strictly monotonic bijective function  $\varphi : [0, 1] \rightarrow [0, 1]$  that maps a uniform mesh  $\xi_i = i/N$ , to a possibly non-uniform mesh  $x_i = \varphi(i/N)$ , for  $i = 0, 1, \dots, N$ .

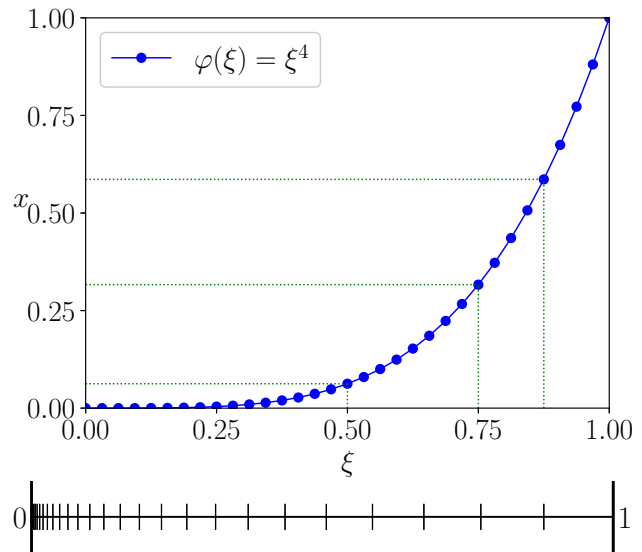
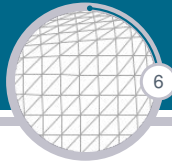


Figure: Mesh generated when  $\varphi(\xi) = \xi^4$

# Bakhvalov mesh (via equidistribution)



One method of generating a Bakhvalov mesh for (2) is by **equidistributing** the function [Linß, 2010],

$$\rho(x) = \max \left\{ 1, K \frac{\beta}{\varepsilon} \exp \left( -\frac{\beta x}{\sigma \varepsilon} \right) \right\}, \quad (3)$$

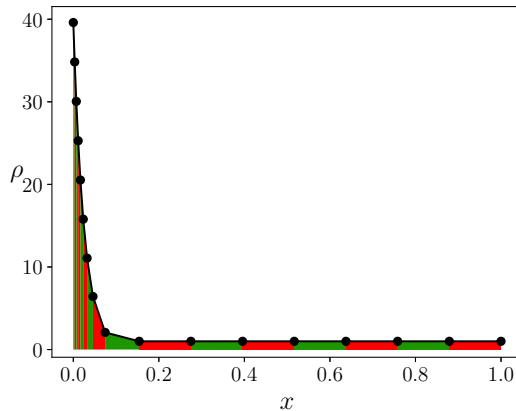
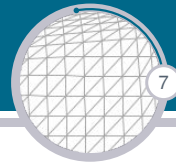
where  $\sigma$ ,  $\beta$  and  $K$  are constants.

That is, one computes the mesh  $\omega^N := \{0 = x_0, x_1, \dots, x_N = 1\}$  such that

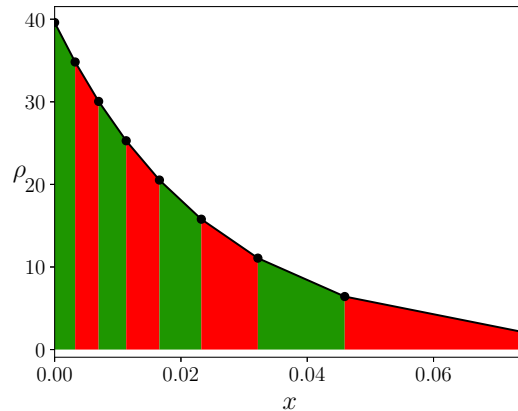
$$\int_{x_i}^{x_{i+1}} \rho(x) dx = \frac{1}{N} \int_0^1 \rho(x) dx \text{ for } i = 0, 1, \dots, N-1.$$

This is a nonlinear problem, as is the classic method of generating a Bakhvalov mesh.

# Equidistribution of $\rho$



(a)  $x \in [0, 1]$

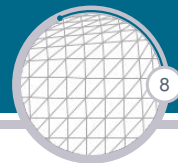


(b) zoomed

**Figure:** integral of  $\rho$  on the resulting mesh.

When  $\varepsilon \ll 1$ ,  $\rho$  decays rapidly near  $x = 0$ . Therefore, we use the Gauss-Lobatto quadrature rule when solving the 1D MPDE. We thank Jørgen Dokken for pointing us towards a nice implementation.





A (moving) mesh PDE is presented in [Huang et al., 1994] as a way to generate specially adapted meshes:

1. A PDE whose solution is a mesh generating function is posed.
2. The PDE features a coefficient,  $\rho$ , that controls the concentration of points in the resulting mesh.
3. Classically,  $\rho$  depends on (local error estimates for) the solution.
4. However, we will use the basic idea to generate *a priori* Bakhvalov-style meshes.

# From equidistribution to an MPDE



We derived the MPDE,

$$-(\rho(x)x(\xi)')' = 0 \text{ for all } \xi \in (0, 1), \quad x(0) = 0 \text{ and } x(1) = 1, \quad (4)$$

using the equidistribution principle.

The BCs are necessary to result in a mesh generating function.

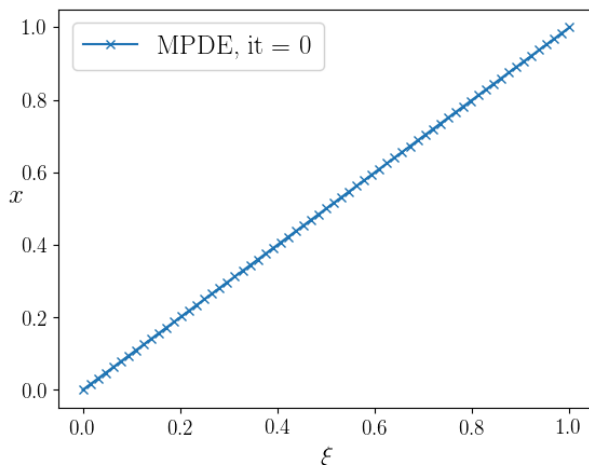
The solution to (4) is a Bakhvalov mesh when  $\rho$  is as defined in (3).

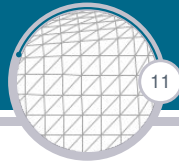
Since this is a **nonlinear** problem we use a fixed point iteration method to find a solution.

# Solving the MPDE by an FEM



Example of how the mesh evolved when  $N = 64$  and  $\varepsilon = 10^{-3}$





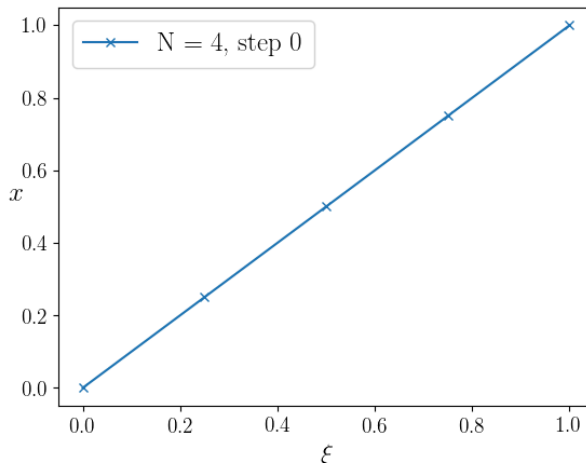
The number of iterations required was  $\mathcal{O}(N)$ . So, to improve the efficiency of the method, we

1. **start** with a mesh with 4 intervals,
2. **apply** 3 iterations of the MPDE,
3. **interpolate** the solution onto a mesh with **twice** the number of intervals,
4. **repeat** steps 2–3 until we reach the required number of mesh intervals, and
5. then iterate until a **stopping** criterion is achieved.

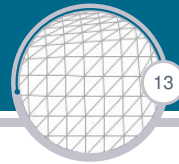
This reduces the number of iterations required to  $\mathcal{O}(\log_2 N)$ , e.g., when  $\varepsilon = 10^{-8}$  and  $N = 1024$ , iterations:  $517 \rightarrow 27$  (3 on final mesh).



Example of how the mesh evolved when  $N = 64$  and  $\varepsilon = 10^{-3}$

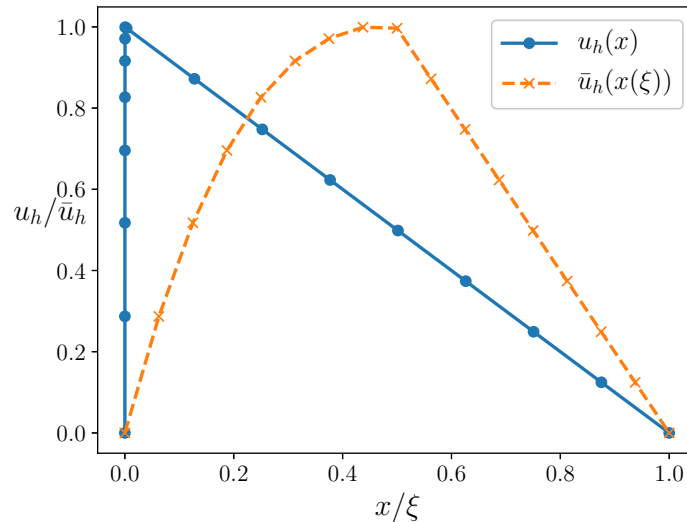


# Scalar 1D reaction-diffusion problem



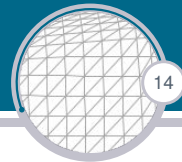
$$-\varepsilon^2 u'' + u = 1 - x, \quad \text{on } (0, 1), \quad u(0) = u(1) = 0.$$

Solution with  $\varepsilon = 10^{-4}$  and  $N = 16$  on the MPDE (physical) mesh  $\omega^N$  and uniform (computational) mesh  $\omega^{[c]}$



# Scalar 1D reaction-diffusion problem

Error measurement



It can be shown (e.g., [Roos et al., 2008]) that the error in the linear-FEM solution generated on a Bakhvalov mesh satisfies

$$\|u - u^h\|_E \leq C(\varepsilon^{1/2} N^{-1} + N^{-2}),$$

where  $u$  is the true solution and  $u^h$  is the FEM solution,  $C$  is a constant independent of  $\varepsilon$  and  $N$ , and  $\|\cdot\|_E$  is the usual energy norm induced by the FEM bilinear form.

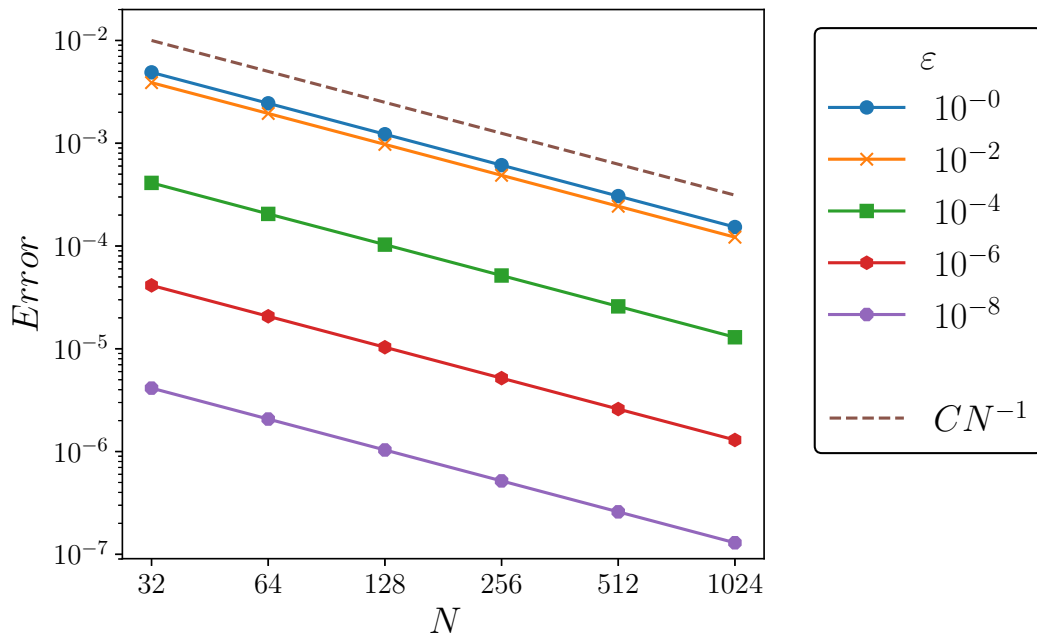
In practice we compare the linear-FEM with the quadratic-FEM solution to compute our errors,  $e^h$ .

# Scalar 1D reaction-diffusion problem

Errors

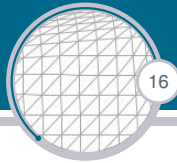


Plot of  $\|e^h\|_E$  for the scalar 1D problem when solved on a MPDE mesh,  $\omega^N$





# 2D example



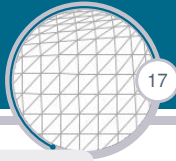
Arguably, in 1D the method outlined has no advantage over other methods of equidistribution.

This would also be true in 2D if we restrict our interest to problems for which tensor product grids are appropriate.

Therefore, we present a scenario where a non-tensor product grid is more appropriate.

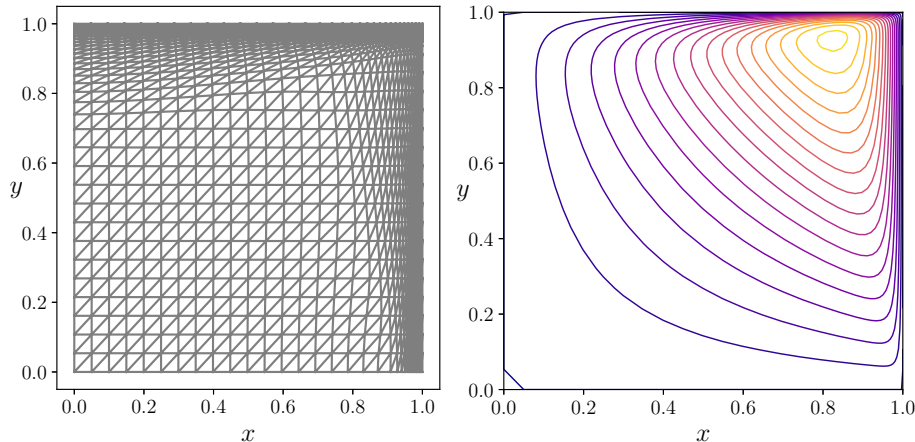
The solution to the MPDE only determines the location of the mesh points, resulting in a unique mesh in 1D. However in 2D we also need **connectivity**, here our adapted-mesh inherits the connectivity from a uniform mesh.

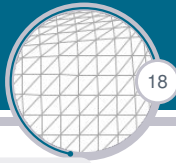
# 2D problem with spatially varying diffusion



$$\begin{aligned} & -\nabla \cdot \left( \begin{pmatrix} \varepsilon(1+2y)^2 & 0 \\ 0 & \varepsilon(3-2x)^2 \end{pmatrix} \nabla u(x, y) \right) + u(x, y) \\ & = (e^x - 1)(e^y - 1), \quad \text{for } (x, y) \in \Omega = (0, 1)^2, \quad \text{with } u|_{\Omega} = 0. \end{aligned}$$

MPDE Mesh and contour plot of solution when  $\varepsilon = 10^{-2}$  and  $N = 32$





The MPDE, for  $\vec{x}(\xi_1, \xi_2) = (x, y)^T$ , is

$$-\nabla \cdot (M(\vec{x}(\xi_1, \xi_2)) \nabla \vec{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]},$$

with boundary conditions,

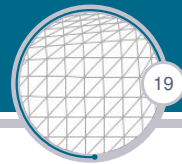
$$\begin{aligned} x(0, \xi_2) = 0, \quad x(1, \xi_2) = 1, \quad \frac{\partial x}{\partial n}(\xi_1, 0) = 0, \quad \frac{\partial x}{\partial n}(\xi_1, 1) = 0, \\ y(\xi_1, 0) = 0, \quad y(\xi_1, 1) = 1, \quad \frac{\partial y}{\partial n}(0, \xi_2) = 0, \quad \frac{\partial y}{\partial n}(1, \xi_2) = 0, \end{aligned}$$

and

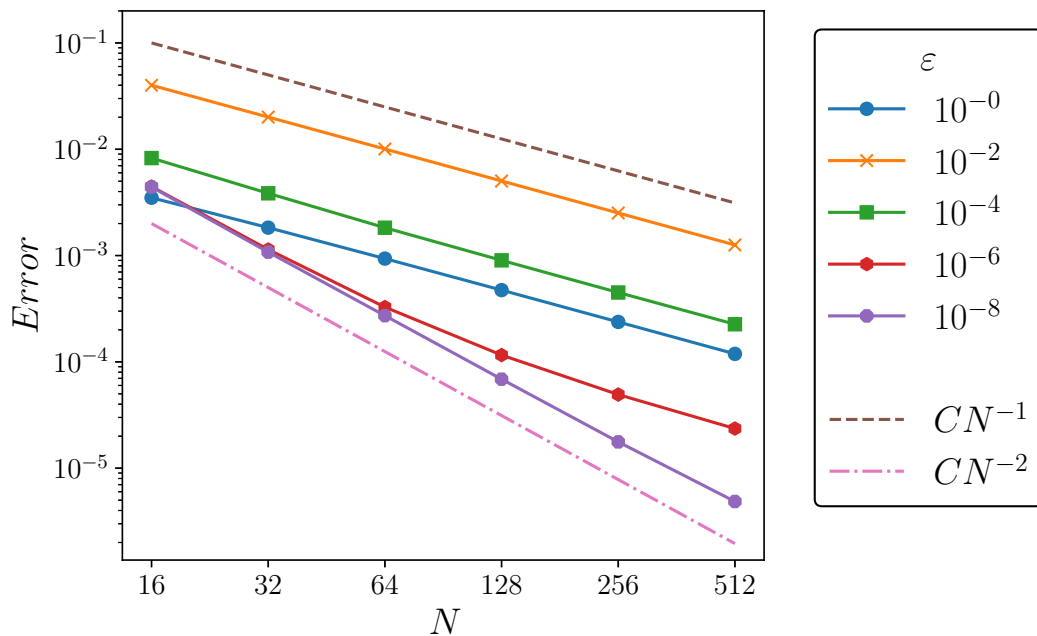
$$M(\vec{x}) = \begin{pmatrix} \max \left\{ 1, K_1 \frac{\beta}{\varepsilon(1+2y)^2} \exp \left( -\frac{\beta(1-x)}{\sigma\varepsilon(1+2y)^2} \right) \right\} & 0 \\ 0 & \max \left\{ 1, K_2 \frac{\beta}{\varepsilon(3-2x)^2} \exp \left( -\frac{\beta(1-y)}{\sigma\varepsilon(3-2x)^2} \right) \right\} \end{pmatrix}.$$

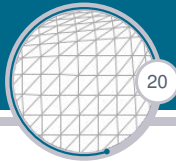
# 2D problem with spatially varying diffusion

Errors

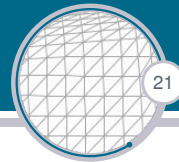


Plot of  $\|e^h\|_E$

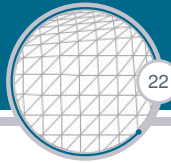




- ▶ The MPDE approach appears to be useful for generating **suitable** layer-adapted meshes, given sufficient *a priori* data.
- ▶ The method extends to 2D problems in situations where **non-tensor product** grids are appropriate.
- ▶ The MPDE is nonlinear and converges slowly for small  $\varepsilon$  and large  $N$ , we resolve this by combining the MPDE with *h*-refinement, the **iteration count** depends only very weakly on  $\varepsilon$ .
- ▶ For the 1D example provided, existing theory proves robust convergence. However, the 2D example provided does not have a theoretical basis.



- ▶ Our original motivation actually comes from solving some convection-diffusion-type problems: **advection-diffusion** and **Navier Stokes**. Initial results are promising.
- ▶ We are particularly interested in modelling dispersion and flow through **constricted channels**. MPDEs on convex sub-domains may be useful.
- ▶ We want to perform a comparison of **alternative** MPDE formulations.
- ▶ And, of course, we would like to incorporate *a posteriori* error estimation. To date, best results are with **hierarchical** error estimators.



[Alnæs et al., 2015] Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N. (2015).

The FEniCS project version 1.5.

*Archive of Numerical Software*, 3(100).

[Bakhvalov, 1969] Bakhvalov, N. S. (1969).

On the optimization of the methods for solving boundary value problems in the presence of a boundary layer.

*Ž. Vyčisl. Mat i Mat. Fiz.*, 9:841–859.

[Hill and Madden, 2021] Hill, R. and Madden, N. (2021).

Generating layer-adapted meshes using mesh partial differential equations.

*Numer. Math. Theor. Meth. Appl.*, to appear.



[Huang et al., 1994] Huang, W., Ren, Y., and Russell, R. D. (1994).

Moving mesh partial differential equations (MMPDES) based on the equidistribution principle.

*SIAM J. Numer. Anal.*, 31(3):709–730.

[Huang and Russell, 2011] Huang, W. and Russell, R. D. (2011).

*Adaptive Moving Mesh Methods*, volume 174 of *Applied Mathematical Sciences*.

Springer, New York.

[Linß, 2010] Linß, T. (2010).

*Layer-Adapted Meshes for Reaction-Convection-Diffusion Problems*, volume 1985 of *Lecture Notes in Mathematics*.

Springer-Verlag, Berlin.



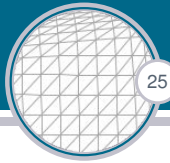
[Roos et al., 2008] Roos, H.-G., Stynes, M., and Tobiska, L. (2008).  
*Robust Numerical Methods for Singularly Perturbed Differential Equations*,  
volume 24 of *Springer Series in Computational Mathematics*.  
Springer-Verlag, Berlin, second edition.  
Convection-Diffusion-Reaction and Flow Problems.

## Acknowledgement

This project is funded by the Irish Research  
Council, (GOIPG/2017/463).



# From equidistribution to an MPDE



We derive an **MPDE** by considering the equidistribution principle as a mapping  $x(\xi) : [0, 1] \rightarrow [a, b]$  from the **computational** coordinate  $\xi$  to the **physical** coordinate  $x$ , which satisfies

$$\int_a^{x(\xi)} \rho(x) dx = \xi \int_a^b \rho(x) dx. \quad (5)$$

Differentiating (5) twice with respect to  $\xi$  we get the **nonlinear** MPDE,

$$-(\rho(x)x(\xi)')' = 0 \text{ for all } \xi \in (0, 1), \quad x(0) = a \text{ and } x(1) = b. \quad (6)$$

The BCs are necessary to result in a mesh generating function. The solution to (4) is a Bakhvalov mesh when  $\rho$  is as defined in (3). Since this is a **nonlinear** problem we use a fixed point iteration method to find a solution.

# CutFEM-style methods in FEniCSx: CAD and level sets

August Johansson, SINTEF Digital, Norway

Vibeke Skytt, SINTEF Digital, Norway

24 March 2021

The interest in finite element methods on non-matching meshes has increased in recent years. Theoretical advances, based on the ideas of Nitsche, have allowed for the construction of methods such as CutFEM for a wide variety of problems.

From the perspective of implementation, one major challenge is to be able to compute integrals over cells that only have a partial intersection with the domain. Often, quadrature rules for both the volume and surface are required by the method.

In this contribution we will present our work on using custom quadrature in the FEniCSx framework. We focus on domains described by level sets as well as domains from CAD, and restrict ourselves to quadrilateral and hexahedral elements.

---

You can cite this talk as:

August Johansson and Vibeke Skytt. “CutFEM-style methods in FEniCSx: CAD and level sets”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 401–412. DOI: 10.6084/m9.figshare.14495382.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/johansson.html>.



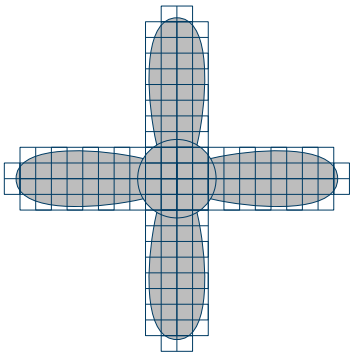
# CutFEM-style methods in FEniCS-X: CAD and level sets

August Johansson, Vibeke Skytt  
Mathematics and Cybernetics, SINTEF Digital

FEniCS 2021

2021-03-24

# Finite element methods on non-matching meshes



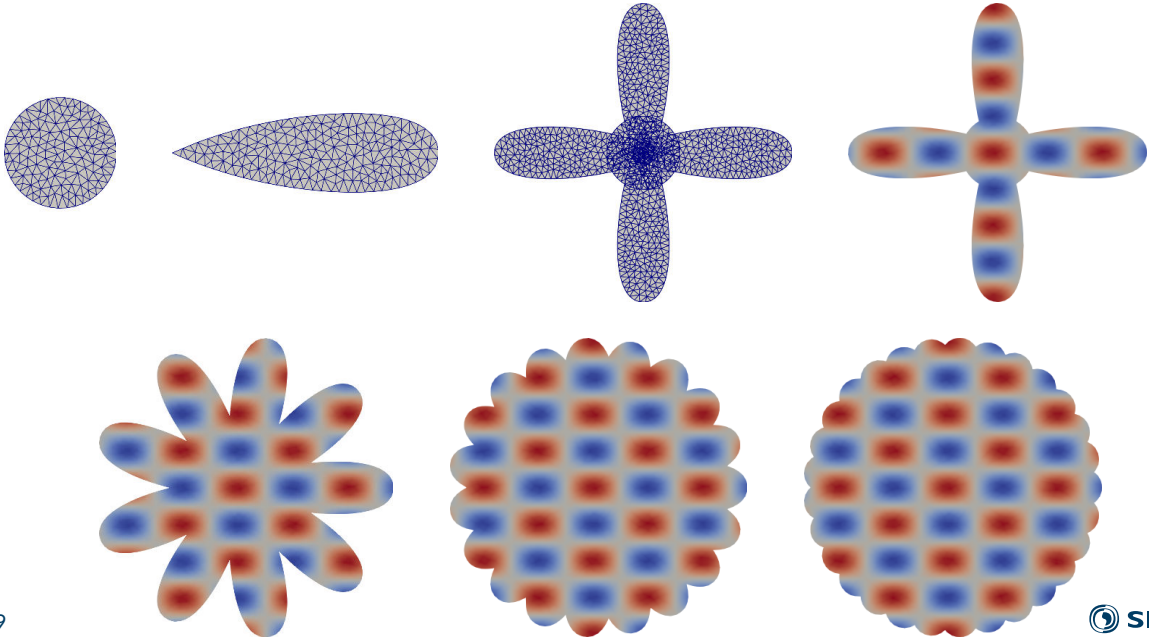
- Non-matching mesh construction is easy.
- The mesh is used for the finite element approximation.
- The mesh is **not** used for geometry approximation.
- Several methods exist: CutFEM,  $\phi$ -FEM, TraceFEM, etc.
- Beneficial for problems with dynamic domains.
- **Cut elements** are  $K : K \cap \partial\Omega \neq \emptyset$ .
- Custom quadrature on the cut elements for

$$\int_{K \cap \Omega} dx \quad \text{and} \quad \int_{K \cap \partial\Omega} ds$$

- **This talk:** a "library" with custom quadrature based on FEniCS-X.

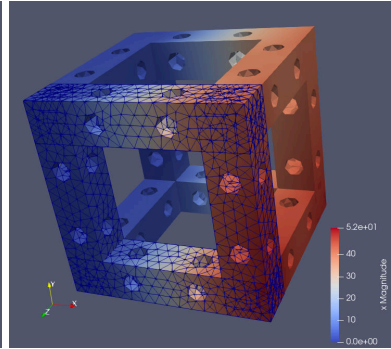
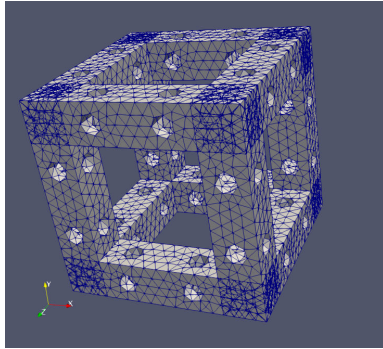
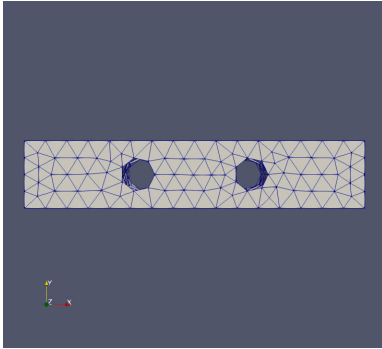
# Custom quadrature in FEniCS: MultiMesh 2D

---



# Custom quadrature in FEniCS: MultiMesh 3D

---



# Extensions to FFC-X & DOLFIN-X

---

The **FEniCS** pipeline has functionality for custom integrals:

```
void tabulate_tensor_custom(A, w, c, coord_dofs,  
                             num_qr, qr_pts, qr_w, normals)
```

Extend **FFC-X**:

- Include metadata as

```
ufl.dx(metadata={'quadrature_rule': 'runtime'}, domain=mesh)
```

- Make such integrals generate code for the custom integral type.
- Generate code for
  - calling `evaluate_basis_derivatives()`.
  - using the provided normals.

Extend **DOLFIN-X**:

- Add support for the custom integral type.
- Mimic setup for cell integrals.



# Important high-level functionality

---

- **Custom assembler** mimicking examples from `test_custom_assembler.py`:

```
custom_assemble_matrix(form, [(cells, qr_pts, qr_w, normals)])
```

- Works as standard FE assembly:
  - Set up sparsity pattern (same as cell integral type).
  - Loop over cells (typically the cut cells).
  - Call the custom integral kernel.
  - Assemble into the global matrix (e.g. calling `MatSetValues`).
- Utilities for setting **mesh tags** from the geometry representation.
- Function for **locking inactive dofs** similar to `DirichletBC`.

## Python example for testing bulk and surface quadrature

```
geom_kernel = Geometry('circle.iges')
cut_cells, uncut_cells = geom_kernel.cells()
qr_pts, qr_w = geom_kernel.bulk_qr()
qr_pts_surf, qr_w_surf, normals = geom_kernel.surf_qr()
bulk_data = (cut_cells, qr_pts, qr_w)
surf_data = (cut_cells, qr_pts_surf, qr_w_surf, normals)

cell_tags = get_cell_tags(mesh, uncut_cells, uncut_cell_tag=1)

dx_cut = ufl.dx(metadata={'quadrature_rule': 'runtime'}, domain=mesh)
dx_uncut = ufl.dx(subdomain_data=cell_tags, domain=mesh)

area = custom_assemble_scalar(1.0*dx_cut, [surf_data])
vol = custom_assemble_scalar(1.0*dx_cut, [bulk_data])
      + dolfinx.assemble_scalar(1.0*dx_uncut(uncut_cell_tag))
```

# Obtain quadrature from external libraries

## CAD: GoTools library

- <https://github.com/SINTEF-Geometry/GoTools>
- Spline geometry libraries by V. Skytt, T. Dokken et al (SINTEF).
- Quadrature created from tensor product-type constructions (work in progress).

## Level set: Algoim library

- <https://algoim.github.io>
- Level set library by R. Saye (LBL).
- Provides accurate quadrature.

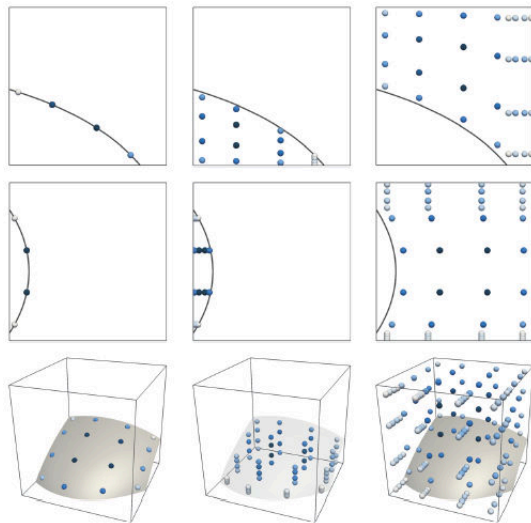


Figure: From <https://algoim.github.io> (with permission).

# Example: CutFEM Poisson on a circle, **spline** geometry

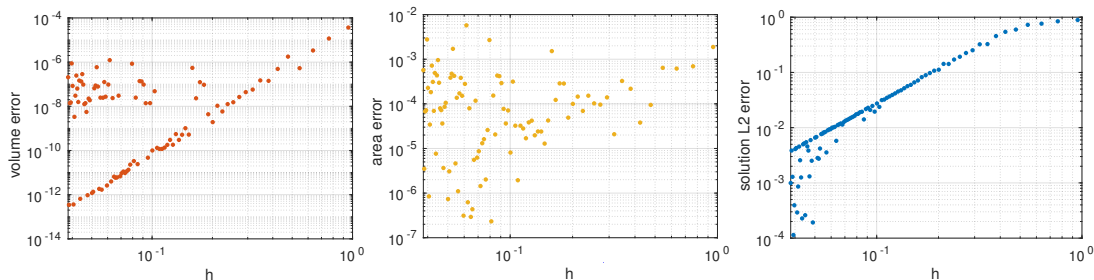


Figure: Relative volume error, relative area error and  $L^2$  error.

## Conclusions:

- Quadrature not correct.
- Do not see all errors in a weak norm.

# Example: CutFEM Poisson on a circle, **level set** geometry

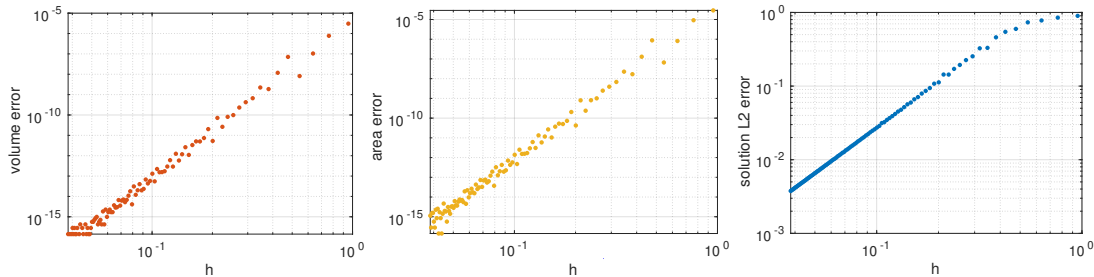


Figure: Relative volume error, relative area error and  $L^2$  error.

## Conclusions:

- Quadrature correct.
- Perfect convergence.



Technology for a better society

# Finite elements on accelerators: an experience using FEniCSx and SYCL

Igor Baratta ( IgorBaratta), University of Cambridge, United Kingdom

Chris Richardson ( chrisrichardson), University of Cambridge, United Kingdom

Garth Wells ( garth-wells), University of Cambridge, United Kingdom

24 March 2021

In this presentation, we are going to talk about our experience implementing the finite element method on different architectures and accelerators using the FEniCSx libraries and the SYCL programming model. Our main focus is on performance portability, we would like the FEM program to get consistent performance on a wide variety of platforms, instead of being very efficient on a single one.

SYCL is a modern kernel-based parallel programming model that allows for one code to be written which can run in multiple types of computational devices (eg CPUs and GPUs). A kernel describes a single operation, that can be instantiated many times and applied to different input data (eg cell-wise matrix assembly). This kernel-based model matches nicely with the new FEniCS data-centric design: DOLFINx generates data to operate in parallel (geometry, topology, and dofmaps) and FFCx generates efficient code that can be used as part of the parallel kernels.

We will discuss how different ways of expressing parallelism can affect the performance we ultimately achieve, for instance, we consider different global assembly strategies and data structures. We will also discuss how carefully arranging memory transfer and allocations can reduce latency and increase throughput in different accelerators. Finally, we will show some performance results of simplified finite element simulations on different architectures, ranging from Intel and AMD CPUs to NVIDIA GPUs.

---

You can cite this talk as:

Igor Baratta, Chris Richardson, and Garth Wells. “Finite elements on accelerators: an experience using FEniCSx and SYCL”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 413–429. DOI: 10.6084/m9.figshare.14495385.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/baratta.html>.



UNIVERSITY OF  
CAMBRIDGE

# Finite elements on accelerators

An experience using **FEniCSx** and **SYCL**

Igor Baratta  
Chris Richardson  
Garth Wells



# Table of Contents



- 01** INTRODUCTION
- 02** FINITE ELEMENT WORKFLOW
- 03** IMPLEMENTATION
- 04** RESULTS
- 05** FUTURE WORK

# What's Performance Portability?

And why do we care about it?

An application is performance portable if it:

- ✓ Achieves reasonable level of performance
- ✓ Requires minimal platform specific code



# Programming Model



SYCL is a high-level single source parallel programming model, that can target a range of heterogeneous platforms:

- uses completely standard C++;
- both host CPU and device code can be written in the same C++ source file;
- open standard coordinated by the **Khronos** group.

SYCL implementations:

Intel  
SYCL\*

hipSYCL\*

Compute  
Cpp

triSYCL\*

*\*open source*

```
cl::sycl::queue q{cl::sycl::gpu_selector()};

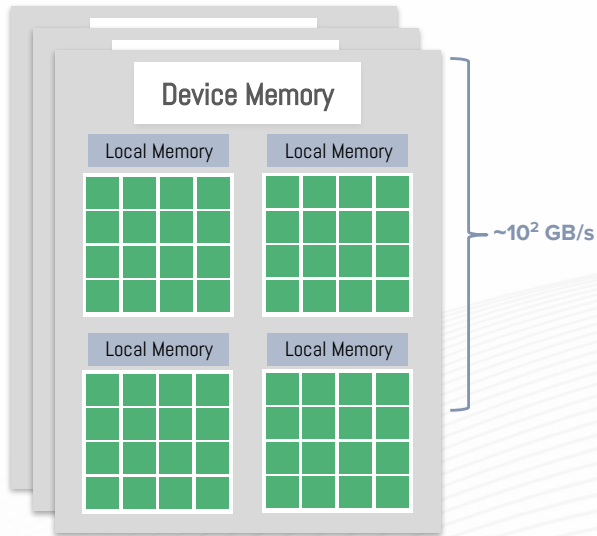
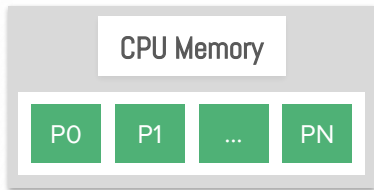
int N = 100;
auto a = cl::sycl::malloc_device<double>(N, q);
auto b = cl::sycl::malloc_shared<double>(N, q);
auto e = q.fill(a, 3.0, N);

q.parallel_for(cl::sycl::range<1>(N), e,
[=](cl::sycl::id<1> Id) {
    int i = Id.get(0);
    b[i] = 2 * a[i];
});

q.wait();

for (int i = 0; i < N; i++)
    assert(b[i] == 6.);
```

# Simple Workflow



Copy input data from **Host** memory to **Device** memory



Launch kernels for execution on the **Device**

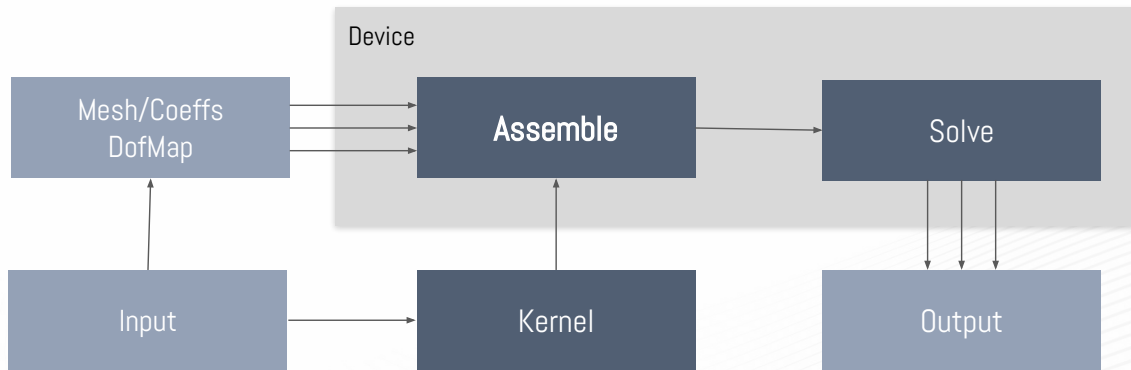


Wait for the execution queue to finish



Copy results back to **Host** from **Device**

# An idealised modular Finite Element workflow

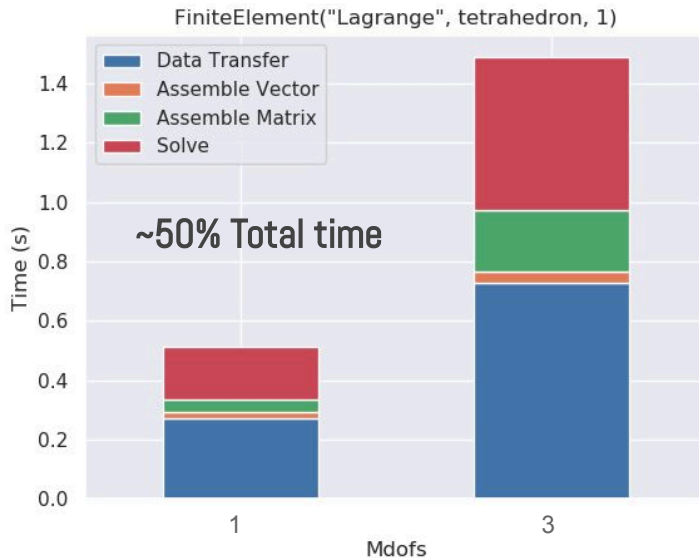


UFL  
File

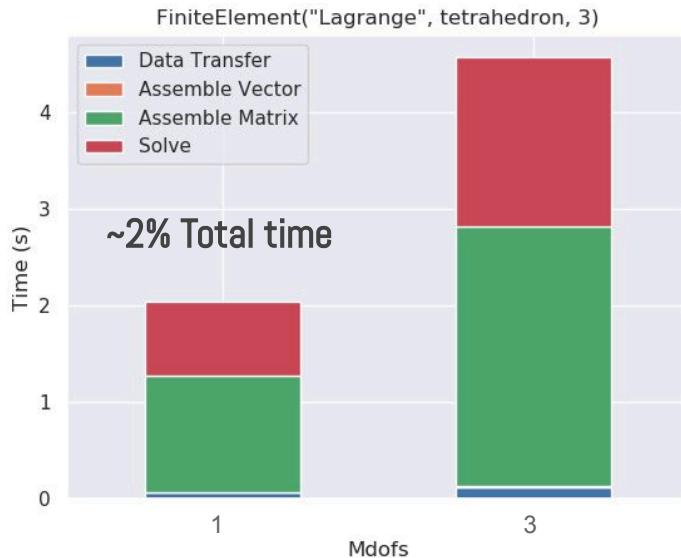
```
element = FiniteElement("Lagrange", tetrahedron, 3)
...
a = inner(grad(u), grad(v)) * dx + k*inner(u, v) * dx
L = inner(f, v) * dx
```

ffcx --sycl\_defines=True problem.ufl

# Data Transfer to Computation Ratio - P1



# Data Transfer to Computation Ratio - P3





# Matrix Assembly

For each cell:

- 01 Gather cell coordinates and coefficients
- 02 Compute element matrix
- 03 Update global CSR matrix

Global assembly strategies:

- Binary Search\*
- Lookup Table\*
- Two Stage

\* atomic operations

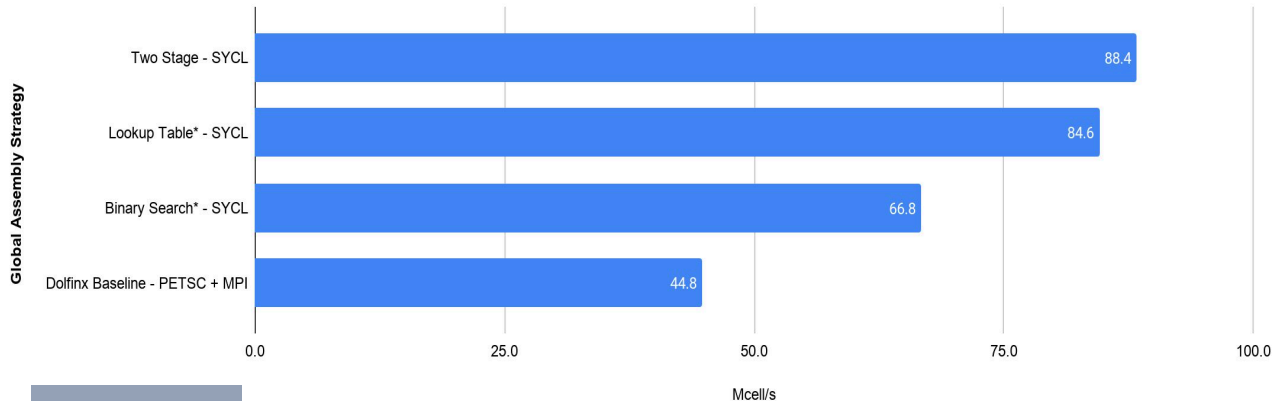
```
auto kernel = [=] (cl::sycl::id<1> ID) {
    const int i = ID.get(0);
    ...
    double Ae [ndofs * nofs];

    // Gather cell coordinates and coefficients
    for (std::size_t j = 0; j < 4; ++j)
    {
        const std::size_t dmi = x_coor[i * 4 + j];
        for (int k = 0; k < gdim; ++k)
            cell_geom[j * gdim + k] = x[dmi * gdim + k];
    }
    ...
    // Compute element matrix
    tabulate_cell_a(Ae, coeffs, cell_geom);

    // Update global matrix - Binary Search
    for (int j = 0; j < ndofs; j++)
        for (int k = 0; k < ndofs; k++)
        {
            int ind = dofs[offset + k];
            int pos = find(indices, first, last, ind);
            atomic_ref atomic_A(data[pos]);
            atomic_A += Ae[j * ndofs + k];
        }
    };
```

# Matrix Assembly - CPU Performance

Performance (MCells/s) P1 - 20 Mcells, 3 Mdofs

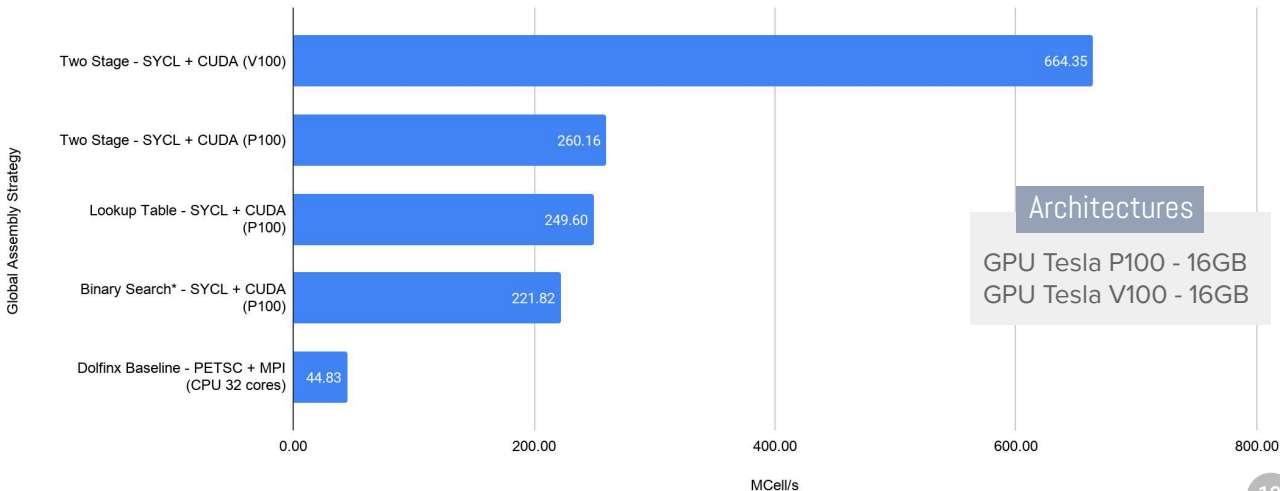


## Architectures

2 x Intel Xeon Skylake 6142 processors, 2.6GHz 16-core  
Theoretical peak performance: 2.7 Tflop/s.  
192GB RAM

# Matrix Assembly - GPU Performance

Performance (in Mcell/s) of assembling a CSR matrix for the Helmholtz problem on a GPU.  
FiniteElement('Lagrange', tetrahedron, 1)



# Matrix Assembly - GPU Performance

Performance (in Mcell/s) of assembling a CSR matrix for the Helmholtz problem on a GPU.  
FiniteElement('Lagrange', tetrahedron, 1)



# Low Achieved Occupancy

Achieved Occupancy: ~25%

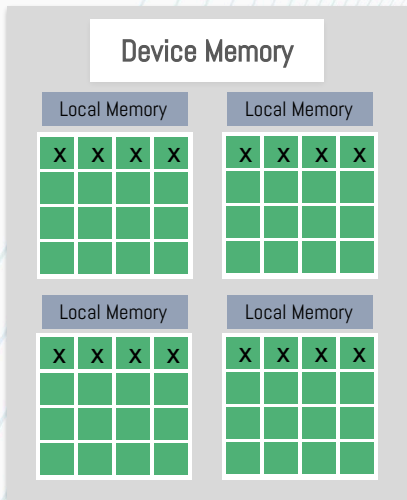
The occupancy limited by register usage.

## Solution:

Use shared memory for precomputed tables.

Each thread block (work-group) has shared memory visible to all threads (work-item) of the block.

	Occupancy	MCell/s
1st Version	25%	664 MCell/s
Shared Memory	63%	1660 MCell/s
Reference CUDA <sup>1</sup>	*	1627 MCells/s



# Thank you!

The code and reproducibility  
instructions can be found at  
<https://github.com/Excalibur-SLE/dolfinx.sycl>



You can reach me via e-mail:  
**ia397@cam.ac.uk**

# Future/Ongoing Work

---

Different problems,  
and meshes

Linear Elasticity,  
Maxwell's equations

Profiling in a wider  
range of devices

AMD GPU, A64FX

Multi-GPU

MPI-based distributed  
memory computations

Code transformation

Improve generated  
code

# Working with complex meshes: The mesh processing pipeline

**U Meenu Krishnan**

([https://computationalmechanics.in/rajib\\_teams/u-meenu-krishnan/](https://computationalmechanics.in/rajib_teams/u-meenu-krishnan/)), Indian Institute of Technology, Roorkee, India

Abhinav Gupta, Indian Institute of Technology, Roorkee, India

Rajib Chowdhury, Indian Institute of Technology, Roorkee, India

**24 March 2021**

In any finite element analysis, it is very important to have an efficient tool to generate the mesh. In our work, we use Gmsh as a meshing software, and it is observed that the main problem with the current implementation of the mesh processing pipeline is that it is facing difficulties to handle models with complex geometries comprising of many boundaries and loading conditions; this means it has more number of physical groups. We use these physical groups to assign material properties, loads, or boundary conditions to the model.

The main objective of this talk is to explain the complete mesh processing pipeline for complex geometry. Starting from modeling of a mesh geometry and marking different boundary conditions in Gmsh, then will discuss about the benefits of using Mesh functions and Mesh Value Collections in FEniCS, and we will also discuss how to use the number tags from the XDMF file for defining various subdomains required for load and boundary conditions applications, and at the end how to visualize the XDMF file for validating if all the boundary conditions and loads are applied appropriately.

This talk will help the community to understand the complete mesh processing pipeline for a complex mesh geometry with increased efficiency.

---

You can cite this talk as:

U Meenu Krishnan, Abhinav Gupta, and Rajib Chowdhury. “Working with complex meshes: The mesh processing pipeline”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 430–438. DOI: 10.6084/m9.figshare.14495403.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/krishnan.html>.



# Working with Complex Meshes : The Mesh Processing Pipeline

FEniCS-2021

**U. Meenu Krishnan**

umeenukrishnan@ce.iitr.ac.in

Indian Institute of Technology, Roorkee

Abhinav Gupta,

Indian Institute of Technology, Roorkee

Dr. Rajib Chowdhury,

Indian Institute of Technology, Roorkee

# Motivation

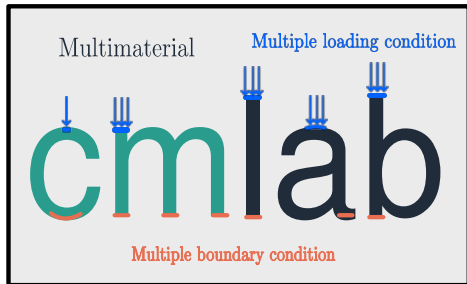
## We wish to use FEniCS with complex geometries

In practice, a real world engineering structure could have :

1. Multiple loading areas
2. Multiple boundary conditions
3. Multiple materials

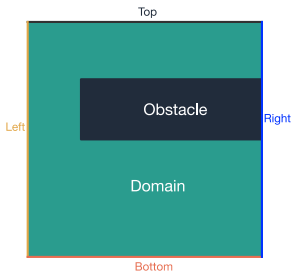
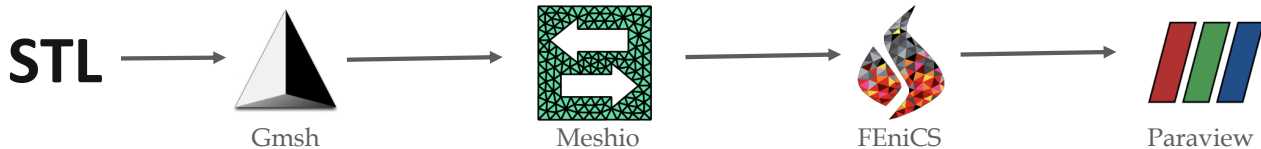
Thus can have 10 – 100's of marked regions in the mesh

**Problem:** This could result in human error in the process of modelling



“Output of a simulation is as good as the accuracy in the mathematical modelling”

# Preferred mesh processing pipeline



```
$PhysicalNames
6
1 3 "Top"
1 4 "Right"
1 5 "Left"
1 6 "Bottom"
2 1 "Domain"
2 2 "Obstacle"
$EndPhysicalNames
```

```
# Define Dirichlet boundary conditions at top and bottom boundaries
```

```
bcs = [DirichletBC(V, 5.0, boundaries, 2),
```

```
DirichletBC(V, 0.0, boundaries, 4)]
```

```
# Define new measures associated with the interior domains and
```

```
# exterior boundaries
```

```
dx = Measure("dx")[domains]
```

```
ds = Measure("ds")[boundaries]
```

```
# Define variational form
```

```
F = (inner(a0*grad(u), grad(v))*dx(0) + inner(a1*grad(u), grad(v))*dx(1)
```

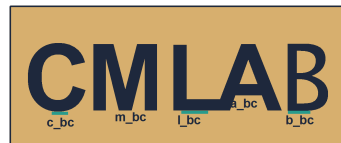
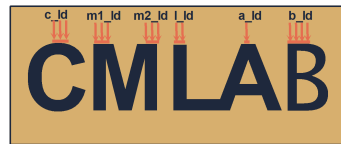
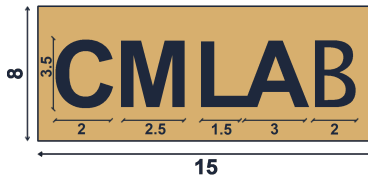
```
- g_L*v*ds(1) - g_R*v*ds(3)
```

```
- f*v*dx(0) - f*v*dx(1))
```

# Basic design approach

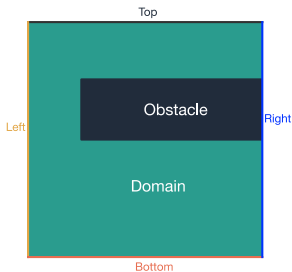
Engineering structures are accompanied with schematic drawings

1. Layout of the structure
2. Details of boundary conditions
3. Details of loading condition
4. Details about material properties



Aim : To use the same tag names which is in the schematic drawing in the FEniCS implementation

# Desired mesh processing pipeline



```
$PhysicalNames
6
1 3 "Top"
1 4 "Right"
1 5 "Left"
1 6 "Bottom"
2 1 "Domain"
2 2 "Obstacle"
$EndPhysicalNames
```

```
# Define Dirichlet boundary conditions at top and bottom boundaries
```

```
bcs = [DirichletBC(V, 5.0, boundaries, tags['Top']),
```

```
        DirichletBC(V, 0.0, boundaries, tags['Bottom'])]
```

```
# Define new measures associated with the interior domains and
```

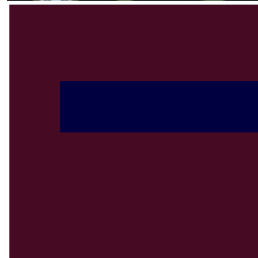
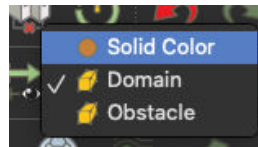
```
# exterior boundaries
```

```
dx = Measure("dx")[domains]
```

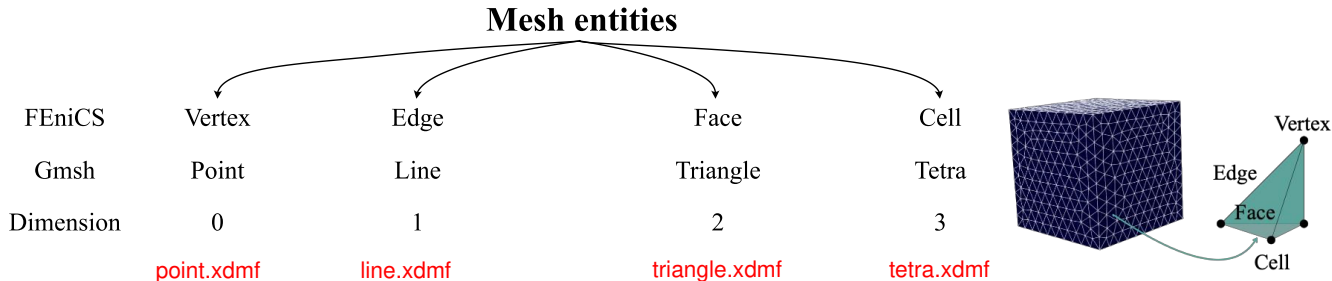
```
ds = Measure("ds")[boundaries]
```

```
# Define variational form
```

```
F = (inner(a0*grad(u), grad(v))*dx(tags['Domain'])
      + inner(a1*grad(u), grad(v))*dx(tags['Obstacle'])
      - g_L*v*ds(tags['Left']) - g_R*v*ds(tags['Right'])
      - f*v*ds(tags['Domain']) - f*v*ds(tags['Obstacle']))
```



# Basis for Meshx



```
{
  "Top": 3,
  "Right": 4,
  "Left": 5,
  "Bottom": 6,
  "Domain": 1,
  "Obstacle": 2
}
```

tags.json

```
domain_mvc = MeshValueCollection("size_t", mesh, dim)
with XDMFFile("mesh/triangle.xdmf") as infile:
    infile.read(domain_mvc, "tag")

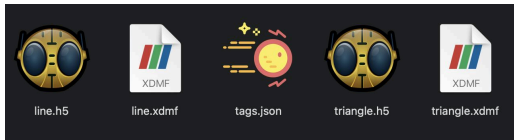
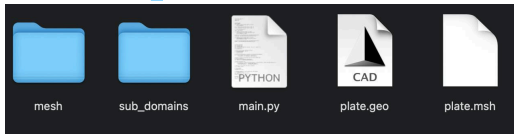
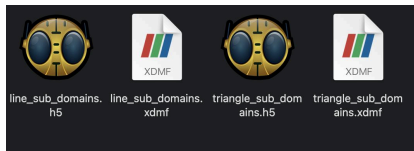
domain = cpp.mesh.MeshFunctionSizet(mesh, domain_mvc)

f = open('mesh/tags.json')
tags = json.load(f)
```

```
# Define variational form
F = (inner(a0*grad(u), grad(v))*dx(tags['Domain'])
    + inner(a1*grad(u), grad(v))*dx(tags['Obstacle'])
    - g_L*v*ds(tags['Left']) - g_R*v*ds(tags['Right'])
    - f*v*dx(tags['Domain']) - f*v*dx(tags['Obstacle']))
```

# Example:

```
→ root cd Codes/  
→ Codes cd poisson/  
→ poisson ls  
main.py plate.geo plate.msh  
→ poisson meshx plate.msh  
String tag      Num Tag      Dim  
-----  
Top              3          1  
Right            4          1  
Left             5          1  
Bottom           6          1  
Domain           1          2  
Obstacle         2          2  
Creating line mesh  
Creating triangle mesh  
XDMF created! 🎉  
→ poisson ls  
main.py mesh plate.geo plate.msh sub_domains  
→ poisson python3 main.py
```



GitHub repository for meshx:

<https://github.com/iitrabhi/meshx>

You can use this Docker image:

<https://github.com/iitrabhi/fenics-docker>

# Thank You....

(computationalmechanics.in)



# Numerical investigation of the interaction of two electrolytic drops under an external electric field

**Shyam Sunder Yadav** (<https://www.bits-pilani.ac.in/pilani/ssyadav/Profile>), Birla Institute of Technology and Science Pilani, India

25 March 2021

In the current work, we study the interaction of two micro droplets containing electrolytes in presence of an external electric field. For this purpose, we use the open source code BERNAlSE which is built on top of FEniCS and capable of simulating the electrohydrodynamics of binary electrolytes. We focus on the movement of the electrolytes in and around the thin liquid bridge which forms during the interaction of droplets and investigate their role in causing coalescence or bouncing of droplets at different strengths of externally applied electric field.

