FEM@LLNL Seminar Series, June 12, 2025

Interpolation-Based Immersed Finite Element and Isogeometric Analysis

John A. Evans

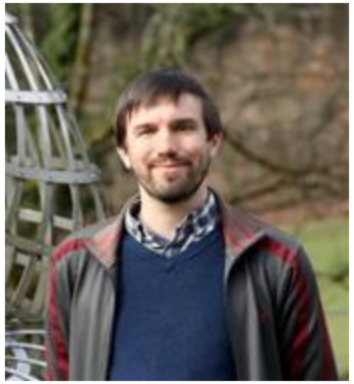
Associate Professor

Smead Aerospace Engineering Sciences

University of Colorado Boulder

john.a.evans@colorado.edu

CU Boulder

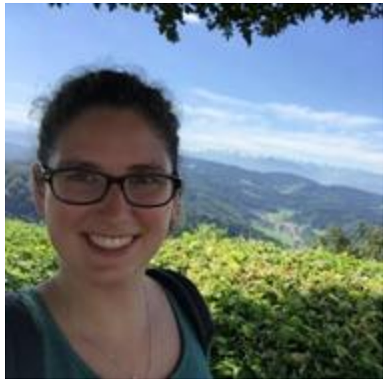

John Evans


Kurt Maute


Nils Wunsch

UCSD

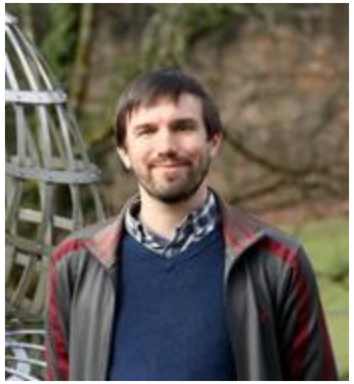

David Kamensky


Jennifer Fromm


Ru Xiang


Han Zhao

CU Boulder



John Evans

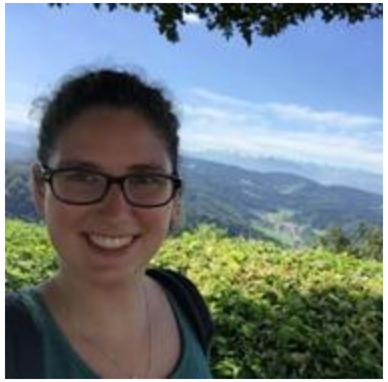
Kurt Maute

Nils Wunsch

Ryan Caputo

UCSD

J.S. Chen