---

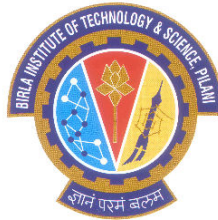
You can cite this talk as:

Shyam Sunder Yadav. “Numerical investigation of the interaction of two electrolytic drops under an external electric field”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 439–457. DOI: 10.6084/m9.figshare.14495406.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/yadav.html>.

# Numerical investigation of the interaction of two electrolytic drops under an external electric field

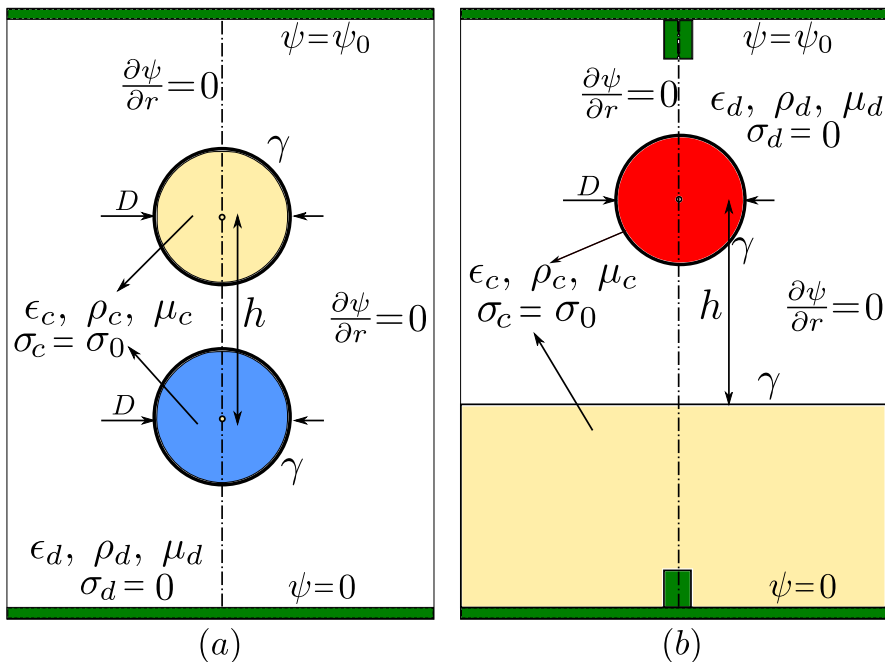
Shyam Sunder Yadav  
*Assistant Professor, Mechanical Engineering*



BITS Pilani, India  
March 25, 2021

# Interaction of charged droplets

## Drop-Interface and Drop-Drop Interactions



**Figure:** Computational domain for (a) Drop-Interface, (b) Drop-Drop interactions

# Interaction of charged droplets

Codes used in the current study

- Two different numerical techniques are used
  - ▶ Finite Differences + CLSVOF method
    - ★ Developed by my advisor and his advisor!
    - ★ I added the charge advection using VOF
  - ▶ Finite Element + Phase field method
    - ★ Developed by Gaute Linga
    - ★ Based on FENICS, Code is called BERNAISE
    - ★ <https://github.com/gautelinga/BERNAISE>

# Governing equations in CLSVOF based code

- **Navier-Stokes equation:**

- ▶  $\rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \nabla \cdot [\mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)] + \rho \mathbf{g} + \mathbf{f}_v^\gamma + \mathbf{f}_v^E$
- ▶  $\nabla \cdot \vec{U} = 0$

- **Interface advection:**

- ▶  $\frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi = 0$
- ▶  $\frac{\partial F}{\partial t} + \mathbf{v} \cdot \nabla F = 0$

- **Equations for quasi-electrostatics**

- ▶  $\nabla \cdot \epsilon_0 \epsilon \vec{E} = q_v$
- ▶  $\mathbf{f}_v^E = q_v \mathbf{E} - \frac{1}{2} \epsilon_0 E^2 \nabla \epsilon$
- ▶  $\frac{\partial q_v}{\partial t} + \mathbf{v} \cdot \nabla q_v + \nabla \cdot \sigma \mathbf{E} = 0$

- **Surface tension forces**

- ▶  $\mathbf{f}_v^\gamma = \gamma \kappa \hat{n} \delta_s$
- ▶  $\hat{n} = -\frac{\nabla \phi}{|\nabla \phi|}$

# Numerical Schemes in CLSVOF based code

## Discretization summary

- **Grid:** Staggered grid, (Harlow and Welch).
- **Viscous terms:** Second order accurate central difference scheme.
- **Convective terms:** Second order ENO scheme, (Harten et al.).
- **Surface tension:** Continuum surface force model, (Brackbill et al.).
- **Electric forces:** Continuum electric force model, (Tomar et al.).
- **Temporal term:** First order accurate explicit Euler method.
- **Pressure:** Second order accurate Projection method, (Chorin).
- **Interface capturing:** CLSVOF algorithm, (Sussman and Puckett).
- **Time step:** Variable time step is used:
  - ▶ **CFL criterion :**  $\Delta t \leq cfl \frac{\Delta x}{u_{max}}$
  - ▶ **Viscous time scale :**  $\Delta t \leq \frac{\rho \Delta x^2}{4\mu}$
  - ▶ **Capillary time scale :**  $\Delta t \leq \left[ \frac{(\rho_1 + \rho_2) \Delta x^3}{\gamma} \right]^{\frac{1}{2}}$
  - ▶ **Charge Relaxation time scale :**  $\Delta t \leq \frac{\epsilon_0 \epsilon}{\sigma}$

# Governing equations in BERNASE

Based on FENICS

- Two-phase electrokinetic flows are described by the coupled problem of solute transport, fluid flow and electrostatics

- $$\begin{aligned} \rightarrow & \frac{\partial}{\partial t} \rho(\phi) \vec{U} + \nabla \cdot \rho(\phi) \vec{U} \vec{U} - \nabla \cdot [2\mu(\phi) \mathbb{D} + \vec{U} \rho'(\phi) M(\phi) \nabla g_\phi] + \nabla p = \\ & -\phi \nabla g_\phi - \sum c_j \nabla g_{c_j} \end{aligned}$$
- $$\rightarrow \nabla \cdot \vec{U} = 0$$
- $$\rightarrow \frac{\partial c_j}{\partial t} + \vec{U} \cdot \nabla c_j - \nabla \cdot (K_j(\phi) c_j \nabla g_{c_j}) = 0$$
- $$\rightarrow \nabla \cdot (\epsilon_0 \epsilon \vec{E}) = \rho_e$$
- $$\rightarrow [2\mu \mathbb{D} - p' \mathbb{I} + \gamma \kappa \mathbb{I} + \epsilon_0 \epsilon \vec{E} \vec{E} - \frac{1}{2} \epsilon_0 \epsilon E^2 \mathbb{I}] \cdot \hat{n} = 0$$
- $$\rightarrow \frac{\partial \phi}{\partial t} + \vec{U} \cdot \nabla \phi - \nabla \cdot (M(\phi) \nabla g_\phi) = 0$$

- Chemical potential of species  $c_j$  and the phase field  $\phi$

- $$\rightarrow g_{c_j}(c_j, \phi) = \alpha'(c_j) + \beta_j(\phi) + z_j V$$
- $$\rightarrow \text{For dilute solutions: } \alpha(c) = c(\log(c) - 1)$$
- $$\rightarrow g_\phi = \frac{\partial f}{\partial \phi} - \nabla \cdot \frac{\partial f}{\partial \nabla \phi} + \sum \beta'_j(\phi) c_j - \frac{1}{2} \epsilon'(\phi) |\nabla V|^2$$
- $$\rightarrow f(\phi, \nabla \phi) = \frac{3\sigma}{2\sqrt{2}} \left[ \frac{\epsilon}{2} |\nabla \phi|^2 + \epsilon^{-1} W(\phi) \right]$$
- $$\rightarrow W(\phi) = \frac{(1-\phi^2)^2}{4}$$

- Phase field mobility:  $M(\phi) = \epsilon M_0$  or  $M(\phi) = M_0 * \max(1 - \phi^2)$

# CLSVOF based simulations

Drop-Drop, Situation before contact

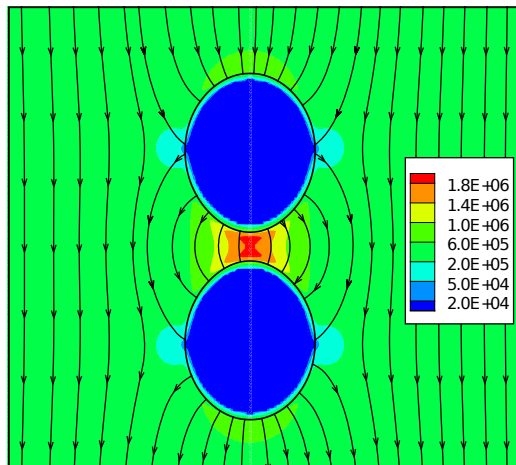


Figure: Efield before contact

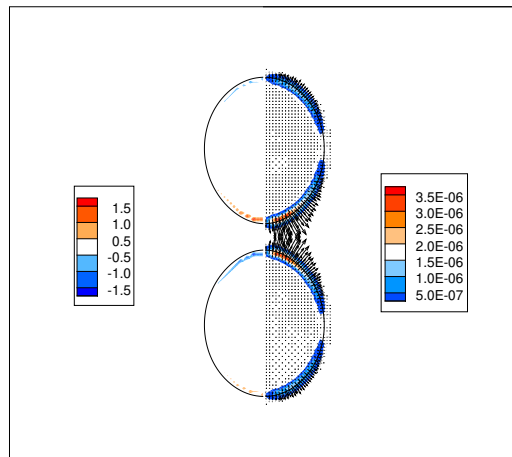


Figure: Charge & Eforce before contact



# CLSVOF based simulations

## Drop-Drop, Situation at contact

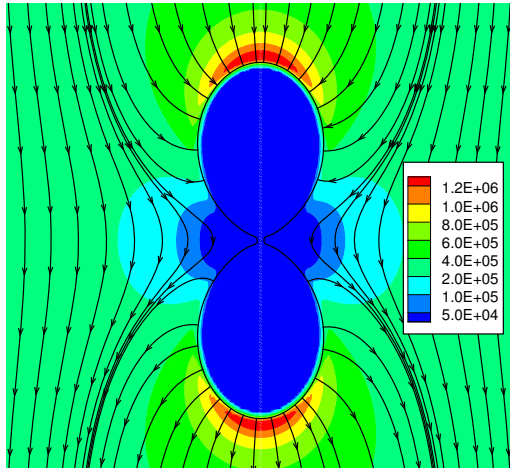


Figure: Efield at contact

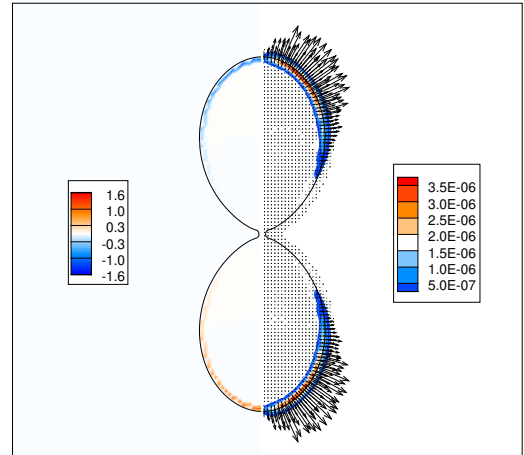


Figure: Charge & Eforce at contact

# CLSVOF based simulations

Drop-Drop, Situation after contact

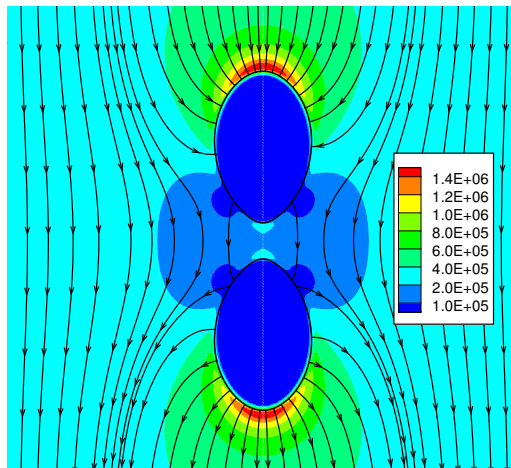


Figure: Efield after contact

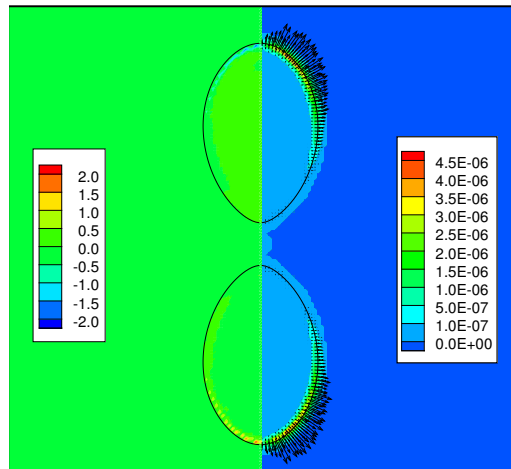


Figure: Charge & Eforce after contact

# Drop-Interface Interactions

Situation before contact

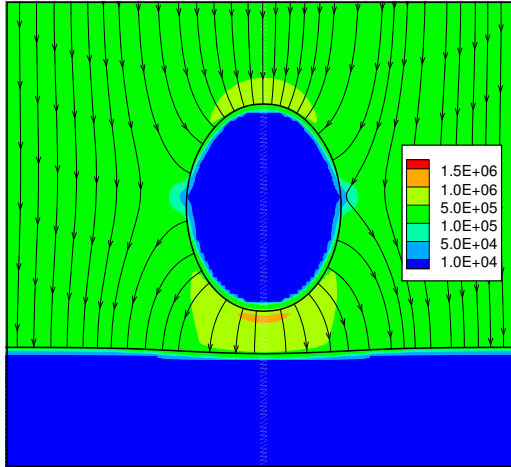


Figure: Efield before contact

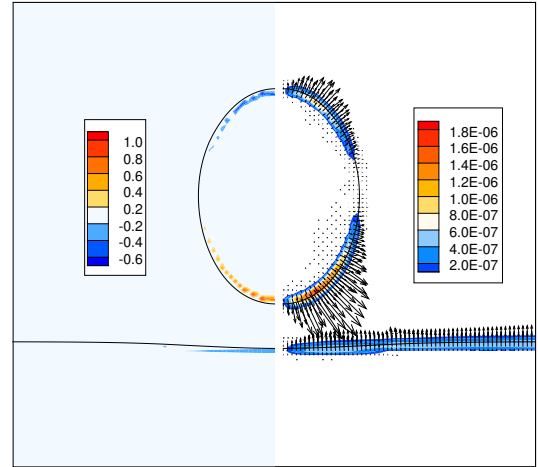


Figure: Charge & Eforce before contact

# Drop-Interface Interactions

Situation at contact

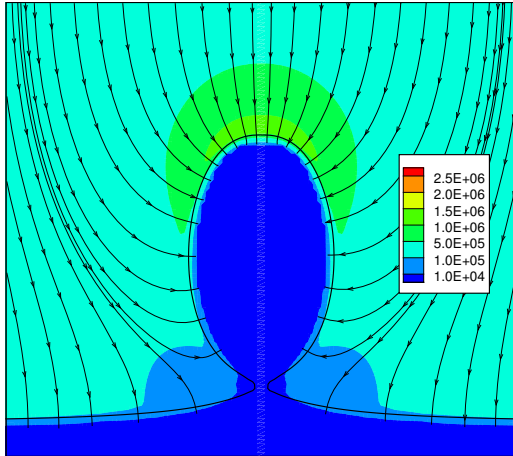


Figure: Efield at contact

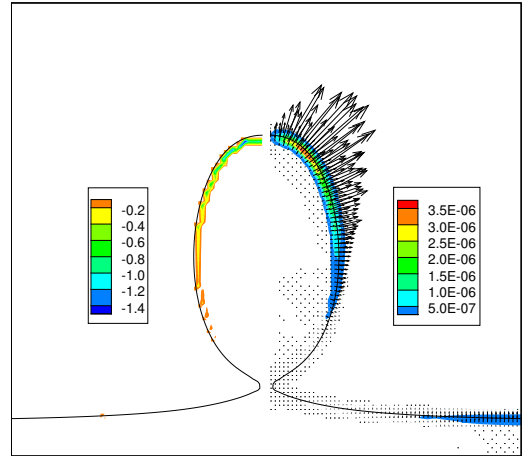


Figure: Charge & Eforce at contact

# Drop-Interface Interactions

Situation after contact

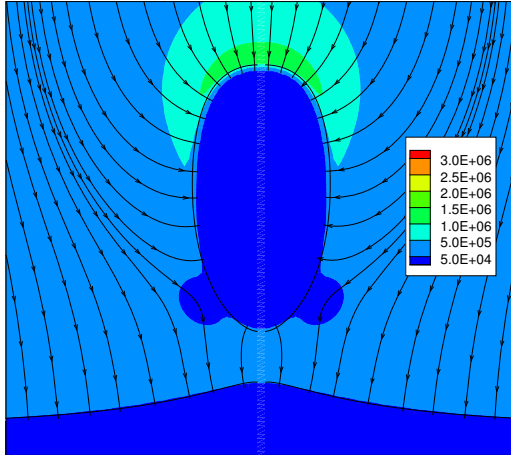


Figure: Efield after contact

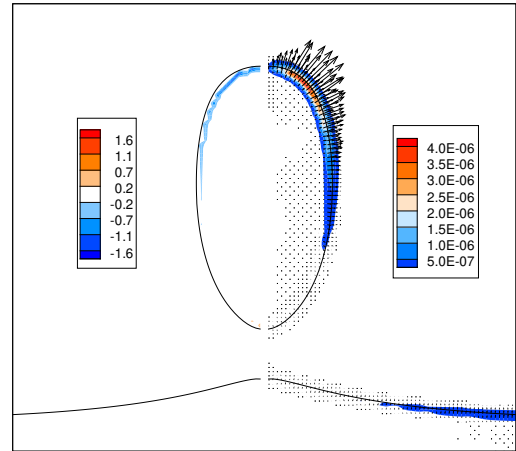


Figure: Charge & Eforce after contact

# Drop-Interface Interactions

Drop-Interface, Greater velocity after contact

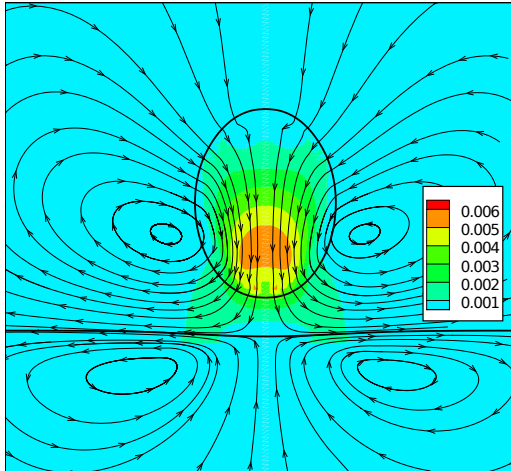


Figure: Velocity before contact

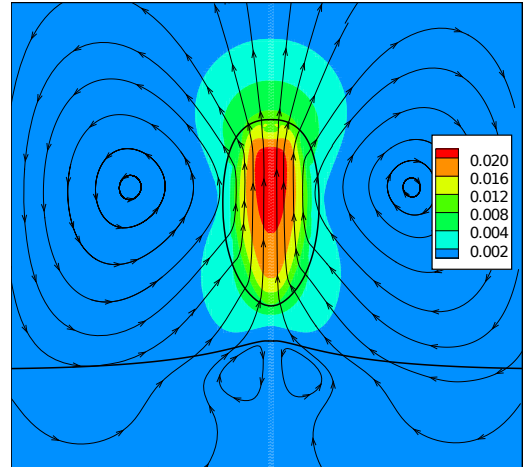
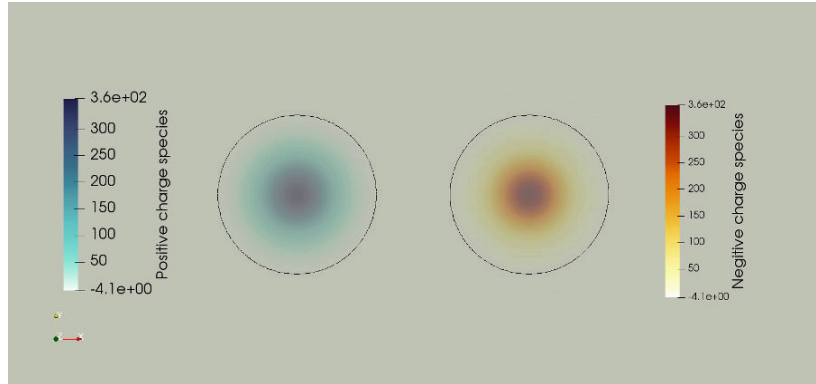


Figure: Velocity after contact

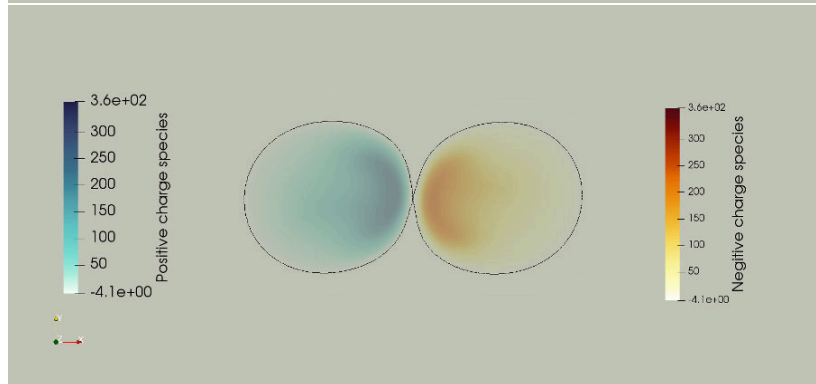
# BERNAISE based results

## Distribution of charged species

- Initial distribution



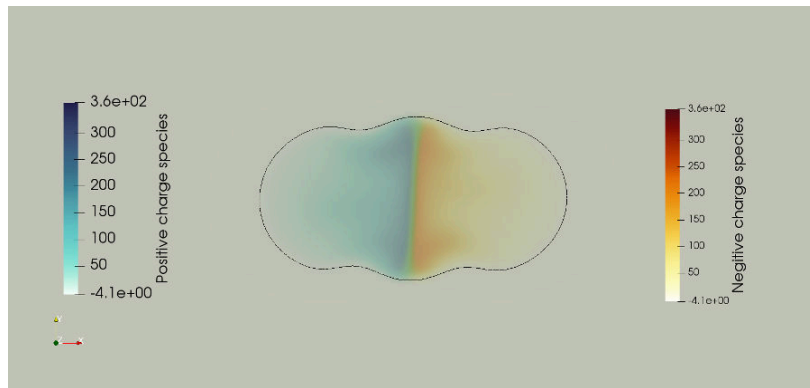
- Just before contact



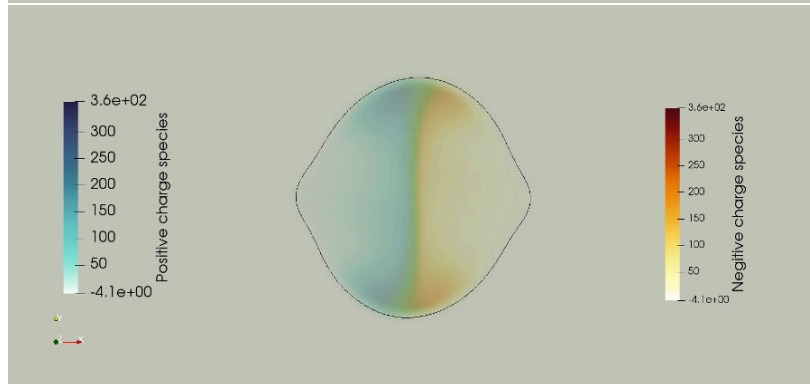
# BERNAISE based results

## Distribution of charged species

- During contact



- During contact

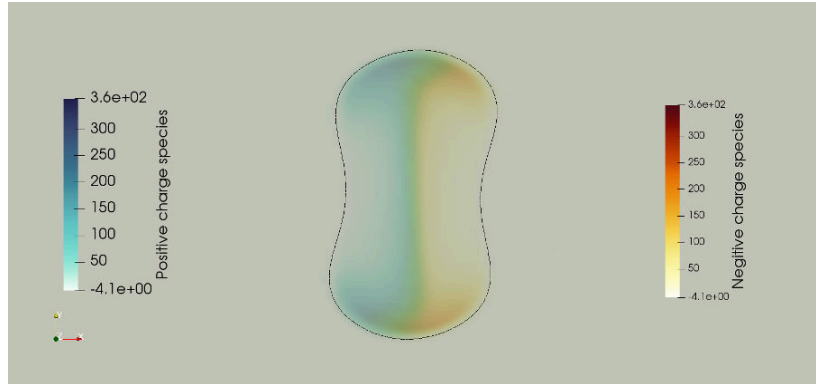




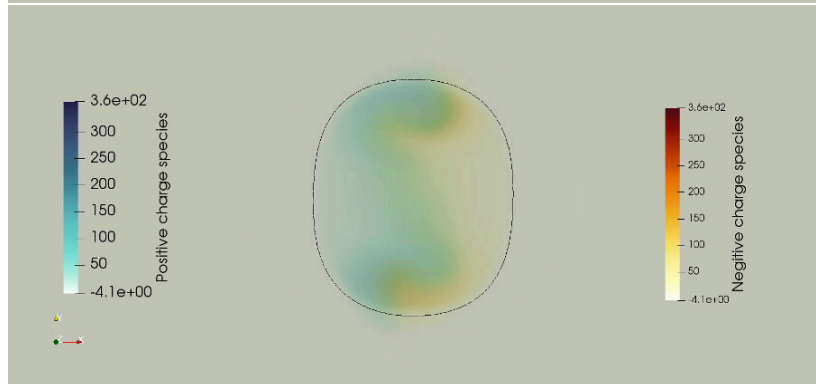
# BERNAISE based results

## Distribution of charged species

- During contact



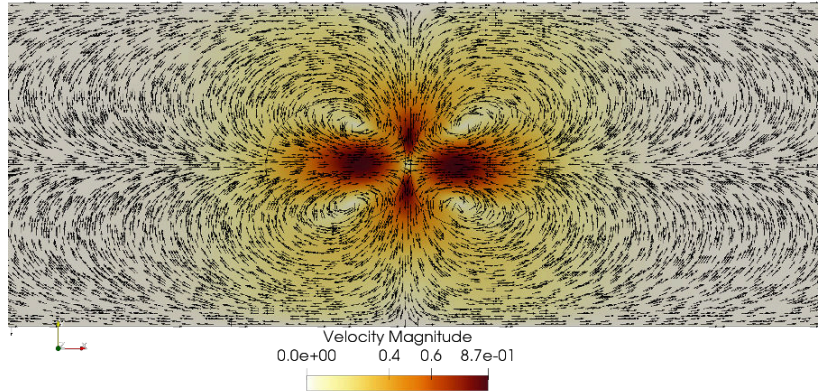
- During contact



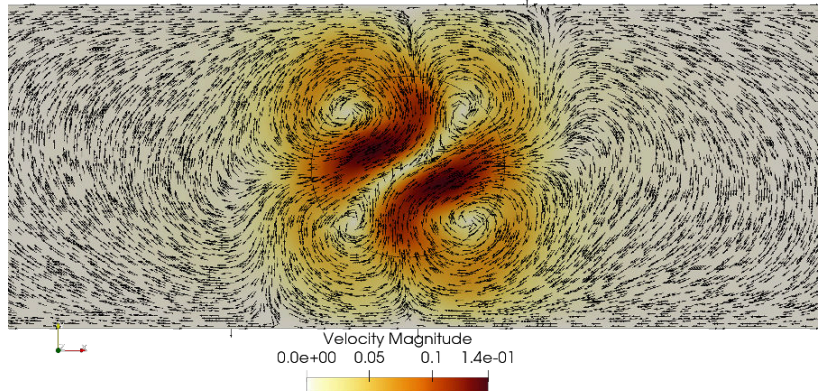
# BERNAISE based results

## Velocity distribution in the domain

- Before contact



- After contact



**Thank you for your attention.  
Questions...**

# FESTIM, a modelling code for hydrogen transport in materials for nuclear fusion applications

Rémi Delaporte-Mathurin, CEA, France

Etienne Hodille, CEA, France

Floriane Leblond, CEA, France

Jonathan Mougenot, LSPM, France

Yann Charles, LSPM, France

Christian Grisolia, CEA, France

James Dark, LSPM, France

25 March 2021

The principle of nuclear fusion is to fuse two hydrogen nuclei to form a helium nucleus and a neutron, releasing incredible amounts of energy in the process. To achieve these fusion reactions, extremely high temperatures are required: more than 10 times the temperature of the Sun's core. The very hot fuel in the plasma is magnetically confined within a chamber called a tokamak. Eventually, hydrogen ions will hit the reactor walls and penetrate in the materials.

In order to simulate hydrogen transport in complex components (multi-material, multidimensional geometries...), a finite element modelling code relying on FEniCS called FESTIM has been developed [2]. FESTIM solves a set of transient Macroscopic Rate Equations (MRE) which accounts for the diffusion (based on Fick's law) and trapping/detrapping of hydrogen isotopes in materials (based on McNabb and Foster's equations [4]) coupled to transient heat transfer.

This talk showcases the use of FESTIM and FEniCS to model key tokamak components such as actively cooled plasma facing components and how results crucial for the International Thermonuclear Experimental Reactor (ITER) [3] operations are extracted from it [1]. The code was verified using the method of manufactured solutions and validated against experimental results. FESTIM was also benchmarked with other codes from the fusion community and with the commercial simulation suite Abaqus.

This talk was awarded a prize: Best talk by a PhD student or undergraduate.

## References

- [1] Rémi Delaporte-Mathurin, Etienne Hodille, Jonathan Mougenot, Gregory De Temmerman, Yann Charles, and Christian Grisolia. "Parametric study of hydrogenic inventory in the ITER divertor based on machine learning". In: *Scientific Reports* 10.1:17798 (2020). DOI: 10.1038/s41598-020-74844-w.
- [2] Rémi Delaporte-Mathurin, Etienne A. Hodille, Jonathan Mougenot, Yann Charles, and Christian Grisolia. "Finite element analysis of hydrogen retention in ITER plasma facing components using FESTIM". 2019.
- [3] "ITER website". <http://www.iter.org>.
- [4] A. McNabb and P. K. Foster. "A new analysis of the diffusion of hydrogen in iron and ferritic steels". In: *Transactions of the Metallurgical Society of AIME* 227 (1963), 618–627.

---

You can cite this talk as:

Rémi Delaporte-Mathurin, Etienne Hodille, Floriane Leblond, Jonathan Mougenot, Yann Charles, Christian Grisolia, and James Dark. "FESTIM, a modelling code for hydrogen transport in materials for nuclear fusion applications". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 458–473. DOI: 10.6084/m9.figshare.14495427.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/delaporte-mathurin.html>.



## **FESTIM, a modelling code for hydrogen transport in materials for nuclear fusion applications**

*R. Delaporte-Mathurin<sup>1,2</sup>, E. A. Hodille<sup>1</sup>, J. Dark<sup>2</sup>, F. Leblond<sup>1</sup>, J. Mougenot<sup>2</sup>, Y. Charles<sup>2</sup>, C. Grisolia<sup>1</sup>*

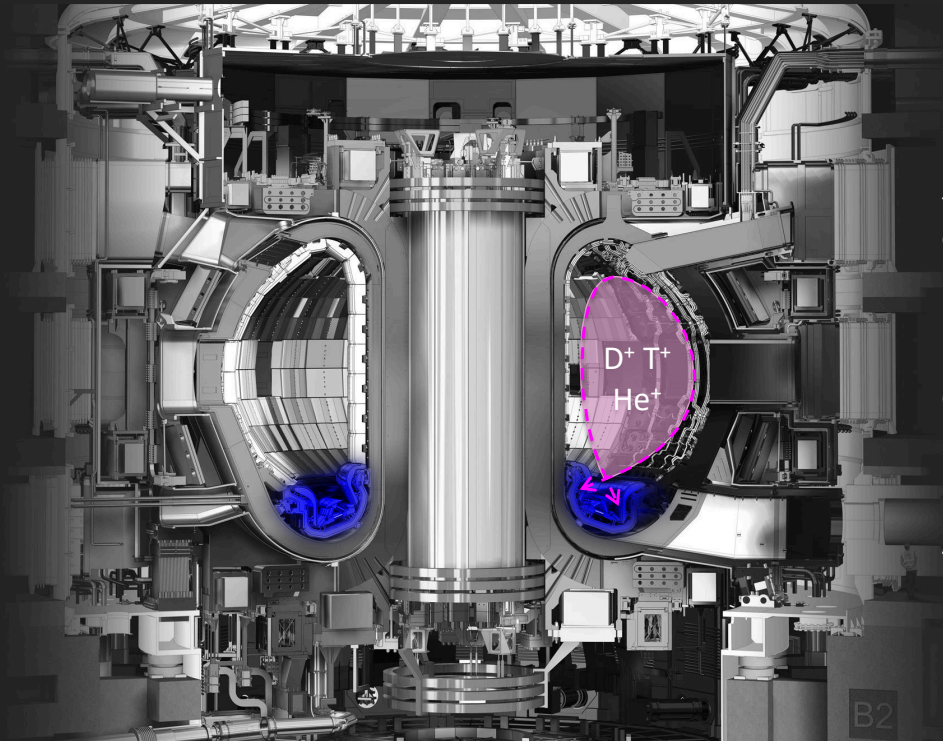
<sup>1</sup> CEA, IRFM/GCFPM, F-13108 Saint-Paul-lez-Durance, France

<sup>2</sup> Université Sorbonne Paris Nord, Laboratoire des Sciences des Procédés et des Matériaux, LSPM, CNRS, UPR 3407, F-93430, Villetaneuse, France



**NUMFOCUS**  
OPEN CODE = BETTER SCIENCE





Tritium  
contamination



Material  
embrittlement



Recycling  
fluxes



B2

$$\begin{aligned} \partial_t c_m &= \nabla(D(T) \cdot \nabla c_m) - \sum_i \partial_t c_{t,i} \quad \text{on } \Omega \\ \partial_t c_{t,i} &= k(T) \cdot c_m (n_i - c_{t,i}) - p(T) \cdot c_{t,i} \quad \text{on } \Omega \end{aligned} \quad \left. \vphantom{\begin{aligned} \partial_t c_m &= \nabla(D(T) \cdot \nabla c_m) - \sum_i \partial_t c_{t,i} \\ \partial_t c_{t,i} &= k(T) \cdot c_m (n_i - c_{t,i}) - p(T) \cdot c_{t,i} \end{aligned}} \right\} \begin{array}{l} \text{Hydrogen transport} \\ \text{McNabb \& Foster - Trans.} \\ \text{Metall. Soc. (1963)} \end{array}$$

$$\frac{c_m^-}{S(T)^-} = \frac{c_m^+}{S(T)^+} \quad \text{on } \Omega_i \cap \Omega_j \quad \left. \vphantom{\frac{c_m^-}{S(T)^-} = \frac{c_m^+}{S(T)^+}} \right\} \begin{array}{l} \text{Conservation of} \\ \text{chemical potential} \\ \text{at interfaces} \end{array}$$

$$\rho C_p \partial_t T = \nabla(\lambda \cdot \nabla T) + Q \quad \text{on } \Omega \quad \left. \vphantom{\rho C_p \partial_t T = \nabla(\lambda \cdot \nabla T) + Q} \right\} \begin{array}{l} \text{Energy equation} \end{array}$$

- ▶  $c_m, c_{t,i}, T$  H concentrations and temperature
- ▶  $i$  corresponds to a type of sink

## FESTIM

- ▶ Finite Element Simulation of Tritium In Materials
- ▶ Based on FEniCS
- ▶ 1/2/3D
- ▶ Multi-materials



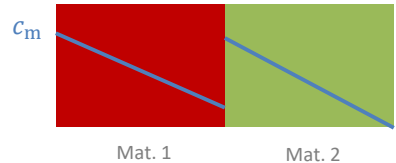
For more info:

Delaporte-Mathurin *et al*, NME (2019)





$$\frac{c_m^-}{S^-} = \frac{c_m^+}{S^+} \quad \text{at interfaces}$$



► Modelling discontinuities in FEniCS

$$\partial_t c_m = \nabla(D \cdot \nabla c_m) - \sum_i \partial_t c_{t,i} \quad \text{on } \Omega$$

$$\theta = c_m / S$$

1. Solve :  $\partial_t(\theta S) = \nabla(D \cdot \nabla(\theta S)) - \sum_i \partial_t c_{t,i} \quad \text{on } \Omega$
2. Post-processing:

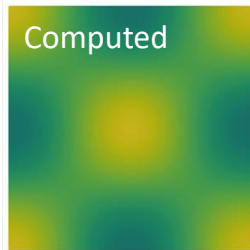
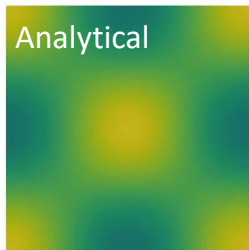
$$c_m = \theta \cdot S$$

project on DG1 space

```
V_DG1 = FunctionSpace(mesh, 'DG', 1)
c_m = project(theta*S, V_DG1)
```

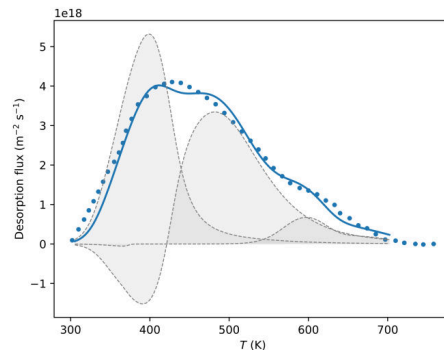
## Verification using MMS

- ▶  $c_{m,D} = 1 + \cos(2\pi x) \cos(2\pi y) + \cos(2\pi t)$
- ▶  $T = 500 + 30 \cos(2\pi x)$
- ▶ Multi-material



## Experimental validation

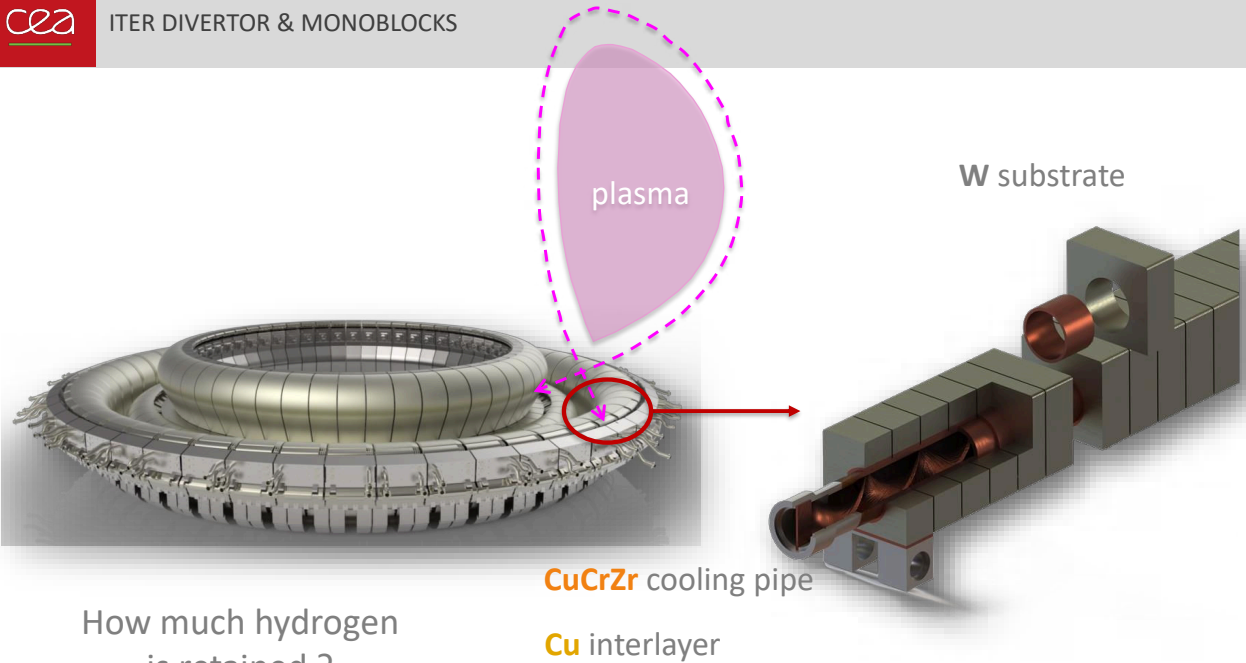
- ▶ Thermo-desorption experiments
- ▶ Parametric optimisation



Delaporte-Mathurin *et al*, NME (2021)

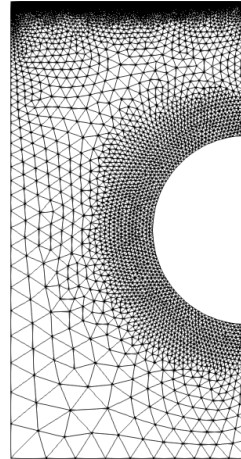
# **Application:**

## Tokamak components



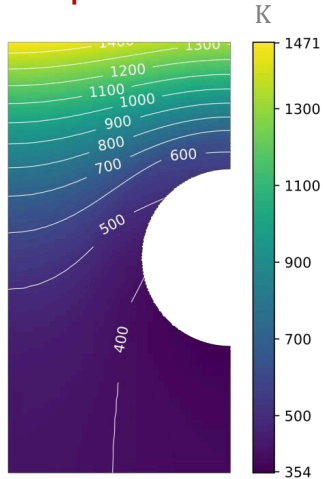
How much hydrogen  
is retained ?

- ▶ Meshed with SALOME (open-source)
- ▶ Converted from .med to .xdmf with meshio [1]
- ▶ High refinement:
  - on the top surface
  - at interfaces
- ▶ Planned: using Adaptive Mesh Refinement

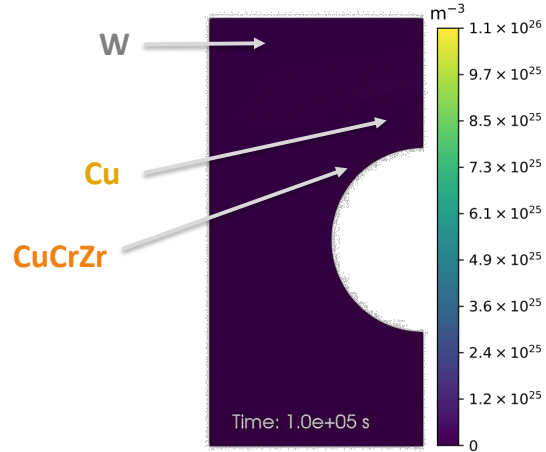


[1] Meshio: [10.5281/zenodo.4590119](https://zenodo.org/record/105281)

## Temperature field



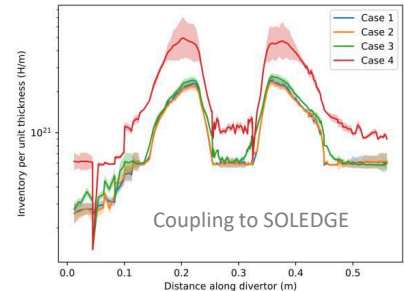
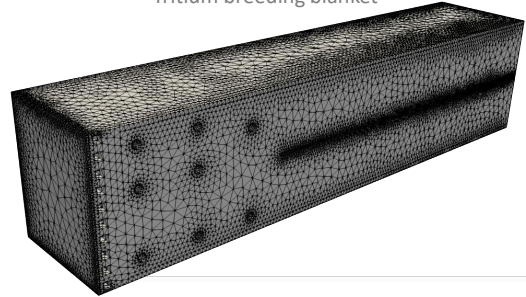
## Hydrogen concentration



- ▶ Total inventory of H :  $\int (c_m + c_t) dx$
- ▶ Coolant contamination :  $\int D(T) \nabla c_m \cdot \mathbf{n} dS$

- ▶ Applications to more complex tokamak components
- ▶ Coupling with other physics
  - CFD
  - MHD,
  - co-deposition models
  - He transport...
- ▶ Coupling with external plasma codes

Tritium breeding blanket





Thank you for your attention

Plots were made with Matplotlib and Paraview



- ▶ **FESTIM** is a FEniCS-based simulation interface
- ▶ Hydrogen transport (including diffusion and trapping) is modelled and coupled to heat transfer.
- ▶ **FESTIM** applications:
  - Simulating fusion reactors components
  - Identifying materials properties
- ▶ **Perspectives:**
  - Applications to more complex tokamak components (tritium breeding blankets)
  - Coupling with other physics (CFD, MHD, co-deposition models, He transport...)
  - Coupling with external plasma codes (SOLEEDGE, SOLPS)

```
import FESTIM

parameters = {
    "materials": [
        {
            "E_D": 0.1,
            "D_0": 1,
            "id": 1
        }
    ],
    "traps": [],
    "mesh_parameters": {
        "initial_number_of_cells": 200,
        "size": 1,
        "refinements": [
        ],
    },
    "boundary_conditions": [
    ],
    "temperature": {
        "type": "expression",
        "value": 300
    },
    "solving_parameters": {
        "final_time": 100,
        "initial_stepsize": 0.1,
        "newton_solver": {
            "absolute_tolerance": 1e-10,
            "relative_tolerance": 1e-9,
            "maximum_iterations": 50,
        }
    },
}

output = FESTIM.run(parameters)
```

```
class UserCoeff(UserExpression):
    def __init__(self, mesh, vm, T, **kwargs):
        super().__init__(kwargs)
        self._mesh = mesh
        self._vm = vm # MeshFunction for volume markers
        self._T = T

    def eval_cell(self, value, x, ufc_cell):
        cell = Cell(self._mesh, ufc_cell.index)
        subdomain_id = self._vm[cell]
        if subdomain_id == 1:
            value[0] = self._T(x)
        else:
            value[0] = 2

    def value_shape(self):
        return ()

S = UserCoeff(mesh, vm, T)

V_DG1 = FunctionSpace(mesh, 'DG', 1)
c_m = project(theta*S, V_DG1)
```

# Interfacing AceGEN and FEniCS for advanced constitutive models

**Jakub Lengiewicz**, University of Luxembourg, Luxembourg

Michal Habera, University of Luxembourg, Luxembourg

Andreas Zilian, University of Luxembourg, Luxembourg

Stephane Bordas, University of Luxembourg, Luxembourg

**25 March 2021**

There are two main difficulties related to FE analysis: (i) the necessity to provide effective FE procedures for complex constitutive models or advanced FE formulations, and (ii) the need to solve big problems. Up to date, no single environment exists to fully tackle both. In the present contribution we combine two existing systems: AceGen and FEniCS. AceGen is an advanced automatic differentiation (AD) system, capable to derive highly efficient low-level computer code for characteristic FE quantities (gradients, tangent operators, sensitivity vectors, etc) for complex FE formulations. FEniCS is a flexible open-source FE framework, designed to perform large-scale analyses on high-performance computer architectures.

The proposed hybrid approach relies on new developments of FEniCSx project, which allows to incorporate external FE code to the environment. As examples, we have generated FE routines for two representative FE formulations: hyperelasticity and non-linear elasto-plasticity with hardening. The former formulation can be also generated by the FEniCS Form Compiler (FFC), thus allows to quantitatively show the runtime speedup of the FE procedures generated by AceGEN w.r.t. FEniCS/FFC. The latter formulation shows the dominance of the hybrid approach over AceGen/AceFem or FEniCS alone, which is the ability to solve complex FE formulations within the HPC-ready framework.

---

You can cite this talk as:

Jakub Lengiewicz, Michal Habera, Andreas Zilian, and Stephane Bordas. “Interfacing AceGEN and FEniCS for advanced constitutive models”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 474. DOI: 10.6084/m9.figshare.14495463.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/lengiewicz.html>.

# FEniCS-preCICE: Coupling FEniCS to other simulation software

**Ishaan Desai** (<https://www.ipvs.uni-stuttgart.de/institute/team/Desai>), University of Stuttgart, Germany

**Benjamin Rodenberg** (<https://www.in.tum.de/i05/personen/personen/benjamin-rodenberg>), Technical University of Munich, Germany

**Richard Hertrich**, Technical University of Munich, Germany

**Alexander Jaust** (<https://www.ipvs.uni-stuttgart.de/institute/team/Jaust-00001>), University of Stuttgart, Germany

**Benjamin Uekermann** (<https://www.ipvs.uni-stuttgart.de/institute/team/Uekermann-00001>), University of Stuttgart, Germany

**25 March 2021**

FEniCS-preCICE facilitates coupling of FEniCS with other software codes using the coupling library preCICE. preCICE enables users to couple different solvers in a partitioned black-box fashion. This talk explains various features of preCICE and its usage. Only a few additional lines of code are necessary to prepare a FEniCS program script for coupling.

FEniCS-preCICE acts as a middle software layer which helps to connect a high-level FEniCS program to the low-level API of preCICE. The package converts between FEniCS and preCICE data structures, provides easy-to-use coupling conditions, and manages checkpointing for implicit coupling. FEniCS-preCICE is able to handle distributed memory parallelization of FEniCS internally. This package is a library itself and follows a FEniCS-native style.

The functionality of FEniCS-preCICE is illustrated by two examples of coupled problems in fluid-structure interaction: a FEniCS heat conduction program coupled to OpenFOAM and a FEniCS elastic solid mechanics program coupled to SU2. In both examples FEniCS is used to solve the solid part of the fluid-solid coupling. The results of both examples are compared with other simulation software showing good agreement. preCICE and FEniCS-preCICE are available under open-source licenses on GitHub.

## References

- [1] Benjamin Rodenberg, Ishaan Desai, Richard Hertrich, Alexander Jaust, and Benjamin Uekermann. “FEniCS-preCICE: Coupling FEniCS to other simulation software”. <https://arxiv.org/abs/2103.11191>.

---

You can cite this talk as:

Ishaan Desai, Benjamin Rodenberg, Richard Hertrich, Alexander Jaust, and Benjamin Uekermann. “FEniCS-preCICE: Coupling FEniCS to other simulation software”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 475–486. doi: 10.6084/m9.figshare.14495469.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/desai.html>.

# FEniCS-preCICE: Coupling FEniCS to other Simulation Software

Ishaan Desai<sup>a</sup>, Benjamin Rodenberg<sup>b</sup>, Richard Hertrich<sup>b</sup>, Alexander Jaust<sup>c</sup>, Benjamin Uekermann<sup>a</sup>

<sup>a</sup>Usability and Sustainability of Simulation Software, Institute for Parallel and Distributed Systems, University of Stuttgart

<sup>b</sup>Scientific Computing in Computer Science, Department of Informatics, Technical University of Munich

<sup>c</sup>Simulation of Large Systems, Institute for Parallel and Distributed Systems, University of Stuttgart



(a) <https://fenicsproject.org/>

# Contents

Introduction to preCICE

FEniCS-preCICE: A preCICE Adapter for FEniCS

Examples of Coupled Problems with FEniCS

Getting FEniCS-preCICE

# preCICE - A Flexible Coupling Library

What is preCICE used for?

Coupling solvers for multi-physics simulations in a partitioned black-box fashion

## Working principles of preCICE

Partitioned approach, black-box coupling, massively parallel, highly flexible, library approach

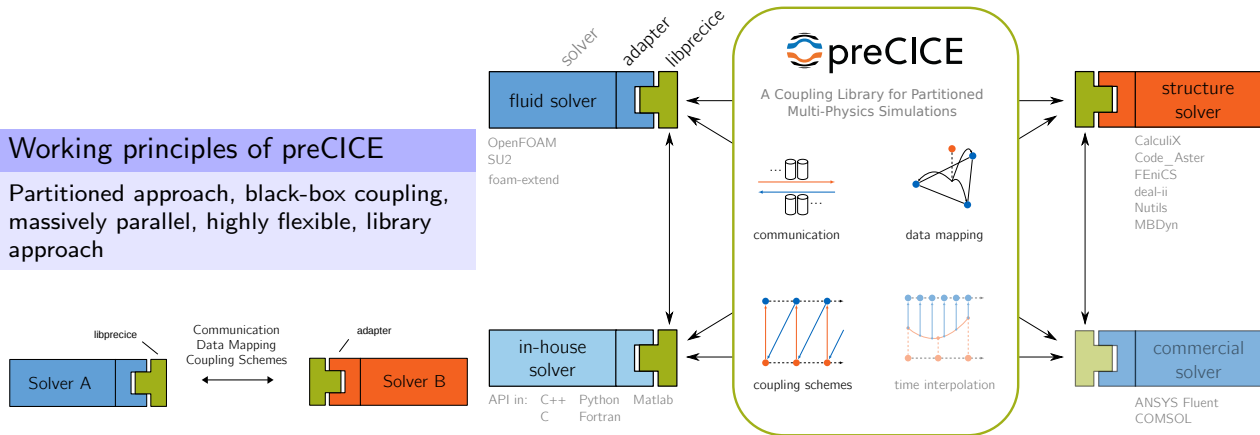


Figure: Overview of preCICE features



## Goal of FEniCS-preCICE

To provide a helper package to a FEniCS user to facilitate easy use of preCICE to setup a coupled problem

### Design Principles:

- To have a middle layer between high-level FEniCS program and low-level C++ preCICE API
- Use python-bindings of preCICE to access C++ API
- To have a highly modular structure which is easy to understand and modify in future
- To handle as many boilerplate tasks as possible inside the adapter

### Features of Adapter:

- Adapter supports 2D cases in FEniCS. Users can define boundary conditions using FEniCS `Expression` or `PointSource`
- Adapter needs to be configured with a JSON file

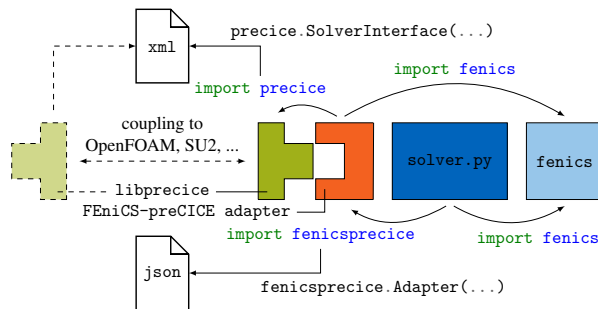


Figure: Functioning of FEniCS-preCICE Adapter

# API Functions of FEniCS-preCICE

- Adapter receives configuration information from JSON file
- Adapter retrieves mesh data from user defined FEniCS `FunctionSpace` and the coupling boundary `SubDomain`
- Converting data from FEniCS format to preCICE format and visa-versa is handled internally in the functions `read_data()` and `write_data()`
- Data at coupling boundary can be of the form of a FEniCS `Expression` or FEniCS `PointSource`
- Distributed parallelization in FEniCS is handled out of the box
- Checkpointing functionality for implicit coupling

```
def __init__(self, adapter_config_filename='precice-adapter-config.json'):
    self._interface = precice.Interface(...)

def initialize(self, coupling_subdomain, read_function_space=None,
    ↪ write_object=None):
    precice_dt = self._interface.initialize()

def read_data(self):
    return data

def write_data(self, write_function):

def create_coupling_expression(self):
    return CouplingExpression(...)

def update_coupling_expression(self, coupling_expression, data):
    coupling_expression.update_boundary_data(nodal_data, x_coordinates,
    ↪ y_coordinates)

def get_point_sources(self, data):
    return x_PointSources, y_PointSources

def store_checkpoint(self, user_u, t, n):
    self._checkpoint = SolverState(user_u.copy(), t, n)
    self._interface.mark_action_fulfilled(
    ↪ precice.action_write_iteration_checkpoint() )

def retrieve_checkpoint(self):
    self._interface.mark_action_fulfilled(
    ↪ precice.action_read_iteration_checkpoint() )
    return self._checkpoint.get_state()
```

## Modifying a FEniCS Program to couple using FEniCS-preCICE

```
from fenics import *

mesh = UnitSquareMesh(10, 10)
class Boundary(SubDomain): ...

V = V_bc = FunctionSpace(mesh, 'P', 2)
u, v = TrialFunction(V), TestFunction(V)
u_D = Expression('...', degree=2)
uncoupled_bc = DirichletBC(V_bc, u_D, Boundary)

# Define initial condition and weak form in FEniCS
...

for t in np.arange(0,T,dT):
    solve(lhs(F) == rhs(F), u, [uncoupled_bc])
```

# Modifying a FEniCS Program to couple using FEniCS-preCICE

```
from fenics import *

mesh = UnitSquareMesh(10, 10)
class Boundary(SubDomain): ...

V = V_bc = FunctionSpace(mesh, 'P', 2)
u, v = TrialFunction(V), TestFunction(V)

u_D = Expression('...', degree=2)
uncoupled_bc = DirichletBC(V_bc, u_D, Boundary)

# Define initial condition and weak form in FEniCS
...

for t in np.arange(0, T, dt):

    solve(lhs(F) == rhs(F), u, [uncoupled_bc])
```

```
from fenics import *
from fenicsprecice import Adapter

mesh = UnitSquareMesh(10, 10)
class Boundary(SubDomain): ...
class CouplingBoundary(SubDomain):

    V = V_bc = FunctionSpace(mesh, 'P', 2)
    u, v = TrialFunction(V), TestFunction(V)
    V_flux = VectorFunctionSpace(mesh, 'P', 1)
    u_D = Expression('...', degree=2)
    uncoupled_bc = DirichletBC(V_bc, u_D, Boundary)

    adapter = Adapter("precice-adapter-config.json")
    precice_dt = adapter.initialize(CouplingBoundary, read_function_space=V_bc,
    ↪ write_object=V_flux)
    u_C = adapter.create_coupling_expression()
    coupled_bc = DirichletBC(V_bc, u_C, CouplingBoundary)

    # Define initial condition and weak form in FEniCS
    ...

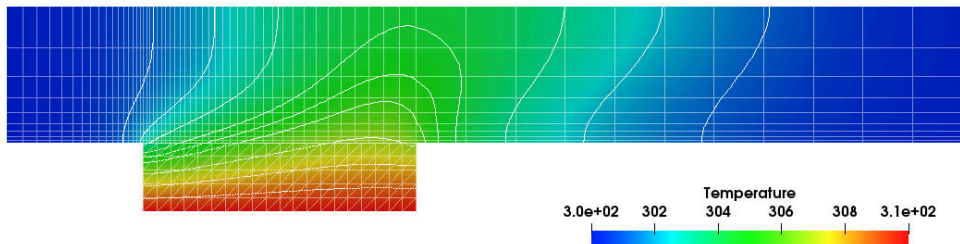
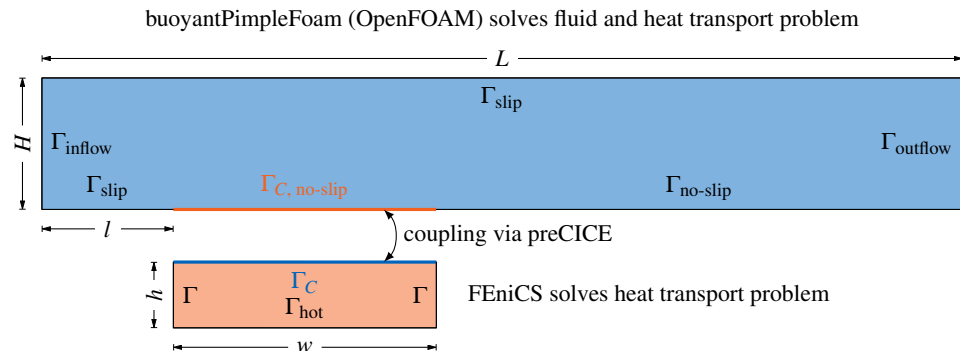
    while adapter.is_coupling_ongoing():
        read_data = adapter.read_data()
        adapter.update_coupling_expression(u_C, read_data)
        dt_assign = np.min([fenics_dt, precice_dt])

        solve(lhs(F) == rhs(F), u_np1, [uncoupled_bc, coupled_bc])
        flux = some_postprocessing(u_np1, V_flux)

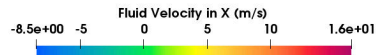
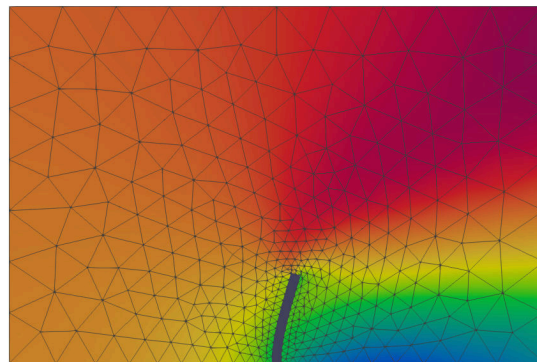
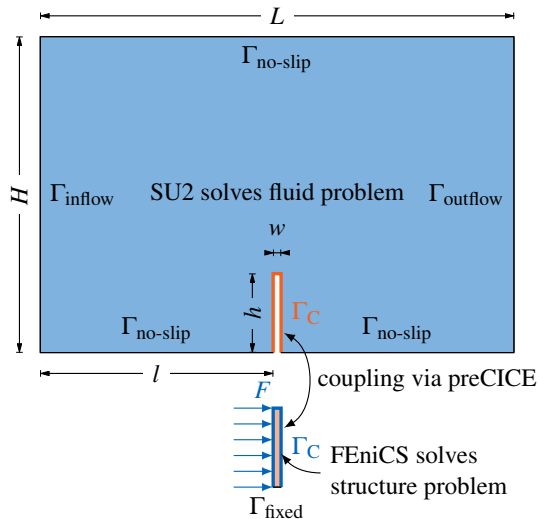
        adapter.write_data(flux)
        precice_dt = adapter.advance(dt_assign)

    u_n.assign(u_np1)
    t += float(dt)
```

## Example Case: Conjugate Heat Transfer Coupling with FEniCS and OpenFOAM



## Example Case: Fluid-Structure Interaction Coupling with FEniCS and SU2



# Getting FEniCS-preCICE and its Dependencies

## Getting the Adapter:

- Maintained under a open-source license here: <https://github.com/precice/fenics-adapter>
- Easy to install: `pip3 install fenicsprecice`
- Latest release can be found here: <https://github.com/precice/fenics-adapter/releases>
- Other dependencies such as Scipy, Numpy, Cython, mpi4py are installed automatically during the adapter installation

## Dependencies

- Python (`python3`)
- preCICE (obviously)
- FEniCS
- Python-bindings for preCICE: `pip3 install --user pyprecice`

## Summary

- preCICE is a coupling library for partitioned, black-box coupling. Designed for highly flexible and massively parallel use
- FEniCS-preCICE is an adapter to couple FEniCS programs with other software codes using preCICE
- Adapter supports 2D FEniCS cases
- Adapter handles FEniCS data structures and distributed parallelization automatically
- Adapter is modular and flexible to use
- Installation is straightforward using `pip`

Always happy with contributions from the community!

Immediate help required:

- Extending adapter to handle 3D FEniCS cases
- Implementing multiple coupling interfaces handling
- Modify adapter to support FENICS-X and DOLFIN-X

Adding tutorials of coupled problems which use FEniCS

Research collaboration with the preCICE team

Pre-print of reference paper: *Benjamin Rodenberg, Ishaan Desai, Richard Hertrich, Alexander Jaust, and Benjamin Uekermann. FEniCS-preCICE: Coupling FEniCS to other Simulation Software: <https://arxiv.org/abs/2103.11191>*



# Motion of synthetic microswimmers at low Reynolds numbers with FEniCS

**Roberto Ausas** (<http://www.lmacc.icmc.usp.br/~ausas>), Institute of Mathematics and Computer Sciences, University of São Paulo, Brazil

Stevens Paz, Universidad del Valle, Colombia

Paula Alvez da Silva, Institute of Mathematics and Computer Sciences, University of São Paulo, Brazil

Gustavo Buscaglia (<http://www.lcad.icmc.usp.br/~buscaglia>), Institute of Mathematics and Computer Sciences, University of São Paulo, Brazil

25 March 2021

Swim propulsion strategies that work well in the macroscopic world turn out to be ineffective at the microscopic scale due to the dominance of viscous forces at very low Reynolds numbers. Understanding the locomotion strategies that can be adopted at the microscopic level by living microorganisms and synthetic microswimmers is of fundamental importance in biology and in biomedical applications. This work deals with the swimming of microscopic bodies in a highly viscous ambient fluid. We solve this fluid-solid interaction problem by means of a fully implicit formulation in which fluid unknowns (velocity and pressure) and positional degrees of freedom of the body are obtained simultaneously. This is accomplished by suitably constructing the space of kinematically admissible fluid-solid motions. A stabilized equal order formulation is adopted for the fluid part and linear triangular elements are used to approximate the swimmer geometry. Lagrangian update of the moving boundaries is performed either by a non-reversible Euler method or by a reversible mid-point scheme. In the latter case, a fixed-point iterative strategy is used to obtain the solution. We present a FEniCS-based finite element implementation of this fluid-swimmer interaction problem. Convergence of the proposed methodology is numerically assessed. Several numerical and implementation details are provided along challenging 2D- and 3D-axisymmetric examples, considering Newtonian and non-Newtonian rheologies.

---

You can cite this talk as:

Roberto Ausas, Stevens Paz, Paula Alvez da Silva, and Gustavo Buscaglia. "Motion of synthetic microswimmers at low Reynolds numbers with FEniCS". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 487. DOI: 10.6084/m9.figshare.14495475.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/ausas.html>.

# Development of an open-source-based framework for multiphysical crystal growth simulations

Arved Enders-Seidlitz, Leibniz Institute for Crystal Growth, Germany

Josef Pal, Leibniz Institute for Crystal Growth, Germany

Kaspars Dadzis, Leibniz Institute for Crystal Growth, Germany

25 March 2021

The NEMOCRYS project in the group “Model experiments” at the IKZ funded by an ERC Starting Grant aims at profoundly validated numerical models for crystal growth. These processes involve a variety of coupled physical phenomena such as heat transfer including radiation and phase change, electromagnetism, melt- and gas flows and thermal stresses. Numerous simulation studies (using eg Comsol, Ansys or OpenFOAM) have been published, however, their applicability remains limited: The validation is mostly insufficient due to missing in-situ measurements, and the models are either implemented in expensive closed-source software or not published at all.

Therefore, a new open-source-based framework for multiphysics simulation in crystal growth is under development. It currently uses Gmsh for FEM mesh generation and Elmer to solve the heat transfer problem, which are wrapped in a python interface. A major challenge in the current implementation is the coupling between Elmer and Gmsh: The transient simulation involves a moving crystal and phase boundary, and thus the mesh needs to be updated. FEniCS is a promising tool providing additional flexibility to implement new models with more advanced coupling algorithms. For example, a dynamic simulation with varying crystal diameter could include heat transfer with phase change and electromagnetic heat induction in FEniCS. External coupling with finite volume libraries such as OpenFOAM could be applied for melt and gas flow calculations.

---

You can cite this talk as:

Arved Enders-Seidlitz, Josef Pal, and Kaspars Dadzis. “Development of an open-source-based framework for multiphysical crystal growth simulations”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 488–501. DOI: 10.6084/m9.figshare.14495487.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/enders-seidlitz.html>.

# Development of an open-source-based framework for multiphysical crystal growth simulations

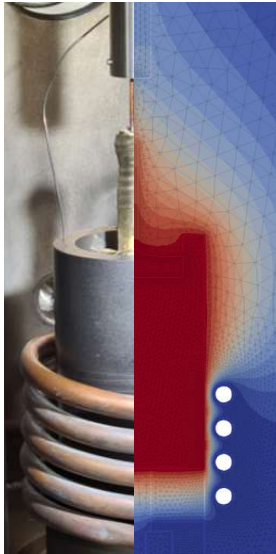
*FEniCS 2021 Conference*

University of Cambridge / online, March 25, 2021

**Arved Enders-Seidlitz**

Josef Pal

Kaspars Dadzis



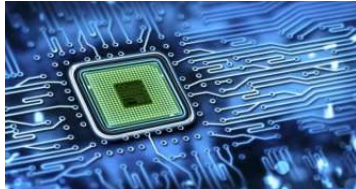
Motivation

Simulation concept

Transient heat transfer simulation

Possible integration of FEniCS

Conclusion



<http://www.knoda.org/back-history-discovery-very-first-silicon-chip-digital-computers/>



<https://cen.acs.org/energy/solar-power/Supercharging-silicon-solar-cell/97/web/2019/07>

**Computer technology,  
solar energy**



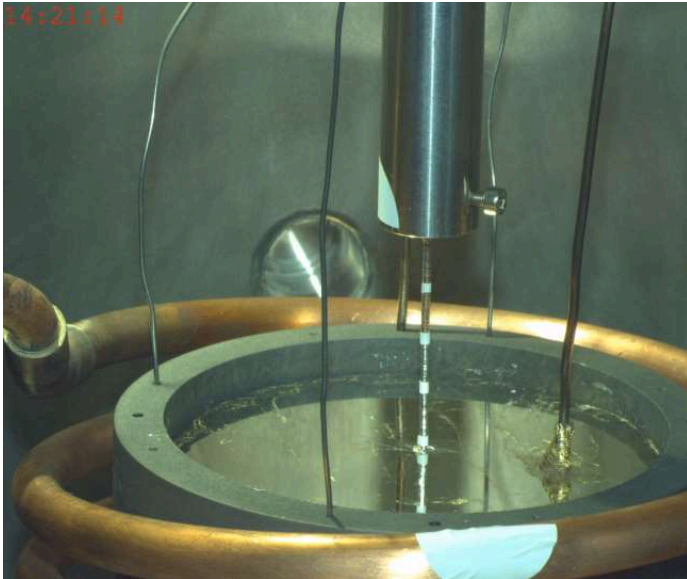
<https://www.sciencedirect.com/topics/chemistry/czochralski-process>

**Silicon single crystal**



<https://www.pvatepla-cgs.com/anlagen/czochralski/>

**Czochralski growth  
furnace**



## Model experiments

Simplified geometry and material

### Materials

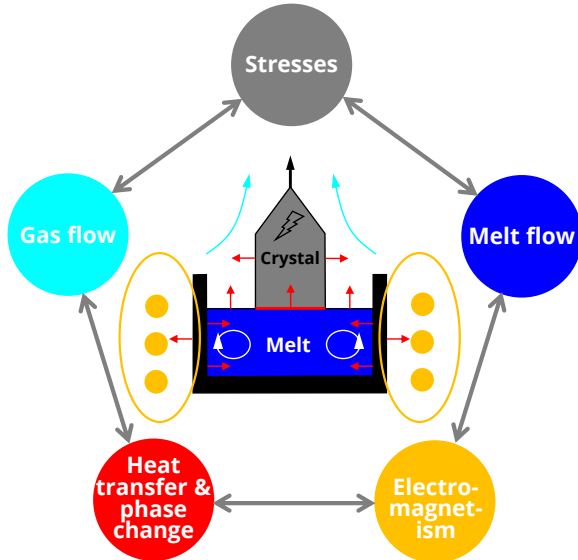
- Tin,  $T_{melt} = 232^{\circ}\text{C}$
- Bismuth,  $\text{NaNO}_3$ , ...

### Conditions

- Air atmosphere
- Vacuum

### Measurements

- Temperatures
  - Thermocouples, Pt100
  - IR Camera
  - Pyrometer
- Electromagnetism
  - Heating power
  - Magnetic field
- Flows, thermal stresses

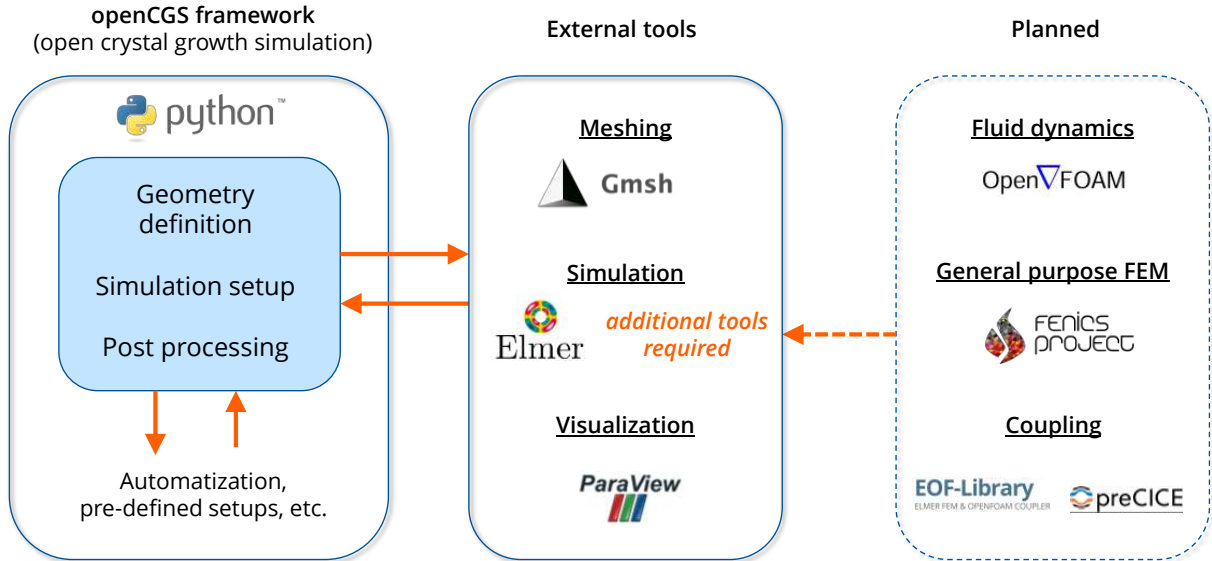


## Numerical challenges

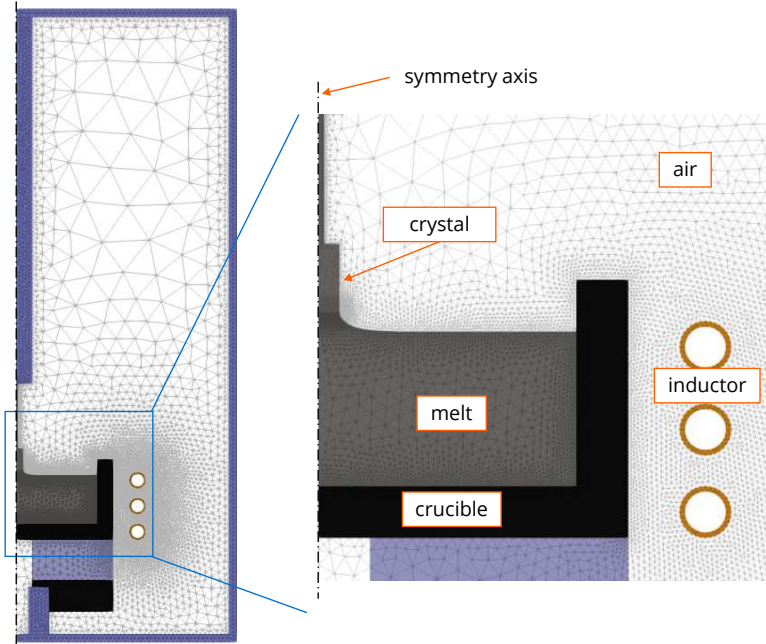
- Complex coupled physics
- Moving geometries
- Different timescales
- ...

## Goals in NEMOCRYS Project

- Validation: Using model experiments
- Open source implementation







## 2D axisymmetric with Elmer

Induction heating (harmonic)

$$\nabla \times \left( \frac{1}{\mu} \nabla \times A_{\varphi} \mathbf{e}_{\varphi} \right) + i\omega\sigma A_{\varphi} \mathbf{e}_{\varphi} = j_{\varphi}$$

Heat transfer

$$\rho c_p \left( \frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla) T \right) - \nabla \cdot (\lambda \nabla T) = \rho h$$

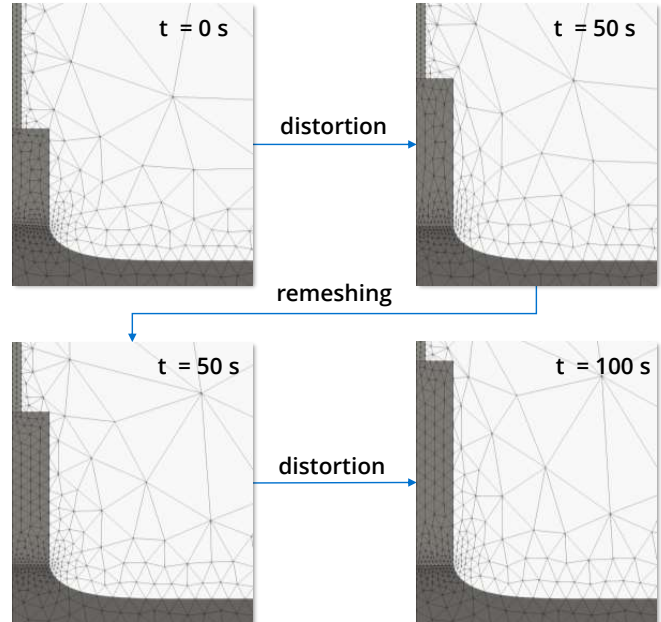
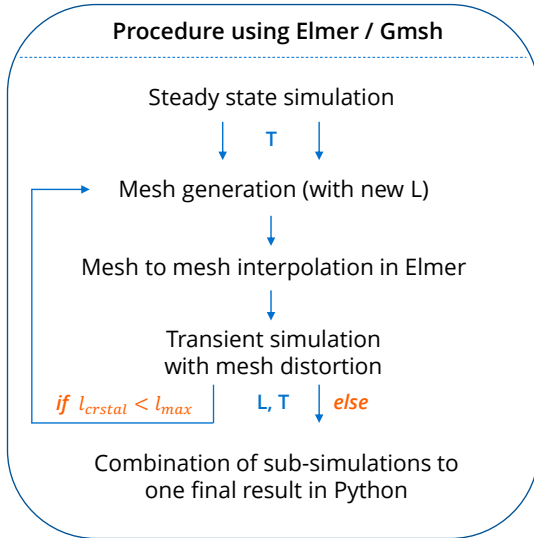
Phase change (steady-state approximation)

$$q = L \rho \mathbf{v} \cdot \mathbf{n}, \quad s_y = (y_{j,1} - y_i) + (x_i - x_{j,1}) \frac{y_{j,2} - y_{i,1}}{x_{j,2} - x_{j,1}}$$

Radiation (at solid/air boundaries)

$$-\lambda_k \frac{\partial T_k}{\partial \mathbf{n}_k} = \sigma_{\varepsilon} \varepsilon_k \left( T_k^4 - \frac{1}{A_k \varepsilon_k} \sum_{i=1}^N G_{ik} \varepsilon_i T_i^4 A_i \right)$$

P. Råback et al.: Elmer Models Manual, CSC – IT Center for Science, 10.11.2020. <https://www.nic.funet.fi/pub/sci/physics/elmer/doc/>



## Mesh update loop in openCGS (simplified)

```
sim = SteadyStateSim(geometry,
                     simulation_setup,
                     start_length)

sim.execute()

while start_length < max_length:
    sim = TransientSim(geometry,
                      simulation_setup,
                      start_length,
                      length_increment,
                      sim)

    sim.execute()
    start_length += length_increment
```

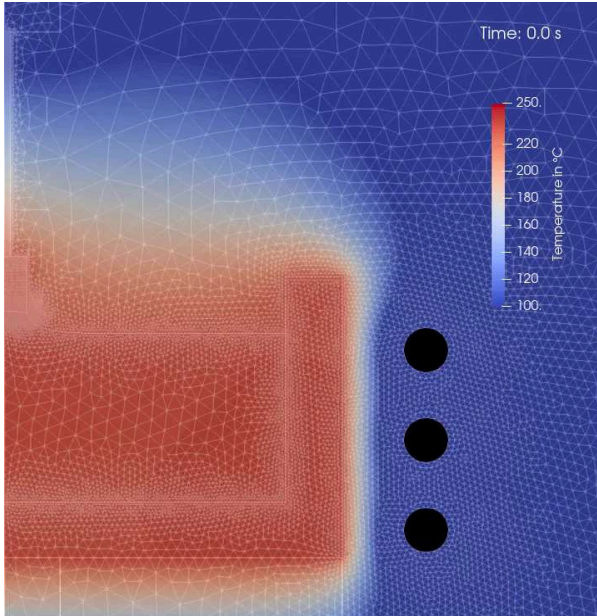
## User input: Two functions (simplified)

```
from opencgs import geo, setup

def geometry(crystal_length):
    geo.crystal(crystal_length, ...)
    geo.crucible(...)
    geo.melt(...)

    ... # boundaries, mesh sizes

def simulation_setup(...):
    setup.add_crystal(...)
    ... # bodies, boundaries
```



## Numerical

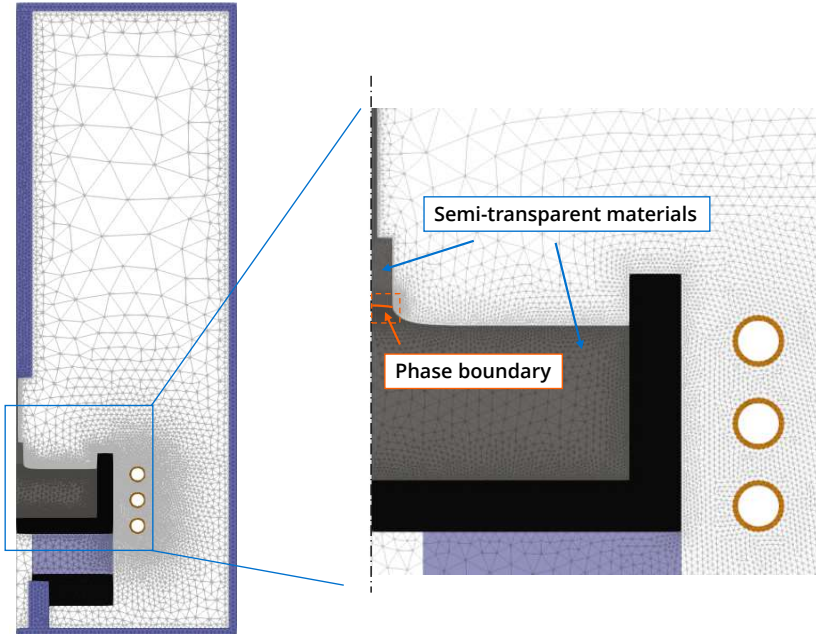
- Simulation numerically stable
- No visible errors introduced by mesh update

## Physical

- Increase in temperature with crystal length, corresponds to experiment
- Validation ongoing: Convective cooling of crystal, etc.

## Future challenges

- Variable crystal diameters
- New models required



## Need for advanced models

- Phase boundary modeling
  - Growth in axial and radial direction
  - Interaction with process control
- Semi-transparent materials
  - Internal radiation
  - Internal absorption

## Possible implementations

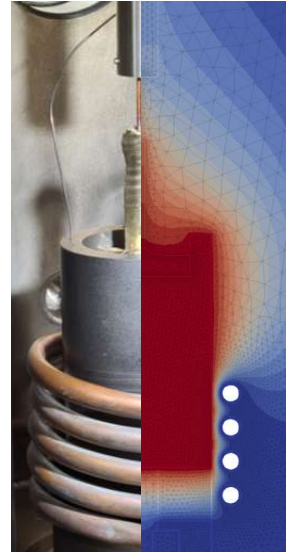
- Coupling to Elmer (preCICE)
- Complete solver in FEniCS –  
radiation model required!

## Transient thermal CZ growth simulation implemented

- Python-based framework using Elmer and Gmsh
- Limited to constant crystal diameters

## Possible integration of FEniCS

- Coupling to Elmer using preCICE: **Under development**
- Complete solver in FEniCS: **Radiation model required**



**Thank you for your attention!**

We'd like to acknowledge Peter Råback for his support regarding the usage of Elmer.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 851768).



# Coupling of thermomechanics with electromagnetism in FEniCS

**Bilen Emek Abali** (<http://bilenemek.abali.org>), Uppsala University, Sweden

**25 March 2021**

In thermomechanics, we utilize balance equations and compute temperature as well as deformation of a continuum body. The theory is well established. In electromagnetism, we use Maxwell's equations for calculating electric field and magnetic flux. As we want to combine all of these fields, there are theoretical and numerical challenges. This talk briefly introduces challenges and addresses some possible solutions for computing electro-magneto-thermo-mechanical systems in FEniCS.

---

You can cite this talk as:

Bilen Emek Abali. "Coupling of thermomechanics with electromagnetism in FEniCS". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 502–517. DOI: 10.6084/m9.figshare.14495493.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/abali.html>.



# Coupling of thermomechanics with electromagnetism in FEniCS

**Bilen Emek Abali**

Associate Professor in Solid Mechanics  
Department of Materials Science and Engineering  
Uppsala University

March 25, 2021, Cambridge



# Thermomechanics and electromagnetism

## Challenges in theory and implementation

- ▶ Coupling of electromagnetism and thermomechanics, ABRAHAM–MINKOWSKI debate
- ▶ Thermodynamically sound derivation of all constitutive equations using MINKOWSKI momentum
- ▶ Balances of mass, momentum, energy, electric charge, and FARADAY law, jump conditions

## Thermomechanics and electromagnetism

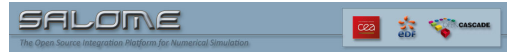
### Challenges in theory and implementation

- ▶ Coupling of electromagnetism and thermomechanics, ABRAHAM–MINKOWSKI debate
- ▶ Thermodynamically sound derivation of all constitutive equations using MINKOWSKI momentum
- ▶ Balances of mass, momentum, energy, electric charge, and FARADAY law, jump conditions
- ▶ Numerical method depends on the chosen gauge conditions
- ▶ Jump conditions to be implemented as terms in the variational formulation rather than element formulation
- ▶ Monolithic computation by using LORENZ gauge and jump conditions

## Implementation

Solving the weak form by using open-source packages:

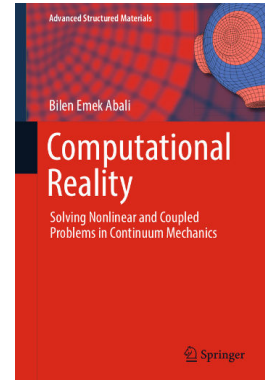
- ▶ CAD in Salome
- ▶ Mesh via NetGen in Salome
- ▶ Code in Python
- ▶ Assembly, linearization, solving via FEM in space and FDM in time by FEniCS
- ▶ Visualization in ParaView



## Implementation

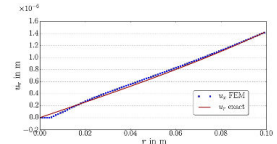
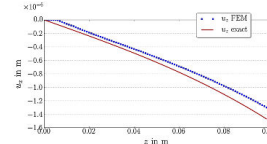
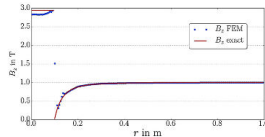
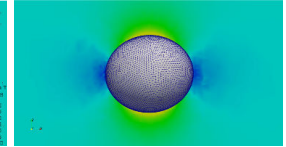
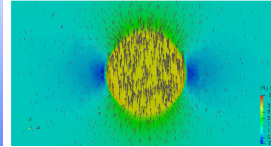
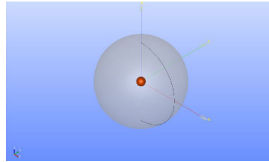
Simulation of multiphysics applications, FEM in space, FDM in time

- ▶ Elastostatics
- ▶ Nonlinear elasticity
- ▶ Plasticity
- ▶ Linear and nonlinear fluid dynamics
- ▶ Fluid-structure interaction
- ▶ Thermomechanics
- ▶ Electromagnetism
- ▶ Thermoelectric coupling
- ▶ Piezoelectricity
- ▶ Magnetohydrodynamics



## Verification of the method

- Thermodynamically sound derivation of all constitutive equations in electromagnetism and thermomechanics
- Computation of displacement,  $\mathbf{u}$ , and magnetic potential,  $\mathbf{A}$ , such that magnetic flux,  $\mathbf{B}$
- Analytical solution for verifying the novel numerical implementation using LORENZ gauge and jump conditions

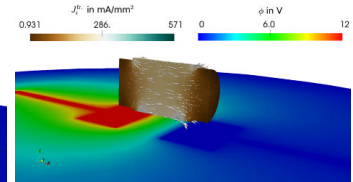
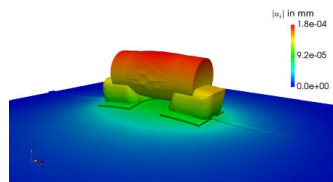
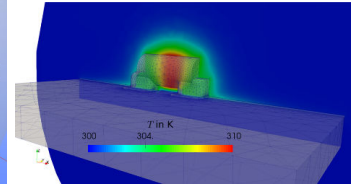
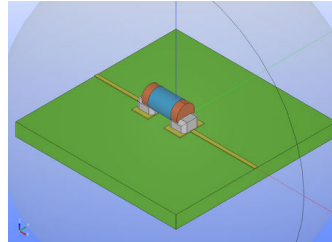


BEA and F. A. Reich. Continuum Mechanics and Thermodynamics 32.3 (2020), pp. 693–708.

BEA and F. A. Reich. Computer Methods in Applied Mechanics and Engineering 319 (2017), pp. 567–595.

## Multiphysics in electronics, transistor on a board

- Coupled constitutive equations in electromagnetism and thermomechanics
- Monolithic computation of displacement,  $\mathbf{u}$ , temperature,  $T$ , electric potential,  $\phi$ , magnetic potential,  $\mathbf{A}$
- Realistic Mini-MELF geometry and comparison to reduced order models



BEA and T. I. Zohdi. Journal of Computational Electronics 17.2 (2018), pp. 625–636.

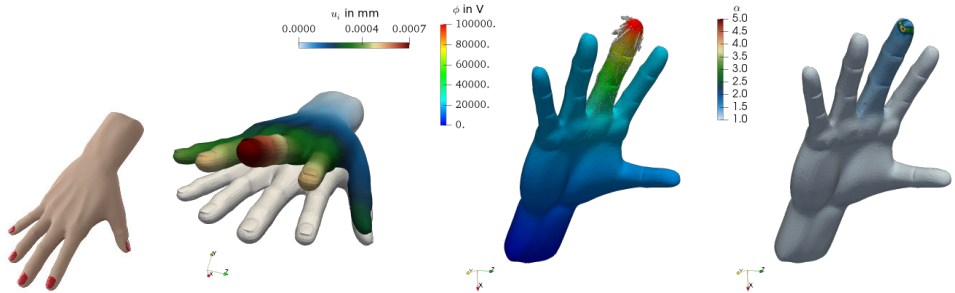
## Thermal damage in lightning



BEA and T. I. Zohdi. Computational Mechanics 65.1 (2020), pp. 149–158.



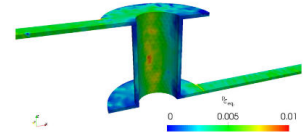
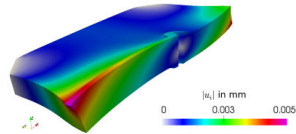
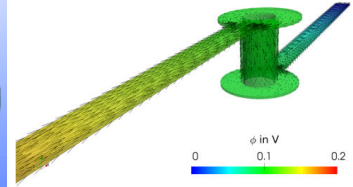
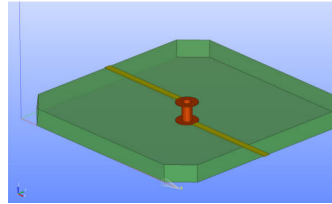
## Thermal damage in lightning



BEA and T. I. Zohdi. Computational Mechanics 65.1 (2020), pp. 149–158.

## Lifetime estimation in fatigue crack growth

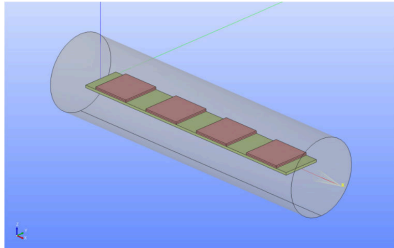
- Plasticity, thermodynamics, and electromagnetism by using experimentally determined material parameters
- Experimental validation of results
- COFFIN–MANSON type fatigue related damage by using accumulated plastic strain



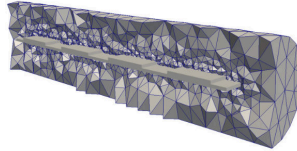
**BEA**, W. H. Müller, H. Walter, O. Wittler, and M. Schneider-Ramelow. GMM-Fachbericht, DVS 340 (2018), pp. 174–179.

**BEA**. Mechanics of Advanced Materials and Modern Processes 3.1 (2017), pp. 1–11

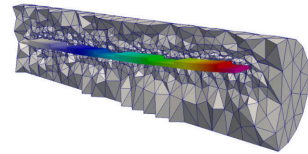
## Piezoceramic fan under large displacement



$|u_i|$  in m  
0.006  
0.003  
0

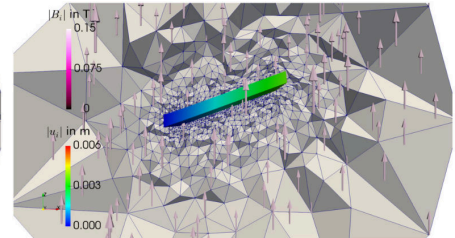
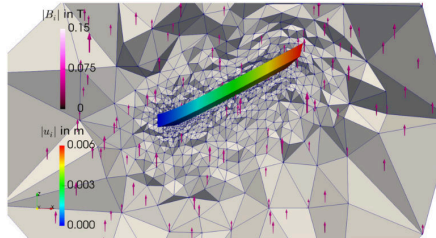
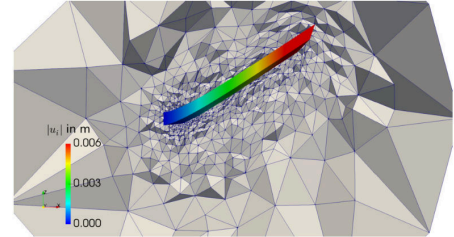
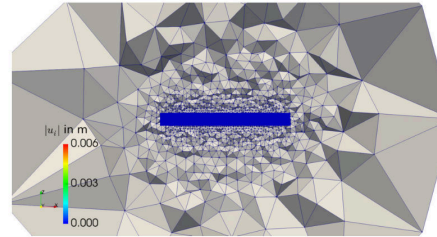
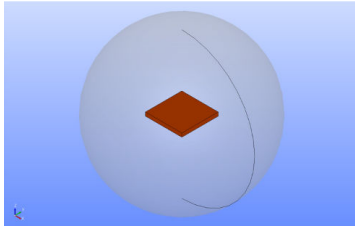


$|u_i|$  in m  
0.006  
0.003  
0



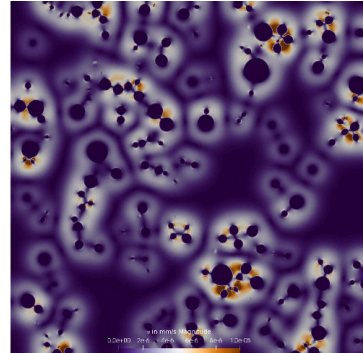
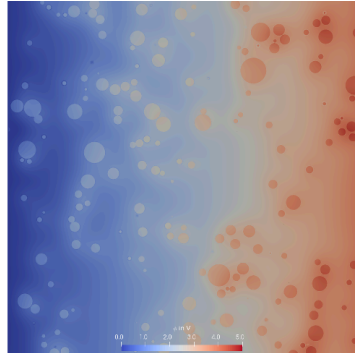
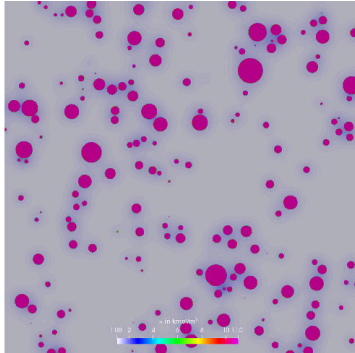
BEA and A. F. Queiruga. Journal of Computational Physics 394 (2019), pp. 200–231.

# Magnetorheological elastomer transducer



BEA and A. F. Queiruga. Journal of Computational Physics 394 (2019), pp. 200–231.

## Multiphysics in batteries, microscale computations



**BEA.** “Modeling mechanochemistry in Li-ion batteries”. In: Scientific Computing in Electrical Engineering. Ed. by G. Nicosia and V. Romano. Vol. 32. Mathematics in Industry. Springer Nature, Cham, 2020. Chap. 8, pp. 79–91.

## What computations we can do?

- ▶ Solving coupled and nonlinear partial differential equations
- ▶ Thermomechanics and electromagnetism in solids and mixtures
- ▶ Reversible phenomena
  - ▶ Piezoelectricity
  - ▶ Pyroelectricity
  - ▶ Magnetothermal coupling
  - ▶ Electromagnetic coupling
- ▶ Irreversible phenomena
  - ▶ Thermoelectric coupling (Peltier elements)
  - ▶ Plasticity and damage

*Thanks a lot!*

<http://bilenemek.abali.org>



# Implementation of a nonlinear anisotropic image denoising model in FEniCS

**Abderrazzak Boufala**, LISTI lab-ENSA, FSJES AM, Ibn Zohr University, Morocco

El Mostafa Kalmoun, Department of Mathematics, Statistics and Physics, College of Arts and Sciences, Qatar University, Qatar

25 March 2021

In this talk, we present a numerical implementation of the following nonlinear anisotropic diffusion-based image denoising model, using the computing platform FEniCS Project:

$$u - u_0 = \frac{1}{2\lambda} \operatorname{div} \left( \frac{1}{(\epsilon^2 + |\nabla u_0|^2)^{1-p/2}} \nabla u \right) \quad \text{in } \Omega,$$
$$\partial_n u = 0 \quad \text{on } \partial\Omega.$$

$u = u(x, y)$  denotes the unknown image to be recovered,  $u_0$  is the observed noisy image,  $\Omega \subset \mathbb{R}^2$  is the spatial image domain and  $\partial_n u$  denotes the derivative of  $u$  in the direction normal to the boundary  $\partial\Omega$ .

We also compare the numerical results with those obtained using finite difference method.

---

You can cite this talk as:

Abderrazzak Boufala and El Mostafa Kalmoun. "Implementation of a nonlinear anisotropic image denoising model in FEniCS". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 518. DOI: 10.6084/m9.figshare.14495499.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/boufala.html>.



# Bistatic polarimetric through-wall SAR public release data set

**Daniel Andre** (<https://www.cranfield.ac.uk/people/dr-daniel-andre-496015>),  
Cranfield University, United Kingdom

Richard Sabbiers, Cranfield University, United Kingdom

**25 March 2021**

This talk presents details of a public release measured radar through-wall dataset suitable for validation of electromagnetic propagation and scattering predictions and for research in scene reconstruction and three-dimensional Synthetic Aperture Radar (SAR) image formation. The collection was fully polarimetric, in a frequency range of 1-4 GHz, employing both monostatic and bistatic two-dimensional synthetic aperture scanning geometries. The scene comprised a concrete walled structure approximately 3 m square and 1.3 m high, containing and surrounded by various metal targets. The Dstl funded experiments were performed at Cranfield University's Ground-Based SAR laboratory in the UK and the dataset will be released in November 2021.

---

You can cite this talk as:

Daniel Andre and Richard Sabbiers. "Bistatic polarimetric through-wall SAR public release data set". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 519. DOI: 10.6084/m9.figshare.14495511.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/andre.html>.

# Nonlocal UFL: Finite elements for Helmholtz equations with a nonlocal boundary condition

**Benjamin Sepanski** (<https://www.cs.utexas.edu/~bmsepan>), University of Texas at Austin,  
Department of Computer Science, United States

**Robert Kirby** ([https://sites.baylor.edu/robert\\_kirby](https://sites.baylor.edu/robert_kirby)), Department of Mathematics,  
Baylor University, United States

**Andreas Kloeckner** (<https://mathematician.de/aboutme>), Department Computer Science,  
University of Illinois at Urbana-Champaign, Urbana, IL, United States

25 March 2021

Numerical resolution of exterior Helmholtz problems require some approach to domain truncation. As an alternative to approximate nonreflecting boundary conditions and invocation of the Dirichlet-to-Neumann map, we introduce new, nonlocal boundary conditions. These conditions are exact and require the evaluation of layer potentials involving Green's functions. The nonlocal boundary conditions are readily approximated by fast multipole methods, and the resulting linear system can be preconditioned by the purely local operator. Integration of the layer potential evaluation library pyntential with the new external operator feature of Firedrake allows us to express these boundary conditions in UFL.

---

You can cite this talk as:

Benjamin Sepanski, Robert Kirby, and Andreas Kloeckner. "Nonlocal UFL: Finite elements for Helmholtz equations with a nonlocal boundary condition". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 520–576. DOI: 10.6084/m9.figshare.14495538.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/sepanski.html>.



---

# Nonlocal UFL: Finite elements for Helmholtz equations with a nonlocal boundary condition

---

Robert Kirby<sup>1</sup>

Andreas Klöckner<sup>2</sup>

Ben Sepanski<sup>3</sup>

<sup>1</sup>Baylor University

<sup>2</sup>University of Illinois at Urbana-Champaign

<sup>3</sup>University of Texas at Austin

25 March 2021



# Order of Presentation

Motivating Problem: Helmholtz scattering

A nonlocal boundary condition

Nonlocal UFL

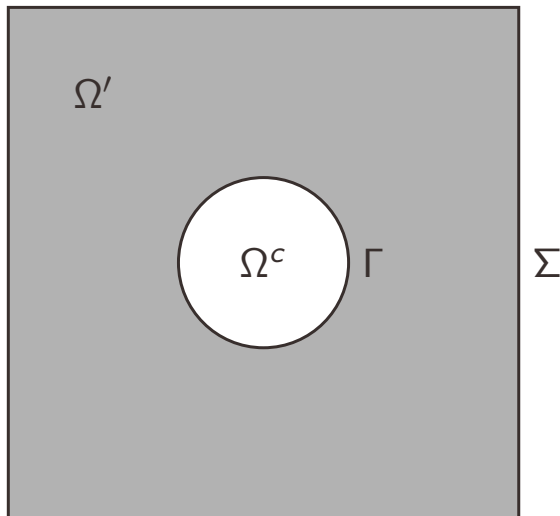
Numerical Results



## Thanks to...

- ▶ NSF 1525697, 1909176
- ▶ The U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0021110
- ▶ Luke Olson (UIUC)

# Exterior scattering<sup>1</sup>



- Model waves reflecting off of obstacle  $\Gamma$

$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \mathbb{R}^d \setminus \Omega^c \\ \frac{\partial u}{\partial n} = f, & \Gamma \end{cases}$$

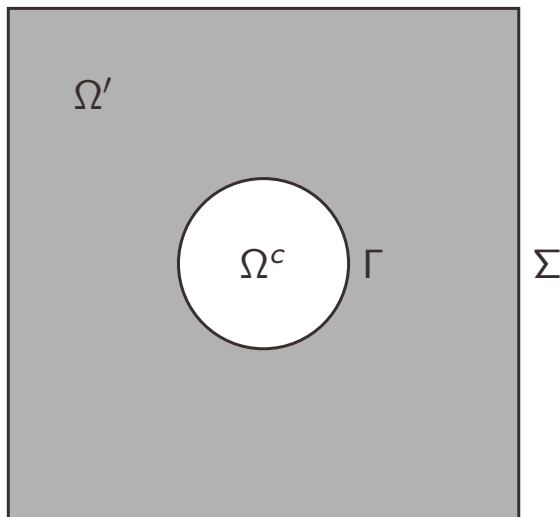
- *Without* any spurious reflections from infinity

$$\lim_{r \rightarrow \infty} r^{(d-1)/2} \left( \frac{\partial u}{\partial r} - i\kappa u \right) = 0$$

- In some finite domain of interest  $\Omega' \subseteq \mathbb{R}^d \setminus \Omega^c$  bounded by  $\Sigma$ .

<sup>1</sup>Colton and Kress 1998; Kress 1999.

# Exterior scattering<sup>1</sup>



- Model waves reflecting off of obstacle  $\Gamma$

$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \mathbb{R}^d \setminus \Omega^c \\ \frac{\partial u}{\partial n} = f, & \Gamma \end{cases}$$

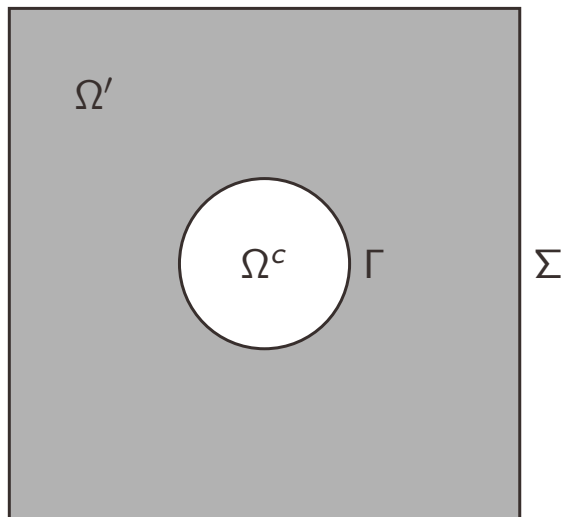
- *Without* any spurious reflections from infinity

$$\lim_{r \rightarrow \infty} r^{(d-1)/2} \left( \frac{\partial u}{\partial r} - i\kappa u \right) = 0$$

- In some finite domain of interest  $\Omega' \subseteq \mathbb{R}^d \setminus \Omega^c$  bounded by  $\Sigma$ .

<sup>1</sup>Colton and Kress 1998; Kress 1999.

# Exterior scattering<sup>1</sup>



- Model waves reflecting off of obstacle  $\Gamma$

$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \mathbb{R}^d \setminus \Omega^c \\ \frac{\partial u}{\partial n} = f, & \Gamma \end{cases}$$

- *Without* any spurious reflections from infinity

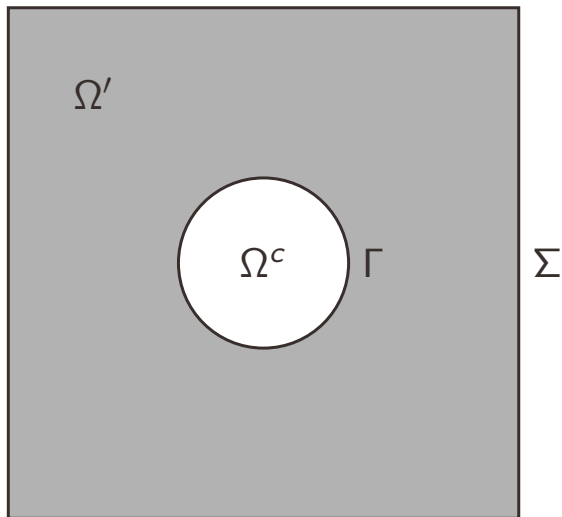
$$\lim_{r \rightarrow \infty} r^{(d-1)/2} \left( \frac{\partial u}{\partial r} - i\kappa u \right) = 0$$

- In some finite domain of interest  $\Omega' \subseteq \mathbb{R}^d \setminus \Omega^c$  bounded by  $\Sigma$ .

<sup>1</sup>Colton and Kress 1998; Kress 1999.



# Exterior scattering: computational problem



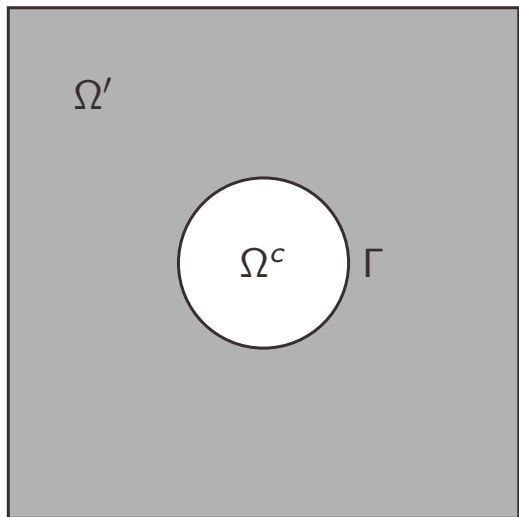
► Problem we want to solve

$$\begin{cases} -\Delta u - \kappa^2 u = 0, \\ \frac{\partial u}{\partial n} = f, \\ \lim_{r \rightarrow \infty} r^{(d-1)/2} \left( \frac{\partial u}{\partial r} - i\kappa u \right) = 0 \end{cases}$$

► Problem we can actually solve

$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ \text{?????}, & \Sigma \end{cases}$$

## Exterior scattering: computational problem



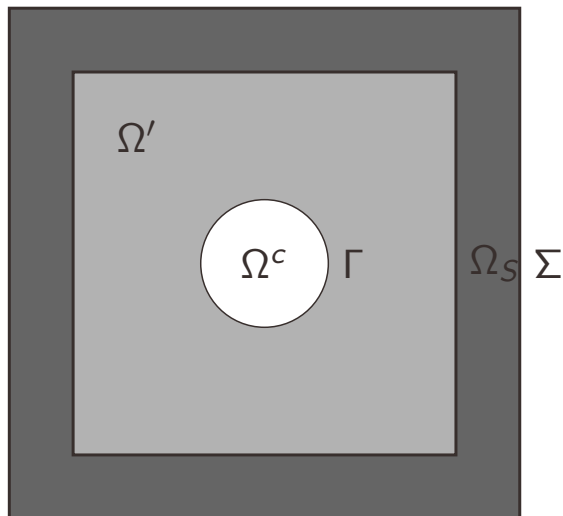
- Problem we want to solve

$$\begin{cases} -\Delta u - \kappa^2 u = 0, \\ \frac{\partial u}{\partial n} = f, \\ \lim_{r \rightarrow \infty} r^{(d-1)/2} \left( \frac{\partial u}{\partial r} - i\kappa u \right) = 0 \end{cases}$$

- Problem we can actually solve

$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ \text{?????}, & \Sigma \end{cases}$$

# Exterior scattering: Perfectly Matched Layers (PML)<sup>3</sup>



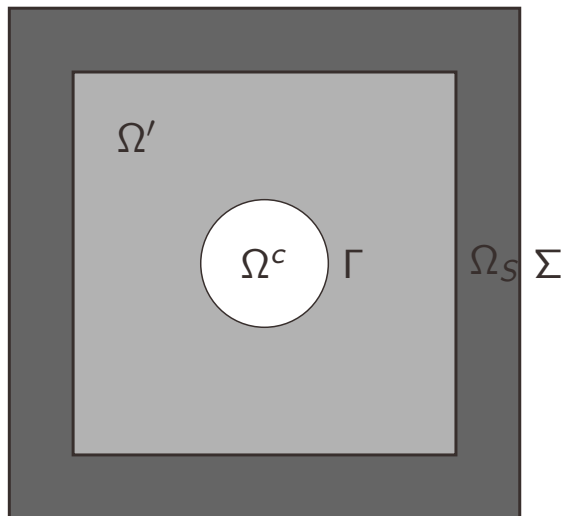
$$\begin{cases} -\nabla \cdot \beta(x) \nabla u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ u = 0, & \Sigma \end{cases}$$

- ▶  $\Omega'$ :  $\beta = I$ , satisfies original equation
- ▶  $\Omega_S$ :  $\beta$  is a complex-valued coordinate transform to cause exponential decay in oscillating waves
- ▶ **Preconditioning is difficult!**<sup>2</sup>

<sup>2</sup>Engquist and Ying 2011; Safin, Minkoff, and Zweck 2018.

<sup>3</sup>Berenger 1994; Erlangga 2006; Bermudez et al. 2006.

# Exterior scattering: Perfectly Matched Layers (PML)<sup>3</sup>



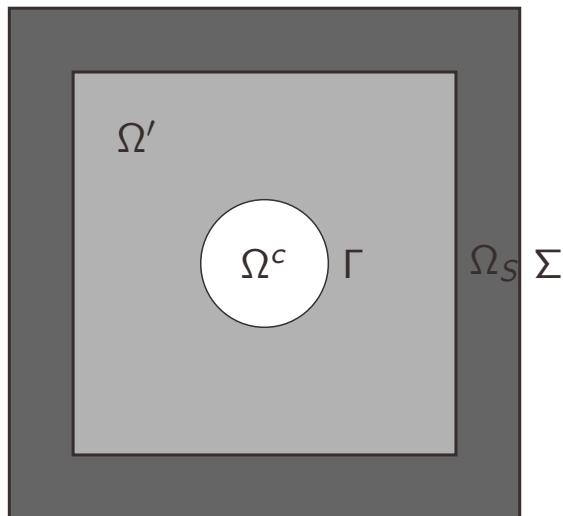
$$\begin{cases} -\nabla \cdot \beta(x) \nabla u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ u = 0, & \Sigma \end{cases}$$

- ▶  $\Omega'$ :  $\beta = I$ , satisfies original equation
- ▶  $\Omega_S$ :  $\beta$  is a complex-valued coordinate transform to cause exponential decay in oscillating waves
- ▶ **Preconditioning is difficult!**<sup>2</sup>

<sup>2</sup>Engquist and Ying 2011; Safin, Minkoff, and Zweck 2018.

<sup>3</sup>Berenger 1994; Erlangga 2006; Bermudez et al. 2006.

# Exterior scattering: Perfectly Matched Layers (PML)<sup>3</sup>



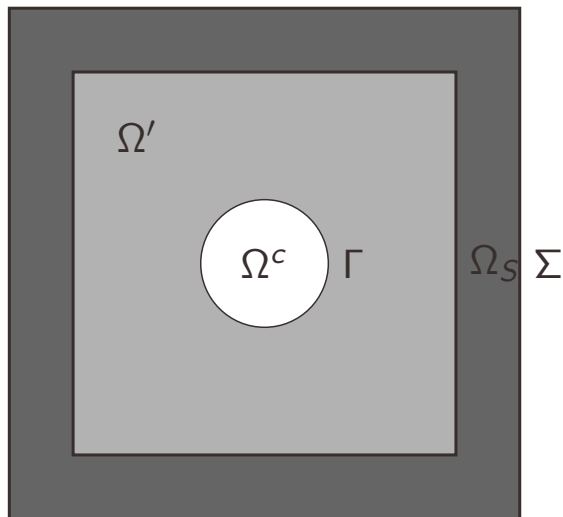
$$\begin{cases} -\nabla \cdot \beta(x) \nabla u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ u = 0, & \Sigma \end{cases}$$

- ▶  $\Omega'$ :  $\beta = I$ , satisfies original equation
- ▶  $\Omega_S$ :  $\beta$  is a complex-valued coordinate transform to cause exponential decay in oscillating waves
- ▶ **Preconditioning is difficult!**<sup>2</sup>

<sup>2</sup>Engquist and Ying 2011; Safin, Minkoff, and Zweck 2018.

<sup>3</sup>Berenger 1994; Erlangga 2006; Bermudez et al. 2006.

# Exterior scattering: Perfectly Matched Layers (PML)<sup>3</sup>



$$\begin{cases} -\nabla \cdot \beta(x) \nabla u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ u = 0, & \Sigma \end{cases}$$

- ▶  $\Omega'$ :  $\beta = I$ , satisfies original equation
- ▶  $\Omega_S$ :  $\beta$  is a complex-valued coordinate transform to cause exponential decay in oscillating waves
- ▶ **Preconditioning is difficult!**<sup>2</sup>

<sup>2</sup>Engquist and Ying 2011; Safin, Minkoff, and Zweck 2018.

<sup>3</sup>Berenger 1994; Erlangga 2006; Bermudez et al. 2006.

# Integral form of the solution

Using the Helmholtz Green's function

$$\mathcal{K}(x) = \frac{i}{4\pi |x|} e^{i\kappa|x|},$$

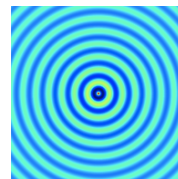


Figure:  $\mathcal{K}$  in 2D

the *true solution* satisfies<sup>4</sup>

$$u(x) = D(u)(x) - S\left(\frac{\partial u}{\partial n}\right)(x), \quad x \in \Omega'$$

where

$$D(u)(x) = \int_{\Gamma} \left( \frac{\partial}{\partial n} \mathcal{K}(x - y) \right) u(y) dy,$$

$$S(u)(x) = \int_{\Gamma} \mathcal{K}(x - y) u(y) dy$$

---

<sup>4</sup>Colton and Kress 1998; Kress 1999.

## Integral form of the solution

Using the Helmholtz Green's function

$$\mathcal{K}(x) = \frac{i}{4\pi |x|} e^{i\kappa|x|},$$

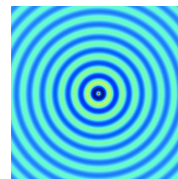


Figure:  $\mathcal{K}$  in 2D

the *true solution* satisfies<sup>4</sup>

$$u(x) = D(u)(x) - S\left(\frac{\partial u}{\partial n}\right)(x), \quad x \in \Omega'$$

where

$$D(u)(x) = \int_{\Gamma} \left( \frac{\partial}{\partial n} \mathcal{K}(x - y) \right) u(y) dy,$$

$$S(u)(x) = \int_{\Gamma} \mathcal{K}(x - y) u(y) dy$$

---

<sup>4</sup>Colton and Kress 1998; Kress 1999.



## Integral form of the solution

Using the Helmholtz Green's function

$$\mathcal{K}(x) = \frac{i}{4\pi |x|} e^{i\kappa|x|},$$

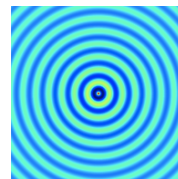


Figure:  $\mathcal{K}$  in 2D

the *true solution* satisfies<sup>4</sup>

$$u(x) = D(u)(x) - S\left(\frac{\partial u}{\partial n}\right)(x), \quad x \in \Omega'$$

where

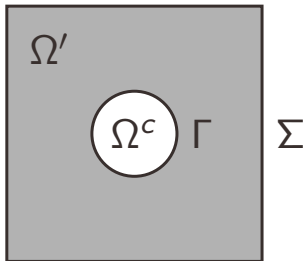
$$D(u)(x) = \int_{\Gamma} \left( \frac{\partial}{\partial n} \mathcal{K}(x - y) \right) u(y) dy,$$

$$S(u)(x) = \int_{\Gamma} \mathcal{K}(x - y) u(y) dy$$

---

<sup>4</sup>Colton and Kress 1998; Kress 1999.

# Exterior scattering



Exact boundary conditions

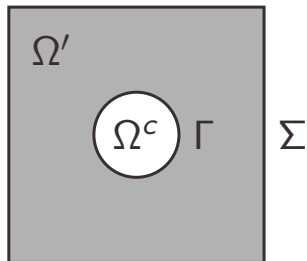
$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ (i\kappa - \frac{\partial}{\partial n})(u - D(u) + S(f)) = 0, & \Sigma \end{cases}$$

Variational Form:

For all  $v \in H^1(\Omega')$

$$\begin{aligned} (\nabla u, \nabla v) - \kappa^2 (u, v) - i\kappa \langle u, v \rangle_{\Sigma} + \langle (i\kappa - \frac{\partial}{\partial n}) D(u), v \rangle_{\Sigma} \\ = \langle f, v \rangle_{\Gamma} + \langle (i\kappa - \frac{\partial}{\partial n}) S(f), v \rangle_{\Sigma} \end{aligned}$$

# Exterior scattering



Exact boundary conditions

$$\begin{cases} -\Delta u - \kappa^2 u = 0, & \Omega' \\ \frac{\partial u}{\partial n} = f, & \Gamma \\ (i\kappa - \frac{\partial}{\partial n})(u - D(u) + S(f)) = 0, & \Sigma \end{cases}$$

## Variational Form:

For all  $v \in H^1(\Omega')$

$$\begin{aligned} (\nabla u, \nabla v) - \kappa^2 (u, v) - i\kappa \langle u, v \rangle_{\Sigma} + \langle (i\kappa - \frac{\partial}{\partial n}) D(u), v \rangle_{\Sigma} \\ = \langle f, v \rangle_{\Gamma} + \langle (i\kappa - \frac{\partial}{\partial n}) S(f), v \rangle_{\Sigma} \end{aligned}$$



# Theory

- ▶  $a$  is a bounded bilinear form on  $H^1 \times H^1$
- ▶  $F$  is a bounded linear functional on  $H^1$
- ▶ Gårding inequality. There exist  $M$  and an  $\alpha > 0$  such that

$$\operatorname{Re}(a(u, u)) + M \|u\|^2 \geq \alpha \|u\|_{H^1(\Omega)}^2.$$

- ▶ For  $h \leq h_0$ , we have optimal-order  $H^1$  and  $L^2$  error estimates.<sup>5</sup>

---

<sup>5</sup>Kirby, Klöckner, and Sepanski 2021.



# Theory

- ▶  $a$  is a bounded bilinear form on  $H^1 \times H^1$
- ▶  $F$  is a bounded linear functional on  $H^1$
- ▶ Gårding inequality. There exist  $M$  and an  $\alpha > 0$  such that

$$\operatorname{Re}(a(u, u)) + M \|u\|^2 \geq \alpha \|u\|_{H^1(\Omega)}^2.$$

- ▶ For  $h \leq h_0$ , we have optimal-order  $H^1$  and  $L^2$  error estimates.<sup>5</sup>

---

<sup>5</sup>Kirby, Klöckner, and Sepanski 2021.



# Theory

- ▶  $a$  is a bounded bilinear form on  $H^1 \times H^1$
- ▶  $F$  is a bounded linear functional on  $H^1$
- ▶ Gårding inequality. There exist  $M$  and an  $\alpha > 0$  such that

$$\operatorname{Re}(a(u, u)) + M \|u\|^2 \geq \alpha \|u\|_{H^1(\Omega)}^2.$$

- ▶ For  $h \leq h_0$ , we have optimal-order  $H^1$  and  $L^2$  error estimates.<sup>5</sup>

---

<sup>5</sup>Kirby, Klöckner, and Sepanski 2021.



# Theory

- ▶  $a$  is a bounded bilinear form on  $H^1 \times H^1$
- ▶  $F$  is a bounded linear functional on  $H^1$
- ▶ Gårding inequality. There exist  $M$  and an  $\alpha > 0$  such that

$$\operatorname{Re}(a(u, u)) + M \|u\|^2 \geq \alpha \|u\|_{H^1(\Omega)}^2.$$

- ▶ For  $h \leq h_0$ , we have optimal-order  $H^1$  and  $L^2$  error estimates.<sup>5</sup>

---

<sup>5</sup>Kirby, Klöckner, and Sepanski 2021.



# Theory

- ▶  $a$  is a bounded bilinear form on  $H^1 \times H^1$
- ▶  $F$  is a bounded linear functional on  $H^1$
- ▶ Gårding inequality. There exist  $M$  and an  $\alpha > 0$  such that

$$\operatorname{Re}(a(u, u)) + M \|u\|^2 \geq \alpha \|u\|_{H^1(\Omega)}^2.$$

- ▶ For  $h \leq h_0$ , we have optimal-order  $H^1$  and  $L^2$  error estimates.<sup>5</sup>

---

<sup>5</sup>Kirby, Klöckner, and Sepanski 2021.





# Nonlocal operations in UFL

► *Recall:*

$$D(u)(x) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(x - y) u(y) dy, \quad x \in \Sigma$$

► *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)

- This makes our stiffness matrix dense, especially in 3D
- *Solution:* Firedrake's matrix-free evaluation

► *Problem:* Naive evaluation of layer potentials is slow:

- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
- *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carrier, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(x) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(x - y) u(y) dy, \quad x \in \Sigma$$

► *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)

- This makes our stiffness matrix dense, especially in 3D
- *Solution:* Firedrake's matrix-free evaluation

► *Problem:* Naive evaluation of layer potentials is slow:

- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
- *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carter, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(x) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(x - y) u(y) dy, \quad x \in \Sigma$$

► *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)

- This makes our stiffness matrix dense, especially in 3D
- *Solution:* Firedrake's matrix-free evaluation

► *Problem:* Naive evaluation of layer potentials is slow:

- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
- *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carter, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(x) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(x - y) u(y) dy, \quad x \in \Sigma$$

► *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)

- This makes our stiffness matrix dense, especially in 3D
- *Solution:* Firedrake's matrix-free evaluation

► *Problem:* Naive evaluation of layer potentials is slow:

- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
- *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Cannier, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(x) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(x - y) u(y) dy, \quad x \in \Sigma$$

- *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)
- This makes our stiffness matrix dense, especially in 3D
  - *Solution:* Firedrake's matrix-free evaluation
- *Problem:* Naive evaluation of layer potentials is slow:
- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
  - *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carrier, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(\mathbf{x}) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(\mathbf{x} - \mathbf{y}) u(\mathbf{y}) d\mathbf{y}, \quad \mathbf{x} \in \Sigma$$

► *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)

- This makes our stiffness matrix dense, especially in 3D
- *Solution:* Firedrake's matrix-free evaluation

► *Problem:* Naive evaluation of layer potentials is slow:

- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
- *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carrier, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(\mathbf{x}) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(\mathbf{x} - \mathbf{y}) u(\mathbf{y}) d\mathbf{y}, \quad \mathbf{x} \in \Sigma$$

► *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)

- This makes our stiffness matrix dense, especially in 3D
- *Solution:* Firedrake's matrix-free evaluation

► *Problem:* Naive evaluation of layer potentials is slow:

- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
- *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carrier, Greengard, and Rokhlin 1988.



## Nonlocal operations in UFL

► *Recall:*

$$D(u)(\mathbf{x}) = \int_{\Gamma} \frac{\partial}{\partial n} \mathcal{K}(\mathbf{x} - \mathbf{y}) u(\mathbf{y}) d\mathbf{y}, \quad \mathbf{x} \in \Sigma$$

- *Problem:* Nonlocal operations have large support (all of  $\Sigma$ !)
- This makes our stiffness matrix dense, especially in 3D
  - *Solution:* Firedrake's matrix-free evaluation
- *Problem:* Naive evaluation of layer potentials is slow:
- $\text{ndof}(\Gamma) \cdot \text{ndof}(\Sigma)$
  - *Solution:* Fast multipole methods (FMM)<sup>6</sup>: use the structure of  $\mathcal{K}$  to compute the potential in *linear time* with low-rank approximations

---

<sup>6</sup>Carrier, Greengard, and Rokhlin 1988.



# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ Firedrake evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.

# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ Firedrake evaluates inner product

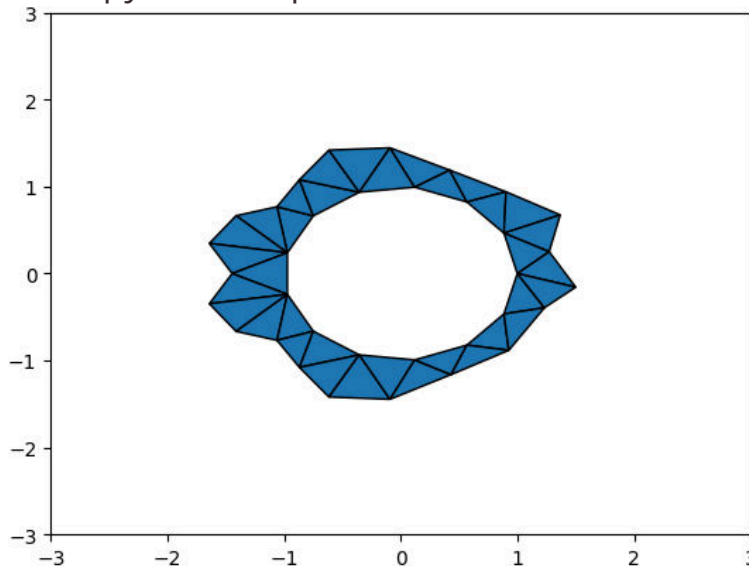
---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.

# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- Build LayerPotential as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest



<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.

# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ Firedrake evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.



# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ Firedrake evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.



# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ Firedrake evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.

# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ `Firedrake` evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.

# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ Firedrake evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.



# Nonlocal operations in UFL: Marshalling pytential<sup>8</sup>

- ▶ Build `LayerPotential` as a UFL External Operator<sup>7</sup>
  - ✓ Build pytential representation of domain of interest
  - ✓ Build pytential representation of function space
  - ✓ Build efficient converter between pytential and firedrake representations
    - Fully support automatic differentiation
- ▶ Evaluation of  $\langle (i\kappa - \frac{\partial}{\partial n})D(u), v \rangle_{\Sigma}$ 
  - ✓ `LayerPotential` evaluates  $D(u)$  (automatically uses pytential, which employs FMM to compute the potential)
  - ✓ `Firedrake` evaluates inner product

---

<sup>7</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>

<sup>8</sup>Klöckner 2020.



## Solving the system with Firedrake

Extend UFL:

$$a(u, v) = (\nabla u, \nabla v) - \kappa^2 (u, v) - i\kappa \langle u, v \rangle_{\Sigma} + \langle (i\kappa - \frac{\partial}{\partial n}) D(u) v \rangle_{\Sigma}$$



## Solving the system with Firedrake

Extend UFL:

$$a(u, v) = (\nabla u, \nabla v) - \kappa^2 (u, v) - i\kappa \langle u, v \rangle_{\Sigma} + \langle (i\kappa - \frac{\partial}{\partial n}) D(u) v \rangle_{\Sigma}$$

Will be written as:

```
Du = DoubleLayerPotential(u, HelmholtzKernel(dim=2),
                           source=gamma, target=sigma)
a = inner(grad(u), grad(v)) * dx - \
    kappa**2 * inner(u, v) * dx - \
    i * kappa * inner(u, v) * ds(sigma) + \
    i * kappa * inner(Du, v) * ds(sigma) - \
    inner(dot(grad(Du), n), v) * ds(sigma)
```



## Solving the system with Firedrake

Extend UFL:

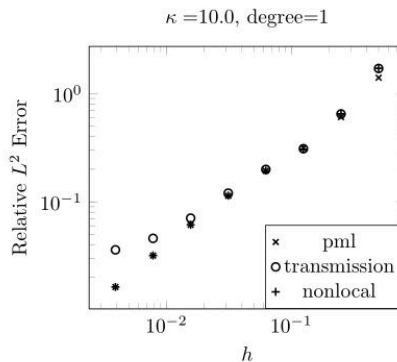
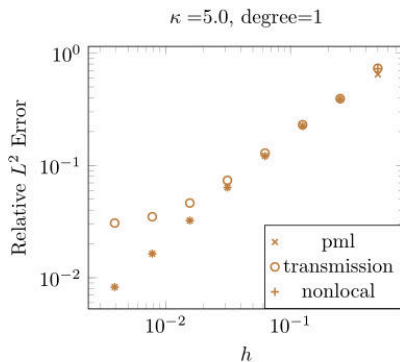
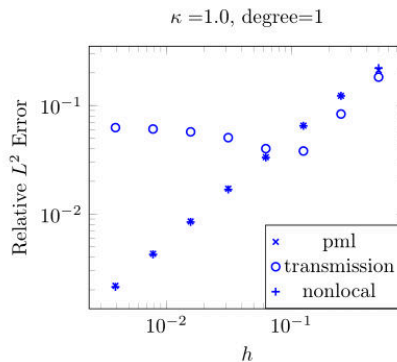
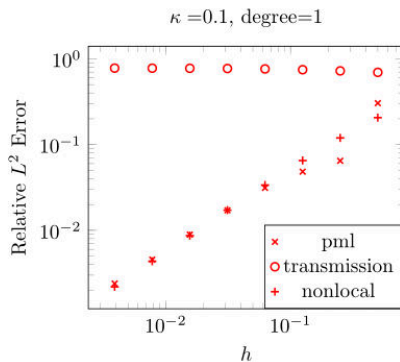
$$a(u, v) = (\nabla u, \nabla v) - \kappa^2 (u, v) - i\kappa \langle u, v \rangle_{\Sigma} + \langle (i\kappa - \frac{\partial}{\partial n}) D(u) v \rangle_{\Sigma}$$

Will be written as:

```
Du = DoubleLayerPotential(u, HelmholtzKernel(dim=2),
                           source=gamma, target=sigma)
a = inner(grad(u), grad(v)) * dx - \
    kappa**2 * inner(u, v) * dx - \
    i * kappa * inner(u, v) * ds(sigma) + \
    i * kappa * inner(Du, v) * ds(sigma) - \
    inner(dot(grad(Du), n), v) * ds(sigma)
```

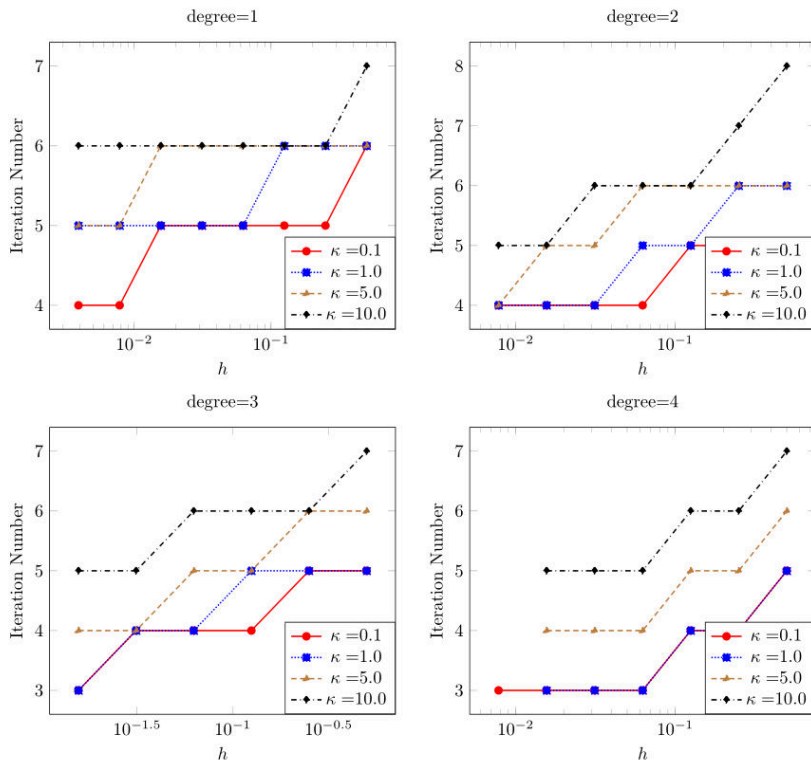


## Numerical results: 2D



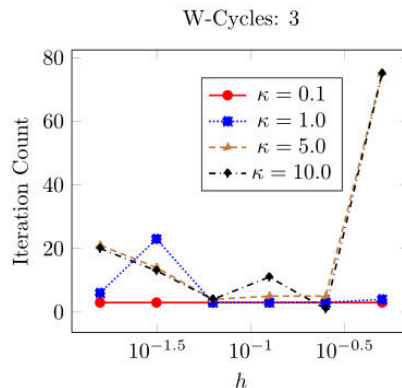
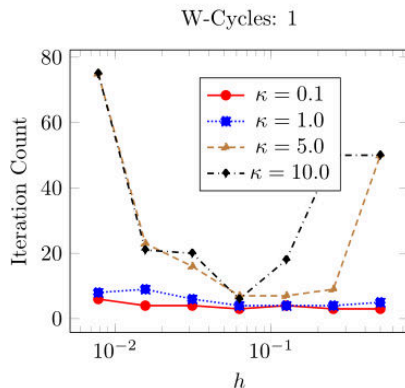


# Preconditioning: LU of local part



## Preconditioning: PyAMG

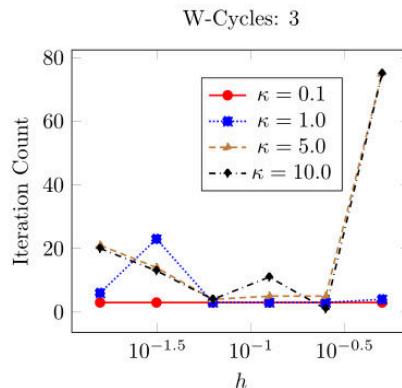
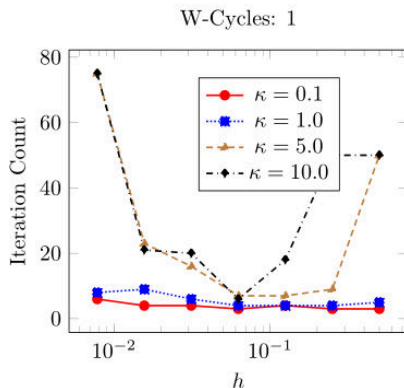
- If we can find a good preconditioner for the local problem, we get a good preconditioner for the nonlocal problem
- PyAMG: precondition with plane waves





## Preconditioning: PyAMG

- ▶ If we can find a good preconditioner for the local problem, we get a good preconditioner for the nonlocal problem
- ▶ PyAMG: precondition with plane waves







# Conclusion

## Results

- ▶ Novel nonlocal boundary condition
  - Error estimates<sup>9</sup>
- ▶ Extension of UFL to efficiently handle nonlocal operators
- ▶ Numerical experiments demonstrating optimal-order convergence
- ▶ Investigation into preconditioners

## Coming Soon

- ▶ Full implementation of LayerPotentials and VolumePotential<sup>10</sup>s in UFL as External Operator<sup>11</sup>s
- ▶ General theory for this method and application to more problems

<sup>10</sup>Kirby, Klöckner, and Sepanski 2021.

<sup>11</sup>X. Wei, *IEM-FEM Coupling*: <https://fenics2021.com/talks/wei.html>

<sup>12</sup>N. Bouziani, *External Operators*: <https://fenics2021.com/talks/bouziani.html>



## References I



Berenger, Jean-Pierre (1994). "A perfectly matched layer for the absorption of electromagnetic waves". In: *Journal of Computational Physics* 114.2, pp. 185–200. ISSN: 0021-9991. DOI: <https://doi.org/10.1006/jcph.1994.1159>. URL: <https://www.sciencedirect.com/science/article/pii/S0021999184711594>.



Bermudez, Alfredo et al. (Jan. 2006). "An optimal finite-element/pml method for the simulation of acoustic wave propagation phenomena". In: *Variational Formulations in Mechanics: Theory and Applications*. Citation Key: Bermudez:2006.



## References II



Carrier, J., Leslie Greengard, and Vladimir Rokhlin (July 1988). "A Fast Adaptive Multipole Algorithm for Particle Simulations". In: *SIAM Journal on Scientific and Statistical Computing* 9.4, pp. 669–686. ISSN: 0196-5204, 2168-3417. DOI: 10.1137/0909044.



Colton, David and Rainer Kress (Jan. 1998). *Inverse Acoustic and Electromagnetic Scattering Theory*. 2nd ed. Springer. ISBN: 3-540-62838-X.



## References III



Engquist, Björn and Lexing Ying (2011). "Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation". In: *Communications on Pure and Applied Mathematics* 64.5, pp. 697–735. DOI: <https://doi.org/10.1002/cpa.20358>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.20358>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.20358>.



Erlangga, Y. A. (2006). "A preconditioner for the Helmholtz equation with perfectly matched layer". In: URL: <https://repository.tudelft.nl/islandora/object/uuid%3A241032bc-7485-41cb-8f2a-e36ad88fbc08>.



## References IV



Kirby, Robert C., Andreas Klöckner, and Ben Sepanski (2021). *Finite elements for Helmholtz equations with a nonlocal boundary condition*. arXiv: 2009.08493 [math.NA].



Klöckner, Andreas (2020). “pytential Source Code Repository”. In: URL: <https://github.com/inducer/pytential>.



Kress, Rainer (Jan. 1999). *Linear Integral Equations*. Springer. ISBN: 978-0-387-98700-2.



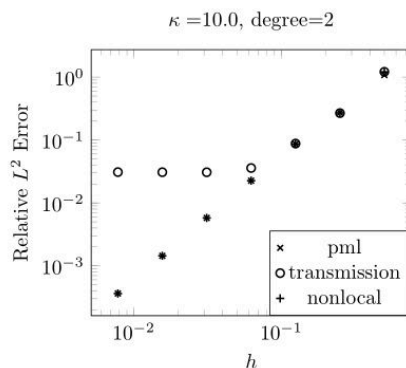
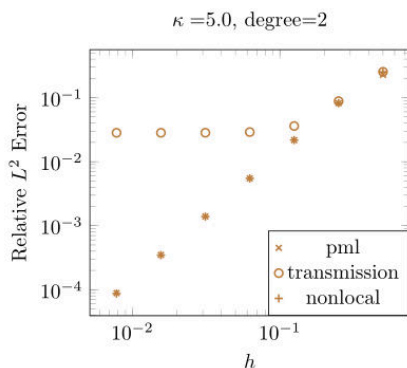
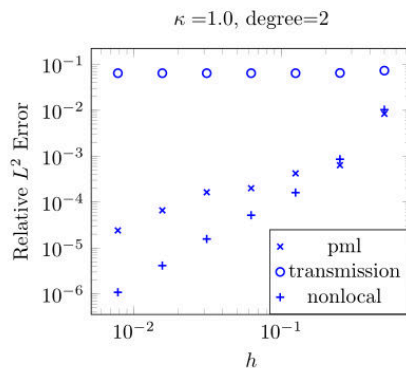
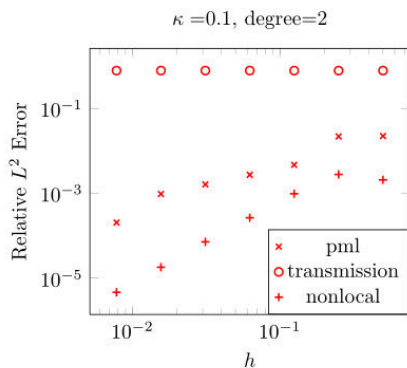
Safin, Artur, Susan Minkoff, and John Zweck (Jan. 2018). “A Preconditioned Finite Element Solution of the Coupled Pressure-Temperature Equations Used to Model Trace Gas Sensors”. In: *SIAM Journal on Scientific Computing* 40.5, B1470–B1493. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/17M1145823.



## Backup Slides

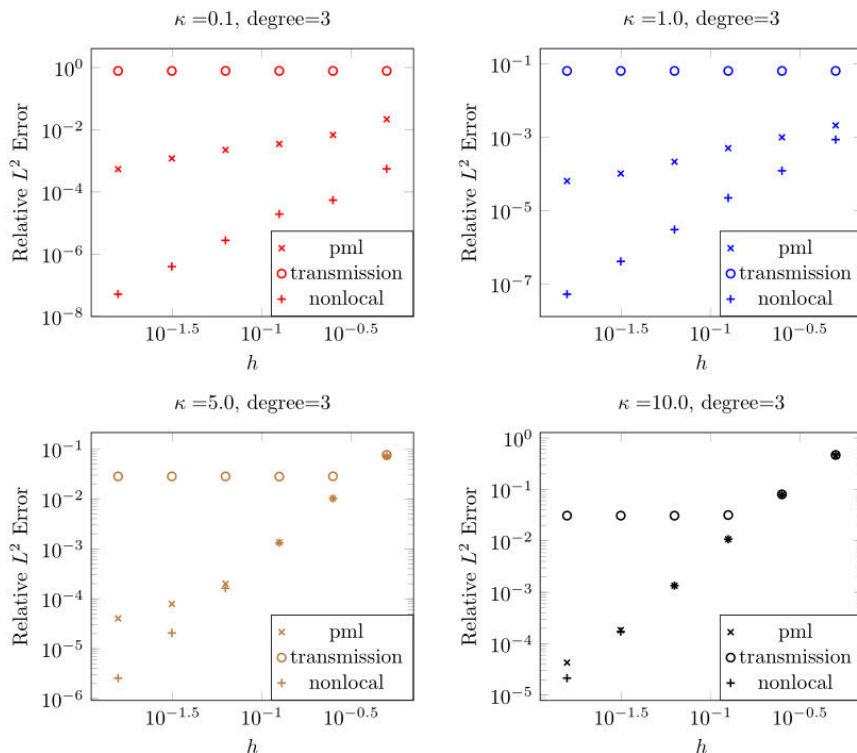


## Numerical results: 2D, degree 2





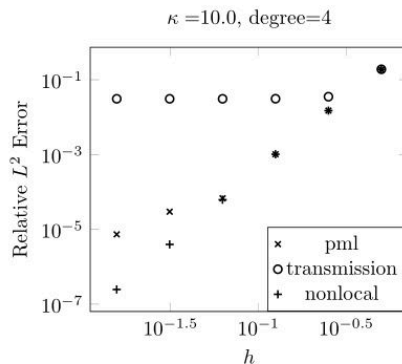
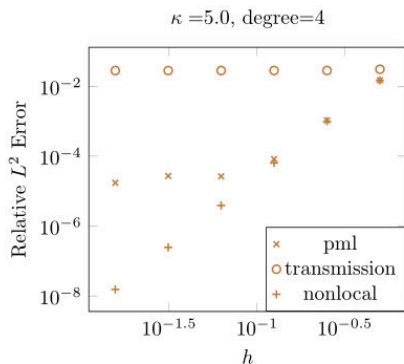
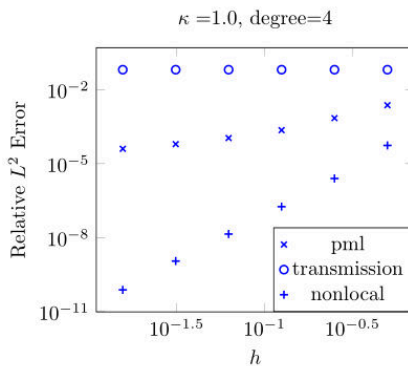
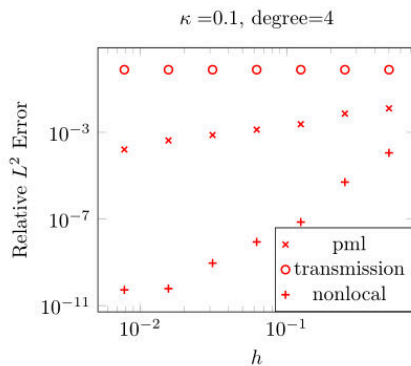
## Numerical results: 2D, degree 3





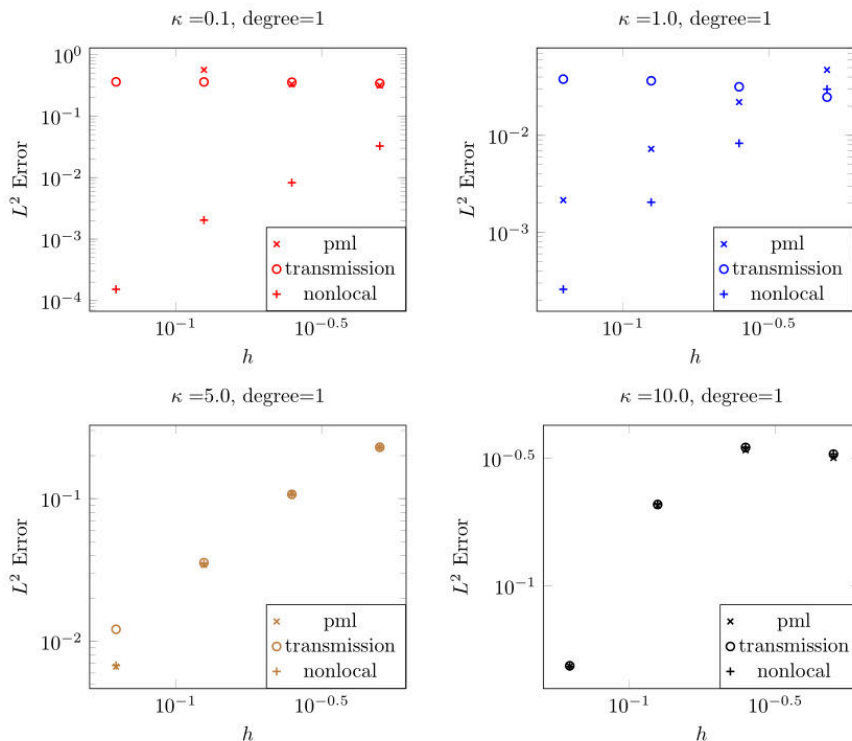


# Numerical results: 2D, degree 4





# Numerical results: 3D, degree 1



# Plasma modelling using FEniCS and FEDM

**Aleksandar P. Jovanovic**, Leibniz Institute for Plasma Science and Technology (INP), Germany

Detlef Loffhagen, Leibniz Institute for Plasma Science and Technology (INP), Germany

Markus M. Becker, Leibniz Institute for Plasma Science and Technology (INP), Germany

25 March 2021

Non-thermal low-temperature plasmas produced by electric discharges are widely used for various kinds of applications, such as chemical processing (like ozone generation), surface processing (such as plasma etching and sputtering), plasma actuators, or biomedical applications. In order to determine the governing physical and chemical processes, which is sometimes ambitious by experimental methods, plasma modelling is an additional option to be applied. For the numerical analysis of atmospheric-pressure plasmas, which are considered here, fluid models are mostly used due to their computational efficiency. Common fluid models for non-thermal plasmas consist of a set of balance equations for the particle number densities of the relevant plasma species and the energy density of electrons. Poisson's equation is typically solved to self-consistently calculate the electric potential and field. Depending on the gas under consideration the number of species that needs to be taken into account spans from tens to hundreds, which results in the same number of balance equations that need to be solved. At the same time, the number of collision and radiation processes to be considered can even reach thousands, which can make setting-up of the model difficult. Moreover, the physical time scales in plasmas range from picoseconds (electron kinetics) to tens of seconds (slow plasma-chemical reactions) so that the implementation of adaptive time stepping methods is necessary. The present FEniCS-based FEDM (Finite Element Discharge Modelling) code addresses these challenges by automating the model set-up procedure and implementing a backward differentiation formula for time stepping. This contribution represents the main features of the code, shows results of verification studies using benchmarking, and highlights how FEniCS can be used for the numerical analysis of dielectric barrier discharges in argon at atmospheric pressure.

The present work was funded by the Deutsche Forschungsgemeinschaft – project number 407462159.

---

You can cite this talk as:

Aleksandar P. Jovanovic, Detlef Loffhagen, and Markus M. Becker. "Plasma modelling using FEniCS and FEDM". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 577–597. DOI: 10.6084/m9.figshare.14495562.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/jovanovic.html>.

# Plasma modelling using FEniCS and FEDM

A. P. Jovanović, D. Loffhagen, and M. M. Becker

Leibniz Institute for Plasma Science and Technology (INP)  
Felix-Hausdorff-Str. 2, 17489 Greifswald

*FEniCS 2021*  
*25 March 2021*



Funded by the Deutsche  
Forschungsgemeinschaft -  
project number 407462159

- Plasma is a gaseous state in which free electrons and ionised atoms or molecules exist.
- Non-thermal low-temperature plasmas considered here are usually produced by electric discharges.
- They are used for different applications, such as chemical and surface processing, or biomedical applications.
- In order to describe physical and chemical processes in plasma, experimental studies are often supplemented by numerical modelling.



*Images obtained from <https://www.inp-greifswald.de/>*

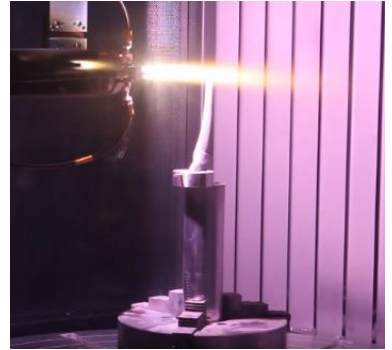
## Introduction

- Plasma is a gaseous state in which free electrons and ionised atoms or molecules exist.
- Non-thermal low-temperature plasmas considered here are usually produced by electric discharges.
- They are used for different applications, such as chemical and surface processing, or biomedical applications.
- In order to describe physical and chemical processes in plasma, experimental studies are often supplemented by numerical modelling.



*Images obtained from <https://www.inp-greifswald.de/>*

- Plasma is a gaseous state in which free electrons and ionised atoms or molecules exist.
- Non-thermal low-temperature plasmas considered here are usually produced by electric discharges.
- They are used for different applications, such as chemical and surface processing, or biomedical applications.
- In order to describe physical and chemical processes in plasma, experimental studies are often supplemented by numerical modelling.



*Images obtained from <https://www.inp-greifswald.de/>*

- Plasma is a gaseous state in which free electrons and ionised atoms or molecules exist.
- Non-thermal low-temperature plasmas considered here are usually produced by electric discharges.
- They are used for different applications, such as chemical and surface processing, or biomedical applications.
- In order to describe physical and chemical processes in plasma, experimental studies are often supplemented by numerical modelling.



*Images obtained from <https://www.inp-greifswald.de/>*



# Governing equations

- Poisson's equation for electric potential

$$-\varepsilon_0 \varepsilon_r \nabla^2 \phi = \sum_p q_p n_p$$

$$\mathbf{E} = -\nabla \phi$$

- Electron energy balance equation

$$\frac{\partial w_e}{\partial t} + \nabla \cdot \mathbf{Q}_e = -e_0 \mathbf{E} \cdot \mathbf{\Gamma}_e + \tilde{S}_e$$

$$\mathbf{Q}_e = -\frac{5}{3} b_e \mathbf{E} w_e - \nabla \left( \frac{5}{3} D_e w_e \right)$$

$$\tilde{S}_e = \sum_{j=1}^{N_r} \Delta \varepsilon_j R_j$$

- Continuity equation for particle densities

$$\frac{\partial n_p}{\partial t} + \nabla \cdot \mathbf{\Gamma}_p = S_p$$

$$\mathbf{\Gamma}_p = \text{sgn}(q_p) b_p \mathbf{E} n_p - \nabla (D_p n_p)$$

$$S_p = \sum_{j=1}^{N_r} (G_{pj} - L_{pj}) k_j \prod_{i=1}^{N_s} n_i^{\beta_{ij}}$$

- In order to solve the equations, appropriate set of boundary conditions is used:
  - Dirichlet and Robin boundary conditions for Poisson's equation
  - Robin boundary conditions for continuity equations, and electron energy balance equation.

# Challenges in plasma modelling

- For appropriate description of the processes in plasma, lots of particles, and consequently, lots of processes need to be taken into account.

**Table 1** Collision processes related to TMS included in the basic reaction kinetics model in addition to the argon model reported in [34]

Index	Reaction	Rate coefficient	References
<i>Elastic electron collisions</i>			
1	$(\text{CH}_3)_3\text{Si} + e \rightarrow (\text{CH}_3)_3\text{Si} + e$	$f(u_e)$	[45]
<i>Electron impact excitation and dissociation</i>			
2	$(\text{CH}_3)_3\text{Si} + e \rightarrow (\text{CH}_3)_3\text{Si}(v_1) + e$	$f(u_e)$	[45]
3	$(\text{CH}_3)_3\text{Si} + e \rightarrow (\text{CH}_3)_3\text{Si}(v_2) + e$	$f(u_e)$	[45]
4	$(\text{CH}_3)_3\text{Si} + e \rightarrow (\text{CH}_3)_3\text{Si} + \text{CH}_3 + e$	$f(u_e)$	[20, 45]
<i>Electron impact ionization and detachment</i>			
5	$(\text{CH}_3)_3\text{Si} + e \rightarrow (\text{CH}_3)_3\text{Si}^+ + \text{CH}_3 + 2e$	$f(u_e)$	[46]
6	$(\text{CH}_3)_3\text{Si}^- + e \rightarrow (\text{CH}_3)_3\text{Si} + 2e$	$f(u_e)$	[47–49]
<i>Dissociative electron attachment</i>			
7	$(\text{CH}_3)_3\text{Si} + e \rightarrow (\text{CH}_3)_3\text{Si}^- + \text{CH}_3$	$f(u_e)$	[45]
<i>Ion-molecule reactions</i>			
8	$\text{Ar}^+ + (\text{CH}_3)_3\text{Si} \rightarrow (\text{CH}_3)_3\text{Si}^+ + \text{CH}_3 + \text{Ar}[1p_0]$	$1.5 \times 10^{-15}$	[36, 50]
9	$\text{Ar}_2^+ + (\text{CH}_3)_3\text{Si} \rightarrow (\text{CH}_3)_3\text{Si}^+ + \text{CH}_3 + 2 \text{Ar}[1p_0]$	$1.2 \times 10^{-15}$	[36, 50]
<i>Quenching of excited argon species leading to Penning ionization</i>			
10–16	$\text{Ar}^* + (\text{CH}_3)_3\text{Si} \rightarrow (\text{CH}_3)_3\text{Si}^+ + \text{CH}_3 + \text{Ar}[1p_0] + e$	$k_{\text{MAr}}^{\text{PI}}$	See text
<i>Quenching of excited argon species leading to neutral products</i>			
17–23	$\text{Ar}^* + (\text{CH}_3)_3\text{Si} \rightarrow (\text{CH}_3)_3\text{Si} + \text{CH}_3 + \text{Ar}[1p_0]$	$k_{\text{MAr}}^{\text{Q}}$	See text
24–27	$\text{Ar}_2^* + (\text{CH}_3)_3\text{Si} \rightarrow (\text{CH}_3)_3\text{Si} + \text{CH}_3 + 2 \text{Ar}[1p_0]$	$k_{\text{MAr}}^{\text{Q}}$	Analogous to $\text{Ar}^*$ [51, 52]

D. Loffhagen et al., *Plasma Chem. Plasma Process.* **41** (2021) 289

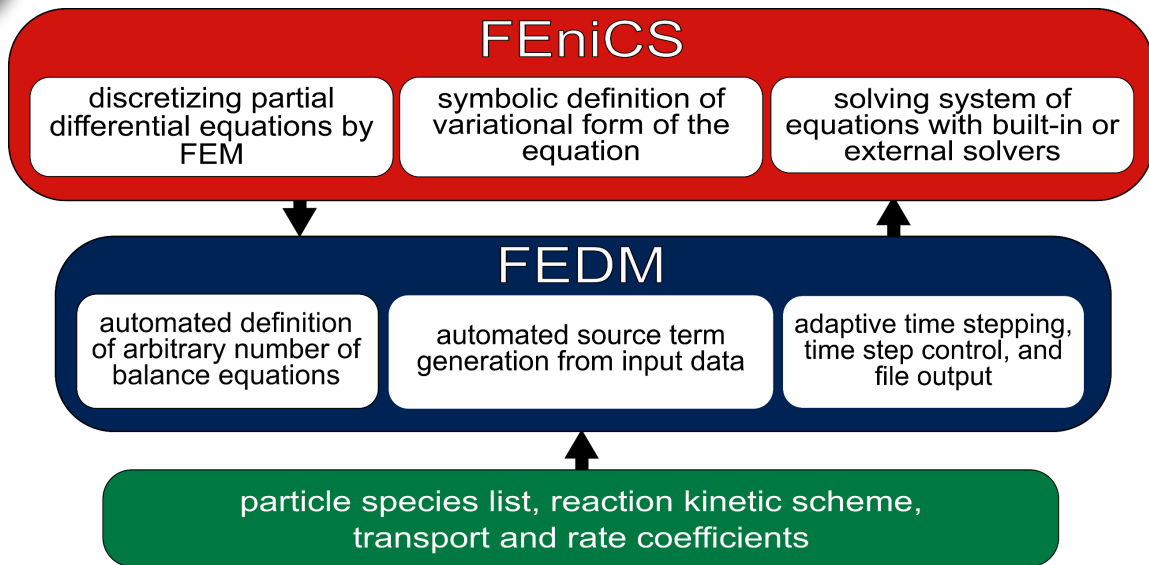
- Chemical reactions in plasma model usually lead to stiff system of equations.
- Time scale of the problem spans from picoseconds to tens of seconds.

Index	Reaction	Rate coefficient	References
552	$\text{CH}_4 + \text{C}_2\text{H} \rightarrow \text{C}_2\text{H}_2 + \text{CH}_3$	$2.3 \times 10^{-10}$	[130]
553	$\text{CH}_4 + \text{C}_2 \rightarrow \text{C}_2\text{H} + \text{CH}_3$	$1.7 \times 10^{-17}$	[131]
554	$\text{CH}_4 + \text{CH} \rightarrow \text{C}_2\text{H}_2 + \text{H}$	$9.0 \times 10^{-17}$	[132]
555	$\text{C}_2\text{H}_4 + \text{C}_2\text{H} \rightarrow \text{C}_2\text{H}_2 + \text{C}_2\text{H}_3$	$3.5 \times 10^{-17}$	[133]
556	$\text{C}_2\text{H}_4 + \text{C}_2 \rightarrow \text{C}_2\text{H} + \text{C}_2\text{H}_3$	$1.6 \times 10^{-16}$	[134]
557	$\text{C}_2\text{H}_4 + \text{CH} \rightarrow \text{C}_2\text{H}_2 + \text{CH}_3$	$1.5 \times 10^{-16}$	[129]
558	$\text{C}_2\text{H}_4 + \text{CH} \rightarrow \text{C}_2\text{H}_2 + \text{H}$	$3.0 \times 10^{-17}$	[129]
559	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 (+M) \rightarrow \text{C}_2\text{H}_4 (+M)$	$1.9 \times 10^{-17}$	[126]
560	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 \rightarrow \text{C}_2\text{H}_4 + \text{C}_2\text{H}_2$	$2.4 \times 10^{-16}$	[136]
561	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 (+M) \rightarrow \text{C}_2\text{H}_4 (+M)$	$2.5 \times 10^{-17}$	[135]
562	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 \rightarrow 2\text{C}_2\text{H}_4$	$8.0 \times 10^{-16}$	[135]
563	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 \rightarrow \text{C}_2\text{H}_4 + \text{C}_2\text{H}_2$	$8.0 \times 10^{-16}$	[135]
564	$\text{C}_2\text{H}_2 + \text{CH}_2 \rightarrow \text{CH}_3 + \text{C}_2\text{H}_3$	$3.0 \times 10^{-17}$	[135]
565	$\text{C}_2\text{H}_2 + \text{H} \rightarrow 2\text{CH}_3$	$6.0 \times 10^{-17}$	[136]
566	$\text{C}_2\text{H}_2 + \text{C}_2\text{H} \rightarrow \text{C}_2\text{H}_3 + \text{C}_2\text{H}_3$	$1.2 \times 10^{-16}$	[133]
567	$\text{C}_2\text{H}_2 + \text{C}_2 \rightarrow 2\text{C}_2\text{H}_2$	$3.3 \times 10^{-16}$	[134]
568	$\text{C}_2\text{H}_2 + \text{H} (+M) \rightarrow \text{C}_2\text{H}_4 (+M)$	$1.1 \times 10^{-16}$	[126, 127]
569	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 (+M) \rightarrow \text{C}_2\text{H}_4 (+M)$	$1.6 \times 10^{-17}$	[135]
570	$\text{C}_2\text{H}_2 + \text{C}_2\text{H}_2 \rightarrow \text{C}_2\text{H}_4 + \text{C}_2\text{H}_2$	$1.6 \times 10^{-16}$	[135]
571	$\text{C}_2\text{H}_2 + \text{CH}_2 \rightarrow \text{CH}_3 + \text{C}_2\text{H}_2$	$3.0 \times 10^{-17}$	[135]
572	$\text{C}_2\text{H}_2 + \text{H} \rightarrow \text{C}_2\text{H}_3 + \text{H}_2$	$2.0 \times 10^{-17}$	[126]
573	$\text{C}_2\text{H} + \text{CH}_2 \rightarrow \text{CH} + \text{C}_2\text{H}_2$	$3.0 \times 10^{-17}$	[135]
574	$\text{C}_2\text{H} + \text{H} (+M) \rightarrow \text{C}_2\text{H}_2 (+M)$	$3.0 \times 10^{-16}$	[135]
575	$\text{C}_2 + \text{H}_2 \rightarrow \text{C}_2\text{H} + \text{H}$	$1.5 \times 10^{-16}$	[136]

110

Plasma Chemistry and Plasma Processing (2021) 41, 289–314

## FEDM (Finite Element Discharge Modelling) code



## FEDM (Finite Element Discharge Modelling) code

- Transport and reaction rate coefficients are imported into model in form of functions or look-up tables.
- Source term definition is automated based on the reaction kinetic scheme.
- Time discretization is done using backward differentiation formula.

$$y_{n+2} - \frac{(1+\omega_{n+1})^2}{1+2\omega_{n+1}}y_{n+1} + \frac{\omega_{n+1}^2}{1+2\omega_{n+1}}y_n = \Delta t_{n+2} \frac{1+\omega_{n+1}}{1+2\omega_{n+1}}f_{n+2}$$

- Time stepping control is done using either *H211b* or PI.3.4 controllers.

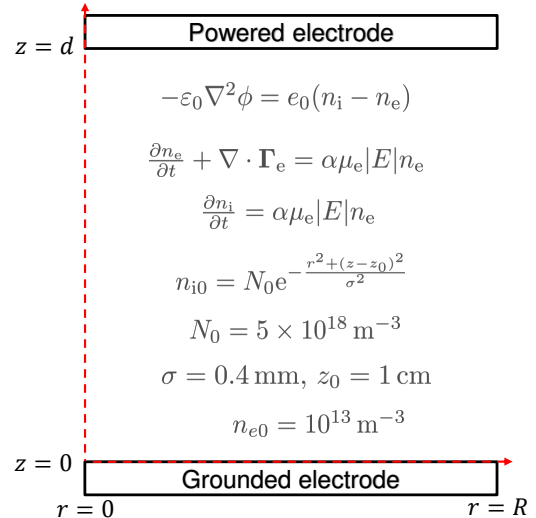
$$\Delta t_{n+1} = \left(\frac{0.8TOL}{\hat{r}_{n+1}}\right)^{0.3/k} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}}\right)^{0.4/k} \Delta t_n$$

$$\Delta t_{n+1} = \left(\frac{0.8TOL}{\hat{r}_n}\right)^{0.25/k} \left(\frac{0.8TOL}{\hat{r}_{n-1}}\right)^{0.25/k} \left(\frac{\Delta t^n}{\Delta t^{n-1}}\right)^{-0.25} \Delta t^n$$

*E. Alberdi Celaya et al., Procedia Comput. Sci., 29, 1014–1026 (2014)*  
*G. Söderlind and L. Wang, J. Comput. Appl. Math., 185, 225–243 (2006)*  
*G. Söderlind, Numer. Algorithms, 31, 281–310 (2002)*

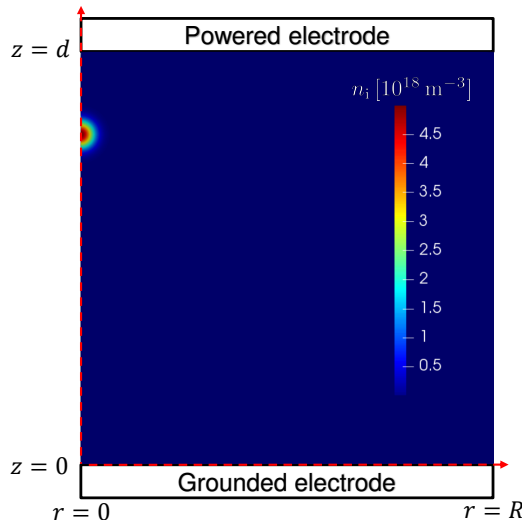
## Code verification by benchmarking

- Axisymmetric positive streamer in air at atmospheric pressure and 300 K is modelled using 2D FEDM code.
- Square domain has radius and gap distance of 1.25 cm.
- Background electric field is 15 kV/cm.
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined towards the axis and streamer region (approx. 500000 elements).
- Linear Lagrange elements are used for all the equations.
- Time-step size is constant:  $\Delta t = 5$  ps.
- Temporal evolution is followed up to 12 ns (2400 time steps).



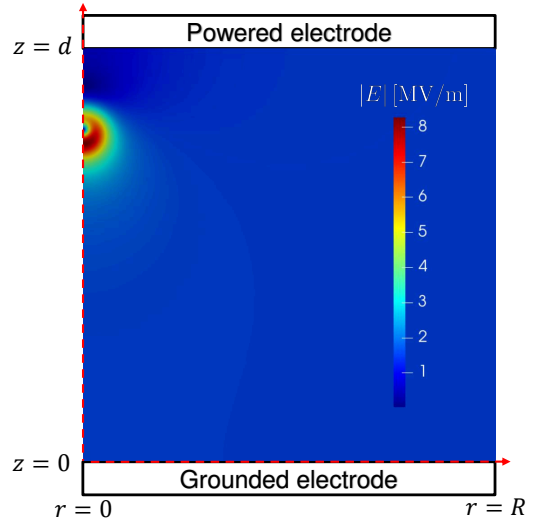
## Code verification by benchmarking

- Axisymmetric positive streamer in air at atmospheric pressure and 300 K is modelled using 2D FEDM code.
- Square domain has radius and gap distance of 1.25 cm.
- Background electric field is 15 kV/cm.
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined towards the axis and streamer region (approx. 500000 elements).
- Linear Lagrange elements are used for all the equations.
- Time-step size is constant:  $\Delta t = 5$  ps.
- Temporal evolution is followed up to 12 ns (2400 time steps).



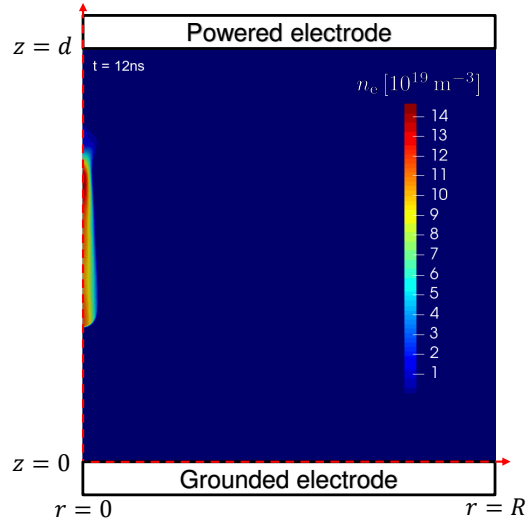
## Code verification by benchmarking

- Axisymmetric positive streamer in air at atmospheric pressure and 300 K is modelled using 2D FEDM code.
- Square domain has radius and gap distance of 1.25 cm.
- Background electric field is 15 kV/cm.
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined towards the axis and streamer region (approx. 500000 elements).
- Linear Lagrange elements are used for all the equations.
- Time-step size is constant:  $\Delta t = 5$  ps.
- Temporal evolution is followed up to 12 ns (2400 time steps).



## Code verification by benchmarking

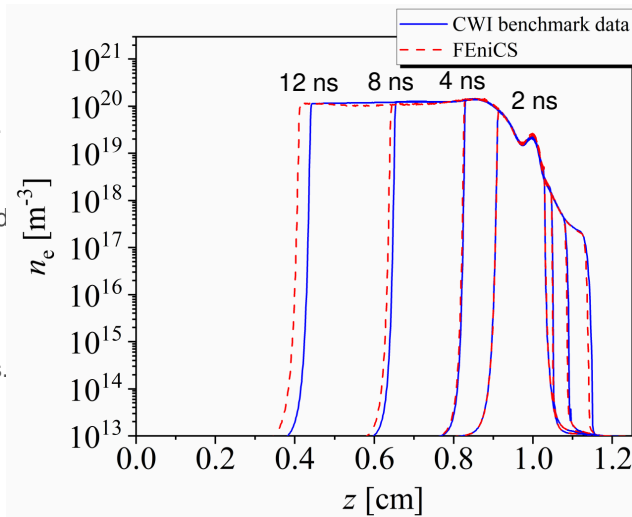
- Axisymmetric positive streamer in air at atmospheric pressure and 300 K is modelled using 2D FEDM code.
- Square domain has radius and gap distance of 1.25 cm.
- Background electric field is 15 kV/cm.
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined towards the axis and streamer region (approx. 500000 elements).
- Linear Lagrange elements are used for all the equations.
- Time-step size is constant:  $\Delta t = 5$  ps.
- Temporal evolution is followed up to 12 ns (2400 time steps).





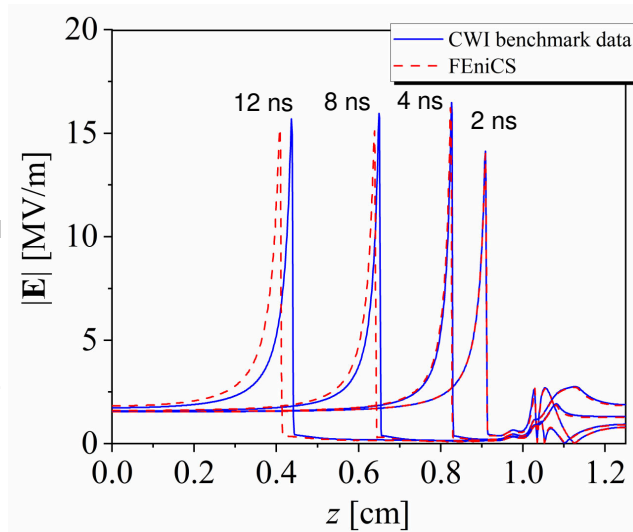
## Code verification by benchmarking

- Axisymmetric positive streamer in air at atmospheric pressure and 300 K is modelled using 2D FEDM code.
- Square domain has radius and gap distance of 1.25 cm.
- Background electric field is 15 kV/cm.
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined towards the axis and streamer region (approx. 500000 elements).
- Linear Lagrange elements are used for all the equations.
- Time-step size is constant:  $\Delta t = 5$  ps.
- Temporal evolution is followed up to 12 ns (2400 time steps).



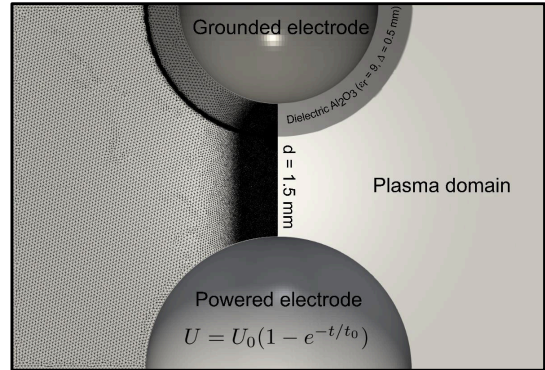
## Code verification by benchmarking

- Axisymmetric positive streamer in air at atmospheric pressure and 300 K is modelled using 2D FEDM code.
- Square domain has radius and gap distance of 1.25 cm.
- Background electric field is 15 kV/cm.
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined towards the axis and streamer region (approx. 500000 elements).
- Linear Lagrange elements are used for all the equations.
- Time-step size is constant:  $\Delta t = 5$  ps.
- Temporal evolution is followed up to 12 ns (2400 time steps).



## Dielectric barrier discharge (DBD) modelling

- Atmospheric-pressure DBD in argon in asymmetric configuration is modelled using 2D FEDM code.
- Electrodes of radius 2 mm are set 1.5 mm apart.
- Grounded electrode (top) is covered by 0.5 mm thick dielectric.
- Pulsed voltage is applied to powered electrode (bottom).
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined near the streamer region and along the dielectric (approx. 350000 elements).
- Linear Lagrange elements are used for all the equations.
- Adaptive time stepping is used ( $1 \text{ ps} < \Delta t < 100 \text{ ps}$ ).
- Temporal evolution is followed up to about 43 ns.



- Atmospheric-pressure DBD in argon in asymmetric configuration is modelled using 2D FEDM code.
- Electrodes of radius 2 mm are set 1.5 mm apart.
- Grounded electrode (top) is covered by 0.5 mm thick dielectric.
- Pulsed voltage is applied to powered electrode (bottom).
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined near the streamer region and along the dielectric (approx. 350000 elements).
- Linear Lagrange elements are used for all the equations.
- Adaptive time stepping is used ( $1 \text{ ps} < \Delta t < 100 \text{ ps}$ ).
- Temporal evolution is followed up to about 43 ns.

$$-\varepsilon_0 \varepsilon_r \nabla^2 \phi = \sum_p q_p n_p$$

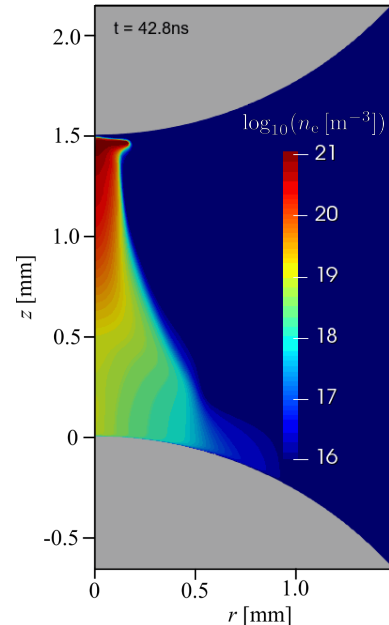
$$\frac{\partial n_p}{\partial t} + \nabla \cdot \mathbf{\Gamma}_p = S_p$$

$$\frac{\partial w_e}{\partial t} + \nabla \cdot \mathbf{Q}_e = -e_0 \mathbf{E} \cdot \mathbf{\Gamma}_e + \tilde{S}_e$$

$$\frac{\partial \sigma}{\partial t} = \sum_p q_p \mathbf{\Gamma}_p \cdot \boldsymbol{\nu}$$

## Dielectric barrier discharge (DBD) modelling

- Atmospheric-pressure DBD in argon in asymmetric configuration is modelled using 2D FEDM code.
- Electrodes of radius 2 mm are set 1.5 mm apart.
- Grounded electrode (top) is covered by 0.5 mm thick dielectric.
- Pulsed voltage is applied to powered electrode (bottom).
- Gaussian seed near the powered electrode is introduced to locally enhance the field and initiate the streamer.
- Mesh is refined near the streamer region and along the dielectric (approx. 350000 elements).
- Linear Lagrange elements are used for all the equations.
- Adaptive time stepping is used ( $1 \text{ ps} < \Delta t < 100 \text{ ps}$ ).
- Temporal evolution is followed up to about 43 ns.



## Conclusion and outlook

---

- FEDM code for automated set-up of the equations is developed.
- The code is verified using benchmarking.
- The challenges in cases where the problem is defined on several subdomains, such as DBDs, could possibly be resolved using mixed-dimensional formulation.
- Handling of electron-energy-dependent and electric-field-dependent coefficients should be further addressed because they can lead to small time-step sizes.

## Contact



Leibniz Institute for Plasma Science and Technology

Address: Felix-Hausdorff-Str. 2, 17489 Greifswald

Phone: +49 - 3834 - 554 3911, Fax: +49 - 3834 - 554 301

E-mail: [aleksandar.jovanovic@inp-greifswald.de](mailto:aleksandar.jovanovic@inp-greifswald.de), Web: [www.leibniz-inp.de](http://www.leibniz-inp.de)

# Implementation of fluid-structure interactions for rigid body motion in FEniCS using immersed finite element method

Chayut Teeraratkul, University of Colorado Boulder, United States

Debanjan Mukherjee, University of Colorado Boulder, United States

25 March 2021

In this work, an implementation of rigid body immersed finite element method in FEniCS is presented. Immersed finite element method was proposed for resolving complex fluid structure interaction problems often encounter in many engineering applications. In immersed finite element method, the structure is represented by a Lagrangian mesh moving on top of a Eulerian fluid mesh. This allow for the fluid mesh to be generated independently from the solid structure and thus greatly simplified the meshing process. The no-slip condition and the FSI force at the fluid-solid interface is enforced using a mesh-to-mesh interpolation of velocity and FSI coupling force. Classically, the interpolation method employed in immersed finite element is to use a discrete delta function; however, in this work a method based on transforming basis function between the two domain is employed. This allow for the support of the FSI force interpolation to be the size elements in the fluid domain touching the structure domain. Support size on which the fluid-structure interaction force is applied is therefore optimal in an element-wise sense. Results from a canonical problem of rigid sphere dropping in a channel is simulated to demonstrate the implementation. Implementation details and performance of the implementation is discussed.

---

You can cite this talk as:

Chayut Teeraratkul and Debanjan Mukherjee. "Implementation of fluid-structure interactions for rigid body motion in FEniCS using immersed finite element method". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 598. doi: 10.6084/m9.figshare.14495568.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/teeraratkul.html>.



# Direct FEM Simulation for air pollution dispersion in building aerodynamics

Linde van Beers, KTH Royal Institute of Technology, Sweden

Johan Jansson, KTH Royal Institute of Technology, Sweden

Måns Andersson, KTH Royal Institute of Technology, Sweden

25 March 2021

We discuss the suitability of time-resolved adaptive Direct FEM Simulations (DFS) as an alternative method to RANS and LES in building aerodynamics. This method has significant advantages over other methods in that it relies less on human-defined parameters and meshes, potentially reducing the need for extensive validation for each new case. Next to that, it uses computational resources very efficiently, as the mesh is adapted to local error estimates. Especially for applications such as pollution dispersion, where RANS does not provide sufficiently detailed flow characteristics and LES is often too slow, adaptive DFS may present a solution.

However, little research is available on the use of this method in the Atmospheric Boundary Layer, which is essential for flow features around buildings. We will present preliminary principles and results for applying time-resolved adaptive DFS in building aerodynamics, with the focus on future application to pollution dispersion. Moreover, experiments will be performed using FEniCS in accessible environments with limited computational resources, to enhance reproducibility.

---

You can cite this talk as:

Linde van Beers, Johan Jansson, and Måns Andersson. "Direct FEM Simulation for air pollution dispersion in building aerodynamics". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 599. doi: 10.6084/m9.figshare.14495574.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/van-beers.html>.

# A tracer's sojourn in a compressible velocity field

**Nate Sime**, Carnegie Institution for Science, United States

Cian Wilson, Carnegie Institution for Science, United States

Peter van Keken, Carnegie Institution for Science, United States

**25 March 2021**

Pointwise satisfaction of the continuity equation is key to precise advection of tracer data through incompressible velocity fields.

In this presentation we explore tracer advection in the context of weakly compressible flows arising in anelastic fluids.

To do this we exploit the hybrid discontinuous Galerkin (HDG) method via the GeoPart and LEoPart libraries in conjunction with the FEniCS project.

We present some esoteric findings and aesthetic images.

---

You can cite this talk as:

Nate Sime, Cian Wilson, and Peter van Keken. "A tracer's sojourn in a compressible velocity field". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 600. doi: 10.6084/m9.figshare.14495586.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/sime.html>.

# A two-level nonlinear beam model using adjoints

**Marco Morandini** (<https://home.aero.polimi.it/morandini>), Politecnico di Milano, Italy

**26 March 2021**

A two-level nonlinear beam model, with the two levels linked by first- and second-order adjoints, allows to successfully deal with beam problems characterized by complex and multi-material cross-sections. Emphasis is given to the approach used for coupling the two levels within the FEniCS framework, and on how to overcome a few issues encountered during the actual implementation of the coupling procedure.

---

You can cite this talk as:

Marco Morandini. “A two-level nonlinear beam model using adjoints”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 601–641. doi: 10.6084/m9.figshare.14495592.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/morandini.html>.



**POLITECNICO**

MILANO 1863

# A two-level nonlinear beam model using adjoints

Marco Morandini

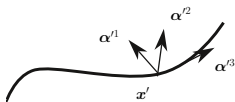
[marco.morandini@polimi.it](mailto:marco.morandini@polimi.it)

Politecnico di Milano

FEniCS 2021, March 22-26

## Beam model

- Domain: line
- Unknowns:  $\mathbf{x}'$ ,  $\boldsymbol{\alpha}'$
- Needs constitutive law  
 $\{\mathbf{T}, \mathbf{M}\}(\boldsymbol{\epsilon}, \boldsymbol{\beta})$

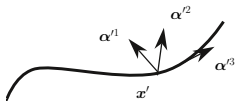


Handling of finite rotations in Dolfin,  
proc. FEniCS Conference 2017

[https://home.aero.polimi.it/morandini/Downloads/  
DolfinFiniteRotations/html/](https://home.aero.polimi.it/morandini/Downloads/DolfinFiniteRotations/html/)

## Beam model

- Domain: line
- Unknowns:  $\mathbf{x}'$ ,  $\boldsymbol{\alpha}'$
- Needs constitutive law  $\{\mathbf{T}, \mathbf{M}\}(\boldsymbol{\epsilon}, \boldsymbol{\beta})$

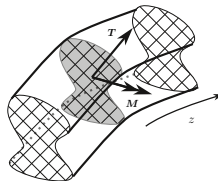


Handling of finite rotations in Dofin,  
proc. FEniCS Conference 2017

<https://home.aero.polimi.it/morandini/Downloads/DofinFiniteRotations/html/>

## Cross-section models

- Domain: cross section (area)
- Unknowns:  $\hat{\mathbf{u}}_i$
- Known 3D constitutive law
- Function of  $\{\mathbf{T}, \mathbf{M}\}$   
(forcing term)



Analysis of beam cross section response  
accounting for large strains and plasticity, IJSS,  
2019.

- Beam model:  
from  $\{\mathbf{x}', \boldsymbol{\alpha}'\}$

$$\int_L (\delta \boldsymbol{\varepsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M}) \, ds - \delta L_e = 0$$

# Two-level model

- Beam model:  
from  $\{\mathbf{x}', \boldsymbol{\alpha}'\}$

$$\int_L (\delta \boldsymbol{\varepsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M}) \, ds - \delta L_e = 0$$

to  $\{\mathbf{x}', \boldsymbol{\alpha}', \mathbf{T}, \mathbf{M}\}$



- Beam model:  
from  $\{\mathbf{x}', \boldsymbol{\alpha}'\}$

$$\int_L (\delta \boldsymbol{\epsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M}) \, ds - \delta L_e = 0$$

to  $\{\mathbf{x}', \boldsymbol{\alpha}', \mathbf{T}, \mathbf{M}\}$

complementary strain energy

$$v(\mathbf{T}, \mathbf{M}) = \boldsymbol{\epsilon} \mathbf{T} + \boldsymbol{\beta} \mathbf{M} - w(\boldsymbol{\epsilon}, \boldsymbol{\beta})$$

- Beam model:  
from  $\{\mathbf{x}', \boldsymbol{\alpha}'\}$

$$\int_L (\delta \boldsymbol{\epsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M}) \, ds - \delta L_e = 0$$

to  $\{\mathbf{x}', \boldsymbol{\alpha}', \mathbf{T}, \mathbf{M}\}$

complementary strain energy

$$v(\mathbf{T}, \mathbf{M}) = \boldsymbol{\epsilon} \mathbf{T} + \boldsymbol{\beta} \mathbf{M} - w(\boldsymbol{\epsilon}, \boldsymbol{\beta})$$

$$\mathcal{H}(\delta\{\boldsymbol{\epsilon}, \boldsymbol{\beta}, \mathbf{T}, \mathbf{M}\}, \{\boldsymbol{\epsilon}, \boldsymbol{\beta}, \mathbf{T}, \mathbf{M}\}) = \int_L (\delta \boldsymbol{\epsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M} + \delta \mathbf{T} \boldsymbol{\epsilon} + \delta \mathbf{M} \boldsymbol{\beta} - \delta v) \, ds - \delta L_e = 0$$

- 1 Beam model:  
from  $\{\mathbf{x}', \boldsymbol{\alpha}'\}$

$$\int_L (\delta \boldsymbol{\epsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M}) ds - \delta L_e = 0$$

to  $\{\mathbf{x}', \boldsymbol{\alpha}', \mathbf{T}, \mathbf{M}\}$

complementary strain energy

$$v(\mathbf{T}, \mathbf{M}) = \boldsymbol{\epsilon} \mathbf{T} + \boldsymbol{\beta} \mathbf{M} - w(\boldsymbol{\epsilon}, \boldsymbol{\beta})$$

$$\mathcal{H}(\delta\{\boldsymbol{\epsilon}, \boldsymbol{\beta}, \mathbf{T}, \mathbf{M}\}, \{\boldsymbol{\epsilon}, \boldsymbol{\beta}, \mathbf{T}, \mathbf{M}\}) = \int_l (\delta \boldsymbol{\epsilon} \mathbf{T} + \delta \boldsymbol{\beta} \mathbf{M} + \delta \mathbf{T} \boldsymbol{\epsilon} + \delta \mathbf{M} \boldsymbol{\beta} - \delta v) ds - \delta L_e = 0$$

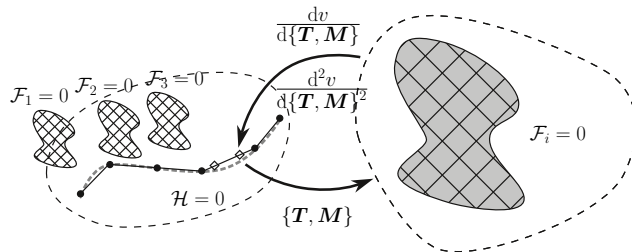
- 2 Cross section model:  $\mathcal{F}(\delta \hat{\mathbf{u}}_i, \hat{\mathbf{u}}_i, \{\mathbf{T}, \mathbf{M}\}) = 0$   
compute

$$v = \int_A \mathbf{S} : \boldsymbol{\epsilon} - \psi(\boldsymbol{\epsilon}, \boldsymbol{\chi}) dA$$

on the cross-section  $\mathcal{F} = 0$  with  $\{\mathbf{T}, \mathbf{M}\}$  from beam model

# Two-level model

Global model:  $\mathcal{H}(\delta\{\epsilon, \beta, \mathbf{T}, \mathbf{M}\}, \{\epsilon, \beta, \mathbf{T}, \mathbf{M}\})$

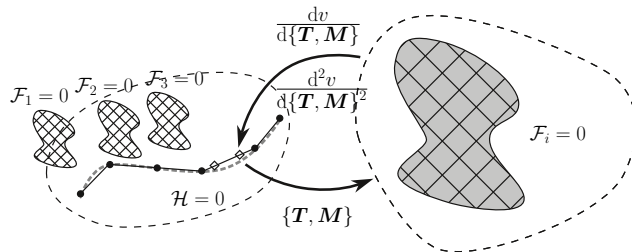


$$\delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$$

$$\partial \delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{d^2v}{d\{\mathbf{T}, \mathbf{M}\}^2} \cdot \partial\{\mathbf{T}, \mathbf{M}\}$$

# Two-level model

Global model:  $\mathcal{H}(\delta\{\epsilon, \beta, \mathbf{T}, \mathbf{M}\}, \{\epsilon, \beta, \mathbf{T}, \mathbf{M}\})$



$$\delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$$

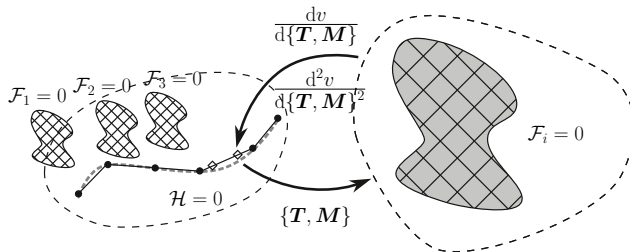
$$\partial \delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{d^2 v}{d\{\mathbf{T}, \mathbf{M}\}^2} \cdot \partial\{\mathbf{T}, \mathbf{M}\}$$

Cross section:  $\mathcal{F}(\delta \hat{\mathbf{u}}_i, \hat{\mathbf{u}}_i, \{\mathbf{T}, \mathbf{M}\}) = 0$

- $\frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$ : first order adjoint

# Two-level model

Global model:  $\mathcal{H}(\delta\{\epsilon, \beta, \mathbf{T}, \mathbf{M}\}, \{\epsilon, \beta, \mathbf{T}, \mathbf{M}\})$



$$\delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$$

$$\partial \delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{d^2v}{d\{\mathbf{T}, \mathbf{M}\}^2} \cdot \partial\{\mathbf{T}, \mathbf{M}\}$$

Cross section:  $\mathcal{F}(\delta \hat{\mathbf{u}}_i, \hat{\mathbf{u}}_i, \{\mathbf{T}, \mathbf{M}\}) = 0$

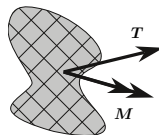
•  $\frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$ : first order adjoint

•  $\frac{d^2v}{d\{\mathbf{T}, \mathbf{M}\}^2}$ : second order adjoint

# Implementation

Cross section:

```
class BeamSection():
    # solve the  $\mathcal{F} = 0$ 
    def solve(self, force, moment):
        ....
    # compute the derivative  $\frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$ 
    def delta_v(self, force, moment):
        ....
    # compute the derivative  $\frac{d^2v}{d^2\{\mathbf{T}, \mathbf{M}\}}$ 
    def de_delta_v(self, force, moment):
        ....
```



where

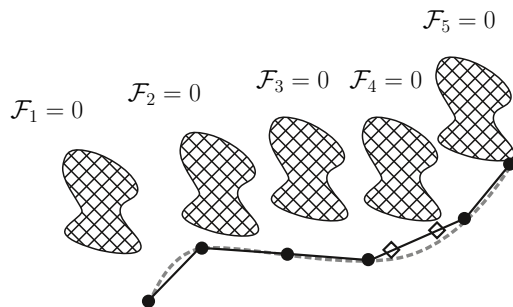
solve find the solution of  $\mathcal{F} = 0$ ;

delta\_v compute the first derivative  $\frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$

de\_delta\_v compute the second derivative  $\frac{d^2v}{d^2\{\mathbf{T}, \mathbf{M}\}}$

# Implementation

Global beam model  $\mathcal{H} = 0$ :  
store array `beams_sec` of `BeamSections`,  
one for each cell





Global beam model  $\mathcal{H} = 0$ , array `beams_sec`

Global beam model  $\mathcal{H} = 0$ , array beams\_sec

```
class delta_v_expression(UserExpression):  
  
    # evaluate  $\frac{dv}{d\{T, M\}}$   
    def eval_cell(self, value, x, ufc_cell):  
  
        ...  
  
        value = beams_sec[ufc_cell.index].delta_v(self.Tc, self.Mc)
```

Global beam model  $\mathcal{H} = 0$ , class `delta_v_expression`

$$\int_I (\delta \epsilon \mathbf{T} + \delta \beta \mathbf{M} + \delta \mathbf{T} \epsilon + \delta \mathbf{M} \beta - \delta v) ds - \delta L_e$$

#  $\delta \mathbf{T}$  and  $\delta \mathbf{M}$

```
test_FM = as_vector([v_F[0], v_F[1], ...])
```

#  $\frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$

```
delta_v_expr = delta_v_expression(u, element = AZ2_EL)
```

#  $\delta v = \delta\{\mathbf{T}, \mathbf{M}\} \cdot \frac{dv}{d\{\mathbf{T}, \mathbf{M}\}}$

```
delta_v = inner(test_FM, delta_v_expr)
```

#  $\mathcal{H}$

```
functional = ... - \  
    delta_v * dx - \  
    ....
```

Linearization?

Same approach but with  $\frac{d^2 v}{d\{\mathbf{T}, \mathbf{M}\}^2}$

# Does it work?

Yes



# Does it work?

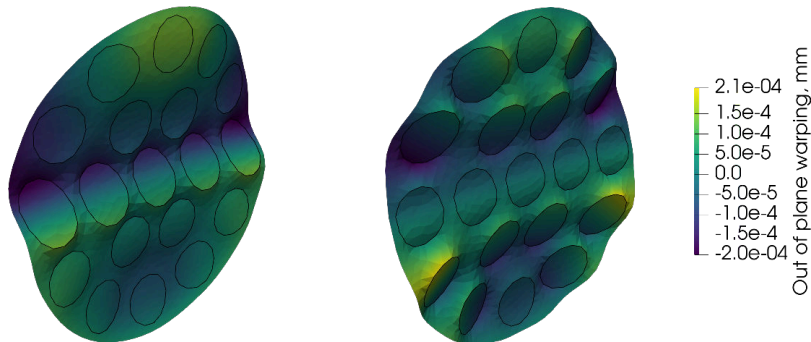
Yes

- quadratic convergence

# Does it work?

Yes

- quadratic convergence
- nice results, also with elasto-plastic & hyperelastic materials



Morandini M, A two-level nonlinear beam analysis method, IJSS, 2020

- $\{T, M\}$  piece-wise constant discontinuous (DG0)



- $\{\mathbf{T}, \mathbf{M}\}$  piece-wise constant discontinuous (DG0)
- repeated calls, for the same cell, with same  $\{\mathbf{T}, \mathbf{M}\}$

- $\{\mathbf{T}, \mathbf{M}\}$  piece-wise constant discontinuous (DG0)
- repeated calls, for the same cell, with same  $\{\mathbf{T}, \mathbf{M}\}$
- (limited) caching  $\rightarrow$  significant speedup

## Issue #2: a tale of coffee mugs

Python Dolphin wrapper

Early in the morning



# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

...

$$\mathcal{F}_1 = 0$$



... 20 min ...

# Issue #2: a tale of coffee mugs

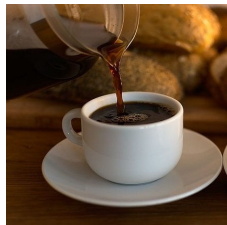
Python Dolfin wrapper

...

$$\mathcal{F}_1 = 0$$



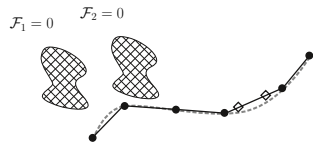
... 20 min ...



# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

...

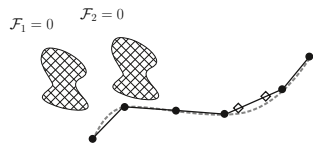


... 20 min ...

# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

...



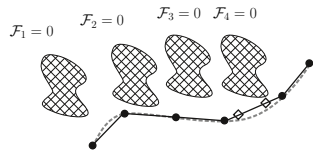
... 20 min ...



# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

...



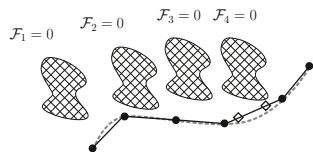
... n x 20 min ...



# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

... late at night



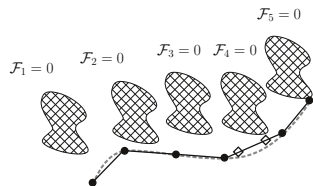
...  $n \times 20$  min ...



# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

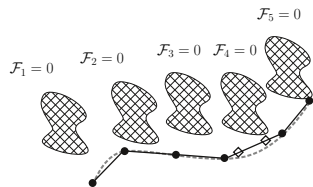
... late at night



# Issue #2: a tale of coffee mugs

Python Dolfin wrapper

... late at night



## Issue #2: a tale of coffee mugs

Why?



## Issue #2: a tale of coffee mugs

Why?

- dijitso caches FFC code generation & compilation



## Issue #2: a tale of coffee mugs

Why?

- dijitso caches FFC code generation & compilation
- caching based of Form “signature”

## Issue #2: a tale of coffee mugs

Why?

- dijitso caches FFC code generation & compilation
- caching based of Form “signature”
- Form.signature() really expensive

## Issue #2: a tale of coffee mugs

Why?

- dijitso caches FFC code generation & compilation
- caching based of Form “signature”
- Form.signature() really expensive
- Form.signature() gets recomputed!



## Issue #2: a tale of coffee mugs

Why?

- dijitso caches FFC code generation & compilation
- caching based of Form “signature”
- Form.signature() really expensive
- Form.signature() gets recomputed!

Stop-gap solution:

## Issue #2: a tale of coffee mugs

Why?

- dijitso caches FFC code generation & compilation
- caching based of Form “signature”
- `Form.signature()` really expensive
- `Form.signature()` gets recomputed!

Stop-gap solution:

replace `Form.signature()`

`hash(.py source file) + form name`





# Interoperability with automatic differentiation libraries through NumPy interface to FEniCS

Ivan Yashchuk, Aalto University, Finland

26 March 2021

Partial differential equations (PDEs) are used to describe a variety of physical phenomena. Computing derivative information of the solution to PDE with respect to the input parameters is important in many tasks in scientific computing. A high-level interface for evaluating derivatives of FEniCS models is developed. It is intended to be used as the backend for extending Automatic Differentiation libraries to support FEniCS solvers. High-level symbolic representation of PDEs allows bypassing differentiating through low-level possibly many iterations of the underlying nonlinear solvers. Automatic tangent linear and adjoint solvers for FEniCS problems are derived with `dolfin-adjoint`/`pyadjoint`. These solvers make it possible to use forward and reverse modes Automatic Differentiation with FEniCS. This package is used for building bridges between FEniCS and JAX, PyMC3 (Theano), PyTorch, Julia's `ChainRule.jl`, `Zygote.jl`. This enables the efficient composition of finite element solvers with arbitrary differentiable programs.

---

You can cite this talk as:

Ivan Yashchuk. "Interoperability with automatic differentiation libraries through NumPy interface to FEniCS". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 642–659. DOI: 10.6084/m9.figshare.14495595.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/yashchuk.html>.

# AD libraries + FEniCS/Firedrake

Ivan Yashchuk

Aalto University | Quansight Labs

FEniCS'21 Conference  
ivan.yashchuk@aalto.fi

$$\int_{\Omega} \text{grad } u \cdot \text{grad } v \, dx - \int_{\Omega} f \cdot v \, dx = 0$$

```
# Create mesh for the unit square domain
n = 10
mesh = UnitSquareMesh(n, n)

# Define discrete function spaces and functions
V = FunctionSpace(mesh, "CG", 1)
W = FunctionSpace(mesh, "DG", 0)

def fenics_solve(f):

    u = Function(V, name="PDE Solution")
    v = TestFunction(V)
    F = (inner(grad(u), grad(v)) - f * v) * dx
    bcs = [DirichletBC(V, 0.0, "on_boundary")]
    solve(F == 0, u, bcs)
    return u
```

$$\int_{\Omega} \text{grad } u \cdot \text{grad } v \, dx - \int_{\Omega} f \cdot v \, dx = 0$$

```
# Create mesh for the unit square domain
n = 10
mesh = UnitSquareMesh(n, n)

# Define discrete function spaces and functions
V = FunctionSpace(mesh, "CG", 1)
W = FunctionSpace(mesh, "DG", 0)

# Define FEniCS template representation of JAX input
templates = (Function(W),)

@build_jax_fem_eval(templates)
def fenics_solve(f):
    # This function inside should be traceable by fenics_adjoint
    u = Function(V, name="PDE Solution")
    v = TestFunction(V)
    F = (inner(grad(u), grad(v)) - f * v) * dx
    bcs = [DirichletBC(V, 0.0, "on_boundary")]
    solve(F == 0, u, bcs)
    return u

# Calculate the solution jacobian using JAX and adjoint PDE
dudf = jax.jacobian(fenics_solve)(f)
```

# Behind the scenes: Tangent and Adjoint PDEs

Symbolic form of the problem is used to derive additional PDEs that are solved for calculating Jacobian-vector and vector-Jacobian products.

Let  $F(u, m) = 0$  represent the PDE,

$u$  represents the solution and  $m$  represents the parameters.

Jacobian-vector product: 
$$\frac{du}{dm} \dot{v} := - \frac{\partial F}{\partial u}^{-1} \frac{\partial F}{\partial m} \dot{v}$$

vector-Jacobian product: 
$$\frac{du}{dm}^* \bar{v} := - \frac{\partial F}{\partial m}^* \left[ \frac{\partial F}{\partial u}^{-*} \bar{v} \right]$$



# Jacobian-vector product

$$\frac{du}{dm} \dot{v} := - \frac{\partial F^{-1}}{\partial u} \frac{\partial F}{\partial m} \dot{v}$$

```
dFdu = ufl.derivative(F, u)
dFdm = ufl.derivative(F, m, vdot)
dudm_vdot = fenics.Function(V)
t1m_F = ufl.action(dFdu, dudm_vdot) + dFdm
t1m_F = ufl.replace(t1m_F, {dudm_vdot: fenics.TrialFunction(V)})
fenics.solve(ufl.lhs(t1m_F) == ufl.rhs(t1m_F), dudm_vdot, bcs=hbc)
```

# Jacobian-transpose-vector product

$$\frac{du}{dm}^* \bar{v} := -\frac{\partial F}{\partial m}^* \left[ \frac{\partial F}{\partial u}^{-*} \bar{v} \right]$$

```
dFdu = ufl.derivative(F, u)
adFdu = ufl.adjoint(dFdu)

u_adj = fenics.Function(V)
adj_F = ufl.action(adFdu, u_adj)
adj_F = ufl.replace(adj_F, {u_adj: fenics.TrialFunction(V)})
fenics.solve(adj_F == v_bar, u_adj)

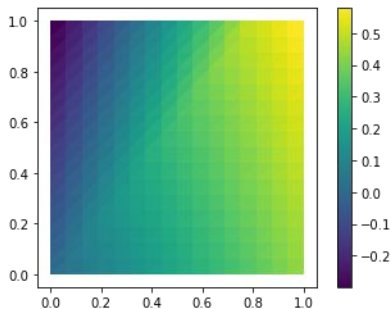
dFdm = ufl.derivative(F, fenics_input, fenics.TrialFunction(V))
adFdm = ufl.adjoint(dFdm)
dudm*_v_bar = fenics.assemble(-adFdm * u_adj)
```

# Composition with other JAX programs

Let's use `jax.stax` to set up network initialization and evaluation functions

```
# Define  $\mathbb{R}^2 \rightarrow \mathbb{R}^1$  function
net_init, net_apply = jax.experimental.stax.serial(Dense(2), Relu, Dense(10), Relu, Dense(1))

nn_predictions = net_apply(net_params, W.tabulate_dof_coordinates())
f_nn = numpy_to_fenics(nn_predictions, fenics.Function(W))
```



```
In [ ]: def eval_nn(net_params):
        f_nn = np.ravel(net_apply(net_params, W.tabulate_dof_coordinates()))
        u = fenics_solve(f_nn)
        norm_u = np.linalg.norm(u)
        return norm_u
```

```
In [ ]: %%time
        jax.grad(eval_nn)(net_params)
```

CPU times: user 340 ms, sys: 7.82 ms, total: 348 ms  
Wall time: 337 ms

```
Out[ ]: [(DeviceArray([[ 0.11934833, -0.08526856],
                        [ 0.11315436, -0.16367367]], dtype=float32),
         DeviceArray([ 0.22669935, -0.25705874], dtype=float32)),
        (),
        (DeviceArray([[ 0.0000000e+00,  0.0000000e+00,  3.4377411e-01,
                        -3.8510424e-01,  4.1671190e-02,  0.0000000e+00,
                        2.8023928e-01, -2.3091170e-06, -4.5939538e-01,
                        0.0000000e+00],
                        [ 0.0000000e+00,  0.0000000e+00,  1.2621611e-02,
                        -1.4139040e-02,  1.5299511e-03,  0.0000000e+00,
                        1.0288941e-02, -3.4431054e-08, -1.6866626e-02,
                        0.0000000e+00]], dtype=float32),
         DeviceArray([-0.0000000e+00, -0.0000000e+00,  3.9731458e-01,
                        -4.4508159e-01,  4.8161194e-02,  0.0000000e+00,
                        3.2388464e-01, -3.3104476e-05, -5.3094304e-01,
                        0.0000000e+00], dtype=float32)),
        (),
        (DeviceArray([[0.0000000e+00],
                        [0.0000000e+00],
                        [2.40548089e-01],
                        [1.17404766e-01],
                        [3.21369737e-01],
                        [0.0000000e+00],
                        [3.51112783e-01],
                        [1.75102848e-08],
                        [1.22634955e-01],
                        [0.0000000e+00]], dtype=float32),
         DeviceArray([0.6358373], dtype=float32)))]
```

# “Physics-Informed” Neural Networks (PINN)

Poisson problem,

Find  $u \in V$  such that  $a(u, v) = L(v)$  for all  $v \in V$

with

$$a(u, v) = \int_{\Omega} \nabla v \cdot \nabla u \, dx, \quad L(v) = \int_{\Omega} v f \, dx$$

$$J(v) = \frac{1}{2}a(v, v) - L(v)$$

Equivalent minimization problem:

$$J(u) \leq J(v) \quad \forall v \in V$$

Usual FEM:  $v(x) = \sum_j c_j \psi_j(x)$     Taking `c = nn(x; coefficients)`

and solving the minimization problem for neural network coefficients we get PINN.

# “Physics-Informed” Neural Networks (PINN)

```
def eval_nn(net_params)
    """
    Given the neural network parameters assemble
    the energy integral using FEniCS and return the value
     $\mathbb{R}^m \rightarrow \mathbb{R}$ ,  $m = \text{size}(\text{net\_params})$ 
    """
    v_nn = net_apply(net_params, W.tabulate_dof_coordinates())
    value = fenics_assemble_energy(v_nn)
    return value

jax.value_and_grad(eval_nn)(net_params) # ( $\mathbb{R}$ ,  $\mathbb{R}^m$ )
jax.hessian(eval_nn)(net_params) #  $\mathbb{R}^m \times \mathbb{R}^m$ 
```

# dolfin-adjoint is not enough but pyadjoint is

pyadjoint/dolfin-adjoint is an automatic differentiation for FEniCS and Firedrake  
+ an interface to selected optimization libraries (SciPy, IPOpt, Moola, PyROL)

The goal is to embed PDE solvers inside other programs for

- composition with other differentiable programs (for example neural networks)
- probabilistic parameter estimation
- interface to optimization and sampling libraries outside of dolfin-adjoint

This work is about serialisation layer using NumPy arrays and API that simplifies embedding of FEniCS/Firedrake in AD libraries

# Finite Element Chain Rules (inspired by ChainRules.jl)

 FEniCS passing Firedrake passing codecov 77%

A serialisation layer using NumPy arrays

+ API that simplifies embedding of FEniCS/Firedrake in AD libraries

```
def evaluate_primal(
    firedrake_function: Callable[..., BackendVariable],
    firedrake_templates: Collection[BackendVariable],
    *numpy_inputs: np.array,
) -> Tuple[np.array, BackendVariable, Collection[BackendVariable], pyadjoint.Tape]

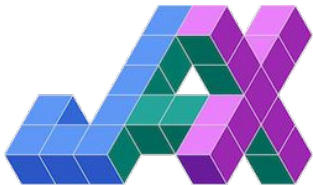
def evaluate_pushforward(
    firedrake_output: BackendVariable,
    firedrake_inputs: Collection[BackendVariable],
    tape: pyadjoint.Tape,
    Δnumpy_inputs: Collection[np.array],
) -> Collection[np.array]

def evaluate_pullback(
    firedrake_output: BackendVariable,
    firedrake_inputs: Collection[BackendVariable],
    tape: pyadjoint.Tape,
    Δnumpy_output: np.array,
) -> Collection[np.array]
```





**FECR**



# PyMC3



```
from fenics_pymc3 import create_fenics_theano_op
# Define FEniCS template representation of Theano/NumPy input
# that is we promise that our arguments are of the following types
# the choice is between Constant and Function
templates = (fenics_adjoint.Constant(0.0), fenics_adjoint.Constant(0.0))

@create_fenics_theano_op(templates)
def solve_elasticity(E, p_g):
    ...
    return solution

import pymc3 as pm
import theano.tensor as tt

loads = [[1.], [2.5], [5.]]
measurements = [0.11338, 0.28346, 0.56693]

with pm.Model() as model:

    E = pm.Normal("E", mu=1.1e5, sigma=0.3e5, shape=(1,))

    maximum_deflections = []
    for i in range(len(measurements)):
        p_g = loads[i]
        predicted_displacement = solve_elasticity(E, p_g)
        maximum_deflection = tt.max(predicted_displacement)
        maximum_deflections.append(maximum_deflection)
    maximum_deflections = tt.stack(maximum_deflections)

    d = pm.Normal("d", mu=maximum_deflections, sd=1e-3, observed=measurements)
```

```
map_estimate = pm.find_MAP(model=model)
print(f"MAP estimate of E is {map_estimate['E']}")
# 100% [22/22 00:04<00:00 logp = 6.6135,
# ||grad|| = 1.1144e-05]
# MAP estimate of E is [125000.40465515]

with model:
    trace = pm.sample(100, chains=1, cores=1, tune=100)
pm.summary(trace)
# Sequential sampling (1 chains in 1 job)
# NUTS: [E]
#
```

	mean	sd	hdi_3%	hdi_97%
# E[0]	132148.824	285.805	131637.637	132484.79

# Julia | Turing.jl

```
function solve_firedrake(kappa0, kappa1)
    ...
    firedrake.solve(F == 0, u, bcs = bcs)
    return u
end

@register_fem_function(
    zygote_solve_firedrake, templates, solve_firedrake)

@model function fit_diffusion_coef(data)
     $\sigma \sim \text{InverseGamma}(3, 0.5)$ 
    kappa0 ~ truncated(Normal(1.0, 0.5), 1e-5, 2)
    kappa1 ~ truncated(Normal(0.7, 0.5), 1e-5, 2)
    predicted_solution = zygote_solve_firedrake([kappa0], [kappa1])
    data ~ MvNormal(predicted_solution,  $\sigma$ )
end

model = fit_diffusion_coef(noisy_solution)
chain = sample(model, NUTS(.65), 1000)
```

## Summary Statistics

parameters	mean	std	naive_se	mcse	ess	rhat
Symbol	Float64	Float64	Float64	Float64	Float64	Float64
kappa0	1.2497	0.3789	0.0120	0.0357	130.3485	1.0001
kappa1	0.5443	0.1711	0.0054	0.0155	139.7087	1.0001
$\sigma$	0.0143	0.0019	0.0001	0.0001	178.8158	1.0043

## Quantiles

parameters	2.5%	25.0%	50.0%	75.0%	97.5%
Symbol	Float64	Float64	Float64	Float64	Float64
kappa0	0.5183	0.9850	1.2590	1.5483	1.9140
kappa1	0.2295	0.4291	0.5424	0.6698	0.8579
$\sigma$	0.0111	0.0130	0.0142	0.0154	0.0188

# Summary | AD + FEniCS/Firedrake

## What?

*Automatic* forward and reverse differentiation of FEniCS/Firedrake composable with JAX | PyMC3 | Julia

## Why?

Reuse existing well established libraries instead of reinventing the wheels in “differentiable physics” fashion

Composability with other libraries of host AD:

Including PDEs in probabilistic modelling using PyMC3 | Turing.jl | NumPyro (JAX)

Interfacing with optimization and sampling libraries

## What's next?

Arbitrary higher-order derivatives for JAX and Julia

Distributed array interface

Compatibility with JAX's JIT compilation

# How to get started?

Step 1: Install Firedrake or FEniCS

For embedding in other AD libraries:

<https://github.com/IvanYashchuk/fecr>

For JAX interface:

<https://github.com/IvanYashchuk/jax-fenics-adjoint>

For PyMC3 interface:

<https://github.com/IvanYashchuk/fenics-pymc3>

For Julia interface:

<https://github.com/IvanYashchuk/PyFenicsAD.jl>

# Developing an automatized optimization problem in FEniCS for parameter determination of metamaterials

**Navid Shekarchizadeh**, Department of Basic and Applied Sciences for Engineering, Sapienza University of Rome, Italy

Alberto Maria Bersani, Department of Mechanical and Aerospace Engineering, Sapienza University of Rome, Italy

**26 March 2021**

In this work, a novel automatized optimization process is developed for the inverse analysis and parameter determination of metamaterials. Metamaterials are the family of materials designed to have tailored material properties, such as high strength-to-weight ratio or extreme elasticity, by using an optimized topology. Due to metamaterials' inner substructure, it is of interest to simulate their mechanical behaviour using reduced-order modelling utilizing the generalized mechanics. We determine the constitutive parameters of such models by developing an automatized optimization process in FEniCS. This process utilizes the Trust Region Reflective optimization method, from Scipy, for minimizing the deviation of the continuum model from a detailed micro-scale model. The parameter identification procedure proves to be robust and reliable by testing it for the pantographic structures as an example of metamaterials.

---

You can cite this talk as:

Navid Shekarchizadeh and Alberto Maria Bersani. "Developing an automatized optimization problem in FEniCS for parameter determination of metamaterials". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 660–679. DOI: 10.6084/m9.figshare.14495607.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/shekarchizadeh.html>.



SAPIENZA  
UNIVERSITÀ DI ROMA

# Developing an automatized optimization problem in FEniCS for parameter determination of metamaterials

**Navid Shekarchizadeh**, Alberto Maria Bersani

Department of Basic and Applied Sciences for Engineering  
Sapienza University of Rome

FEniCS 2021, Cambridge

26 Mar. 2021



# Introduction

## What are metamaterials?

- engineered materials, with properties not found in natural materials
- usually arranged in repeating patterns
- at scales smaller than the wavelengths of the phenomena they influence
- derive their properties from their designed structures

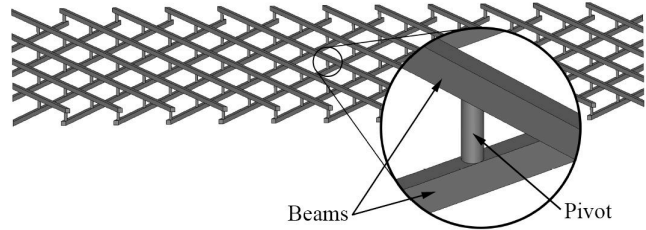
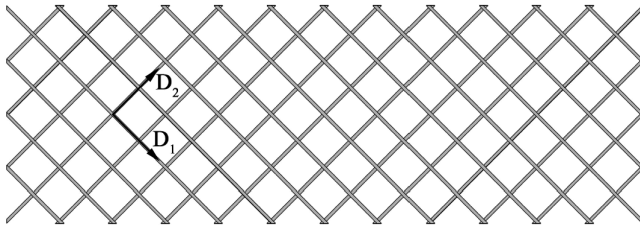
We need to identify the parameters of metamaterials' models



# Introduction

An example of metamaterials:

## Pantographic structures



Shekarchizadeh, N, Abali, BE, Barchiesi, E, Bersani, AM. Inverse analysis of metamaterials and parameter determination by means of an automatized optimization problem. *Z Angew Math Mech.* 2021;e202000277

# Introduction

## Pantographic Structures

- **Properties:**

- Large deformation in the elastic region
- High toughness: absorbing large amount of energy in the elastic and plastic regimes
- Extraordinarily high specific strength

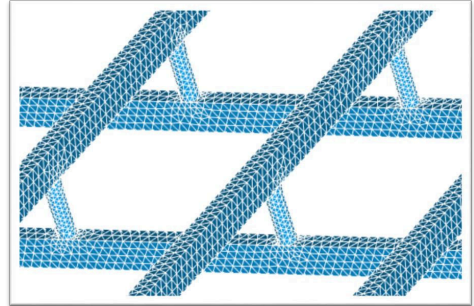
- **Main Deformation Energy Mechanisms:**

- Shear deformation of the elastic pivots
- Bending of beams
- Stretching of beams

# Introduction

## Modeling Pantographic Structures

- **Micro-scale Model**
  - Using Cauchy first-gradient continuum theory
- **Macro-scale Model**
  - Using a strain-gradient energy model



# Micro-scale Model

- **Nonlinear Elasticity**

- Deformation of a body

$$x_i = X_i + u_i$$

- Deformation gradient

$$F_{ij} = \frac{\partial x_i}{\partial X_j}$$

- Green-Lagrange strain tensor

$$E_{ij} = \frac{1}{2}(F_{ki}F_{kj} - \delta_{ij}) = \frac{1}{2}(u_{k,i}u_{k,j} + u_{i,j} + u_{j,i})$$

- Strain energy density:

$$W_m(\mathbf{E}) = \frac{\lambda}{2}E_{kk}^2 + \mu E_{ij}E_{ij}$$

- Elasticity action functional:

$$\mathcal{A} = \int_{B_0} \left( \frac{1}{2} \rho_0 \dot{u}_i \dot{u}_i - W_m + \rho_0 f_i u_i \right) dV + \int_{\partial B_0^N} \hat{t}_i u_i dA$$

- Weak form:

$$- \int_{B_0} \frac{\partial W_m}{\partial u_{i,j}} \delta u_{i,j} dV + \int_{\partial B_0^N} \hat{t}_i \delta u_i dA = 0$$

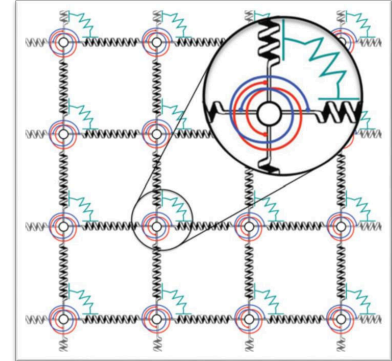
# Macro-scale Model

A macro-scale model for planar pantographic structures

- A homogenized model with **strain-gradient** terms

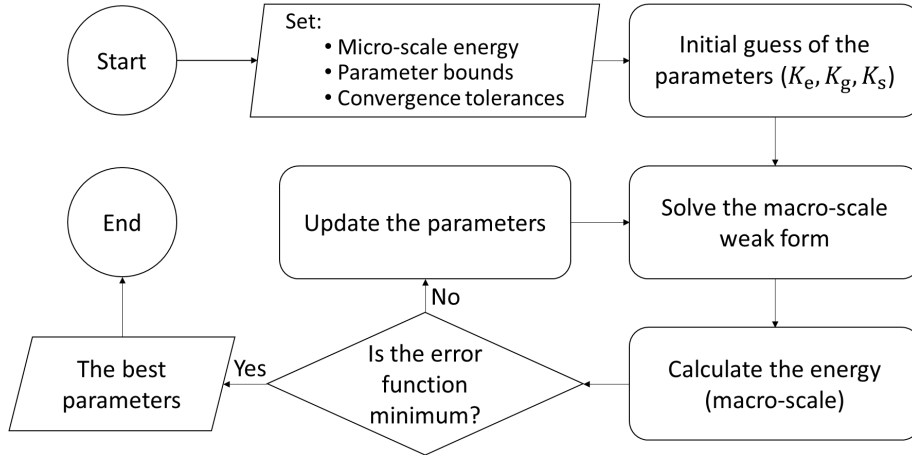
$$W_M(\boldsymbol{\varepsilon}, \boldsymbol{\kappa}, \gamma) = \frac{1}{2} \mathbf{K}_e (\varepsilon_1^2 + \varepsilon_2^2) + \frac{1}{2} \mathbf{K}_g (\kappa_1^2 + \kappa_2^2) + \frac{1}{2} \mathbf{K}_s \gamma^2$$

$$-\int_{B_0} \frac{\partial W_M(\boldsymbol{\varepsilon}, \boldsymbol{\kappa}, \gamma)}{\partial u_{i,j}} \delta u_{i,j} dV + \int_{\partial B_0^N} \hat{t}_i \delta u_i dA = 0$$



# Optimization problem

## Numerical Identification



# Optimization

## Numerical Identification

- Optimization function: *scipy.optimize.least\_squares* (from Python)
- Optimization method: Trust Region Reflective (*trf*) algorithm

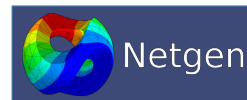
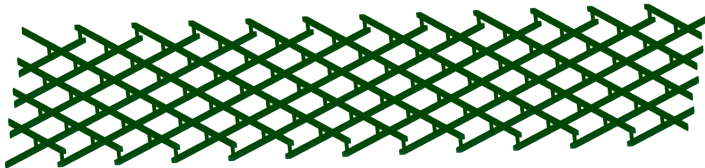


FENICS  
PROJECT

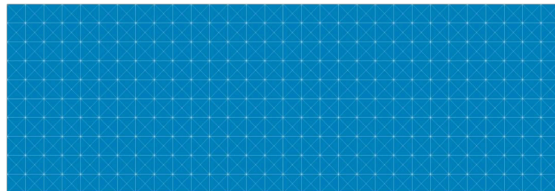
[https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least\\_squares.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html)

# Modeling

- Creating 3D CAD model and meshing in SALOME
- 230k degrees of freedom



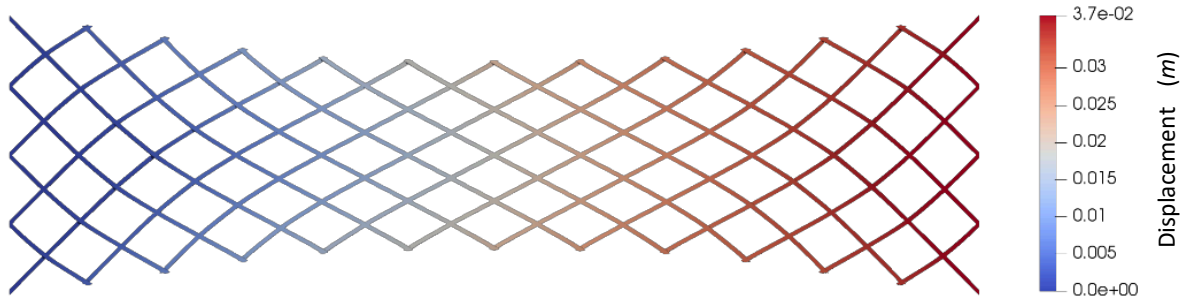
- Creating 2D homogenized model in FEniCS
- 5k degrees of freedom
- Simulate a tensile test





# Results

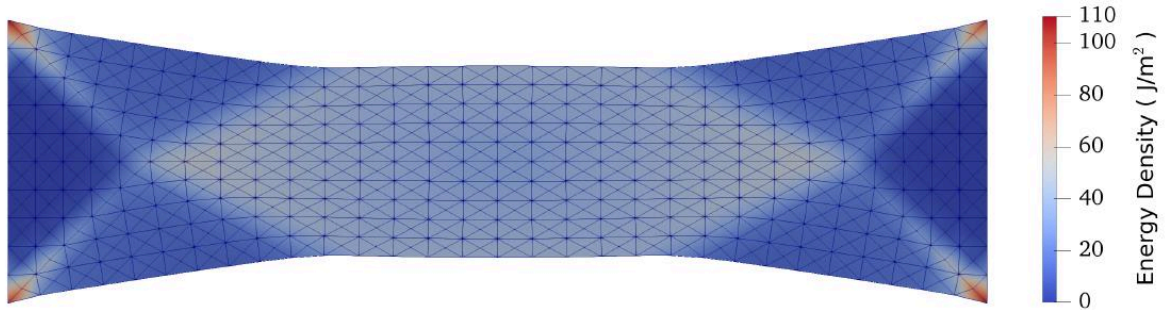
- Micro-scale model results:
  - Plot of displacement (17.6 % normal strain)



Shekarchizadeh, N, Abali, BE, Barchiesi, E, Bersani, AM. Inverse analysis of metamaterials and parameter determination by means of an automatized optimization problem. *Z Angew Math Mech.* 2021;e202000277

# Results

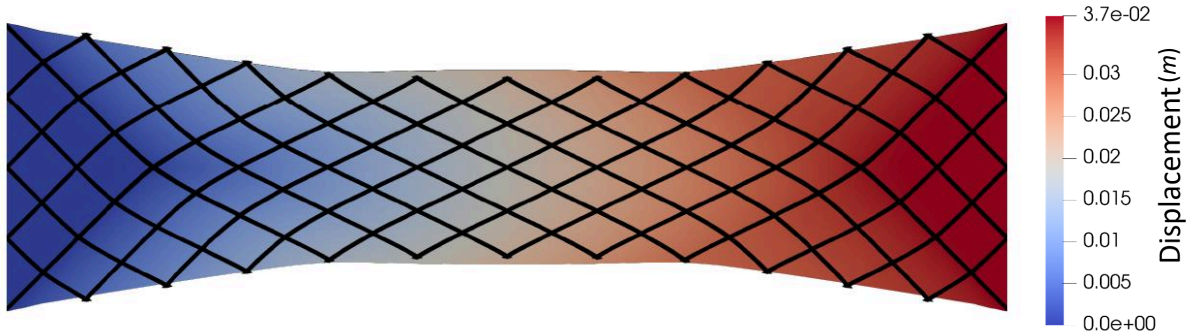
- Macro-scale model results:
  - Plot of energy



Shekarchizadeh, N, Abali, BE, Barchiesi, E, Bersani, AM. Inverse analysis of metamaterials and parameter determination by means of an automatized optimization problem. *Z Angew Math Mech.* 2021;e202000277

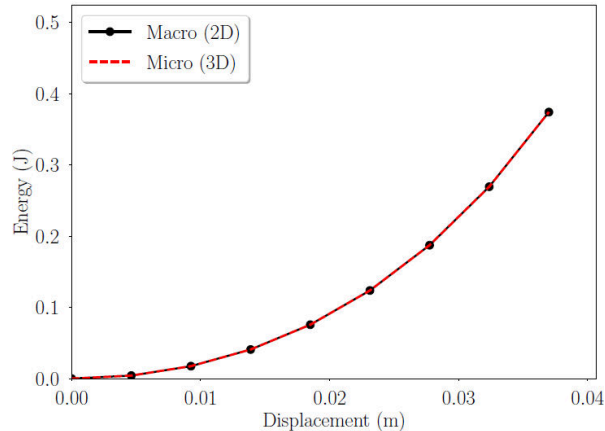
# Results

- Comparing the models:
  - Displacement plot: micro-scale (in black), macro-scale (in color)



# Results

- Numerical identification results:



# Results

- Numerical identification results:

## Constitutive Parameters

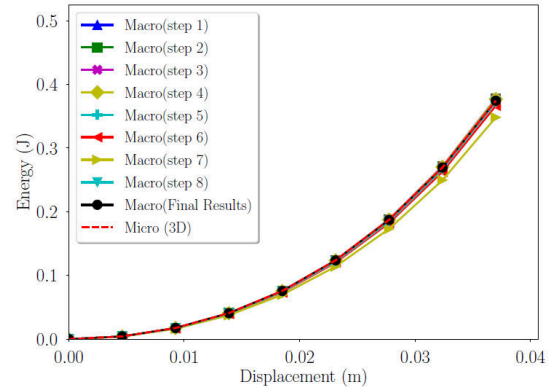
Parameter	Initial Guess	Final Results
$K_e$ (N/m)	$K_e^0 = \frac{E w_b h_b}{p_b} = 2.107 \times 10^5$	$1.406 \times 10^5$
$K_g$ (Nm)	$K_g^0 = \frac{E I_z}{p_b} = 1.756 \times 10^{-2}$	$2.699 \times 10^{-2}$
$K_s$ (N/m)	$K_s^0 = \frac{G \pi d_p^4}{32 h_p p_b^2} = 1.364 \times 10^2$	$2.138 \times 10^2$

# Results

- Numerical identification results:
  - Sensitivity analysis

Table 2: Sensitivity analysis

Parameter	Displacement (mm)							
	4.6	9.2	13.9	18.5	23.1	27.7	32.4	37.0
$K_e/K_e^0$	1.011	1.021	1.029	1.042	1.065	1.100	1.094	1.097
$K_g/K_g^0$	1.026	1.035	1.037	1.036	1.032	1.025	1.018	1.032
$K_s/K_s^0$	1.550	1.573	1.572	1.568	1.555	1.519	1.440	1.543



# Results

- Macro-scale model results:
  - Mesh convergence

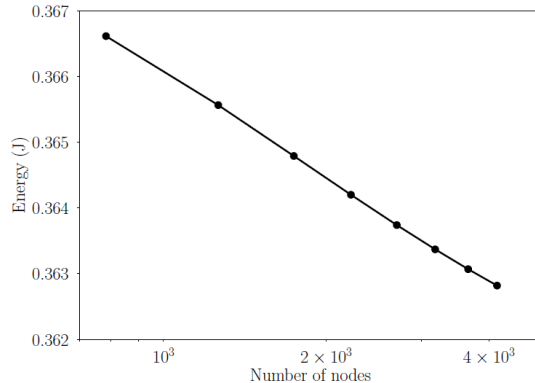


Table 3: Convergence results

	Number of nodes							
	783	1097	1469	1899	2379	2909	3497	4143
Energy (J)	0.3666	0.3655	0.3647	0.3641	0.3637	0.3633	0.3630	0.3628
Error (%)		0.29	0.21	0.16	0.13	0.10	0.08	0.07



# Conclusion

- Implementing a novel optimization procedure for the numerical identification of the parameters
- Consistency of the micro-scale and the macro-scale models in terms of deformation and energy
- Efficiency and robustness of the Trust Region Reflective Algorithm
- Robustness of the developed code by checking the sensitivity

**Shekarchizadeh, N**, Abali, BE, Barchiesi, E, Bersani, AM. Inverse analysis of metamaterials and parameter determination by means of an automatized optimization problem. *Z Angew Math Mech.* 2021;e202000277





**THANK YOU**

FOR YOUR KIND ATTENTION

# Total generalized variation for piecewise constant functions

Lukas Baumgärnter, Humboldt-Universität zu Berlin, Germany

Stephan Schmidt, Humboldt-Universität zu Berlin, Germany

Roland Herzog, Technische Universität Chemnitz, Germany

Ronny Bergmann, Technische Universität Chemnitz, Germany

José Vidal-Núñez, University of Alcalá, Spain

26 March 2021

The total generalized variation (TGV) was introduced as a generalization to the total variation regularizer to avoid the staircasing effect. In particular, the kernel of second-order TGV consists of linear polynomials.

Strictly speaking, the functional measures TV on piecewise constant data, however, the concept has become state of the art for reconstruction of pixel images using an appropriate discretization.

A discrete version of second-order TGV for piecewise constant functions on triangulated meshes is presented in the FEniCS framework. The non-smooth regularizer prefers equally distributed jumps over larger constant areas and thus prevents the visible staircasing effect compared to TV.

This is demonstrated for image denoising problems on structured as well as unstructured planar meshes using an implementation of the split Bregman method in FEniCS.

The functional is suitable for data on surfaces and can be extended to manifold valued data to measure the total generalized variation of the normal vector.

---

You can cite this talk as:

Lukas Baumgärnter, Stephan Schmidt, Roland Herzog, Ronny Bergmann, and José Vidal-Núñez. “Total generalized variation for piecewise constant functions”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 680. doi: 10.6084/m9.figshare.14495610.

BibTeX for this citation can be found at <https://mcsroggs.github.io/fenics2021/talks/baumgartner.html>.

# Efficient Hessian computation in deterministic and Bayesian inverse problems

Daniel I Gendin, Boston University, United States

Paul Barbone, Boston University, United States

26 March 2021

Inverse problem applications often require finding the minimum and Hessian of an optimization problem with partial differential equation constraints. Computing the Hessian of the cost functional is useful to estimate the uncertainty in the inverse problem solution from both a deterministic and Bayesian point of view. Direct computation of the Hessian, however, is prohibitively expensive for inverse problems with high dimensionality. We present a computational algorithm that computes the Hessian as a by-product of solving the inverse problem at practically no additional cost. It is based on solving using conjugate gradient (CG) inner iterations to solve for Newton updates in outer iterations to find the minimum. As an iterative matrix solver, an advantage of CG is that of short term recurrence preserves global conjugacy of the search directions, and therefore prior searches may be discarded. By saving conjugate directions and the action of the Hessian on those directions, we show that we can recover the full Hessian while computing the minimum. We present the algorithm in weak form in Hilbert space, and implement it in FEniCS. We verify the implementation in simulated inverse problems of modest size, and demonstrate its applicability to real data in an application of ultrasound elastography.

---

You can cite this talk as:

Daniel I Gendin and Paul Barbone. “Efficient Hessian computation in deterministic and Bayesian inverse problems”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 681–692. DOI: 10.6084/m9.figshare.14495613.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/gendin.html>.

# Efficient Hessian computation in deterministic and Bayesian inverse problems

Daniel I. Gendin, Paul E. Barbone

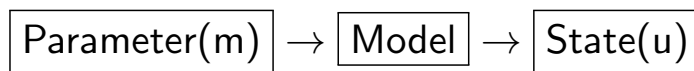
Boston University

Acknowledgment: NIH R01 CA195527

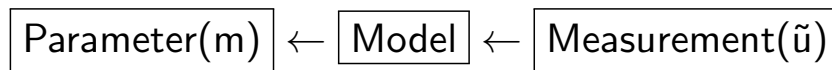
March 26, 2021

# Introduction

- Forward problem



- Inverse problem



- Bayes's Theorem

$$P_{\text{post}}(m|\tilde{u}) \propto P_{\text{like}}(\tilde{u}|m)P_{\text{prior}}(m) \quad (1)$$

- Posterior probability is easy to evaluate but difficult to interpret.
- How do we characterize the posterior?

# Background

- Sampling full posterior (e.g. MCMC)
  - [Petra et al., 2014, Bardsley et al., 2020, Chen and Ghattas, 2020, Vigliotti et al., 2018, Zou et al., 2019]
- Laplace approximation:  $P_{\text{post}}(\mathbf{m}) \approx N(\bar{\mathbf{m}}, H^{-1}[\bar{\mathbf{m}}])$ 
  - [Bui-Thanh et al., 2013, Saibaba et al., 2020, Chang et al., 2014, Fatehiboroujeni et al., 2020, Cui et al., 2016]
- Approximating the Hessian
  - [Saibaba et al., 2020, Ambartsumyan et al., 2020, Flath et al., 2011]
- In this work we find the MAP through Newton-CG and approximate the Hessian by using the Krylov basis found in computing the MAP.
  - This gives us the Hessian “for free.”
  - Finding the MAP is a constrained optimization problem.

# Optimization formulation

- Cost = -Log posterior

$$\underbrace{C(m)}_{-\log(P_{\text{post}}(m|\tilde{u}))} = \underbrace{\frac{1}{2} \|\tilde{u} - u(m)\|_{\text{noise}}^2}_{-\log(P_{\text{like}}(\tilde{u}|m))} + \underbrace{\frac{1}{2} R(m, m)}_{-\log(P_{\text{prior}}(m))} . \quad (2)$$

- Constraint equation (weak form).

$$a(\hat{w}, u; m) = l(\hat{w}) \quad \forall \hat{w} \in \mathcal{W}. \quad (3)$$

- Laplace approximation

- Close to the MAP point

$$C(m) = C(\bar{m}) + \cancel{(G[\bar{m}], m - \bar{m})_m} + \frac{1}{2} H[\bar{m}](m - \bar{m}, m - \bar{m}) + O(\|m - \bar{m}\|^3). \quad (4)$$

$$P_{\text{post}}(m) \propto \exp\left(\frac{1}{2} H[\bar{m}](m - \bar{m}, m - \bar{m})\right) \sim N(\bar{m}, H^{-1}[\bar{m}]) \quad (5)$$

# MAP evaluation

- We use a Newton-CG method to find the MAP point.
- Newton (outer) iterations
  - Tend to converge in few iterations
  - Consistent with Laplace approximation.
  - Explicit construction of full Hessian is prohibitive
- Preconditioned-CG (inner) iterations
  - Requires only the action of the Hessian in the search directions.
  - Constructs a Krylov space of H-conjugate search directions  $\{p\}$  and R-orthogonal gradients  $\{r\}$ .
  - Algorithm theoretically converges in  $K_d + 1$  steps, where  $K_d$  is the rank of the data part of the Hessian.



# Efficient Hessian evaluation

- Given preconditioned-CG products  $p_a$ ,  $r_a$  and  $q_a$  and  $s_a$ :

## Main result

$$\begin{aligned} H[m^n](\delta m_a, \delta m_b) = & \sum_{j=1}^k \frac{1}{D_{jj}} (\delta m_a, q_j)(q_j, \delta m_b) - \sum_{j=0}^{k-1} \frac{1}{C_{jj}} (\delta m_a, s_j)(s_j, \delta m_b) \\ & + R(\delta m_a, \delta m_b) \end{aligned} \quad (6)$$

- Where:

$$(v, q_a) = H[m^n](v, p_a) \quad \forall v \in \mathcal{M} \quad (7)$$

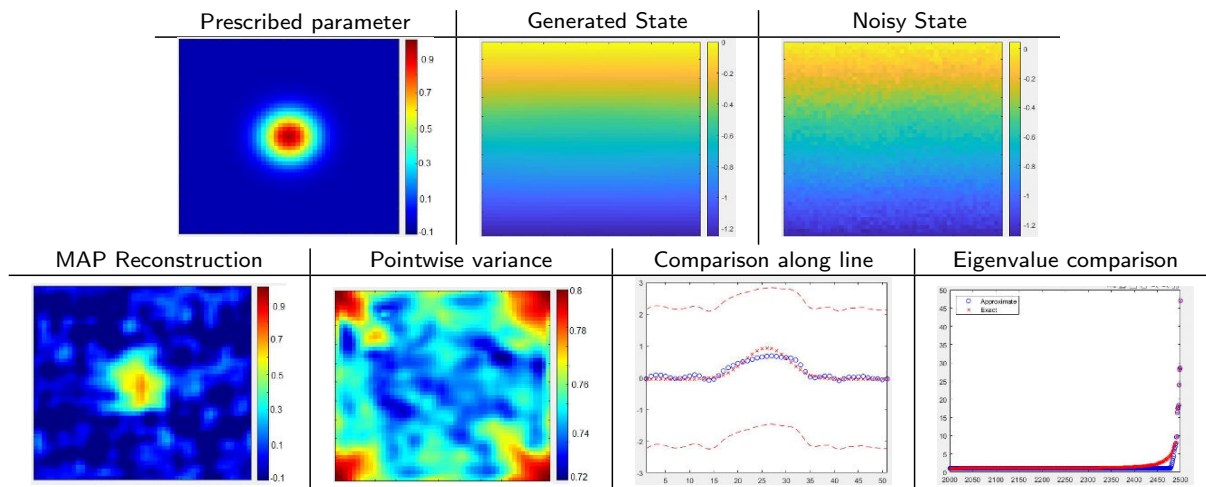
$$(v, s_a) = R(v, r_a) \quad \forall v \in \mathcal{M} \quad (8)$$

$$D_{aa} = H[m^n](p_a, p_a) \quad (\text{no sum}) \quad \forall a \in \{1, \dots, k\} \quad (9)$$

$$C_{aa} = R(r_a, r_a) \quad (\text{no sum}) \quad \forall a \in \{0, \dots, k-1\}. \quad (10)$$

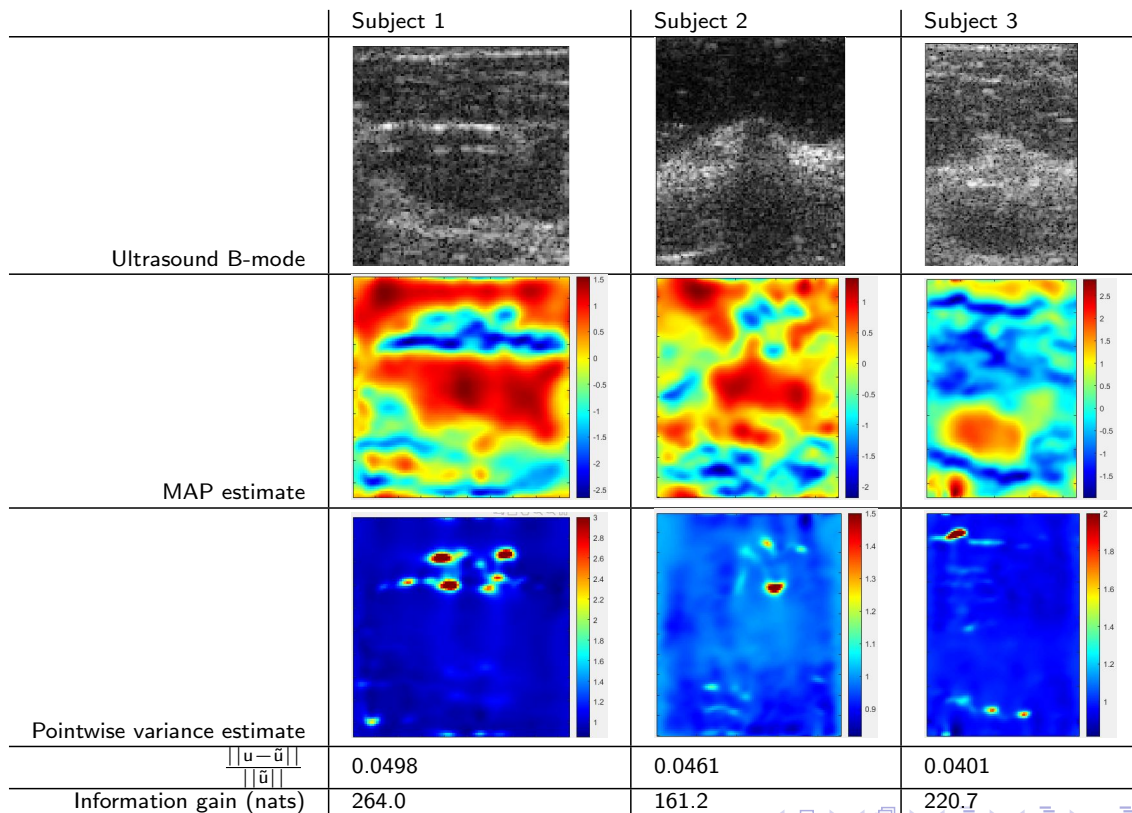
# Simulated data

- We consider the inverse elasticity problem where the state  $u$  is displacements and parameter  $m = \log(\text{Shear Modulus})$ .



# Elastic modulus maps of breast masses: UQ

Data courtesy of M. Fatemi at Mayo Clinic and T.J. Hall at University of Wisconsin



# Conclusions

- We develop a method to construct an approximation for the Hessian “for free” using components obtained during the process of optimization.
- Our method takes advantage of the conjugacies of the directions that comprise the Krylov space used to build the solution.
- The UFL interface in FEniCS facilitates easy implementation of our method.
- Thank you!

# Sources I

- [Ambartsumyan et al., 2020] Ambartsumyan, I., Boukaram, W., Bui-Thanh, T., Ghattas, O., Keyes, D., Stadler, G., Turkiyyah, G., and Zampini, S. (2020).  
Hierarchical matrix approximations of hessians arising in inverse problems governed by pdes.  
*SIAM Journal on Scientific Computing*, 42(5):A3397–A3426.
- [Bardsley et al., 2020] Bardsley, J. M., Cui, T., Marzouk, Y. M., and Wang, Z. (2020).  
Scalable optimization-based sampling on function space.  
*SIAM Journal on Scientific Computing*, 42(2):A1317–A1347.
- [Bui-Thanh et al., 2013] Bui-Thanh, T., Ghattas, O., Martin, J., and Stadler, G. (2013).  
A computational framework for infinite-dimensional bayesian inverse problems part i: The linearized case, with application to global seismic inversion.  
*SIAM Journal on Scientific Computing*, 35(6):A2494–A2523.
- [Chang et al., 2014] Chang, J. C., Savage, V. M., and Chou, T. (2014).  
A path-integral approach to bayesian inference for inverse problems using the semiclassical approximation.  
*Journal of Statistical Physics*, 157(3):582–602.
- [Chen and Ghattas, 2020] Chen, P. and Ghattas, O. (2020).  
Projected stein variational gradient descent.  
*arXiv preprint arXiv:2002.03469*.
- [Cui et al., 2016] Cui, T., Marzouk, Y., and Willcox, K. (2016).  
Scalable posterior approximations for large-scale bayesian inverse problems via likelihood-informed parameter and state reduction.  
*Journal of Computational Physics*, 315:363–387.
- [Fatehiboroujeni et al., 2020] Fatehiboroujeni, S., Petra, N., and Goyal, S. (2020).  
Linearized bayesian inference for young's modulus parameter field in an elastic model of slender structures.  
*Proceedings of the Royal Society A*, 476(2238):20190476.

# Sources II

- [Flath et al., 2011] Flath, H. P., Wilcox, L. C., Akçelik, V., Hill, J., van Bloemen Waanders, B., and Ghattas, O. (2011). Fast algorithms for bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial hessian approximations.  
*SIAM Journal on Scientific Computing*, 33(1):407–432.
- [Petra et al., 2014] Petra, N., Martin, J., Stadler, G., and Ghattas, O. (2014). A computational framework for infinite-dimensional bayesian inverse problems, part ii: Stochastic newton mcmc with application to ice sheet flow inverse problems.  
*SIAM Journal on Scientific Computing*, 36(4):A1525–A1555.
- [Saibaba et al., 2020] Saibaba, A. K., Chung, J., and Petroske, K. (2020). Efficient krylov subspace methods for uncertainty quantification in large bayesian linear inverse problems.  
*Numerical Linear Algebra with Applications*, 27(5):e2325.
- [Spantini et al., 2015] Spantini, A., Solonen, A., Cui, T., Martin, J., Tenorio, L., and Marzouk, Y. (2015). Optimal low-rank approximations of bayesian linear inverse problems.  
*SIAM Journal on Scientific Computing*, 37(6):A2451–A2487.
- [Vigliotti et al., 2018] Vigliotti, A., Csányi, G., and Deshpande, V. (2018). Bayesian inference of the spatial distributions of material properties.  
*Journal of the Mechanics and Physics of Solids*, 118:74–97.
- [Villa et al., 2019] Villa, U., Petra, N., and Ghattas, O. (2019). hippylib: an extensible software framework for large-scale inverse problems governed by pdes; part i: deterministic inversion and linearized bayesian inference.  
*arXiv preprint arXiv:1909.03948*.
- [Zou et al., 2019] Zou, Z., Mukherjee, S., Antil, H., and Aquino, W. (2019). Adaptive particle-based approximations of the gibbs posterior for inverse problems.  
*arXiv preprint arXiv:1907.01551*.

# Dissipative materials models with Materiaux and FEniCS

**Matthias Rambausek**, LMS, C.N.R.S., École Polytechnique, Institut Polytechnique de Paris, France  
Dipayan Mukherjee, LMS, C.N.R.S., École Polytechnique, Institut Polytechnique de Paris, France  
Konstantinos Danas, LMS, C.N.R.S., École Polytechnique, Institut Polytechnique de Paris, France

26 March 2021

In this contribution we introduce the material modeling framework “Materiaux” [3], which is partly based on the FEniCS technologies. Materiaux allows for the convenient development of complicated material models (including plasticity and dissipative ferromagnetism) in Python and provides the necessary integration bits to directly use the models in FEniCS/dolfin simulations. While the present implementation of the FEniCS bindings are based on subclassing `dolfin::Expression`, the structure of the models created is much more general such that they in principle allow for bindings in `dolfinx` and other FE packages. After outlining the basic structure of Materiaux we demonstrate the capabilities of Materiaux in combination with FEniCS by an application to magnetically hard [1] [2] and soft viscoelastic magnetorheological elastomers.

## References

- [1] Mukherjee, D., Danas, and K. “An evolving switching surface model for ferromagnetic hysteresis”. In: *Journal of Applied Physics* 125.3:033902 (2019). DOI: 10.1063/1.5051483.
- [2] Mukherjee, D., Rambausek, M., Danas, and K. “An explicit dissipative model for isotropic hard magnetorheological elastomers”. In: *Journal of the Mechanics and Physics of Solids* (2021). DOI: 10.1016/j.jmps.2021.104361.
- [3] Rambausek and M. “The Materiaux material modeling suite (Version v0.1.post1)”. In: *Zenodo* (2021). DOI: 10.5281/zenodo.4479295.

---

You can cite this talk as:

Matthias Rambausek, Dipayan Mukherjee, and Konstantinos Danas. “Dissipative materials models with Materiaux and FEniCS”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 693. DOI: 10.6084/m9.figshare.14495619.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/rambausek.html>.

# Reduced order methods for optimal flow control: FEniCS-based applications

**Maria Strazzullo** (<https://people.sissa.it/~mstrazzu>), SISSA (International School for Advanced Studies), Italy

Francesco Ballarin (<https://www.francescoballarin.it>), Catholic University of the Sacred Heart, Italy

Gianluigi Rozza (<https://people.sissa.it/~grozza>), SISSA (International School for Advanced Studies), Italy

26 March 2021

Optimal control is a powerful mathematical tool that can be used to fill the gap between collected data and equations, making the model more reliable and precise in the prediction of physical phenomena. However, optimal control problems are usually costly, most of all in a parametrized setting where many evaluations of the problem must be run to have a more comprehensive knowledge of the whole system. Reduced order methods (ROMs) help us to tackle this issue. Indeed, they aim at describing the parametric nature of the optimality system in a low-dimensional framework, accelerating the system solutions, maintaining the model accuracy. The talk aims at showing an overview of several applications in this topic through FEniCS-based [1] libraries, RBniCS and Multiphenics [4] [2], developed to deal with parametric partial differential equations. After describing the general problem formulation and the basic ideas behind ROMs, we will show many numerical results in the optimal control field highlighting the potential of FEniCS and of the two libraries for these very complex problems, moving from steady linear problems to nonlinear time-dependent ones [3] [5] [6].

## References

- [1] A. Logg, K. Mardal, and G. Wells. “Automated Solution of Differential Equations by the Finite Element Method”. 2012.
- [2] “Multiphenics project”. <https://mathlab.sissa.it/multiphenics>.
- [3] F. Pichi, M. Strazzullo, F. Ballarin, and G. Rozza. “Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: Application to Navier–Stokes equations and model reduction”. In: *Submitted* (2020).
- [4] “RBniCS - reduced order modelling in FEniCS”. <https://www.rbnicsproject.org>. 2015.
- [5] M. Strazzullo, F. Ballarin, and G. Rozza. “POD-Galerkin model order reduction for parametrized nonlinear time dependent optimal flow control: an application to Shallow Water Equations”. In: *Submitted* (2020).
- [6] M. Strazzullo, F. Ballarin, and G. Rozza. “POD-Galerkin model order reduction for parametrized time dependent linear quadratic optimal control problems in saddle point formulation”. In: *Journal of Scientific Computing* 83.55 (2020). DOI: 10.1007/s10915-020-01232-x.

---

You can cite this talk as:

Maria Strazzullo, Francesco Ballarin, and Gianluigi Rozza. “Reduced order methods for optimal flow control: FEniCS-based applications”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 694–709. DOI: 10.6084/m9.figshare.14495628.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/strazzullo.html>.



# Reduced order methods for optimal flow control: FEniCS-based applications



**SISSA**

**Maria Strazzullo**

mathLab, Mathematics Area, SISSA International School for  
Advanced Studies, Trieste, Italy

26 March 2021

FEniCS 2021



# Motivations

## Starting Point

Reduced Order Methods (ROMs) for parameterized Optimal Flow Control Problems (OFCPs) in environmental sciences

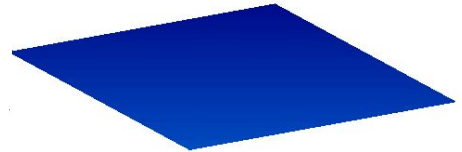
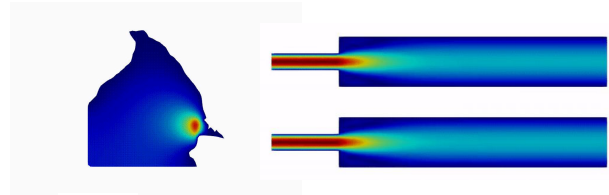
### PDEs-based

Several simulations for different values of physical and/or geometrical parameters  
(Uncertainty Quantification, Parameter Estimation...)

OFCPs

### DATA-based

Scattered  
expensive and difficult to collect  
complex to interpret



# Motivations

## Starting Point

Reduced Order Methods (ROMs) for parameterized Optimal Flow Control Problems (OFCPs) in environmental sciences

### ROMs

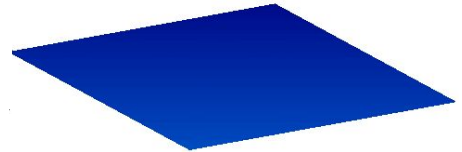
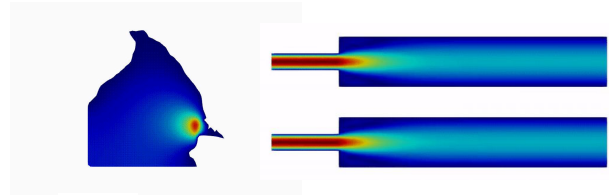
#### PDEs-based

Several simulations for different values of physical and/or geometrical parameters  
(Uncertainty Quantification, Parameter Estimation...)

#### OFCPs

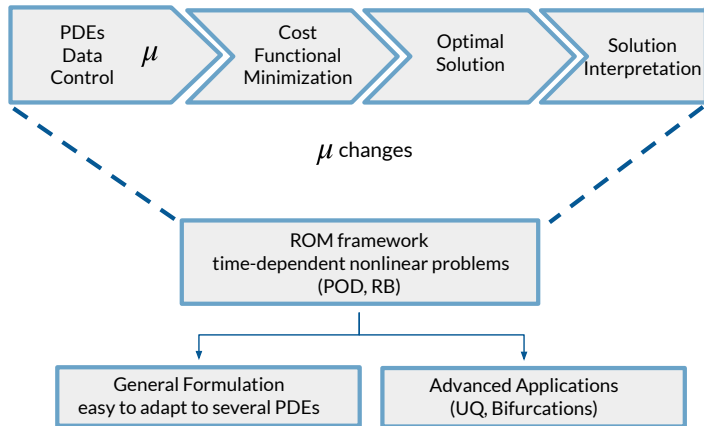
#### DATA-based

Scattered  
expensive and difficult to collect  
complex to interpret



By Dan Copsey (DanCopsey1 at English Wikipedia) — Own work, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=1692219>

# The Optimal Control Pipeline



<https://mathlab.sissa.it/multiphenics>

<https://www.rbnicsproject.org/>

## Outline:

1. Problem Formulation
2. Ideas behind ROMs
3. Advanced Applications
  - Shallow Waters Equations (SWE)
  - Steering Bifurcations
  - Uncertainty Quantification (UQ)

## Advisors:

Prof. Gianluigi Rozza (SISSA)  
Dr. Francesco Ballarin (UNICATT)

## Collaborators:

Prof. Rob Stevenson (UvA)  
Giuseppe Carere (UvA)  
Dr. Federico Pichi (SISSA/EPFL)

# Problem Formulation

$$\min_{(y,u) \in \mathcal{Y} \times \mathcal{U}} J(y, u; \boldsymbol{\mu}) := \min_{(y,u) \in \mathcal{Y} \times \mathcal{U}} \frac{1}{2} \int_0^T \|y(\boldsymbol{\mu}) - y_d(\boldsymbol{\mu})\|_{Y(\Omega_{\text{OBS}})}^2 dt + \int_0^T \frac{\alpha}{2} \|u(\boldsymbol{\mu})\|_{U(\Omega_u)}^2 dt$$

such that  $\mathcal{E}(y, u; \boldsymbol{\mu}) = 0$

$$(\text{DIM} = 3\mathcal{N}) \begin{cases} D_y \mathcal{L}((y, u, p); y_d, \boldsymbol{\mu})[\omega] = 0 & \forall \omega \in \mathbb{Y}, \\ D_u \mathcal{L}((y, u, p); y_d, \boldsymbol{\mu})[\kappa] = 0 & \forall \kappa \in \mathbb{U}, \\ D_p \mathcal{L}((y, u, p); y_d, \boldsymbol{\mu})[\zeta] = 0 & \forall \zeta \in \mathbb{Y}. \end{cases}$$

Linear

Non-linear

Time-dependent

## PDEs

Graetz Flows (temperature)  
Advection Diffusion (pollutant)  
Shallow Waters (coastal management)  
Navier-Stokes (haemodynamics)

## Controls

Boundary, Part of the domain, the whole domain, forcing terms...  
Control **changes** the usual behaviour of the solution.

Optimal Control is a very powerful mathematical tool

- more reliable solutions
- great versatility

However it is **costly** (based on the solution of three equations)

# Problem Formulation

$$\min_{(y,u) \in \mathcal{Y} \times \mathcal{U}} J(y, u, \mu) := \min_{(y,u) \in \mathcal{Y} \times \mathcal{U}} \frac{1}{2} \int_0^T \|y(\mu) - y_d(\mu)\|_{Y(\Omega_{\text{OBS}})}^2 dt + \int_0^T \frac{\alpha}{2} \|u(\mu)\|_{U(\Omega_u)}^2 dt$$

such that  $\mathcal{E}(y, u, \mu) = 0$

$$(\text{DIM} = 3N) \begin{cases} D_y \mathcal{L}((y, u, p); y_d, \mu)[\omega] = 0 & \forall \omega \in \mathbb{Y}, \\ D_u \mathcal{L}((y, u, p); y_d, \mu)[\kappa] = 0 & \forall \kappa \in \mathbb{U}, \\ D_p \mathcal{L}((y, u, p); y_d, \mu)[\zeta] = 0 & \forall \zeta \in \mathbb{Y}. \end{cases}$$

Linear  
Non-linear  
Time-dependent

Optimal Control is a very powerful mathematical tool

- more reliable solution
- great versatility

However it is **costly** (based on the solution of three equations)  
**most of all in a parametrized setting**

## PDEs

Graetz Flows (temperature)  
Advection Diffusion (pollutant)  
Shallow Waters (coastal management)  
Navier-Stokes (haemodynamics)

## Controls

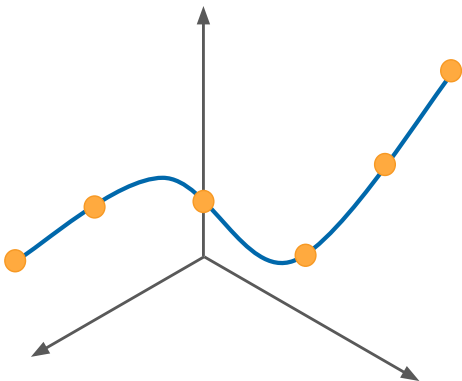
Boundary, Part of the domain, the whole domain, forcing terms...  
Control **changes** the usual behaviour of the solution.

## ROMs

Exploit the parametric structure of the problem

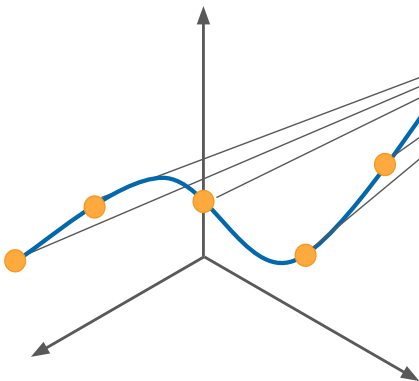
# Ideas behind ROMs

Explore the solutions **snapshots** and how they change with respect to the parameter



# Ideas behind ROMs

Explore the solutions **snapshots** and how they change with respect to the parameter



**Building phase**  
POD (SVD, PCA)  
RB (Adaptive technique)

$Z^T$

$A_N$

$Z^T$

$A$

$Z$

use  $\mu$

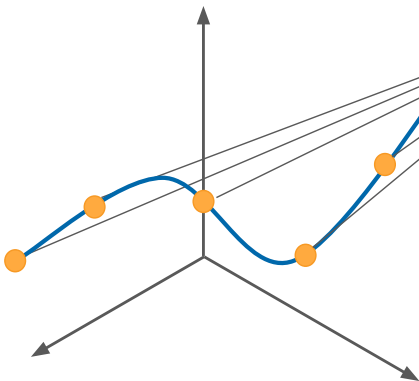
$A_N$

for each new parameter evaluation



# Ideas behind ROMs

Explore the solutions **snapshots** and how they change with respect to the parameter



**Building phase**  
POD (SVD, PCA)  
RB (Adaptive technique)

$Z^T$

$A_N$

$Z^T$

$A$

$Z$

Saving computational time

use  $\mu$   $A_N$  for each new parameter evaluation

# How do you use RBniCS?

## Example of Steady Coercive State Equation

```
class EllipticOptimalControl(EllipticOptimalControlProblem):
```

```
    # Default initialization of members
    def __init__(self, V, **kwargs):
        # Call the standard initialization
        EllipticOptimalControlProblem.__init__(self, V, **kwargs)
        # ... and also store FEniCS data structures for assembly
        assert "subdomains" in kwargs
        assert "boundaries" in kwargs
        self.subdomains, self.boundaries = kwargs["subdomains"], kwargs["boundaries"]
        yup = TrialFunction(V)
        (self.y, self.u, self.p) = split(yup)
        zvq = TestFunction(V)
        (self.z, self.v, self.q) = split(zvq)
        self.dx = Measure("dx")(subdomain_data=subdomains)
        self.ds = Measure("ds")(subdomain_data=boundaries)
        # Regularization coefficient
        self.alpha = 0.01
        # Desired state
        self.y_d = Constant(1.0)
        # Customize linear solver parameters
        self._linear_solver_parameters.update({
            "linear_solver": "mumps"
        })
```

In the class you define the parameters multiplied by the various forms defined as in FEniCS

Simple modifications for more complicated problems or to use other algorithms

```
# 1. Read the mesh for this problem
```

```
mesh = Mesh("data/mesh1.xml")
subdomains = MeshFunction("size_t", mesh, "data/mesh1_physical_region.xml")
boundaries = MeshFunction("size_t", mesh, "data/mesh1_facet_region.xml")
```

```
# 2. Create Finite Element space (Lagrange P1)
```

```
scalar_element = FiniteElement("Lagrange", mesh.ufl_cell(), 1)
element = MixedElement(scalar_element, scalar_element, scalar_element)
V = FunctionSpace(mesh, element, components=["y", "u", "p"])
```

```
# 3. Allocate an object of the EllipticOptimalControl class
```

```
elliptic_optimal_control = EllipticOptimalControl(V, subdomains=subdomains, boundaries=boundaries)
mu_range = [(1.0, 3.5), (0.5, 2.5)]
elliptic_optimal_control.set_mu_range(mu_range)
```

```
# 4. Prepare reduction with a reduced basis method
```

```
pod_galerkin_method = PODGalerkin(elliptic_optimal_control)
pod_galerkin_method.set_Nmax(10)
```

```
# 5. Perform the offline phase
```

```
lifting_mu = (1.0, 1.0)
elliptic_optimal_control.set_mu(lifting_mu)
pod_galerkin_method.initialize_training_set(100)
reduced_elliptic_optimal_control = pod_galerkin_method.offline()
```

```
# 6. Perform an online solve
```

```
online_mu = (2.0, 2.0)
reduced_elliptic_optimal_control.set_mu(online_mu)
reduced_elliptic_optimal_control.solve()
reduced_elliptic_optimal_control.export_solution(filename="online_solution")
print("Reduced output for mu =", online_mu, "is", reduced_elliptic_optimal_control.compute_output())
```

# How do you use RBniCS?

## Example of Steady Coercive State Equation

```
class EllipticOptimalControl(EllipticOptimalControlProblem):
```

```
    # Default initialization of members
    def __init__(self, V, **kwargs):
        # Call the standard initialization
        EllipticOptimalControlProblem.__init__(self, V, **kwargs)
        # ... and also store FEniCS data structures for assembly
        assert "subdomains" in kwargs
        assert "boundaries" in kwargs
        self.subdomains, self.boundaries = kwargs["subdomains"], kwargs["boundaries"]
        yup = TrialFunction(V)
        (self.y, self.u, self.p) = split(yup)
        zvq = TestFunction(V)
        (self.z, self.v, self.q) = split(zvq)
        self.dx = Measure("dx")(subdomain_data=subdomains)
        self.ds = Measure("ds")(subdomain_data=boundaries)
        # Regularization coefficient
        self.alpha = 0.01
        # Desired state
        self.y_d = Constant(1.0)
        # Customize linear solver parameters
        self._linear_solver_parameters.update({
            "linear_solver": "mumps"
        })
```

In the class you define the parameters multiplied by the various forms defined as in FEniCS

Simple modifications for more complicated problems or to use other algorithms

```
# 1. Read the mesh for this problem
```

```
mesh = Mesh("data/mesh1.xml")
subdomains = MeshFunction("size_t", mesh, "data/mesh1_physical_region.xml")
boundaries = MeshFunction("size_t", mesh, "data/mesh1_facet_region.xml")
```

```
# 2. Create Finite Element space (Lagrange P1)
```

```
scalar_element = FiniteElement("Lagrange", mesh.ufl_cell(), 1)
element = MixedElement(scalar_element, scalar_element, scalar_element)
V = FunctionSpace(mesh, element, components=["y", "u", "p"])
```

```
# 3. Allocate an object of the EllipticOptimalControl class
```

```
elliptic_optimal_control = EllipticOptimalControl(V, subdomains=subdomains, boundaries=boundaries)
mu_range = [(1.0, 3.5), (0.5, 2.5)]
elliptic_optimal_control.set_mu_range(mu_range)
```

```
# 4. Prepare reduction with a reduced basis method
```

```
pod_galerkin_method = PODGalerkin(elliptic_optimal_control)
pod_galerkin_method.set_Nmax(10)
```

```
# 5. Perform the offline phase
```

```
lifting_mu = (1.0, 1.0)
elliptic_optimal_control.set_mu(lifting_mu)
pod_galerkin_method.initialize_training_set(100)
reduced_elliptic_optimal_control = pod_galerkin_method.offline()
```

```
# 6. Perform an online solve
```

```
online_mu = (2.0, 2.)
reduced_elliptic_optimal_control.set_mu(online_mu)
reduced_elliptic_optimal_control.solve()
reduced_elliptic_optimal_control.export_solution(filename="online_solution")
print("Reduced output for mu =", online_mu, "is", reduced_elliptic_optimal_control.compute_output())
```

# Coastal Height Management

Minimisation of

$$\frac{1}{2} \int_0^T \int_{\Omega} (h - h_d(\mu_3))^2 dx dt + \frac{1}{2} \int_0^T \int_{\Omega} (\mathbf{v} - \mathbf{v}_d(\mu_3))^2 dx dt + \frac{\alpha}{2} \int_0^T \int_{\Omega} \mathbf{u}^2 dx dt$$

constrained to

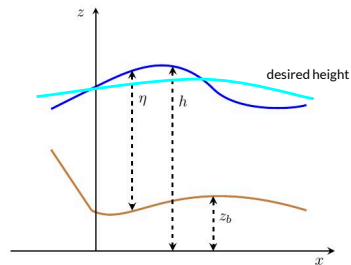
$$\mathbf{v}_t + \mu_1 \Delta \mathbf{v} + \mu_2 (\mathbf{v} \cdot \nabla) \mathbf{v} + g \nabla \eta - \mathbf{u} = 0 \quad \text{in } \Omega \times [0, T],$$

$$h_t + \operatorname{div}(\eta \mathbf{v}) = 0 \quad \text{in } \Omega \times [0, T],$$

$$\mathbf{v} = \mathbf{v}_0 \quad \text{on } \Omega \times \{0\},$$

$$h = h_0 \quad \text{on } \Omega \times \{0\},$$

$$\mathbf{v} = \mathbf{0} \quad \text{on } \partial\Omega \times [0, T].$$



**GOAL:** recover parametrized desired height and velocity profiles with distributed control



**Straightforward implementation in  
RBniCS with standard modifications  
of Nonlinear problems classes**

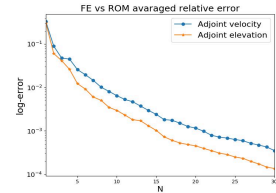
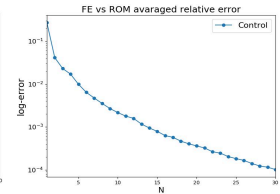
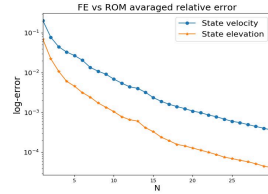
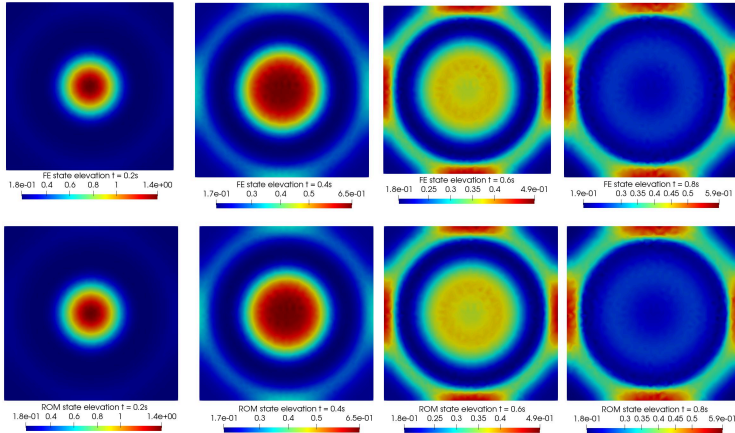
$$[0, T] = [0, 0.8],$$

$$\Omega = [0, 10] \times [0, 10],$$

$$\alpha = 10^{-5},$$

$$\Delta t = 0.1.$$

# Coastal Height Management



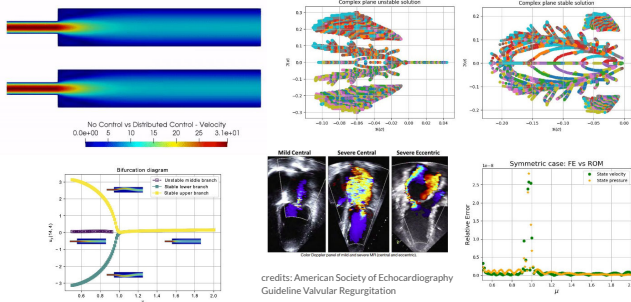
[Strazzullo, Ballarin, Rozza, POD-Galerkin Model Order Reduction for Parametrized Nonlinear Time Dependent Optimal Flow Control: an Application to Shallow Water Equations. *Submitted*, 2021]

Errors  $\sim 1e-4$ , Speedup  $\sim 30$   
ROM vs FE dim = 270 vs 94'016.

# Advanced Applications

## Optimal control + Bifurcations in Navier-Stokes:

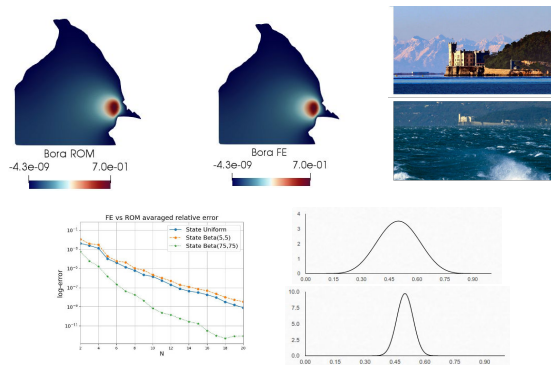
- Does optimal control affect the system?
- Does optimal control change stability?
- Does it change the stability analysis?
- What is the most “natural” configuration?



[Pichi, Strazzullo, Ballarin, Rozza, Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier-Stokes equations with model order reduction. *Submitted, 2020*]

## Optimal control + Uncertainty Quantification:

- Input-output relation?
- **Weighted-POD**
- More knowledge is better (less basis need)



[Carere, Strazzullo, Ballarin, Rozza, Stevenson, Weighted POD-reduction for parametrized PDE-constrained Optimal Control Problems with random inputs and its applications to environmental sciences *Submitted, 2021*]

# Conclusions

## ROMs and OCPs

- Great mathematical tool to reach a desired configuration
- Expensive to solve
- RBniCS and multiphenics application to OFCPs
  - great versatility
  - simple to code



## Applications

- POD for nonlinear time-dependent problems (SWEs)
- OFCPs and bifurcations
- OFCPs and UQ

<https://www.rbnicsproject.org/>

### Acknowledgements

European Union Funding for Research and Innovation -- Horizon 2020 Program -- in the framework of European Research Council Executive Agency: Consolidator Grant H2020 ERC CoG 2015 AROMA-CFD project 681447  
"Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics".



*Thank you for your attention!*

# A generic FEniCS-framework for moment approximations of Boltzmann's equation

Edilbert Christhuraj, RWTH Aachen University, Germany

Manuel Torrilhon, RWTH Aachen University, Germany

26 March 2021

The moment method is a technique which relates a microscopic description of a kinetic model to a macroscopic continuum model. In the context of nonequilibrium gas flows, the moment method is used to derive fluid dynamics equations from the Boltzmann equation which provides a kinetic description of the nonequilibrium. The resulting moment equations are a set of partial differential equations (PDEs) and inherently form a hierarchy due to the nature of the moment method. Using more moment variables result in a larger system with better physical accuracy. Following a similar procedure, a corresponding set of boundary conditions can be directly derived from the kinetic description. Ultimately the moment equations can be reformulated and rewritten as a hyperbolic system of first order PDEs. Our aim is to develop a generic framework which solves arbitrary moment systems and we hitherto developed a solver called 'FEniCS For Moment Equations' (F2ME). In this talk, we briefly look at a set of moment systems and describe how FEniCS is employed to solve these systems. Furthermore, we discuss the implementation of our approach in detail and demonstrate the validity of the developed solver by comparing it with analytical and experimental solutions.

---


You can cite this talk as:

Edilbert Christhuraj and Manuel Torrilhon. "A generic FEniCS-framework for moment approximations of Boltzmann's equation". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 710–721. DOI: 10.6084/m9.figshare.14495631.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/christhuraj.html>.



# A Generic Fenics-Framework For Moment Approximations of Boltzmann equation

 **Edilbert Christhuraj** and Prof. Manuel Torrilhon

Applied and Computational Mathematics  
RWTH Aachen  
Germany

FEniCS 2021



**RWTH**AACHEN  
UNIVERSITY

## Setting The Scene

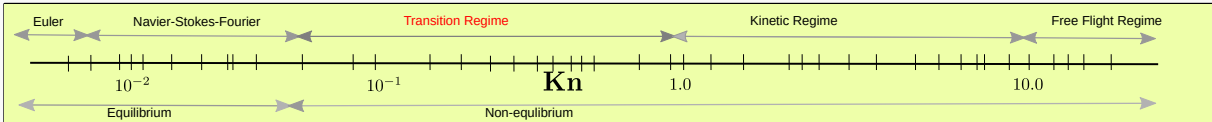
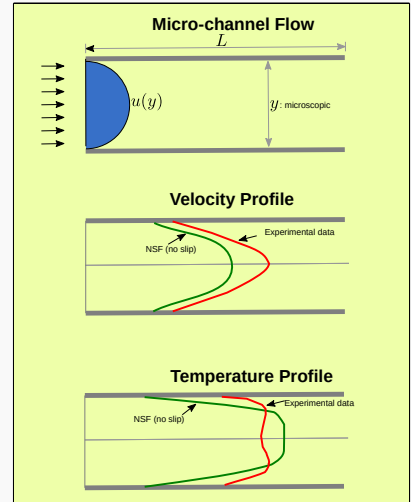
### 1. Need for new models

- We want to model non-equilibrium gas flows.
- E.g. consider a gas flow with temperature  $\theta$  and velocity  $u$ .
  - Gas velocity follows *Stokes* problem,  $\nabla \cdot \mathbf{u} = 0$ ,  $\nabla p = \nabla \mathbf{u}$ .
  - Gas temperature follows *Poisson* problem,  $\Delta \theta = 0$
- Classical models use equilibrium assumptions in their closure equations. E.g.
  - Fourier's heat conduction law  $q \sim \nabla \theta$
  - Stoke's law  $\sigma \sim (\nabla \mathbf{u})_{\text{dev}}$
- In non-equilibrium scenarios, these models are inadequate. (E.g., see micro-channel flow example)

### 2. Knudsen number

- is a dimensionless quantity in gas kinetic theory.
- is defined as  $\frac{\text{Mean free path}}{\text{Characteristic length scale}} = \frac{\lambda}{L}$ .
- is a good indicator of gas flow characterisation.
- When **Kn** is large,
  - it implies non-equilibrium gas flow.
  - $\lambda$  is very large, e.g. atmospheric entry flows, vacuum devices, or  $L$  is very small, e.g. micro-electro-mechanical devices.

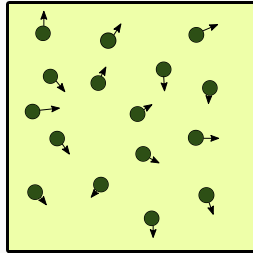
Find a continuum mechanics theory based on a system of partial differential equations to describe non-equilibrium gas flows accurately.



## Theoretical Foundation In A Nutshell

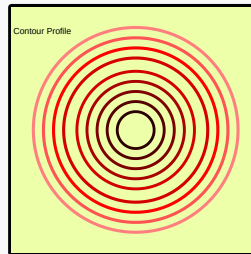
- Gas kinetic theory approach
- Gas as collection of particles
- Each particle has attributes
  - mass  $m$
  - position  $\mathbf{x}$
  - velocity  $\mathbf{c}$
- Computationally expensive

Microscopic Setting



Continuum Limit

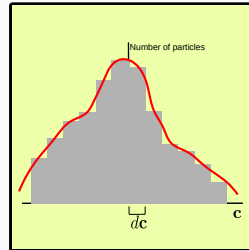
Macroscopic Setting



- Continuum equations
- Macroscopic variables
  - fluid velocity  $\mathbf{u}$
  - temperature  $\theta$
  - density  $\rho$
- Typically, partial differential equations with boundary conditions

Statistics

- Probability density function  $f(\mathbf{x}, \mathbf{c}, t)$
- Equilibrium distribution
 
$$f^{(eq)}(\mathbf{c}) = \frac{\rho/m}{\sqrt{2\pi\theta}^3} \exp\left(-\frac{(c_i - u_i)^2}{2\theta}\right)$$
- Boltzmann equation
 
$$\frac{\partial f}{\partial t} + c_i \frac{\partial f}{\partial x_i} = Q(f, f)$$



Mesoscopic Setting

Moment Method

1. Moment method is an approximative method to Boltzmann equation
2. It connects microscopic setting to macroscopic setting through statistical description

## Moment Method - An Overview

1. Choose a constant background process with reference values of our choice.
2. Approximate the distribution function  $f$  by

$$\tilde{f} \approx \sum_{\mathcal{I}} w_{\mathcal{I}} \psi_{\mathcal{I}}(\mathbf{c}) f_r(\mathbf{c}; \rho_r, \theta_r, u_i^{(r)}), \quad (1)$$

where

- $\mathcal{I}$  is an index set,
  - $w_{\mathcal{I}}$  are the coefficients,
  - $\psi_{\mathcal{I}}$  are the associated legendre polynomials,
  - $f_r$  is a reference maxiwellian (pseudo-equilibrium).
3. Relations between moment variables  $w_{\mathcal{I}}$  and macroscopic variables are possible:

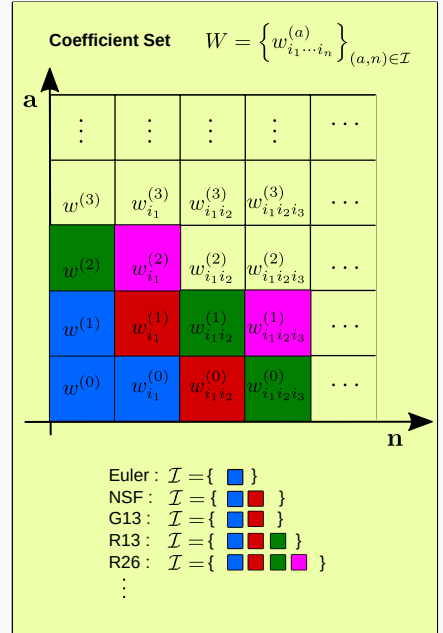
$$\{w^{(0)}, w_{i_1}^{(0)}, w^{(1)}, w_{i_1 i_2}^{(0)} w_{i_1}^{(1)}, \dots\} \Longleftrightarrow \{\rho, u_{i_1}, \theta, \sigma_{i_1 i_2}, q_{i_1}, \dots\}.$$

4. Galerkin-type orthogonal projection on the Boltzmann equation leads to moment equations.

$$\int_{\mathbb{R}^3} \bullet \left( \frac{\partial \tilde{f}}{\partial t} + c_i \frac{\partial \tilde{f}}{\partial x_i} \right) dc = \int_{\mathbb{R}^3} \bullet S(\tilde{f}, \tilde{f}) dc,$$

where  $\bullet$  is any suitable test function.

5. Choose different ansatz and index set  $\mathcal{I}$  to obtain different moment models.



## A Generic First order Formulation

- **Toy example:** Consider the following Poisson equation

$$-\Delta\theta = f \quad \longrightarrow \quad \begin{array}{l} \nabla \cdot q = f \\ \nabla\theta = -q \end{array} \quad \longrightarrow \quad \begin{array}{l} \partial_x q_x + \partial_y q_y = f \\ \partial_x \theta = -q_x \\ \partial_y \theta = -q_y \end{array}$$

- The above system can be written as a system of first order equations.

$$W = \begin{pmatrix} \theta \\ q_x \\ q_y \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \partial_x \begin{pmatrix} \theta \\ q_x \\ q_y \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \partial_y \begin{pmatrix} \theta \\ q_x \\ q_y \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta \\ q_x \\ q_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- **Key idea:** Formulate moment system as first order system. In two dimensions  $(x, y)$ , a system of  $m$  moment variables reads

$$A^{(x)} \partial_x W + A^{(y)} \partial_y W + P W = F,$$

where

- $U \in \mathbb{R}^m$  is the vector of  $m$  scalar moment variables,
- $A^{(i)} \in \mathbb{R}^{m \times m}$  is the transport matrix in  $i^{\text{th}}$  direction,
- $P \in \mathbb{R}^{m \times m}$  is the coefficient matrix of collision terms,
- $F \in \mathbb{R}^m$  is the vector of external forces.

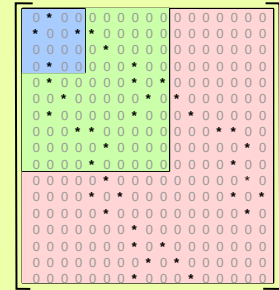
- Corresponding boundary conditions read

$$W^{(o)} = L \mathbb{A} W^{(e)} + g,$$

where

- $W^{(o)} \in \mathbb{R}^p$  is the vector of odd moments,
- $L \in \mathbb{R}^{p \times q}$  is the given boundary matrix,
- $W^{(e)} \in \mathbb{R}^q$  is the vector of even moments,
- $g \in \mathbb{R}^p$  is the vector of inhomogeneities.
- $\mathbb{A} \in \mathbb{R}^{p \times q}$  is defined in the slide 7

### System Matrix Hierarchy



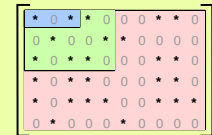
$$= A^{(x)}$$

G3

G13

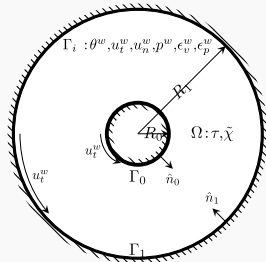
G26

### Boundary Matrix Hierarchy



$$= L$$

## Problem Setup - User's Perspective



Process : Flow over a cylinder  
 Quantities of interest : Velocity  $\mathbf{u}$ , Temperature  $\theta$ , Heat flux  $\mathbf{q}$   
 Geometry boundary : inner cylinder  $\Gamma_0$ , outer cylinder  $\Gamma_1$

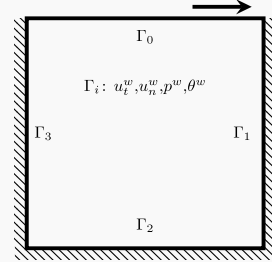
Typical questions a user may ask:

Given some values, e.g., pressure  $p$ , inflow, heat flux  $\mathbf{q}$ , tangential velocity  $u_t$ , on a boundary  $\Gamma_i$ ,

what is the velocity distribution?

what is the temperature distribution ?

⋮



Process : Lid driven cavity  
 Quantities of interest : Velocity  $\mathbf{u}$ , Temperature  $\theta$ , pressure  $\mathbf{p}$   
 Geometry boundary : moving wall  $\Gamma_0$ , stationary walls  $\Gamma_1, \Gamma_2, \Gamma_3$

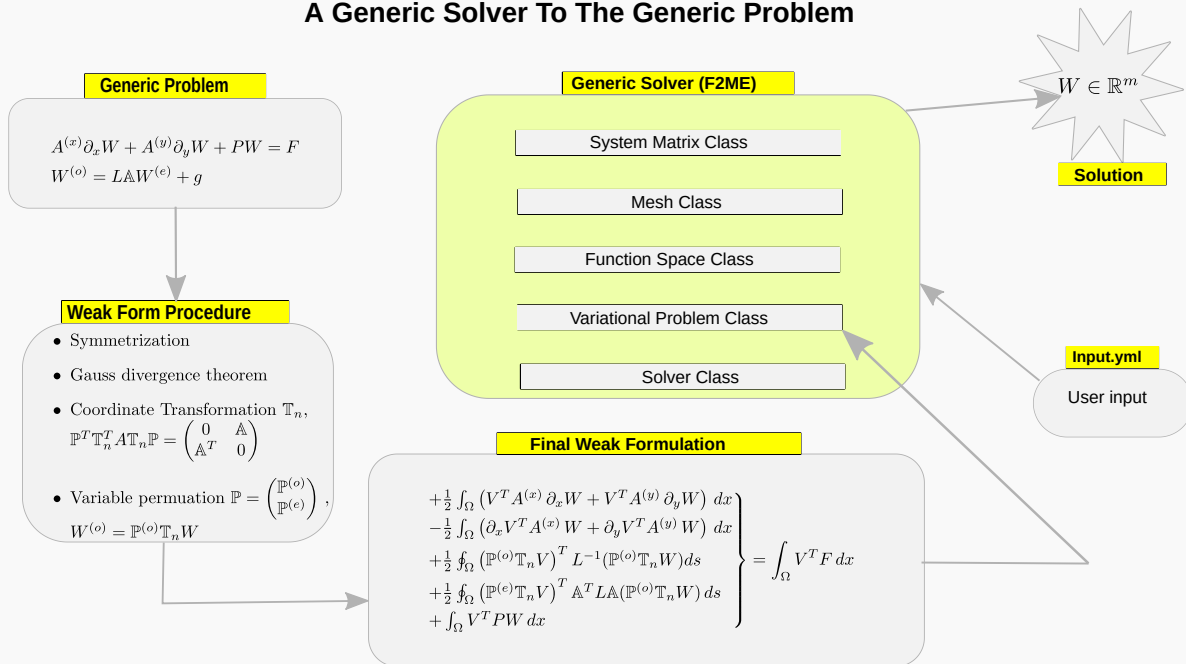
1. Processes are equipped with  **$\mathbf{Kn}$**  number.

- When  **$\mathbf{Kn}$**  is low, use classical models, e.g., NSF.
- When  **$\mathbf{Kn}$**  is (moderately) large, no need to solve full Boltzmann. Instead, solve moment models.

Choose a model so that it is accurate enough to describe the process.

2. Prescribe desired values on different boundaries.

## A Generic Solver To The Generic Problem



## F2ME

### f2me.py (main file)

**Name** Fenics For Moment Equations , hence the abbreviation **F2ME**

**Location** <https://www.gitlab.com/19ec94/f2me>  
or  
<https://git.rwth-aachen.de/19ec94/f2me>

**Usage** `$ git clone git@gitlab.com:19ec94/f2me.git`  
`$ cd f2me/f2me`  
`$ python3 f2me.py input.yml`

### Features

Built upon FEniCS  
Generalised solver  
Highly modularised  
Easily extendable to other moment models  
Easily customisable to needs of an user  
Docker support available

Import custom modules

Read user input

Mesh using Gmsh

VectorFunctionSpace()

Prepare system matrices

Create & assemble system

Inbuilt solver (mumps)

```

...
import dolfin as df
...
df.parameters['ghost_mode'] = 'shared_facet'

# Import custom modules
from modules.SystemMatrix import SystemMatrix
from modules.Mesh import H2Mesh
from modules.FunctionSpace import FunctionSpace
from modules.FormulateVariationalProblem import \
    FormulateVariationalProblem
from modules.Solver import Solver

# Read and Store user input
if len(sys.argv) > 1:
    user_given_input_file = sys.argv[1]
else:
    print("Provide an input file")
    quit()
with open(user_given_input_file, 'r') as input_file:
    my_input_cls = yaml.load(input_file, \
        Loader=yaml.FullLoader)

# Main code starts
for current_mesh in range(len(my_input_cls['mesh_list'])):
    my_current_mesh = my_input_cls['mesh_list'][current_mesh]

    # Read and Process MESH info
    my_mesh_cls = H2Mesh(my_current_mesh)

    # Set up FUNCTION SPACE
    problem_type = my_input_cls['problem_type']
    number_of_moments = my_input_cls['number_of_moments']
    my_function_space_cls = FunctionSpace(my_input_cls, \
        my_mesh_cls)
    my_function_space_cls.set_function_space()

    # Create SYSTEM MATRIX
    my_system_matrix_cls = SystemMatrix(my_input_cls, \
        my_mesh_cls)
    my_system_matrix_cls.convert_to_ufl_form()

    # Create and Assemble VARIATIONAL PROBLEM
    my_var_prob_cls = FormulateVariationalProblem(
        my_input_cls, my_mesh_cls,
        my_system_matrix_cls, my_function_space_cls
    )
    my_var_prob_cls.create_lhs() #Left Hand Side
    my_var_prob_cls.create_rhs() #Right Hand Side

    # Call SOLVER
    my_solver_cls = Solver(my_input_cls, \
        my_function_space_cls, my_var_prob_cls)
    if problem_type == 'nonlinear':
        my_solver_cls.inbuilt_newton_solver()
        u = my_solver_cls.u
    else:
        my_solver_cls.inbuilt_linear_solver()
        u = my_solver_cls.u.Function

    # Start POST-PROCESSING
    sol = u.split()

```



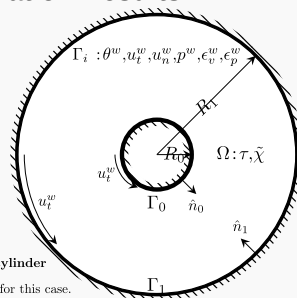
## Simulation Results

### input.yml

```

problem_type: 'linear' # 'linear' or 'nonlinear'
moment_order: 13 # 3 or 6 or 13 or 'grad13' or 'ns'
number_of_moments: 17 # 6 or 9 or 10 or 17
mesh_list: # Provide mesh list
- ../mesh/ring0.h5
- ../mesh/ring1.h5
- ../mesh/ring2.h5
- ../mesh/ring3.h5
- ../mesh/ring4.h5
- ../mesh/ring5.h5
- ../mesh/ring6.h5
stabilization: # Set FEM stabilization parameters
enable: True
stab_type: cip
cip:
  DELTA_T: 1.0
  DELTA_P: 0.01
  DELTA_U: 1.0
  ht: 3
  hp: 3
  hu: 3
  gla:
    h_power: 1
    const: 10
Problem specific
Kn: 0.1 # Knudsen number
Ma: 0.0 # Mach number
newton_abs_tol: 0.00 # Solver parameters
newton_rel_tol: 0.0
newton_max_itr: 0
newton_relaxation_parameter: 0.0
chi: 1.0
epsilon_w: 1.0
bc: # Boundary conditions
3000:
  theta_w: 1.0
  u_t_w: -10.0 * sin(phi)
  u_n_w: 0.0
  p_w: 0.0
3100:
  theta_w: 0.5
  u_t_w: -1.0 * sin(phi)
  u_n_w: +1.0 * cos(phi)
  p_w: -0.27 * cos(phi)

```



### Moment System R13

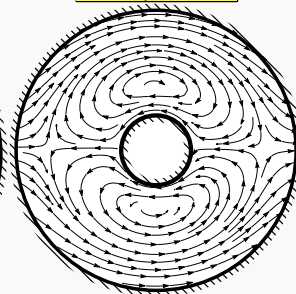
\* Test case: **Flow over a cylinder**

\* Analytical solutions exist for this case.

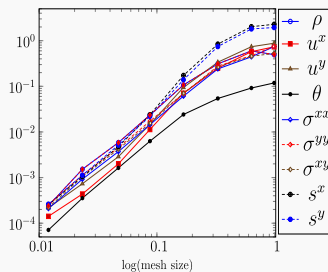
\* **Convergence tests** are performed.

\* Order of convergence is found to be between first and second order for all variables.

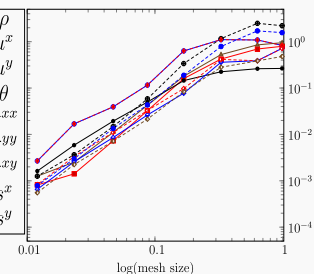
### Velocity Profile



### Normalised $L^2$ Errors



### Normalised $L^\infty$ Errors



## Simulation Results

input.yml	
Model specific	problem_type: 'nonlinear' # linear or nonlinear moment_order: 'grad13' # 6 or 9 or 13 or 'grad13' or 'ns' number_of_moments: 9 # 6 or 9 or 10 or 17 mesh_list: # Provide mesh list
Problem specific	stabilization: # Set FEM stabilisation parameters enable: True stab_type: gls cip: DELTA_T: 1.0 DELTA_P: 0.01 DELTA_U: 1.0 ht: 3 hp: 3 hu: 3 gle: h_power: 1 const: 10 Ks: 0.00001 # Knudsen number Ma: 0.01 # Mach number newton_abs_tol: 0.001 # Solver parameters newton_rel_tol: 0.01 newton_max_itr: 5000 newton_relaxation_parameter: 0.3 chi: 1.0 epsilon_w: 0.0000001 bc: # Boundary conditions 3000: theta_w: 0.0 u_t_w: -1.0 u_b_w: 0.0 p_w: 0.0 3100: theta_w: 0.0 u_t_w: 0.0 u_b_w: 0.0 p_w: 0.0
Stabilisation	
Problem specific	
Solver specific	
Model specific	
Problem specific	

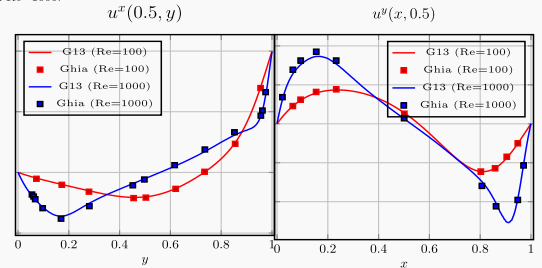
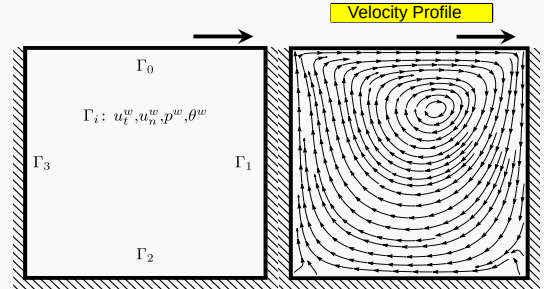
### Moment System G13

\* Test case: **Lid driven cavity**

\* **Comparison** with Ghia et al. is performed.

\* Results are presented for two different Reynolds numbers (Re), namely Re=100 and Re=1000.

\* G13 results are in good agreement with the results presented in Ghia's paper.



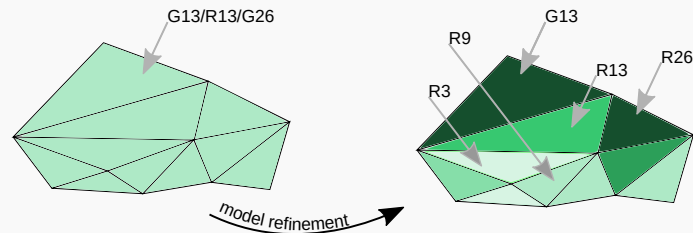
## Outlook

### Summary

- Non-equilibrium gas flows
- Kinetic picture - Boltzmann equation
- Moment approximation to Boltzmann equation
- A generic solver to generic moment equations  
(Many moment models but one solver for all)

### What is next ?

- Extend to other moment models
- Include more realistic physics such as body force, gravity
- Make some parameters space dependent
- Implement model refinement
- $\vdots$



# Hybridized discontinuous Galerkin methods for the Stokes and Navier–Stokes equations in FEniCSx: non-simplex cells and curved geometries

Joseph P. Dean (✉ [jpdean](mailto:jpdean)), University of Cambridge, United Kingdom

Sander Rhebergen, University of Waterloo, Canada

Chris N. Richardson (✉ [chrisrichardson](mailto:chrisrichardson)), University of Cambridge, United Kingdom

Garth N. Wells, University of Cambridge, United Kingdom

26 March 2021

We investigate hybridized discontinuous Galerkin (HDG) methods for the Stokes and incompressible Navier–Stokes equations which yield approximate velocity fields that are pointwise divergence free in each cell and globally  $H(\text{div})$ -conforming. The analysis of a recently developed method is restricted to simplex cells and affine geometries. Here, we explore the extension of the method to non-simplex cells and curved boundaries, both of which are important for engineering applications. Static condensation is used to reduce the size of the global system of equations. For the implementation, we make use of some new features of FEniCSx, which is composed of DOLFINx, FFCx, Basix, and UFL. We use UFL and FFCx to compile kernels for each block of the global matrix, which are then exposed to the Python interface using CFFI. These kernels are called from a custom kernel (compiled by Numba) to carry out the static condensation process. The smaller statically condensed system can then be solved using a block preconditioned iterative solver. We present analysis and numerical results demonstrating that the approximate velocity field is pointwise divergence free in each cell and globally  $H(\text{div})$ -conforming on meshes with non-simplex cells and curved boundaries.

---

You can cite this talk as:

Joseph P. Dean, Sander Rhebergen, Chris N. Richardson, and Garth N. Wells. “Hybridized discontinuous Galerkin methods for the Stokes and Navier–Stokes equations in FEniCSx: non-simplex cells and curved geometries”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 722–741. DOI: 10.6084/m9.figshare.14495634.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/dean.html>.



# Hybridized discontinuous Galerkin methods for the Stokes and Navier-Stokes equations in FEniCSx: non-simplex cells and curved geometries

Joseph P. Dean<sup>1</sup>  
Garth N. Wells<sup>1</sup>

Sander Rhebergen<sup>2</sup>

Chris N. Richardson<sup>1</sup>

<sup>1</sup>*University of Cambridge*    <sup>2</sup>*University of Waterloo*

# Outline

1. The Stokes problem
2. Why not use conforming methods?
3. Hybridized discontinuous Galerkin
4. Non-simplex and curved cells
5. Implementation
6. Numerical results
7. The Navier-Stokes equations
8. Open questions

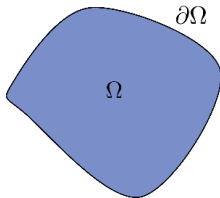
# Problem statement

**Stokes problem** (weak form): Given  $f \in [L^2(\Omega)]^d$ , find  $u \in V := [H_0^1(\Omega)]^d$  and  $p \in Q := L_0^2(\Omega)$  such that

$$\begin{aligned} a(u, v) + b(v, p) &= F(v) \quad \forall v \in V, \\ b(u, q) &= 0 \quad \forall q \in Q, \end{aligned}$$

where

$$a(u, v) := \int_{\Omega} \nu \nabla u : \nabla v \, dx, \quad b(v, p) := - \int_{\Omega} p \nabla \cdot v \, dx, \quad \text{and} \quad F(v) := \int_{\Omega} f \cdot v \, dx.$$



# Some observations

1. The problem is well-posed and  $\exists \beta > 0$  such that

$$\inf_{q \in Q} \sup_{v \in V} \frac{\int_{\Omega} q \nabla \cdot v \, dx}{\|v\|_{1,\Omega} \|q\|_{0,\Omega}} \geq \beta$$

2. The following invariance property<sup>1</sup> holds:

$$f \rightarrow f + \nabla \phi \implies (u, p) \rightarrow (u, p + \phi)$$

---

<sup>1</sup>Volker John et al. "On the Divergence Constraint in Mixed Finite Element Methods for Incompressible Flows". In: *SIAM Review* 59.3 (2017), pp. 492–544. DOI: [10.1137/15m1047696](https://doi.org/10.1137/15m1047696).



# Mass conservation?

**Mass conservation** (weak statement):

$$b(u, q) = 0 \quad \forall q \in Q$$

- The weak statement implies exact mass conservation, meaning  $\|\nabla \cdot u\|_{0,\Omega} = 0$ .

**Mass conservation** (discrete statement): *Let  $u_h \in V_h \subset V$ , then*

$$b(u_h, q_h) = 0 \quad \forall q_h \in Q_h \subset Q$$

- The discrete statement could imply global, local (cell), or exact mass conservation depending on  $V_h$  and  $Q_h$ . If  $\nabla \cdot V_h \subseteq Q_h$ , mass is conserved exactly.

**With conforming methods, it is difficult to balance stability and incompressibility**

# Hybridized discontinuous Galerkin<sup>2</sup>

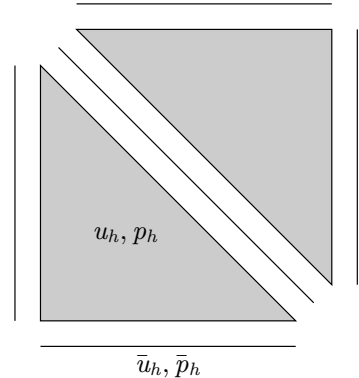
Let  $\mathbf{u}_h := (u_h, \bar{u}_h) \in \mathbf{V}_h$  and  $\mathbf{p}_h := (p_h, \bar{p}_h) \in \mathbf{Q}_h$ , where  $\mathbf{V}_h := V_h \times \bar{V}_h$ ,  $\mathbf{Q}_h := Q_h \times \bar{Q}_h$ , and

$$V_h := \left\{ v_h \in [L^2(\mathcal{T}_h)]^d; v_h|_K \in V_h(K) \forall K \in \mathcal{T}_h \right\},$$

$$\bar{V}_h := \left\{ \bar{v}_h \in [L^2(\mathcal{F}_h)]^d; \bar{v}_h|_F \in \bar{V}_h(F) \forall F \in \mathcal{F}_h, \bar{v}_h = 0 \text{ on } \partial\Omega \right\},$$

$$Q_h := \left\{ q_h \in L^2(\mathcal{T}_h); q_h|_K \in Q_h(K) \forall K \in \mathcal{T}_h \right\},$$

$$\bar{Q}_h := \left\{ \bar{q}_h \in L^2(\mathcal{F}_h); \bar{q}_h|_F \in \bar{Q}_h(F) \forall F \in \mathcal{F}_h \right\}.$$



<sup>2</sup>S. Rhebergen and G. N. Wells. "A hybridizable discontinuous Galerkin method for the Navier–Stokes equations with pointwise divergence-free velocity field". In: *J. Sci. Comput.* 76.3 (2018), pp. 1484–1501. DOI: [10.1007/s10915-018-0671-4](https://doi.org/10.1007/s10915-018-0671-4).

# HDG formulation

**Stokes problem** (HDG formulation): Find  $(\mathbf{u}_h, \mathbf{p}_h) \in \mathbf{V}_h \times \mathbf{Q}_h$  such that

$$\begin{aligned} a_h(\mathbf{u}_h, \mathbf{v}_h) + b_h(v_h, \mathbf{p}_h) &= F(v_h) \quad \forall \mathbf{v}_h \in \mathbf{V}_h, \\ b_h(u_h, \mathbf{q}_h) &= 0 \quad \forall \mathbf{q}_h \in \mathbf{Q}_h, \end{aligned}$$

where

$$\begin{aligned} a_h(\mathbf{u}_h, \mathbf{v}_h) &:= \sum_{K \in \mathcal{T}_h} \int_K \nu \nabla u_h : \nabla v_h \, dx - \sum_{K \in \mathcal{T}_h} \int_{\partial K} \nu \left( (u_h - \bar{u}_h) \cdot \partial_n v_h + \partial_n u_h \cdot (v_h - \bar{v}_h) \right) \, ds \\ &\quad + \sum_{K \in \mathcal{T}_h} \int_{\partial K} \nu \frac{\alpha}{h_K} (u_h - \bar{u}_h) \cdot (v_h - \bar{v}_h) \, ds, \end{aligned}$$

and

$$b_h(v_h, \mathbf{p}_h) := - \sum_{K \in \mathcal{T}_h} \int_K p_h \nabla \cdot v_h \, dx + \sum_{K \in \mathcal{T}_h} \int_{\partial K} v_h \cdot n \bar{p}_h \, ds.$$

# Mapping functions

Let  $\psi_K : V_h(K) \rightarrow V_h(\hat{K})$ .

## Lemma

If  $\psi_K$  is the pullback by the geometric mapping (as in the original method), and if  $\nabla \cdot V_h(K) \subseteq Q_h(K)$  and  $\bar{Q}_h(F) \supseteq \{v_h|_F \cdot n; v_h \in V_h(K)\}$ , then the discrete velocity field is exactly divergence free.

**Problem: what if the geometric mapping is not affine?**

## Lemma

If  $\psi_K$  is the contravariant Piola transform, then the above conditions can be relaxed; if  $\nabla \cdot V_h(\hat{K}) \subseteq Q_h(\hat{K})$  and  $\bar{Q}_h(\hat{F}) \supseteq \{\hat{v}_h|_{\hat{F}} \cdot \hat{n}; \hat{v}_h \in V_h(\hat{K})\}$  then the discrete velocity field is exactly divergence free.

A similar idea can be applied to Scott–Vogelius elements on curved domains.<sup>3</sup>

---

<sup>3</sup>Michael Neilan and M. Baris Otus. “Divergence-free Scott–Vogelius elements on curved domains”. In: (2020), pp. 1–23. arXiv: 2008.06429. URL: <http://arxiv.org/abs/2008.06429>.

# Suitable spaces

**Simplex cells:** If  $\hat{K}$  is the reference simplex and if  $\psi_K$  is the contravariant Piola transform, then the spaces

$$V_h(\hat{K}) := [\mathbb{P}_k(\hat{K})]^d, \quad \bar{V}_h(\hat{F}) := [\mathbb{P}_k(\hat{F})]^d, \quad Q_h(\hat{K}) := \mathbb{P}_{k-1}(\hat{K}) \quad \text{and} \quad \bar{Q}_h(\hat{F}) := \mathbb{P}_k(\hat{F})$$

give an exactly divergence free velocity field even if the geometric mapping is not affine.

**Non-simplex cells:** If  $\hat{K}$  is the reference quadrilateral or hexahedron and if  $\psi_K$  is the contravariant Piola transform, then the spaces

$$V_h(\hat{K}) := \mathbb{RT}_k(\hat{K}), \quad \bar{V}_h(\hat{F}) := [\mathbb{Q}_k(\hat{F})]^d, \quad Q_h(\hat{K}) := \mathbb{Q}_k(\hat{K}), \quad \text{and} \quad \bar{Q}_h(\hat{F}) := \mathbb{Q}_k(\hat{F})$$

give an exactly divergence free velocity field even if the geometric mapping is not affine.

# More about the non-simplex case

- $H(\text{div})$ -conforming finite elements are introduced following the same ideas as divergence conforming DG<sup>4</sup> and HDG<sup>5</sup> methods.
- Other  $H(\text{div})$ -conforming finite elements can be used, but care must be taken as some lose optimal order approximation in  $[L^2(\Omega)]^d$  on general quadrilateral meshes.<sup>6</sup>

---

<sup>4</sup>Bernardo Cockburn, Guido Kanschat, and Dominik Schötzau. "A Note on Discontinuous Galerkin Divergence-free Solutions of the Navier-Stokes Equations". In: *Journal of Scientific Computing* 31:1-2 (2007), pp. 61–73. DOI: [10.1007/s10915-006-9107-7](https://doi.org/10.1007/s10915-006-9107-7).

<sup>5</sup>Christoph Lehrenfeld and Joachim Schöberl. "High order exactly divergence-free Hybrid Discontinuous Galerkin Methods for unsteady incompressible flows". In: *Computer Methods in Applied Mechanics and Engineering* 307 (2016), pp. 339–361. DOI: [10.1016/j.cma.2016.04.025](https://doi.org/10.1016/j.cma.2016.04.025).

<sup>6</sup>Douglas N. Arnold, Daniele Boffi, and Richard S. Falk. "Quadrilateral H (div) Finite Elements". In: *SIAM Journal on Numerical Analysis* 42:6 (2005), pp. 2429–2451. DOI: [10.1137/S0036142903431924](https://doi.org/10.1137/S0036142903431924).

# Static condensation

The block structure of the element tensor is of the form

$$\begin{bmatrix} A_{uu} & B_{pu}^T & A_{\bar{u}u}^T & B_{\bar{p}u}^T \\ B_{pu} & 0 & 0 & 0 \\ A_{\bar{u}u} & 0 & A_{\bar{u}\bar{u}} & 0 \\ B_{\bar{p}u} & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} U \\ P \\ \bar{U} \\ \bar{P} \end{pmatrix} = \begin{pmatrix} F_u \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Eliminating the cell degrees of freedom gives the condensed element tensor

$$\begin{bmatrix} A_{\bar{u}\bar{u}} - BA^{-1}B^T & -BA^{-1}C^T \\ -CA^{-1}B^T & -CA^{-1}C^T \end{bmatrix} \begin{pmatrix} \bar{U} \\ \bar{P} \end{pmatrix} = \begin{pmatrix} -BA^{-1}F \\ -CA^{-1}F \end{pmatrix},$$

where

$$A = \begin{bmatrix} A_{uu} & B_{pu}^T \\ B_{pu} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} A_{\bar{u}u} & 0 \end{bmatrix}, \quad C = \begin{bmatrix} B_{\bar{p}u} & 0 \end{bmatrix}, \quad \text{and} \quad F = \begin{pmatrix} F_u \\ 0 \end{pmatrix}.$$

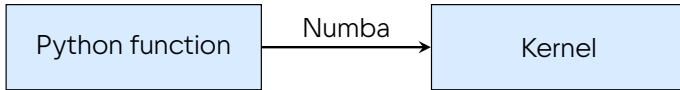
# Implementation

Features of FEniCSx:

- Create kernels generated from UFL that are callable from python



- Create user defined kernels written in Python



- User defined kernels can call generated kernels



# Implementation

Create kernels for each block of the element tensor ( $A_{uu}, \dots, A_{\bar{u}\bar{u}}$ ):

```
1 # UFL expressions for each block of the element tensor
2 A_uu_form = nu * inner(grad(u), grad(v)) * dx + nu * gamma * inner(u, v) * ds \
3     - nu * (inner(u, dot(grad(v), n)) + inner(v, dot(grad(u), n))) * ds
4 ...
5 A_uubar_uubar_form = nu * gamma * inner(ubar, vbar) * ds
6
7 # Compile forms with FFCx and expose to Python
8 forms = [A_uu_form, ..., A_uubar_uubar_form]
9 compiled_forms = ffcx.codegeneration.jit.compile_forms(forms)
10 A_uu_cell_kernel = compiled_forms[0][0].create_cell_integral().tabulate_tensor
11 A_uu_facet_kernel = \
12     compiled_forms[0][0].create_exterior_facet_integral().tabulate_tensor
13 ...
14
```

# Implementation

Define a custom kernel to compute the top left block of the condensed element tensor ( $K_{00} := A_{\bar{u}\bar{u}} - BA^{-1}B^T$ ):

```
1 @numba.cfunc(c_signature)
2 def tabulate_K00(K00_, w_, c_, coords_, entity_local_index, ...):
3     K00 = numba.carray(K00_, (ubar_size, ubar_size))
4     A_uu = np.zeros((u_size, u_size))
5     ...
6     # Compute cell integrals
7     A_uu_cell_kernel(ffl.from_buffer(A_uu), w_, c_, coords_, entity_local_index, ...)
8     ...
9     for j in range(n_facets):
10         # Compute facet integrals
11         A_uu_facet_kernel(ffl.from_buffer(A_uu), w_, c_, coords_, fj, ...)
12         ...
13     # Static condensation
14     K00 += A_uu_bar_uu_bar - B @ np.linalg.solve(A, B.T)
15
```

This kernel is passed to DOLFINx to assemble over the mesh.

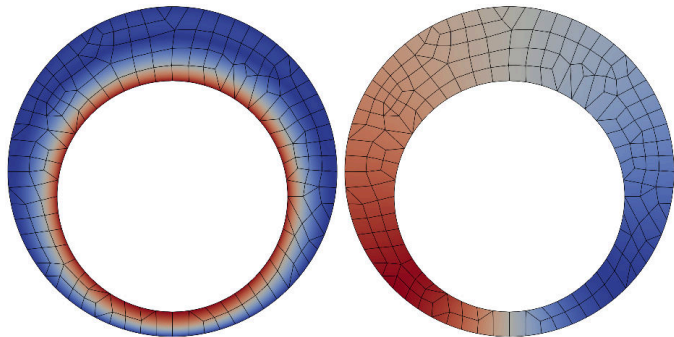
# Implementation: further work

- The above FEniCSx implementation has been tested on simplices.
- Until recently, FEniCSx did not have support for quadrilateral/hexahedral  $H(\text{div})$ -conforming finite elements.
- Basix supports these elements, but some work is required to implement facet function spaces in a more general manner.
- To demonstrate the HDG scheme on meshes containing quadrilaterals, the method was also implemented in NGSolve.<sup>7</sup>

---

<sup>7</sup>Joachim Schöberl. "C++ 11 implementation of finite elements in NGSolve". In: *Technical Report ASC-2014-30, Institute for Analysis and Scientific Computing* (2014). URL: <https://www.asc.tuwien.ac.at/~schoeberl/wiki/publications/ngs-cpp11.pdf>.

# Results: curved cells



(a) Velocity magnitude

(b) Pressure

Figure: Computed solution

	$N$	$e_u$	$e_{\nabla \cdot u}$	$e_{[u]}$
Present method	3870	$6.17 \times 10^{-4}$	<b><math>5.45 \times 10^{-15}</math></b>	$4.68 \times 10^{-14}$
Original method	3870	$6.71 \times 10^{-4}$	<b><math>3.02 \times 10^{-2}</math></b>	$8.51 \times 10^{-13}$

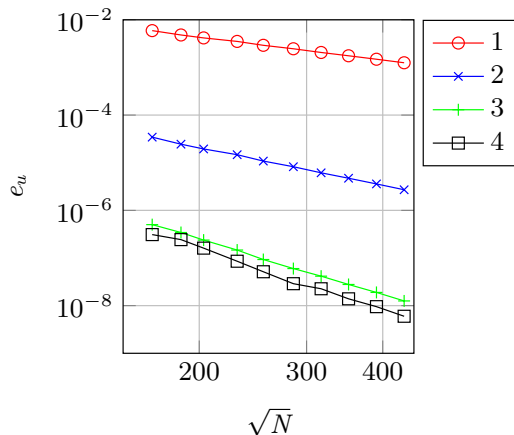


Figure:  $e_u$  against  $\sqrt{N}$  for  $k = 3$  with piecewise polynomial geometric mappings of degrees 1, 2, 3, and 4.

# Extension to the Navier-Stokes equations

- ✓ Straightforward extension to the Navier-Stokes equations
- ✓ Divergence free velocity field on affine and non-affine simplex and non-simplex cells
- ✓ Local momentum conservation
- ✓ Arbitrarily high order

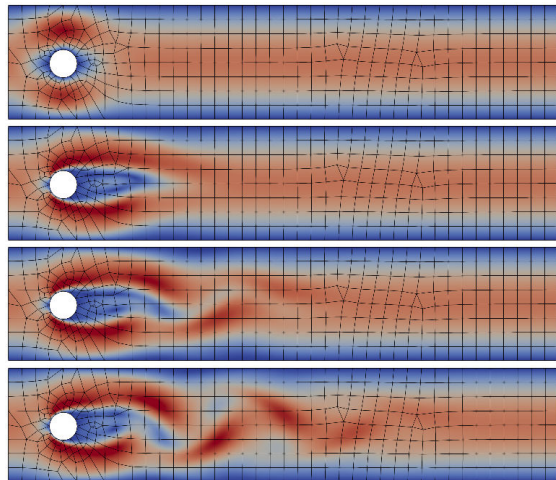


Figure: Velocity magnitude

# Open questions

We are currently working on:

- Implementing a FEniCSx version of the method for meshes with quadrilateral and hexahedral cells.
- Rigorous proofs of the discrete inf-sup condition and error estimates on non-affine meshes.
- Optimal preconditioners and investigating the performance of the method at large scale.

**Any suggestions/advice about these topics would be very much appreciated!**



Thank you. Any questions?

# Semismooth Newton method for Bingham flow

Alexei Gazca, FAU Erlangen-Nürnberg, Germany

26 March 2021

We propose a semismooth Newton method for non-Newtonian models of incompressible flow where the constitutive relation between the shear stress and the symmetric velocity gradient is given implicitly; as a motivating example we consider the Bingham model for viscoplastic flow. The proposed method avoids the use of variational inequalities and is based on a particularly simple regularisation for which the (weak) convergence of the approximate stresses is known to hold. The system is analysed at the function space level and results in mesh-independent behaviour of the nonlinear iterations.

---

You can cite this talk as:

Alexei Gazca. “Semismooth Newton method for Bingham flow”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 742–764. doi: 10.6084/m9.figshare.14495637.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/gazca.html>.





# Semismooth Newton method for Bingham flow

FEniCS 2021

Alexei Gazca

FAU Erlangen-Nürnberg

March 2021

\* Work supported by the Alexander von Humboldt Stiftung

# 1 Incompressible Fluids

| 1

Let  $\Omega \subset \mathbb{R}^d$ ,  $d \in \{2, 3\}$  be a **bounded Lipschitz polyhedral domain** and consider the system:

$$\begin{aligned} \alpha \mathbf{u} - \operatorname{div} \mathbf{S} + \operatorname{div}(\mathbf{u} \otimes \mathbf{u}) + \nabla p &= \mathbf{f}, & \Omega, \\ \operatorname{div} \mathbf{u} &= 0, & \Omega, \\ &+ \text{BCs} \end{aligned}$$

Here

- ▶  $\mathbf{u}: \Omega \rightarrow \mathbb{R}^d$  represents the **velocity field**;
- ▶  $p: \Omega \rightarrow \mathbb{R}$  is the **pressure**;
- ▶  $\mathbf{S}: \Omega \rightarrow \mathbb{R}_{\text{sym}, \text{tr}}^{d \times d}$  is the **shear stress tensor**;

# 1 Constitutive relation (Bingham/Herschel–Bulkley)

| 2

Denote  $\mathbf{D} := \mathbf{D}(\mathbf{u}) := \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^\top)$ .

$$\begin{cases} \mathbf{S} = 2\nu_*(|\mathbf{D}|)\mathbf{D} + \tau_* \frac{\mathbf{D}}{|\mathbf{D}|} & \text{if } |\mathbf{S}| \geq \tau_*, \\ \mathbf{D} = \mathbf{0} & \text{if } |\mathbf{S}| \leq \tau_*. \end{cases}$$

Here  $\tau_* \geq 0$  is the **yield stress** and  $\nu_* > 0$  is the **viscosity**.

# 1 Constitutive relation (Bingham/Herschel–Bulkley)

| 2

Denote  $\mathbf{D} := \mathbf{D}(\mathbf{u}) := \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^\top)$ .

$$\begin{cases} \mathbf{S} = 2\nu_*(|\mathbf{D}|)\mathbf{D} + \tau_* \frac{\mathbf{D}}{|\mathbf{D}|} & \text{if } |\mathbf{S}| \geq \tau_*, \\ \mathbf{D} = \mathbf{0} & \text{if } |\mathbf{S}| \leq \tau_*. \end{cases}$$

Here  $\tau_* \geq 0$  is the **yield stress** and  $\nu_* > 0$  is the **viscosity**.

It can be naturally written using an **implicit function**:

$$\mathbf{G}(\mathbf{S}, \mathbf{D}) := (|\mathbf{S}| - \tau_*)^+ \mathbf{S} - 2\nu_*(\tau_* + (|\mathbf{S}| - \tau_*)^+) \mathbf{D} = \mathbf{0}.$$

# 1 Constitutive relation (Bingham/Herschel–Bulkley)

| 2

Denote  $\mathbf{D} := \mathbf{D}(\mathbf{u}) := \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^\top)$ .

$$\begin{cases} \mathbf{S} = 2\nu_*(|\mathbf{D}|)\mathbf{D} + \tau_* \frac{\mathbf{D}}{|\mathbf{D}|} & \text{if } |\mathbf{S}| \geq \tau_*, \\ \mathbf{D} = \mathbf{0} & \text{if } |\mathbf{S}| \leq \tau_*. \end{cases}$$

Here  $\tau_* \geq 0$  is the **yield stress** and  $\nu_* > 0$  is the **viscosity**.

It can be naturally written using an **implicit function**:

$$\mathbf{G}(\mathbf{S}, \mathbf{D}) := (|\mathbf{S}| - \tau_*)^+ \mathbf{S} - 2\nu_*(\tau_* + (|\mathbf{S}| - \tau_*)^+) \mathbf{D} = \mathbf{0}.$$

📖 M. BULÍČEK, P. GWIAZDA, J. MÁLEK, AND A. ŚWIERCZEWSKA-GWIAZDA. *On unsteady flows of implicitly constituted incompressible fluids*. *SIAM J. Math. Anal.* 44(4):2756–2801, 2012.

📖 P.E. FARRELL, P.A. GAZCA-OROZCO, AND E. SÜLI. *Numerical analysis of unsteady implicitly constituted incompressible fluids: 3-field formulation*. *SIAM J. Numer. Anal.* 58(1):757–787, 2020.

# 1 Regularisation

| 3

A common regularisation [Bercovier, Engelman 1980]:

$$\mathbf{S}_\varepsilon = \tilde{\mathbf{S}}_\varepsilon(\mathbf{D}) := 2\nu_* \mathbf{D} + \tau_* \frac{\mathbf{D}}{\sqrt{\varepsilon^2 + |\mathbf{D}|^2}} \quad \varepsilon > 0.$$

# 1 Regularisation

| 3

A common regularisation [Bercovier, Engelman 1980]:

$$\mathbf{S}_\varepsilon = \tilde{\mathbf{S}}_\varepsilon(\mathbf{D}) := 2\nu_* \mathbf{D} + \tau_* \frac{\mathbf{D}}{\sqrt{\varepsilon^2 + |\mathbf{D}|^2}} \quad \varepsilon > 0.$$

Problems:

- It is **unclear** whether  $\mathbf{S}_\varepsilon \rightarrow \mathbf{S}$ .

# 1 Regularisation

| 3

A common regularisation [Bercovier, Engelman 1980]:

$$\mathbf{S}_\varepsilon = \tilde{\mathcal{S}}_\varepsilon(\mathbf{D}) := 2\nu_* \mathbf{D} + \tau_* \frac{\mathbf{D}}{\sqrt{\varepsilon^2 + |\mathbf{D}|^2}} \quad \varepsilon > 0.$$

Problems:

- It is **unclear** whether  $\mathbf{S}_\varepsilon \rightarrow \mathbf{S}$ .

The main alternative (AL) also has issues:

- In its basic form it can be **slow**.
- Needs more **sophisticated tools**;



# 1 New Regularisation

| 4

We employ here the **simple** regularisation from:

 M. BULÍČEK, J. MÁLEK, AND E. MARINGOVÁ. *On nonlinear problems of parabolic type with implicit constitutive equations involving flux*. [ArXiv Preprint: 2009.06917, 2020](#).

$$\mathbf{G}_\varepsilon(\mathbf{S}, \mathbf{D}) := \mathbf{G}(\mathbf{S} - \varepsilon \mathbf{D}, \mathbf{D} - \varepsilon \mathbf{S}) \quad \varepsilon > 0.$$

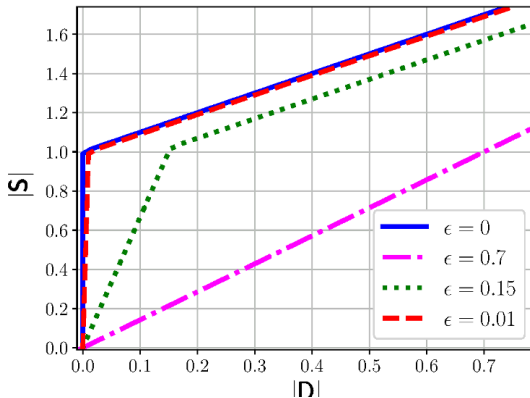
# 1 New Regularisation

| 4

We employ here the **simple** regularisation from:

📄 M. BULÍČEK, J. MÁLEK, AND E. MARINGOVÁ. *On nonlinear problems of parabolic type with implicit constitutive equations involving flux*. *ArXiv Preprint*: 2009.06917, 2020.

$$G_\varepsilon(S, D) := G(S - \varepsilon D, D - \varepsilon S) \quad \varepsilon > 0.$$



# 1 New Regularisation

| 5

We employ here the **simple** regularisation:

📖 M. BULÍČEK, J. MÁLEK, AND E. MARINGOVÁ. *On nonlinear problems of parabolic type with implicit constitutive equations involving flux*. **ArXiv Preprint: 2009.06917, 2020.**

$$\mathbf{G}_\varepsilon(\mathbf{S}, \mathbf{D}) := \mathbf{G}(\mathbf{S} - \varepsilon \mathbf{D}, \mathbf{D} - \varepsilon \mathbf{S}) \quad \varepsilon > 0.$$

► The stresses **converge** (weakly)  $\mathbf{S}_\varepsilon \rightharpoonup \mathbf{S}$ .

# 1 New Regularisation

| 5

We employ here the **simple** regularisation:

📖 M. BULÍČEK, J. MÁLEK, AND E. MARINGOVÁ. *On nonlinear problems of parabolic type with implicit constitutive equations involving flux*. **ArXiv Preprint: 2009.06917, 2020.**

$$\mathbf{G}_\varepsilon(\mathbf{S}, \mathbf{D}) := \mathbf{G}(\mathbf{S} - \varepsilon \mathbf{D}, \mathbf{D} - \varepsilon \mathbf{S}) \quad \varepsilon > 0.$$

- ▶ The stresses **converge** (weakly)  $\mathbf{S}_\varepsilon \rightharpoonup \mathbf{S}$ .
- ▶ The graph defined by  $\mathbf{G}_\varepsilon$  is **strongly monotone** and **2-coercive**:

$$\begin{aligned} \mathbf{S}_1 : \mathbf{D}_1 &\geq c(|\mathbf{S}_1|^2 + |\mathbf{D}_1|^2) - \tilde{c}, \\ (\mathbf{S}_1 - \mathbf{S}_2) : (\mathbf{D}_1 - \mathbf{D}_2) &\geq c_\varepsilon(|\mathbf{S}_1 - \mathbf{S}_2|^2 + |\mathbf{D}_1 - \mathbf{D}_2|^2). \end{aligned}$$

# 1 New Regularisation

| 5

We employ here the **simple** regularisation:

📖 M. BULÍČEK, J. MÁLEK, AND E. MARINGOVÁ. *On nonlinear problems of parabolic type with implicit constitutive equations involving flux*. **ArXiv Preprint: 2009.06917, 2020.**

$$\mathbf{G}_\varepsilon(\mathbf{S}, \mathbf{D}) := \mathbf{G}(\mathbf{S} - \varepsilon \mathbf{D}, \mathbf{D} - \varepsilon \mathbf{S}) \quad \varepsilon > 0.$$

- ▶ The stresses **converge** (weakly)  $\mathbf{S}_\varepsilon \rightharpoonup \mathbf{S}$ .
- ▶ The graph defined by  $\mathbf{G}_\varepsilon$  is **strongly monotone** and **2-coercive**:

$$\begin{aligned} \mathbf{S}_1 : \mathbf{D}_1 &\geq c(|\mathbf{S}_1|^2 + |\mathbf{D}_1|^2) - \tilde{c}, \\ (\mathbf{S}_1 - \mathbf{S}_2) : (\mathbf{D}_1 - \mathbf{D}_2) &\geq c_\varepsilon(|\mathbf{S}_1 - \mathbf{S}_2|^2 + |\mathbf{D}_1 - \mathbf{D}_2|^2). \end{aligned}$$

The function  $\mathbf{G}_\varepsilon$  is still **not continuously differentiable**.

We need a **semismooth Newton** method!

## 2 Semismooth Newton method

| 6

Classical Newton iteration for  $F(z) = 0$ :

$$z^{k+1} = z^k - DF(z^k)^{-1}F(z^k).$$

## 2 Semismooth Newton method

| 6

Classical Newton iteration for  $F(z) = 0$ :

$$z^{k+1} = z^k - DF(z^k)^{-1}F(z^k).$$

Semismooth Newton iteration for  $F(z) = 0$ :

$$z^{k+1} = z^k - M_k^{-1}F(z^k).$$

Here  $M_k$  is an element of the **generalised gradient** of  $F$ , e.g. Clarke's differential (if  $F: \mathbb{R}^m \rightarrow \mathbb{R}^n$ ):

$$\partial F(z) := \text{co}\{M \in \mathbb{R}^{n \times m} : \exists \{z_i\} \subset \mathbb{R}^m \setminus U_R \text{ with } z_i \rightarrow z, \nabla F(z_i) \rightarrow M\}$$

## 2 Semismooth Newton method

| 7

### Example

For  $H(\mathbf{S}) = (|\mathbf{S}| - \tau_*)^+$  one has:

$$\partial H(\mathbf{S}) = \begin{cases} \{\mathbb{1}_{\{|\mathbf{S}| > \tau_*\}} \frac{\mathbf{S}}{|\mathbf{S}|}\} & \text{if } |\mathbf{S}| \neq \tau_*, \\ \{\phi \in \mathbb{R}^{d \times d} : |\phi| \leq 1\} & \text{if } |\mathbf{S}| = \tau_*. \end{cases}$$

For the positive part, UFL makes the choice:

$$\nabla \max\{f, 0\} = \begin{cases} \nabla f & \text{if } f > 0, \\ 0 & \text{if } f \leq 0. \end{cases}$$



## 2 Semismooth Newton method

| 8

We write the original problem as  $F(\mathbf{S}, \mathbf{u}, p) = 0$ , where  $F: Z \rightarrow W$  for some appropriate Banach spaces  $Z$  and  $W$ .

## 2 Semismooth Newton method

| 8

We write the original problem as  $F(\mathbf{S}, \mathbf{u}, p) = 0$ , where  $F: Z \rightarrow W$  for some appropriate Banach spaces  $Z$  and  $W$ .

### Proposition [Ulbrich,2003]

Suppose that in a nbd of the solution  $z$  we have  $\|M^{-1}\|_{\mathcal{L}(X;Z)} \leq c$ , and that

$$\sup_{M \in \partial F(z+h)} \|F(z+h) - F(z) - Mh\|_X = o(\|h\|_Z) \quad \text{as } h \rightarrow 0.$$

Then the semismooth Newton iteration converges locally superlinearly.

## 2 Semismooth Newton method

| 8

We write the original problem as  $F(\mathbf{S}, \mathbf{u}, p) = 0$ , where  $F: Z \rightarrow W$  for some appropriate Banach spaces  $Z$  and  $W$ .

Proposition [Ulbrich,2003]

Suppose that in a nbd of the solution  $z$  we have  $\|M^{-1}\|_{\mathcal{L}(X;Z)} \leq c$ , and that

$$\sup_{M \in \partial F(z+h)} \|F(z+h) - F(z) - Mh\|_X = o(\|h\|_Z) \quad \text{as } h \rightarrow 0.$$

Then the semismooth Newton iteration converges locally superlinearly.

Need to **carefully** check that semismoothness of  $\mathbf{G}: \mathbb{R}_{\text{sym}}^{d \times d} \times \mathbb{R}_{\text{sym}}^{d \times d} \rightarrow \mathbb{R}_{\text{sym}}^{d \times d}$  implies that

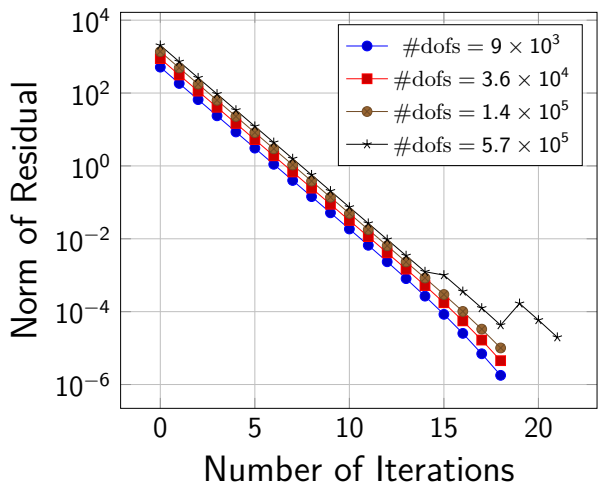
$$(\mathbf{S}, \mathbf{u}) \in L_{\text{sym}}^{r'}(\Omega)^{d \times d} \times W^{1,r}(\Omega)^d \mapsto \mathbf{G}(\mathbf{S}, \mathbf{D}(\mathbf{u})) \in L_{\text{sym}}^q(\Omega)^{d \times d},$$

is semismooth.

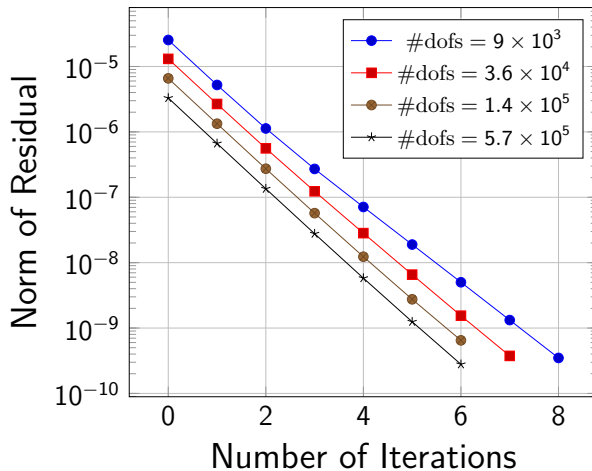
## 2 Examples

| 9

Using a stabilised  $\mathbb{P}_0^{d \times d} - \mathbb{P}_1^d - \mathbb{P}_1$  element with [firedrake](#):

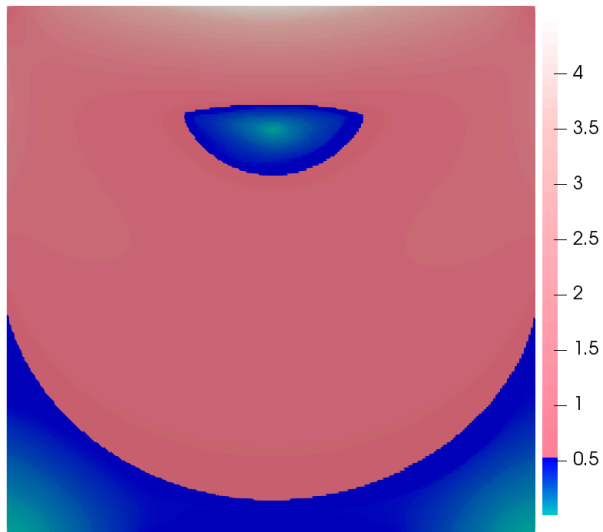


(a)  $\varepsilon = 0.5$ .

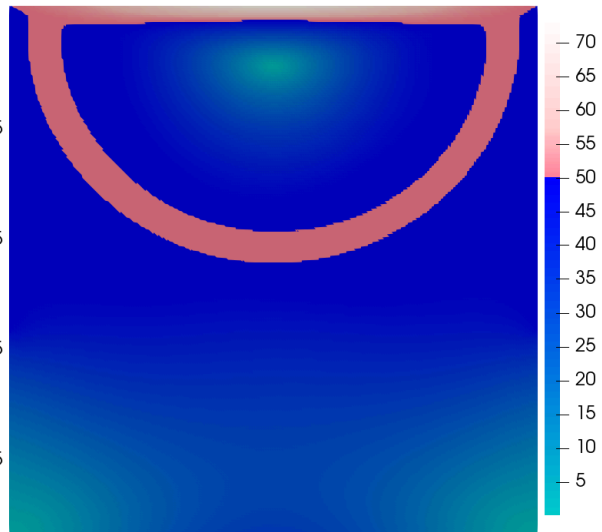


(b)  $\varepsilon = 0.0001$ .

## 2 Examples



(c)  $\tau_* = 0.5$ .



(d)  $\tau_* = 50$ .



# Non-intrusive reduced order modeling of linear poroelasticity in heterogeneous porous media

Teeratorn Kadeethum, Cornell University, United States

Francesco Ballarin, SISSA, Italy

Nikolaos Bouklas, Cornell University, United States

26 March 2021

A simulation tool capable of speeding up the calculation for linear poroelasticity problems in heterogeneous porous media is of large practical interest for engineers, in particular, to effectively perform sensitivity analyses, uncertainty quantification, optimization, or control operations on the fluid pressure and bulk deformation fields. Towards this goal, we present here a non-intrusive model reduction framework built on FEniCS, RBniCS, and Multiphenics using proper orthogonal decomposition (POD) and neural networks, based on the usual offline-online paradigm. As the conductivity of porous media can be highly heterogeneous and span several orders of magnitude, we utilize the interior penalty discontinuous Galerkin (DG) method as a full order solver to handle discontinuity and ensure local mass conservation during the offline stage.

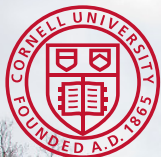
We then use POD as a data compression tool and compare the nested POD technique, in which time and uncertain parameter domains are compressed consecutively, to the classical POD method in which all domains are compressed simultaneously. The neural networks are finally trained to map the set of uncertain parameters, which could correspond to material properties, boundary conditions, or geometric characteristics, to the collection of coefficients calculated from an  $L^2$  projection over the reduced basis. We then perform a non-intrusive evaluation of the neural networks to obtain coefficients corresponding to new values of the uncertain parameters during the online stage. We show that our framework provides reasonable approximations of the DG solution, but it is significantly faster. Moreover, the reduced order framework can capture sharp discontinuities of both displacement and pressure fields resulting from the heterogeneity in the media conductivity, which is generally challenging for intrusive reduced order methods. The sources of error are presented, showing that the nested POD technique is computationally advantageous and still provides comparable accuracy to the classical POD method. We also explore the effect of different choices of the hyperparameters of the neural network on the framework performance.

---

You can cite this talk as:

Teeratorn Kadeethum, Francesco Ballarin, and Nikolaos Bouklas. “Non-intrusive reduced order modeling of linear poroelasticity in heterogeneous porous media”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 765–782. doi: 10.6084/m9.figshare.14495643.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/kadeethum.html>.



# **Non-intrusive ROM of linear poroelasticity in porous media** (<https://arxiv.org/abs/2101.11810>)

Teeratorn Kadeethum, Francesco Ballarin, and Nikolaos Bouklas



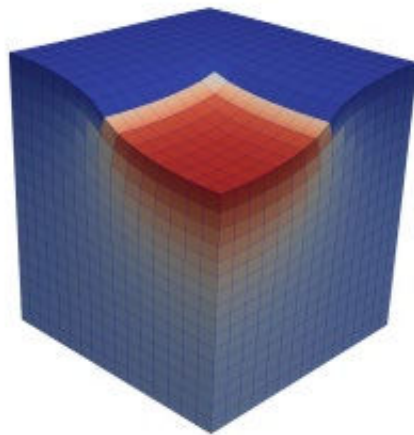
# Governing equations

## Momentum balance equation

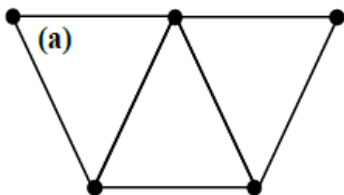
$$\begin{aligned}\nabla \cdot \sigma'(u) - \alpha \nabla \cdot (p\mathbf{I}) + f &= \mathbf{0} \quad \text{in } \Omega \times \mathbb{T}, \\ u &= u_D \quad \text{on } \partial\Omega_u \times \mathbb{T}, \\ \sigma(u) \cdot \mathbf{n} &= t_D \quad \text{on } \partial\Omega_t \times \mathbb{T}, \\ u &= u_0 \quad \text{in } \Omega \text{ at } t = 0,\end{aligned}$$

## Mass balance equation

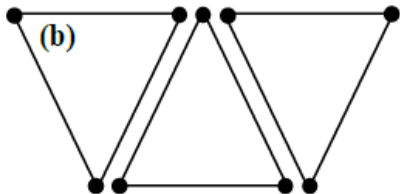
$$\begin{aligned}\left(\frac{1}{M} + \frac{\alpha^2}{K}\right) \frac{\partial p}{\partial t} + \frac{\alpha}{K} \frac{\partial \sigma_v}{\partial t} - \kappa \nabla p &= g \quad \text{in } \Omega \times \mathbb{T}, \\ p &= p_D \quad \text{on } \partial\Omega_p \times \mathbb{T}, \\ -\kappa \nabla p \cdot \mathbf{n} &= q_D \quad \text{on } \partial\Omega_q \times \mathbb{T}, \\ p &= p_0 \quad \text{in } \Omega \text{ at } t = 0,\end{aligned}$$



# Full-order model (FOM) – finite element



**CG:** for displacement field



**DG:** for pressure field

**Monolithic**

$$\begin{bmatrix} \mathcal{J}_{uu}^{\text{CG}_2 \times \text{CG}_2} & \mathcal{J}_{up}^{\text{CG}_2 \times \text{DG}_1} \\ \mathcal{J}_{pu}^{\text{DG}_1 \times \text{CG}_2} & \mathcal{J}_{pp}^{\text{DG}_1 \times \text{DG}_1} \end{bmatrix} \begin{Bmatrix} (\delta u_h^n)^{\text{CG}_2} \\ (\delta p_h^n)^{\text{DG}_1} \end{Bmatrix} = - \begin{Bmatrix} R_u^{\text{CG}_2} \\ R_p^{\text{DG}_1} \end{Bmatrix}$$

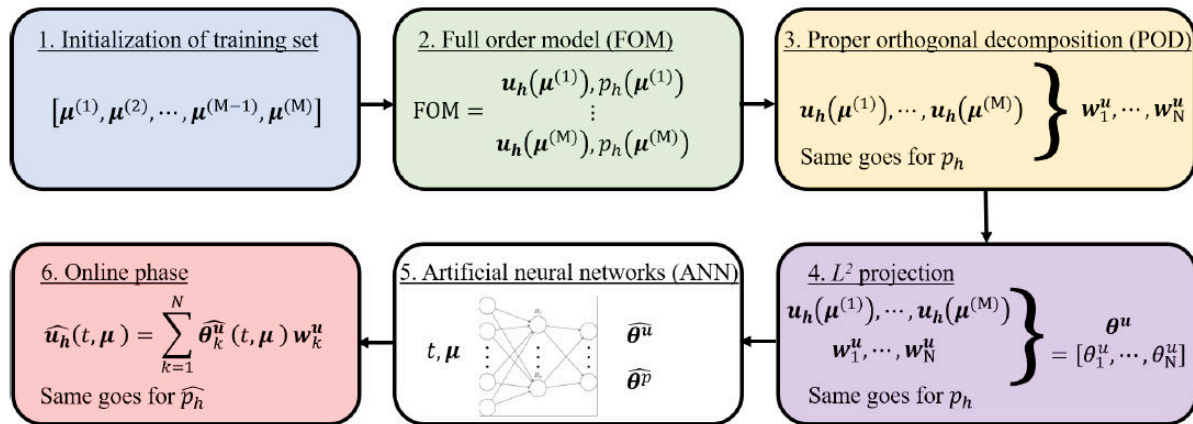
**Time-stepping**

$$\text{BDF}_1(\varphi^n) := \frac{1}{\Delta t^n} (\varphi^n - \varphi^{n-1})$$

<https://www.sciencedirect.com/science/article/pii/S0309170819312576>

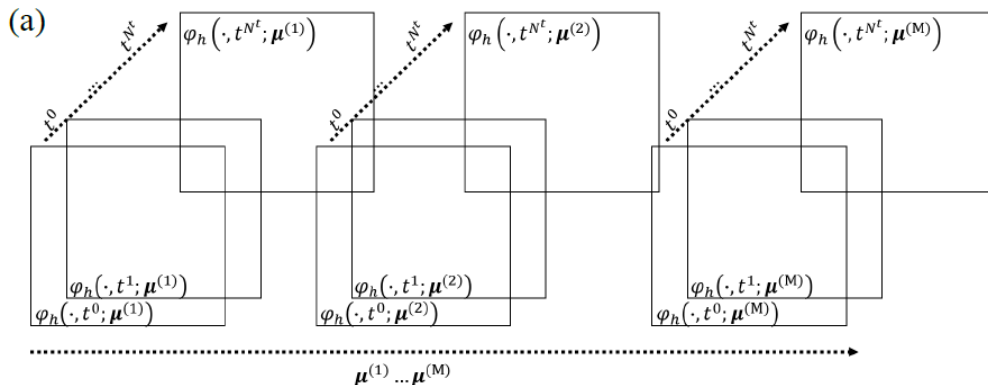
<https://link.springer.com/article/10.1007/s11004-020-09893-y>

<https://www.sciencedirect.com/science/article/pii/S0021999120308044>



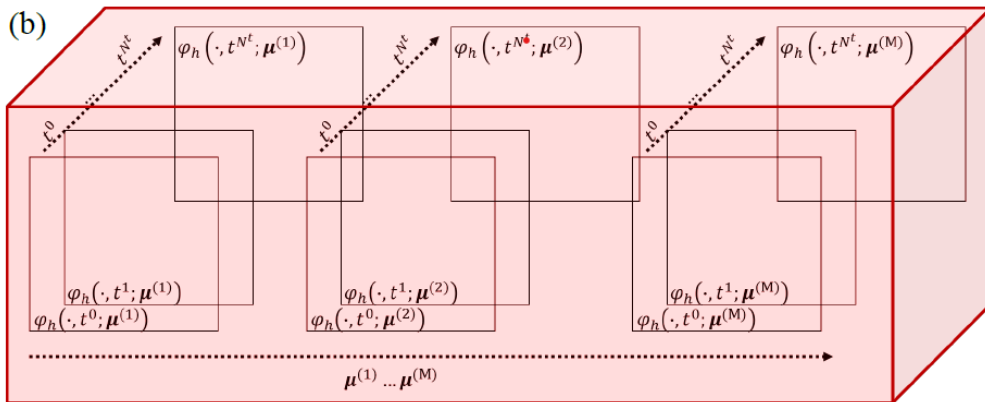
# Proper orthogonal decomposition (POD)

## Finite element snapshots



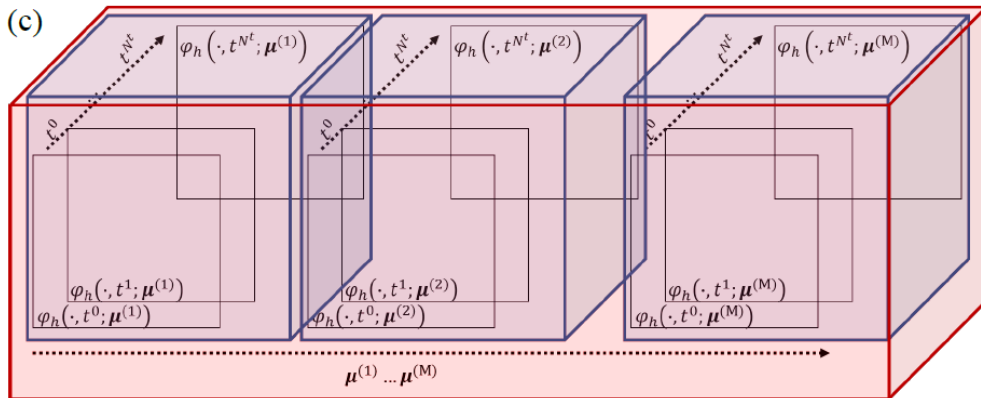
# Proper orthogonal decomposition (POD)

## Single compression

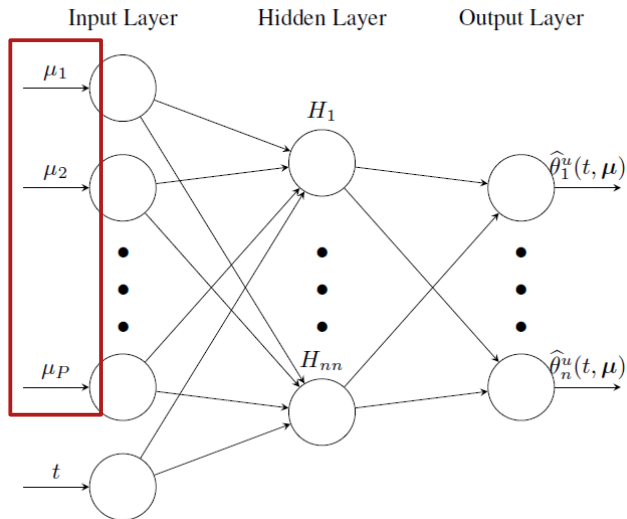


# Proper orthogonal decomposition (POD)

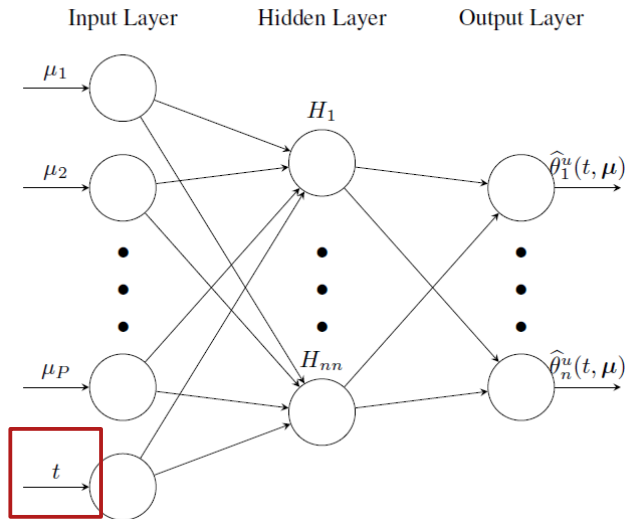
## Nested compression



# Artificial neural networks (ANN)

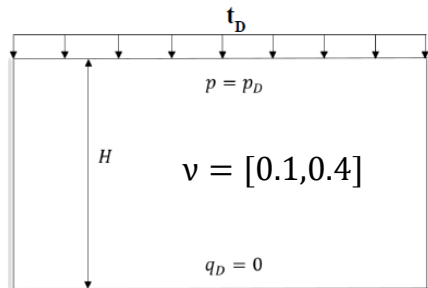


# Artificial neural networks (ANN)



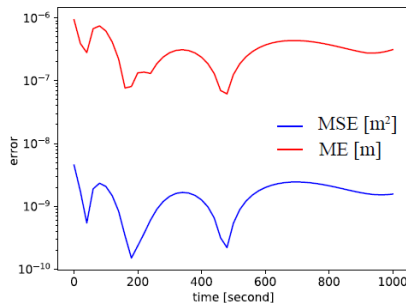


# Terzaghi's consolidation problem



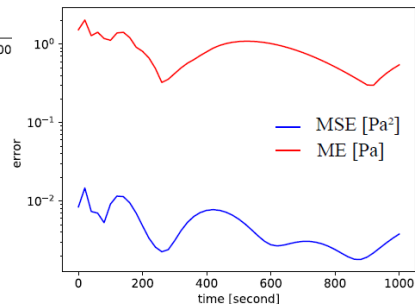
$$\text{MSE}_\varphi(t, \mu) := \|\varphi_h(\cdot; t, \mu) - \hat{\varphi}_h(\cdot; t, \mu)\|_\varphi^2$$

$$\text{ME}_\varphi(t, \mu) := \|\varphi_h(\cdot; t, \mu) - \hat{\varphi}_h(\cdot; t, \mu)\|_\varphi^\infty$$

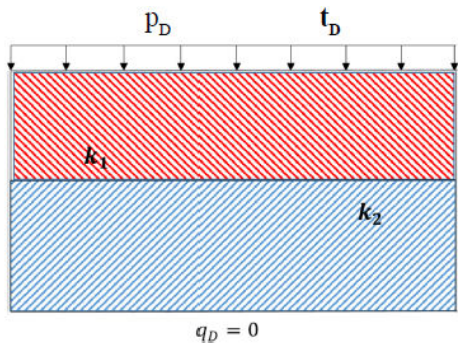


**Displacement**  
( $1 \times 10^{-4}$ )

**Pressure**  
( $1 \times 10^3$ )

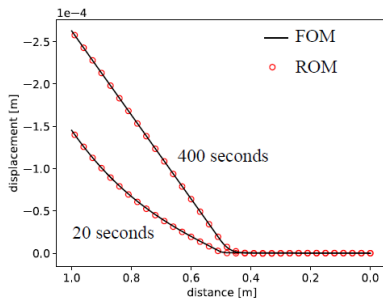


# Consolidation problem with 2-layered material



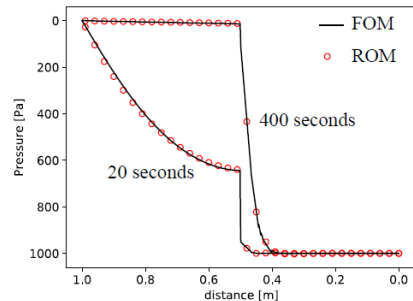
$$k_1 := \begin{bmatrix} 1.0 \times 10^{-12} & 0.0 \\ 0.0 & 1.0 \times 10^{-12} \end{bmatrix}$$

$$k_2 := \begin{bmatrix} k_{xx} & 0.0 \\ 0.0 & k_{xx} \end{bmatrix} \cdot \quad k_{xx} = [1.0 \times 10^{-16}, 1.0 \times 10^{-15}]$$



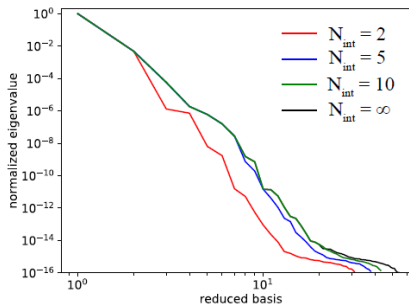
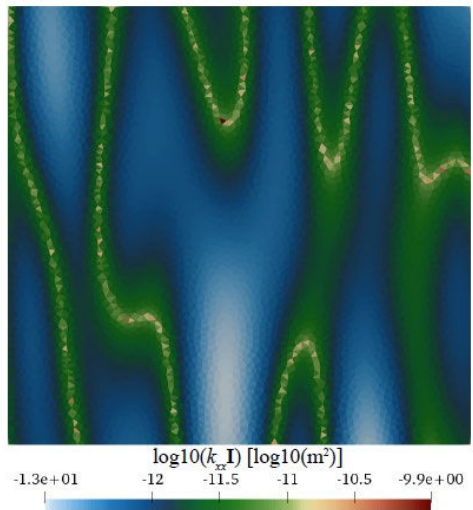
**Pressure**

**Displacement**

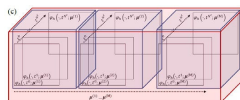


$$\mu = (\nu, \alpha) \in [0.1, 0.4] \times [0.4, 1.0]$$

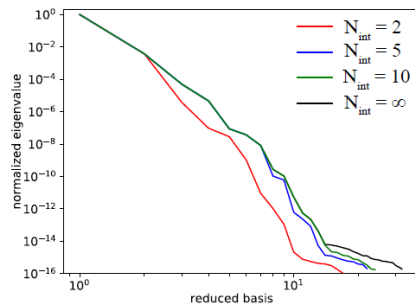
# Heterogeneous media - POD



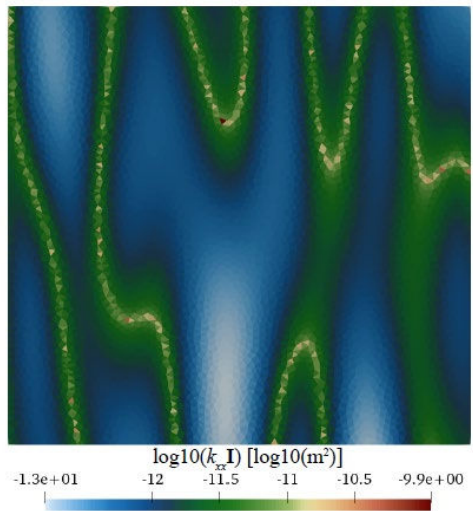
Pressure



Displacement



# Heterogeneous media - POD

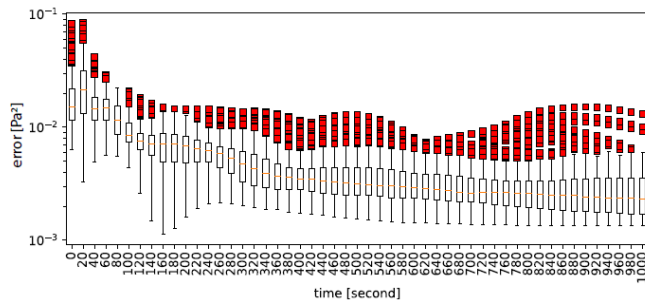


## Wall time used to perform POD

	$N_{\text{int}} = 2$	$N_{\text{int}} = 5$	$N_{\text{int}} = 10$	$N_{\text{int}} = \infty$
$M = 100$	100	125	170	1574
$M = 400$	437	650	1437	36705
$M = 900$	1168	2319	6475	268754

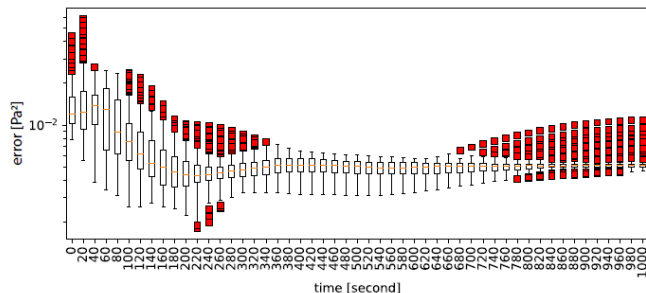
Nested compression could save a lot of time

# Heterogeneous media – 1000 test cases



snapshot = 400, reduced basis = 10 (5),  
hidden layers = 3, and neurons = 7

snapshot = 900, reduced basis = 20 (10),  
hidden layers = 5, and neurons = 10



## Heterogeneous media – costs

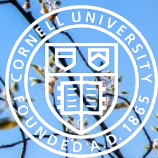
Table 9: Example 4: Comparison of the wall time (seconds) used for sensitivity analysis

	M = 400 (model 1)	M = 900 (model 2)	FOM
Train FOM snapshots	7160	16020	-
Perform POD	1437	6475	-
Train ANN	7064	18492	-
Prediction - 1000 testing $\mu$	2895	3160	17790
Prediction - per testing $\mu$	2.9	3.2	17.8

Taking the training time into account, we need to perform at least 1050 and 2850 inquiries (online phase) to have a break-even point for model 1 and model 2, respectively.

## Current works

- Nonlinear compression – autoencoder and its variants
- Adaptive mesh and timestep
- Physics-informed neural networks



Thank you ☺

<https://arxiv.org/abs/2101.11810>

<https://gitlab.com/multiphenics/multiphenics>

<https://gitlab.com/RBniCS/RBniCS>





# Consensus ADMM for inverse problems governed by multiple PDE models

Luke Lozenski, Washington University in St. Louis, United States

Umberto Villa, Washington University in St. Louis, United States

26 March 2021

The alternating direction method of multipliers (ADMM) provides a natural way of solving inverse problems with multiple forward models and nonsmooth regularization. ADMM allows splitting these large-scale inverse problems into smaller, simpler sub-problems, for which computationally efficient solvers are available. In this work, we are interested in the case in which the forward models stem from partial differential equations and the inversion parameter is a scalar or vector field belonging to an infinite-dimensional Hilbert space. Then, the ADMM methods allow us to split the original inverse problems into several (one for each forward model) single-PDE inverse problems with a smooth Tikhonov-like regularization and, an unconstrained denoising-like problem with non-smooth regularization to update the consensus variable. We discuss several adaptations of the ADMM needed to maintain consistency with the underlining infinite-dimensional problem and ensure scalability. Specifically, we show how using the correct norm in the consensus equations (ie, the one of the underlining Hilbert space) improves both the accuracy and computational efficiency of ADMM. Moreover, we use the Lagrangian formalism to derive expressions of first and second-order optimality conditions for the continuous form of each subproblem, which are then discretized with the finite element method using FEniCS. In particular, for solving the decoupled Tikhonov regularized inverse problems we utilized an inexact Newton conjugate gradient solver in hIPPYlib, an extensible software framework for large-scale inverse problems governed by PDEs. To handle the denoising problem stemming from the consensus variable update we apply the PETScTAOSolver built into FEniCS. We present two imaging applications inspired by electrical impedance tomography and quantitative photoacoustic tomography to demonstrate the proposed method's effectiveness.

---

You can cite this talk as:

Luke Lozenski and Umberto Villa. "Consensus ADMM for inverse problems governed by multiple PDE models". In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 783–805. DOI: 10.6084/m9.figshare.14495652.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/lozenski.html>.

# Consensus ADMM for Inverse Problems Governed by Multiple PDE Models

**Luke Lozenski**, Umberto Villa  
Washington University in St. Louis  
Department of Electrical and Systems Engineering

FEniCS Conference  
March 26 2021

This work was partially supported by the National Science Foundation under Grant No ACI-1550593.

# Motivation

- PDEs are often dependent on unknown (or difficult to measure) parameters associated with physical systems and can be estimated via an inverse problem
- Inverse problems are often-illposed: there's not enough data to recover the parameter
- Regularization selects one solution among many possible solutions
- Non-smooth regularization reinforces certain "nice" properties in solutions: TV enforces sharp edges
- ADMM provides a natural way of splitting these inverse problems into smaller problems.
  - ▶ The subproblems related to the PDEs can be solved efficiently using INCG, which requires a smooth objective term
  - ▶ The term related to the regularization can be solved for separately using other proximal methods
- FEniCS is used for efficient discretization of these variational problems

# ADMM Description

- Equality between solutions of subproblems is reinforced with a consensus term
- ADMM will only reach moderate accuracy in a few iterations and requires many following iterations for high-precision convergence<sup>1</sup>
- This is sufficient for most large-scale applications including
  - ▶ Machine learning
  - ▶ Continuum mechanics<sup>2</sup>
  - ▶ Imaging<sup>3</sup>

---

<sup>1</sup>S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, Foundation and Trends in Machine Learning, Vol. 3, No. 1 (2010).

<sup>2</sup>D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximations, Computers and Mathematics with Applications, Vol. 2, No. 1 (1976)

<sup>3</sup>Y. Wang, J. Yang, W. Yin, and Y. Zhang, A New Alternating Minimization Algorithm for Total Variation Image Reconstruction, SIAM Journal on Imaging Sciences, (2007)

# Setting

Consider the minimization problem

$$\min_{m \in \mathcal{M}} \mathcal{L}(m) + \mathcal{R}(m)$$

- $\mathcal{M}$  is a possibly infinite-dimensional Hilbert space.
- $\mathcal{L} : \mathcal{M} \mapsto \mathbb{R}$  is twice differentiable, may be expensive to evaluate
- $\mathcal{R} : \mathcal{M} \mapsto \mathbb{R}$  is assumed convex and non-smooth

Introduce a consensus variable  $z \in \mathcal{M}$

$$\begin{aligned} \min_{m, z \in \mathcal{M}} \quad & \mathcal{L}(m) + \mathcal{R}(z), \\ \text{s.t.} \quad & m - z = 0 \end{aligned}$$

# Consensus ADMM

We introduce the *augmented Lagrangian* for some  $\rho > 0$

$$L_\rho(m, z, y) = \mathcal{L}(m) + \mathcal{R}(z) + \langle y, m - z \rangle + \frac{\rho}{2} \|m - z\|^2$$

---

## Algorithm 1: Consensus ADMM

---

Begin with starting points  $(m^0, z^0, y^0)$

**while** *While convergence criterion is not met* **do**



$m^{k+1} = \operatorname{argmin}_m L_\rho(m, z^k, y^k)$   
     $z^{k+1} = \operatorname{argmin}_z L_\rho(m^{k+1}, z, y^k)$   
     $y^{k+1} = y^k + \rho(m^{k+1} - z^{k+1})$

**end**

---

4

---

<sup>4</sup>L. Lozanski, U. Villa, "Consensus ADMM for Inverse Problems Governed by Multiple PDE Models", in preparation 2021  

# Inverse problems governed by PDEs

- Goal: Estimate a parameter  $m$  given a measurement  $\mathbf{d} \in \mathcal{D}$  where

$$\mathbf{d} = \mathcal{F}(m) + \mathbf{e},$$

- $\mathcal{F}$  is the composition of a PDE solver and observation operator  $\mathcal{B} : \mathcal{U} \rightarrow \mathcal{D}$ .
- Introduce the *state variable*  $u \in \mathcal{U}$  s.t.

$$\mathcal{F}(m) = \mathcal{B}(u(m)), \quad r(m, u) = 0$$

$$\min_{m \in \mathcal{M}} \mathcal{J}(m) = \frac{1}{2} \|\mathcal{B}(u(m)) - \mathbf{d}\|^2 + \mathcal{R}(m)$$

For a Newton type solution method

- ▶ Using the Lagrangian formalism, gradient computation requires solving two PDEs: the forward & adjoint problems
- ▶ Each Hessian action requires solving two linearized PDEs: the incremental forward & incremental adjoint problems

# The proposed consensus ADMM

---

**Algorithm 2:** The mean based scaled ADMM for parameter inversion with multiple PDEs

---

Let  $q$  be the number of PDEs

Begin with starting points  $(\{m_i^0\}_{i=1}^q, z^0, \{y_i^0\}_{i=1}^q)$

**while** *While convergence criterion is not met*,  $k = 1, \dots$  **do**

**for**  $i = 1, \dots, q$  **do**

$$m_i^{k+1} = \operatorname{argmin}_{m_i} \frac{1}{2q} \|\mathcal{F}_i(m_i) - \mathbf{d}_i\|^2 + \frac{\rho^k}{2q} \|m_i - z^k + y_i^k\|^2$$

**end**

$$\text{Set } \bar{m} = \frac{1}{q} \sum_{i=1}^q m_i^{k+1} \text{ and, } \bar{y} = \frac{1}{q} \sum_{i=1}^q y_i^{k+1}$$

$$z^{k+1} = \operatorname{argmin}_z \mathcal{R}(z) + \frac{\rho}{2} \|\bar{m} - z + \bar{y}\|^2$$

**for**  $i = 1, \dots, q$  **do**

$$y_i^{k+1} = y_i^k + (m_i^{k+1} - z^{k+1})$$

**end**

    Update  $\rho^{k+1}$  adaptively

**end**



# hIPPYlib: Inverse Problem PYthon

- An extensible software framework for PDE-constrained deterministic and Bayesian inverse problems
- Implements state of the art scalable adjoint based algorithms
- Built on FEniCS for discretization of PDEs and PETSc for scalable and efficient linear algebra
- Employs use of advanced structure-exploiting algorithms and approximations
- Maintains consistency with underlying infinite-dimensional problem
- Facilitates experimentation with different priors, observation operators, noise covariance models, model parameter representations, etc.

<https://hippylib.github.io/><sup>5</sup>

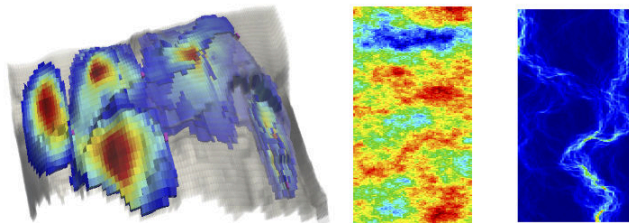
---

<sup>5</sup>Villa et al., (2018). hIPPYlib: An Extensible Software Framework for Large-Scale Inverse Problems. Journal of Open Source Software, 3(30), 940, <https://doi.org/10.21105/joss.00940>

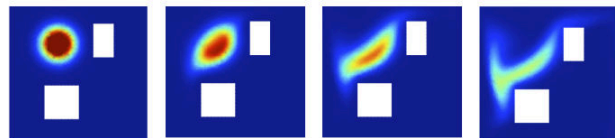
# hIPPYlib features

- Friendly, compact, near-mathematical FEniCS notation to express PDE and likelihood in weak form.
- Automatic generation of efficient code.
- Scalable algorithms
  - ▶ MAP point computation
  - ▶ Low rank representation of posterior covariance via randomized algorithms
  - ▶ Scalable sampling from prior and posterior
  - ▶ Forward/inverse propagation of Uncertainty Quantification

Inference for reservoir modeling



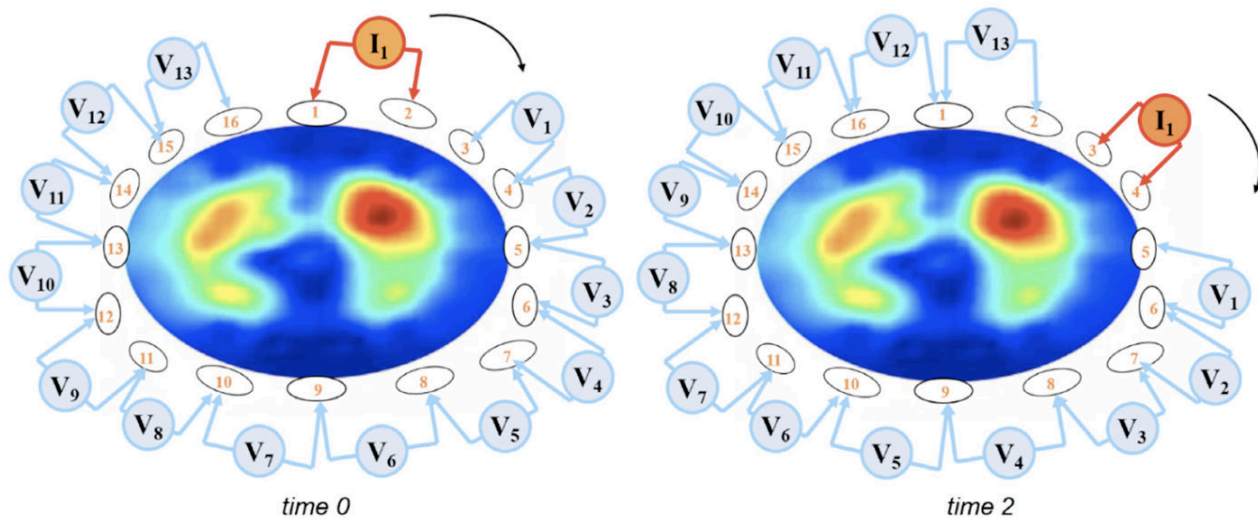
Localization of atmospheric pollution sources



# Electrical Impedance Tomography(EIT)

Electrical Impedance Tomography (EIT) is an imaging modality in which

- An electrical current is introduced on the boundary of an object
- The electric potential is measured on the boundary.
- The potential measurements are used to reconstruct for conductivity



6

# Formulating EIT in the continuous setting

Goal to minimize

$$\frac{1}{q} \sum_{i=1}^q \mathcal{L}_i(m) + \mathcal{R}(m), \quad \mathcal{L}_i(m) = \frac{1}{2} \int_{\Gamma_i} (u_i - \mathbf{d}_i)^2 ds$$

The regularization used was a combination of TV and  $L^2$

The potential  $u_i$  solves the electrostatic Maxwell equation

$$\begin{cases} -\nabla \cdot e^m \nabla u_i = 0 & x \in \Omega \\ \frac{\partial}{\partial \eta} u_i = g_i & x \in \Gamma_N^i \\ u_i = 0 & x \in \Gamma_D^i \end{cases}$$

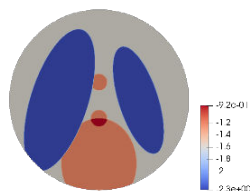
where  $\sigma := e^m$  is the conductivity domain and  $u_i$  is the electric potential resulting from introducing the current  $g_i$

# Discretization

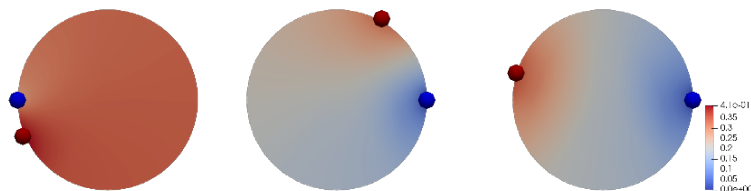
For discretization we applied the finite element method(FEM) used in FEniCS

- $\Omega = D^2$
- Coarsest mesh had 8044 degrees of freedom on  $\mathcal{M}$  and  $\mathcal{U}$
- Parameter updates were accomplished using the INCG algorithm in hIPPYlib
- Consensus updates were found using the PETScTAOSolver built into Fenics

# Ground Truth

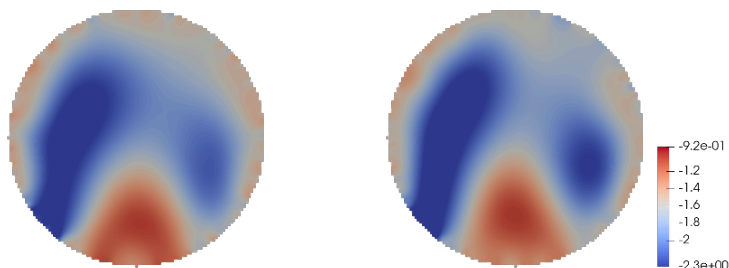


True parameter

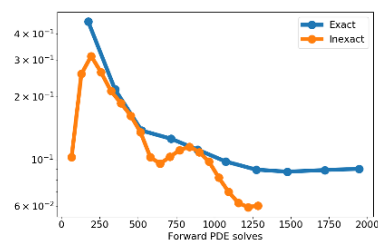
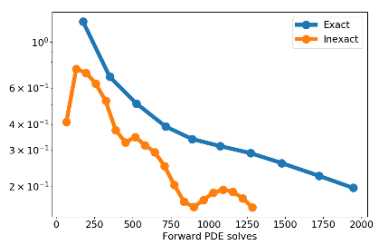
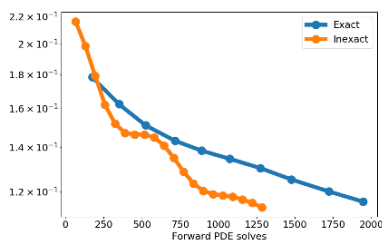


True states 1,11,16 for EIT problem with  $q = 16$

# $H^1$ reconstruction with inexact subproblem solutions



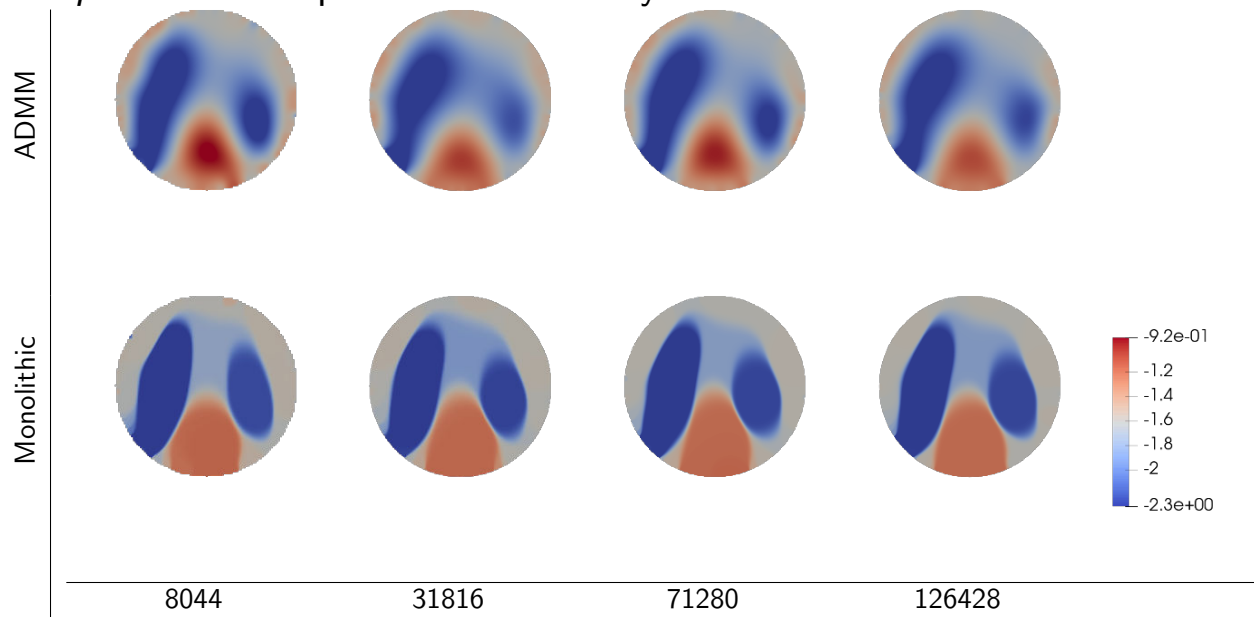
Inverted consensus for EIT problem using exact and inexact  $m$  solves



The relative error, primal and dual residuals wrt Forward PDE solves

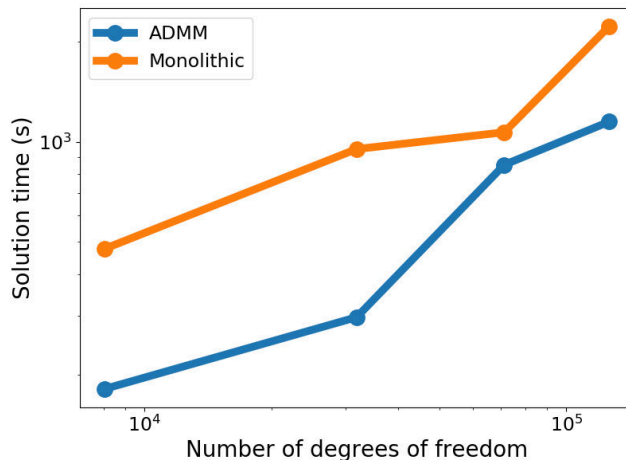
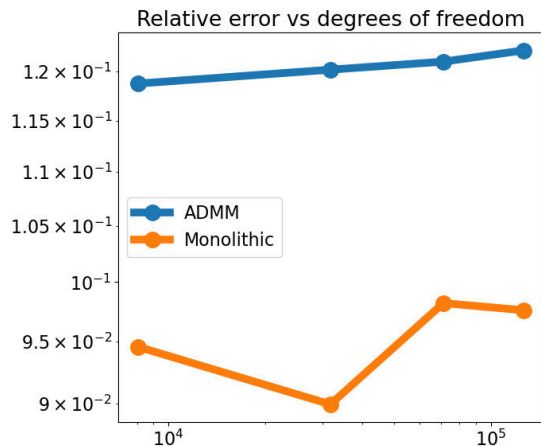
# Scalability with respect to problem size

Fix  $q = 16$  and sequences of uniformly refined meshes



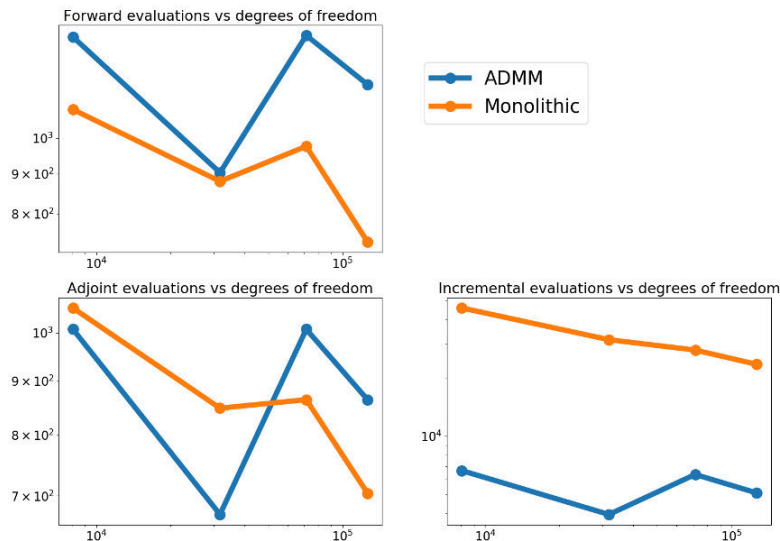


# Accuracy with respect to problem size



Relative error and state misfit for ADMM and monolithic approaches vs number of degrees of freedom

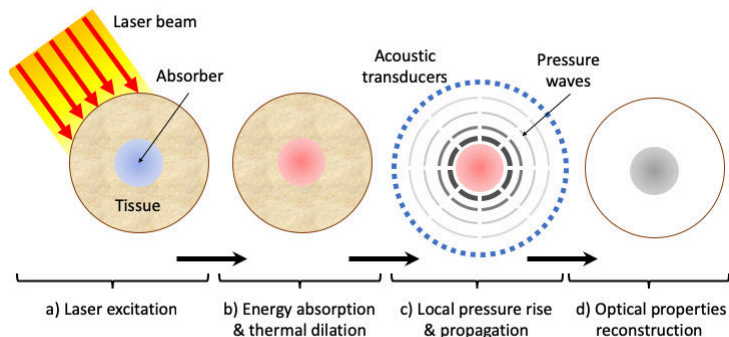
# Scalability with respect to problem size (number of PDE solves)



- Similar results hold for scaling by number of forward models

# Quantitative photoacoustic tomography (qPACT)

- 1 A fast laser pulse is sent into an object
- 2 Underlying material absorbs this energy generating heat and a local increase pressure distribution
- 3 Pressure distribution transitions into acoustic waves and measured on boundary



# Formulation of the qPACT problem

We focused on reconstructing for optical properties given the initial pressure distribution

$$\text{Observation operator } d = \frac{p_0}{\Gamma} = \mu_a \phi + \mathbf{e}$$

Diffusion approximation to radiative transport

$$-\nabla \cdot \frac{1}{3(\mu_a + \mu'_s)} \nabla \phi + \mu_a \phi = 0 \quad x \in \Omega$$

with Robin boundary condition

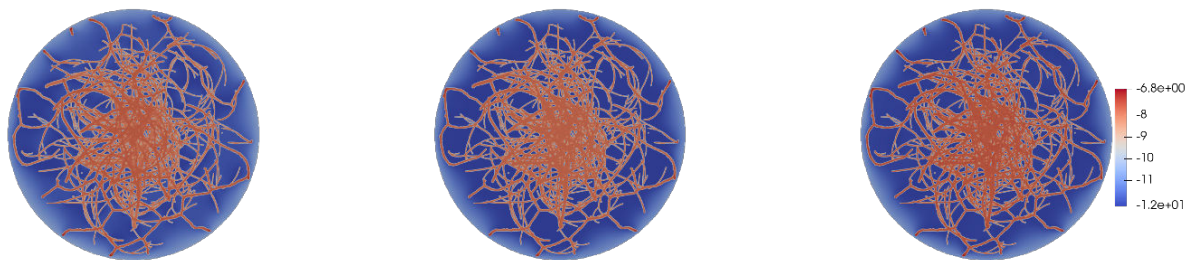
$$\frac{1}{3(\mu_a + \mu'_s)} \frac{\partial \phi}{\partial \eta} + \frac{1}{2} \phi = \frac{1}{2} \phi_0 \quad x \in \partial\Omega$$

Form the data fidelity term

$$\frac{1}{q} \sum_{i=1}^q \mathcal{L}_i(s, c_{thb}, \mu'_s) = \frac{1}{q} \sum_{i=1}^q \|\ln(\mu_{a,i} \phi_i) - \ln(d_i)\|^2$$

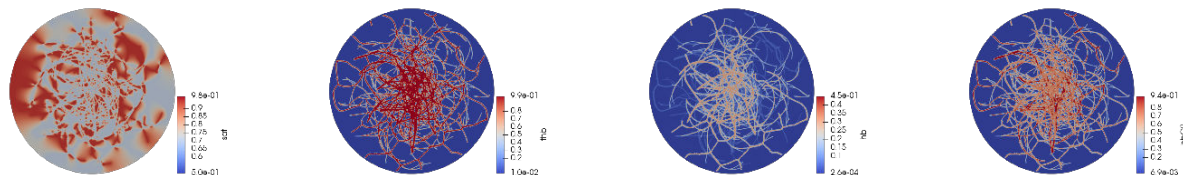
and use regularization with a mixture of Tikhonov, TV, and L1

# Forward Results

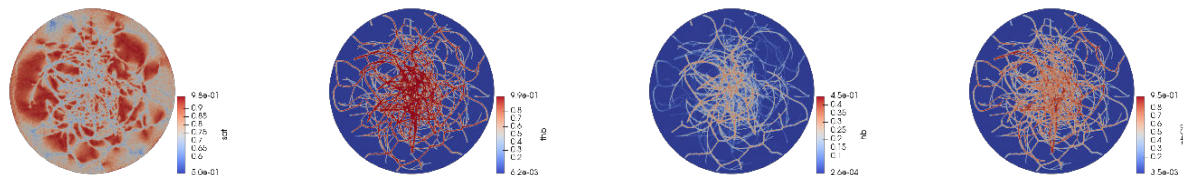


Measurements corresponding to 757, 800, 850 nm

# Reconstruction Results



True  $s$ ,  $c_{thb}$ ,  $c_{hb}$ , and  $c_{hbO_2}$



Reconstructed  $s$ ,  $c_{thb}$ ,  $c_{hb}$ , and  $c_{hbO_2}$

# Conclusions

We presented a framework for solving inverse problems governed by PDE forward models using ADMM

- ADMM is well suited for solving problems involving several large-scale PDE models with nonsmooth regularization
- ADMM solution method significantly reduced computational costs while still achieving satisfactory accuracy

In the future, we plan to improve upon this framework by

- Implementing a primal-dual solver for updating the consensus variable
- Implementing the ADMM process on several processors, with each PDE model being handled by its own set of processors.

The code and EIT example will be included in hIPPYlib

---

This work was partially supported by the National Science Foundation under Grant No ACI-1550593.

# Automating the formulation and resolution of convex variational problems with the `fenics_optim` package

**Jeremy Bleier** (<https://scholar.google.no/citations?user=p9iRhZgAAAAJ>), Laboratoire Navier, Ecole des Ponts ParisTech, Université Gustave Eiffel, France

26 March 2021

In this work, we present a Python package called `fenics_optim` which enables to easily formulate convex variational problems as conic optimization problems within FEniCS. It relies on the conic optimization solver Mosek which uses state-of-the-art interior-point methods for solving large-scale linear, second-order cone and semi-definite programming problems. These are particularly suited for solving non-smooth optimization problems which arise in contact or elasto-plasticity problems for instance but also in the image processing community. In particular, we will present an application to solving limit analysis problems, ie computing directly the limit load of a perfectly plastic structure as a convex optimization problem and without relying on an incremental elasto-plastic procedure until final collapse. I will finish by a recent extension towards limit-analysis based topology optimization.

---

You can cite this talk as:

Jeremy Bleier. “Automating the formulation and resolution of convex variational problems with the `fenics_optim` package”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 806–830. DOI: 10.6084/m9.figshare.14495655.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/bleier.html>.



# Automating the formulation and resolution of convex variational problems with the `fenics_optim` package

Jeremy Bleyer

*Laboratoire Navier, UMR 8205  
Ecole des Ponts ParisTech-Univ Gustave Eiffel-CNRS*



FEniCS 2021 Conference

## Convex variational problems

variational inequalities arise in presence of contact, unilateral conditions (phase-field), plasticity...

$$\begin{array}{ll} \inf_{u \in V} & J(u) \\ \text{s.t.} & u \in \mathcal{K} \end{array}$$

$J$  convex function,  $\mathcal{K}$  convex set

# Convex variational problems

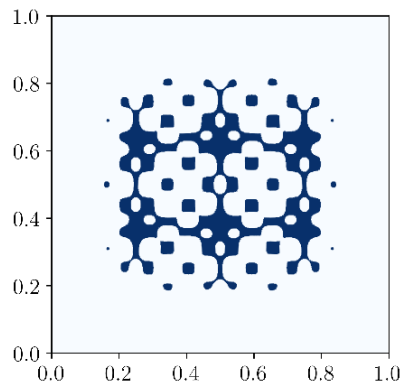
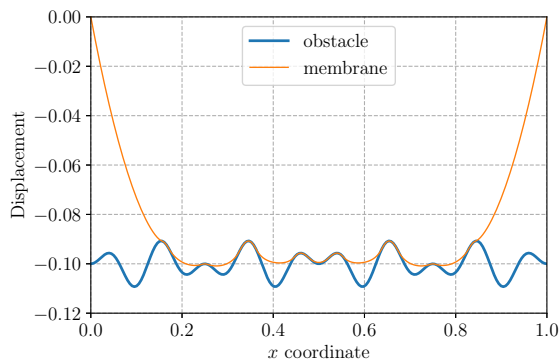
variational inequalities arise in presence of contact, unilateral conditions (phase-field), plasticity...

$$\begin{array}{ll} \inf_{u \in V} & J(u) \\ \text{s.t.} & u \in \mathcal{K} \end{array}$$

$J$  convex function,  $\mathcal{K}$  convex set

e.g. **obstacle problem**:

$$\begin{array}{ll} \inf_{u \in V} & \int_{\Omega} \frac{1}{2} \|\nabla u\|_2^2 \, dx - \int_{\Omega} f u \, dx \\ \text{s.t.} & u \geq g \text{ on } \Omega \end{array}$$



# Conic optimization problems

problems become difficult to solve when  $J$  is **non-smooth** (or  $\mathcal{K}$  complicated)

$$\begin{array}{ll} \inf_{u \in V} & J(u) \\ \text{s.t.} & u \in \mathcal{K} \end{array}$$

# Conic optimization problems

problems become difficult to solve when  $J$  is **non-smooth** (or  $\mathcal{K}$  complicated)

$$\inf_{u \in V} J(u) + \delta_{\mathcal{K}}(u) =: \tilde{J}(u)$$

# Conic optimization problems

problems become difficult to solve when  $J$  is **non-smooth** (or  $\mathcal{K}$  complicated)

$$\begin{array}{ll} \inf_{u \in V, t} & t \\ \text{s.t.} & \tilde{J}(u) \leq t \end{array}$$

# Conic optimization problems

problems become difficult to solve when  $J$  is **non-smooth** (or  $\mathcal{K}$  complicated)

$$\begin{array}{ll} \inf_{u \in V, t} & t \\ \text{s.t.} & \tilde{J}(u) \leq t \end{array}$$

## Conic optimization

$$\begin{array}{ll} \min_{\mathbf{x} \in \mathbb{R}^n} & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & \mathbf{b}_l \leq \mathbf{A}\mathbf{x} \leq \mathbf{b}_u \\ & \mathbf{x} \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p \end{array}$$

where  $\mathcal{K}^j$  are **simple cones**

# Conic optimization problems

problems become difficult to solve when  $J$  is **non-smooth** (or  $\mathcal{K}$  complicated)

$$\begin{array}{ll}\inf_{u \in V, t} & t \\ \text{s.t.} & \tilde{J}(u) \leq t\end{array}$$

## Conic optimization

$$\begin{array}{ll}\min_{\mathbf{x} \in \mathbb{R}^n} & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & \mathbf{b}_l \leq \mathbf{A}\mathbf{x} \leq \mathbf{b}_u \\ & \mathbf{x} \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p\end{array}$$

where  $\mathcal{K}^j$  are **simple cones**

- positive orthant :  $\mathcal{K}^j = \mathbb{R}^{m^+} = \{\mathbf{z} \in \mathbb{R}^m \text{ s.t. } z_i \geq 0\} \Rightarrow \text{LP}$
- Lorentz second-order ("ice-cream") cone :

$$\mathcal{K}^j = \mathcal{Q}_m = \{\mathbf{z} = (z_0, \bar{\mathbf{z}}) \in \mathbb{R} \times \mathbb{R}^{m-1} \text{ s.t. } \|\bar{\mathbf{z}}\| \leq z_0\} \Rightarrow \text{SOCP}$$

- cone of positive semi-definite matrix  $\mathbf{X} \succeq 0 \Rightarrow \text{SDP}$



# Conic optimization problems

problems become difficult to solve when  $J$  is **non-smooth** (or  $\mathcal{K}$  complicated)

$$\begin{aligned} \inf_{u \in V, t} \quad & t \\ \text{s.t.} \quad & \tilde{J}(u) \leq t \end{aligned}$$

## Conic optimization

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{b}_l \leq \mathbf{A}\mathbf{x} \leq \mathbf{b}_u \\ & \mathbf{x} \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p \end{aligned}$$

where  $\mathcal{K}^j$  are **simple cones**

- positive orthant :  $\mathcal{K}^j = \mathbb{R}^{m^+} = \{\mathbf{z} \in \mathbb{R}^m \text{ s.t. } z_i \geq 0\} \Rightarrow \text{LP}$
- Lorentz second-order ("ice-cream") cone :

$$\mathcal{K}^j = \mathcal{Q}_m = \{\mathbf{z} = (z_0, \bar{\mathbf{z}}) \in \mathbb{R} \times \mathbb{R}^{m-1} \text{ s.t. } \|\bar{\mathbf{z}}\| \leq z_0\} \Rightarrow \text{SOCP}$$

- cone of positive semi-definite matrix  $\mathbf{X} \succeq 0 \Rightarrow \text{SDP}$

State-of-the-art **interior point** solvers: CPLEX, MOSEK, CVXOPT

## A more advanced problem

$$c_{\Omega} = \inf_{u \in V_0} \int_{\Omega} \|\nabla u\|_2 \, dx$$

s.t.  $\int_{\Omega} f u \, dx = 1$

antiplane limit analysis, Cheeger problem, first eigenvalue of the 1-Laplacian

## A more advanced problem

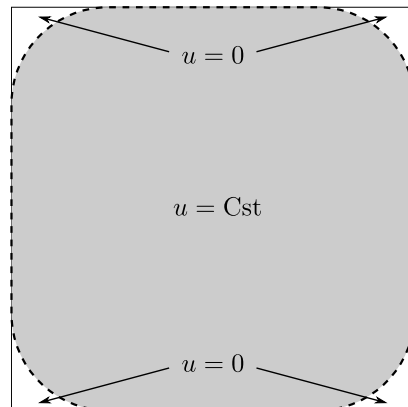
$$c_{\Omega} = \inf_{u \in V_0} \int_{\Omega} \|\nabla u\|_2 \, dx$$

s.t.  $\int_{\Omega} f u \, dx = 1$

antiplane limit analysis, Cheeger problem, first eigenvalue of the 1-Laplacian

**Difficulties:**

- $\|\nabla u\|_2^2$  but  $\|\nabla u\|_2 \Rightarrow$  **non-smooth**
- **discontinuous** solution  $\Rightarrow$  discretization ?



## A more advanced problem

$$c_{\Omega} = \inf_{u \in V_0} \int_{\Omega} \|\nabla u\|_2 \, dx$$

s.t.  $\int_{\Omega} f u \, dx = 1$

antiplane limit analysis, Cheeger problem, first eigenvalue of the 1-Laplacian

**Difficulties:**

- $\|\nabla u\|_2^2$  but  $\|\nabla u\|_2 \Rightarrow$  **non-smooth**
- **discontinuous** solution  $\Rightarrow$  discretization ?

**Conic reformulation:**

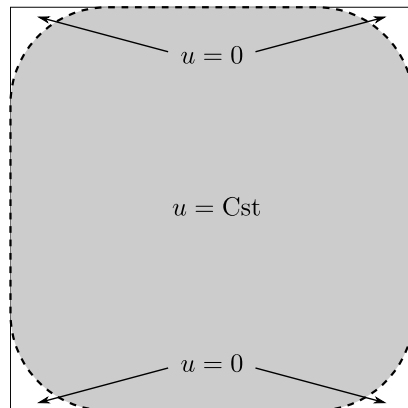
$$\inf_{u \in V_0, \mathbf{z}} \int_{\Omega} z_0 \, dx$$

s.t.  $\int_{\Omega} f u \, dx = 1$

$$\bar{\mathbf{z}} = \nabla u$$

$$\|\bar{\mathbf{z}}\|_2 \leq z_0 \Leftrightarrow \mathbf{z} \in \mathcal{Q}_{d+1}$$

(SOCP problem)



# Conic-representable functions and the `fenics_optim` package

A convex function  $F(\mathbf{x})$  will be *conic-representable* if it can be written as:

$$\begin{aligned} F(\mathbf{x}) = \min_{\mathbf{y}} \quad & \mathbf{c}_x \mathbf{x} + \mathbf{c}_y \mathbf{y} \\ \text{s.t.} \quad & \mathbf{b}_l \leq \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} \leq \mathbf{b}_u \\ & \mathbf{y} \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p \end{aligned}$$

# Conic-representable functions and the `fenics_optim` package

A convex function  $F(\mathbf{x})$  will be *conic-representable* if it can be written as:

$$\begin{aligned} F(\mathbf{x}) = \min_{\mathbf{y}} \quad & \mathbf{c}_x \mathbf{x} + \mathbf{c}_y \mathbf{y} \\ \text{s.t.} \quad & \mathbf{b}_l \leq \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}_u \\ & \mathbf{y} \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p \end{aligned}$$

`fenics_optim` package dedicated to solving problems involving:

$$J(u) = \sum_{i=1}^n \int_{\Omega} F_i(\ell_i(u)) \, dx$$

where  $F_i$  are *conic-representable* and  $\ell_i$  are UFL-representable linear operators

# Conic-representable functions and the `fenics_optim` package

A convex function  $F(\mathbf{x})$  will be *conic-representable* if it can be written as:

$$\begin{aligned} F(\mathbf{x}) = \min_{\mathbf{y}} \quad & \mathbf{c}_x \mathbf{x} + \mathbf{c}_y \mathbf{y} \\ \text{s.t.} \quad & \mathbf{b}_l \leq \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} \leq \mathbf{b}_u \\ & \mathbf{y} \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p \end{aligned}$$

`fenics_optim` package dedicated to solving problems involving:

$$J(u) = \sum_{i=1}^n \int_{\Omega} F_i(\ell_i(u)) \, dx$$

where  $F_i$  are *conic-representable* and  $\ell_i$  are UFL-representable linear operators

Choice of a **quadrature rule**:  $J(u) = \int_{\Omega} F(\ell(u)) \, dx \approx \sum_{g=1}^{N_g} \omega_g F(\mathbf{L}_g \mathbf{u})$

$$\begin{aligned} \Rightarrow \quad \min_{\mathbf{u}} J(\mathbf{u}) = \min_{\mathbf{u}, \mathbf{y}_g} \quad & \sum_{g=1}^{N_g} \omega_g (\mathbf{c}_x \mathbf{L}_g \mathbf{u} + \mathbf{c}_y \mathbf{y}_g) \\ \text{s.t.} \quad & \mathbf{b}_l \leq \mathbf{A} \mathbf{L}_g \mathbf{x}_g + \mathbf{B} \mathbf{y}_g \leq \mathbf{b}_u \\ & \mathbf{y}_g \in \mathcal{K}^1 \times \dots \times \mathcal{K}^p \end{aligned}$$

## Example on the Cheeger problem

auxiliary variables will be implicitly declared on a Quadrature space

```
class L2Norm(ConvexFunction):  
    def conic_repr(self, X):  
        d = self.dim_x  
        z = self.add_var(d+1, cone=Quad(d+1))  
        zbar = as_vector([z[i]  
                           for i in range(1, d+1)])  
        self.add_eq_constraint(X - zbar)  
        self.set_linear_term(z[0])
```



CG1

```
V = FunctionSpace(mesh, "CG", 1)  
prob = MosekProblem("Cheeger problem")  
u = prob.add_var(V, bc=bc)  
  
F = L2Norm(grad(u), degree=0)  
prob.add_convex_term(F)  
  
f = Constant(1.)  
R = FunctionSpace(mesh, "Real", 0)  
def constraint(l):  
    return l*f*u*dx  
prob.add_eq_constraint(R, A=constraint, b=1)  
  
prob.optimize()
```



## Example on the Cheeger problem

auxiliary variables will be implicitly declared on a Quadrature space

```
class L2Norm(ConvexFunction):  
    def conic_repr(self, X):  
        d = self.dim_x  
        z = self.add_var(d+1, cone=Quad(d+1))  
        zbar = as_vector([z[i]  
                           for i in range(1, d+1)])  
        self.add_eq_constraint(X - zbar)  
        self.set_linear_term(z[0])
```



CG2

```
V = FunctionSpace(mesh, "CG", 1)  
prob = MosekProblem("Cheeger problem")  
u = prob.add_var(V, bc=bc)  
  
F = L2Norm(grad(u), degree=0)  
prob.add_convex_term(F)  
  
f = Constant(1.)  
R = FunctionSpace(mesh, "Real", 0)  
def constraint(l):  
    return l*f*u*dx  
prob.add_eq_constraint(R, A=constraint, b=1)  
  
prob.optimize()
```

## Example on the Cheeger problem

auxiliary variables will be implicitly declared on a Quadrature space

```
class L2Norm(ConvexFunction):  
    def conic_repr(self, X):  
        d = self.dim_x  
        z = self.add_var(d+1, cone=Quad(d+1))  
        zbar = as_vector([z[i]  
                           for i in range(1, d+1)])  
        self.add_eq_constraint(X - zbar)  
        self.set_linear_term(z[0])
```

```
V = FunctionSpace(mesh, "CG", 1)  
prob = MosekProblem("Cheeger problem")  
u = prob.add_var(V, bc=bc)  
  
F = L2Norm(grad(u), degree=0)  
prob.add_convex_term(F)  
  
f = Constant(1.)  
R = FunctionSpace(mesh, "Real", 0)  
def constraint(l):  
    return l*f*u*dx  
prob.add_eq_constraint(R, A=constraint, b=1)  
  
prob.optimize()
```



DG1

also works with **facet measures**

## Example on the dual Cheeger problem

**dual problem** with the same objective

$$\begin{aligned} c_{\Omega} = \sup_{\lambda \in \mathbb{R}, \boldsymbol{\sigma} \in W} \quad & \lambda \\ \text{s.t.} \quad & \lambda f = \operatorname{div} \boldsymbol{\sigma} \quad \text{in } \Omega \\ & \|\boldsymbol{\sigma}\|_2 \leq 1 \end{aligned}$$

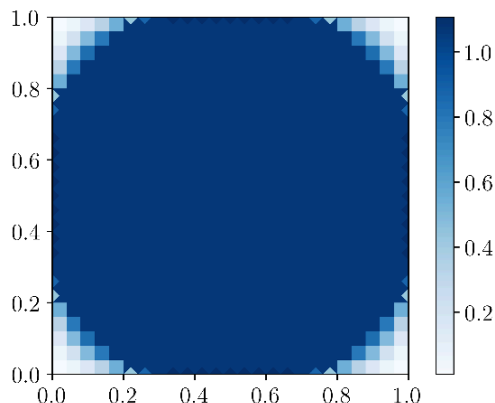
# Example on the dual Cheeger problem

dual problem with the same objective

$$c_{\Omega} = \sup_{\lambda \in \mathbb{R}, \sigma \in W} \lambda$$

s.t.  $\lambda f = \operatorname{div} \sigma \quad \text{in } \Omega$   
 $\|\sigma\|_2 \leq 1$

$\Rightarrow H(\operatorname{div})$ -conforming discretization  
with **RT** elements (lower bound)

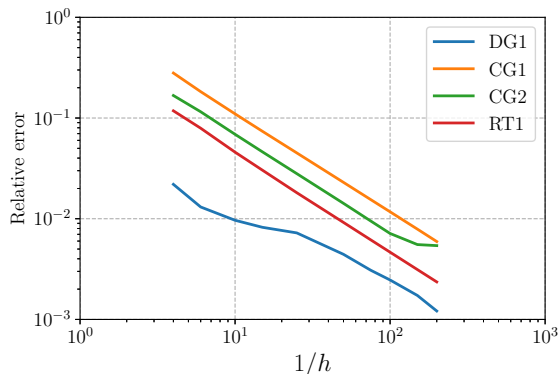


```
prob = MosekProblem("Cheeger dual")
lambda, sig = prob.add_var([R, VRT])

f = Constant(1.)
def constraint(u):
    return u*(lambda*f+div(sig))*dx
prob.add_eq_constraint(VDG0, A=constraint,
    name="u")

F = L2Ball(sig, quadrature_scheme="vertex")
prob.add_convex_term(F)

prob.add_obj_func([1, None])
```



# Variational cartoon/texture decomposition

**Image**  $y = u$  (cartoon) +  $v$  (texture)

Meyer's model (TV + G-norm):

$$\begin{array}{ll} \inf_{u,v} & \int_{\Omega} \|\nabla u\|_2 \, dx + \alpha \|v\|_G \\ \text{s.t.} & y = u + v \end{array}$$

$$\text{where } \|v\|_G = \inf_{\mathbf{g} \in L^\infty(\Omega; \mathbb{R}^2)} \{ \|\sqrt{g_1^2 + g_2^2}\|_\infty \mid v = \operatorname{div} \mathbf{g} \}$$

reformulated as:

$$\begin{array}{ll} \inf_{u,\mathbf{g}} & \int_{\Omega} \|\nabla u\|_2 \, dx \\ \text{s.t.} & y = u + \operatorname{div}(\mathbf{g}) \\ & \|\sqrt{g_1^2 + g_2^2}\|_\infty \leq \alpha \end{array}$$

$L_2$  and  $L_{\infty,2}$ -norms are **conic-representable**  $\Rightarrow$  SOCP problem

# Variational cartoon/texture decomposition

**Image**  $y$  : represented by a  $DG0$  field on a  $512 \times 512$  finite-element mesh

$u, g \in \mathbb{CR} \times \mathbb{RT}$

```
prob = MosekProblem("Cartoon/texture decomposition")
Vu = FunctionSpace(mesh, "CR", 1)
Vg = FunctionSpace(mesh, "RT", 1)
u, g = prob.add_var([Vu, Vg])

def constraint(l):
    return dot(l, u + div(g))*dx
def rhs(l):
    return dot(l, y)*dx
prob.add_eq_constraint(Vu, A=constraint, b=rhs)

tv_norm = L2Norm(grad(u))
prob.add_convex_term(tv_norm)

g_norm = L2Ball(g, k=alpha)
prob.add_convex_term(g_norm)

prob.optimize()
```

# Variational cartoon/texture decomposition

**Image**  $y$  : represented by a  $DG0$  field on a  $512 \times 512$  finite-element mesh

$u, g \in \mathbb{CR} \times \mathbb{RT}$

Original image



Cartoon layer



Texture layer

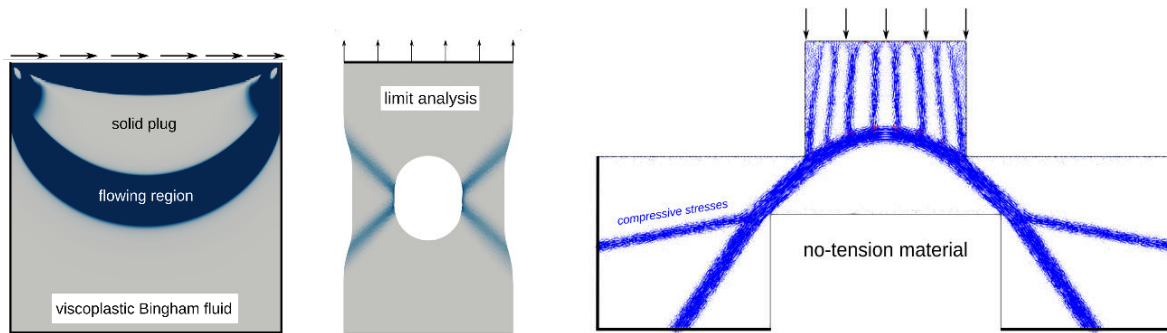


*Barbara* image

# Conclusions

Package available at <https://gitlab.enpc.fr/navier-fenics/fenics-optim>

- UFL syntax for **conic-representable functions**
- supports **LP**, **SOCP**, **SDP**, exponential and power cones via Mosek
- **other applications** : viscoplastic fluids, limit analysis, topology optimization, nonlinear membranes/shells, inpainting, optimal transport, etc.



## Perspectives

- other IPM solvers, custom solver ?
- first-order solvers (**proximal algorithms**)
- porting to `dolfin-x`

*Bleyer J., TOMS, 46(3), 1-33. 2020*



# Generating high-order time stepping methods

**Robert Kirby**, Baylor University, United States

Jorge Marchena-Menendez, Baylor University, United States

Patrick Farrell, University of Oxford, United Kingdom

26 March 2021

Code generation systems greatly simplify the formulation of physical problems, allowing efficient and accurate discretizations to be rapidly deployed for challenging problems. However, domain-specific languages like UFL currently lack abstractions to describe time-dependence, leaving users to hand-code multistep or Runge–Kutta methods if they wish to obtain high temporal accuracy or utilize special time-stepping strategies.

In this talk, we describe IRKsome, a simple package that, given a UFL description of a semidiscrete problem and a Butcher tableau, produces UFL for the associated Runge–Kutta method. In this way, we can obtain high-order time-stepping methods with appropriate stability and/or symplecticity properties. On the down side, implicit Runge–Kutta methods lead to algebraic systems coupling together the several stages, which presents greater challenges to the linear solvers. We also present preliminary results on preconditioners.

Although Irksome currently sits on top of Firedrake, the critical aspects of UFL manipulation (the hard part) should work well with FEniCS or other UFL-based codes.

---

You can cite this talk as:

Robert Kirby, Jorge Marchena-Menendez, and Patrick Farrell. “Generating high-order time stepping methods”. In: *Proceedings of FEniCS 2021, online, 22–26 March* (eds Igor Baratta, Jørgen S. Dokken, Chris Richardson, Matthew W. Scroggs) (2021), 831. DOI: 10.6084/m9.figshare.14495661.

BibTeX for this citation can be found at <https://mscroggs.github.io/fenics2021/talks/kirby.html>.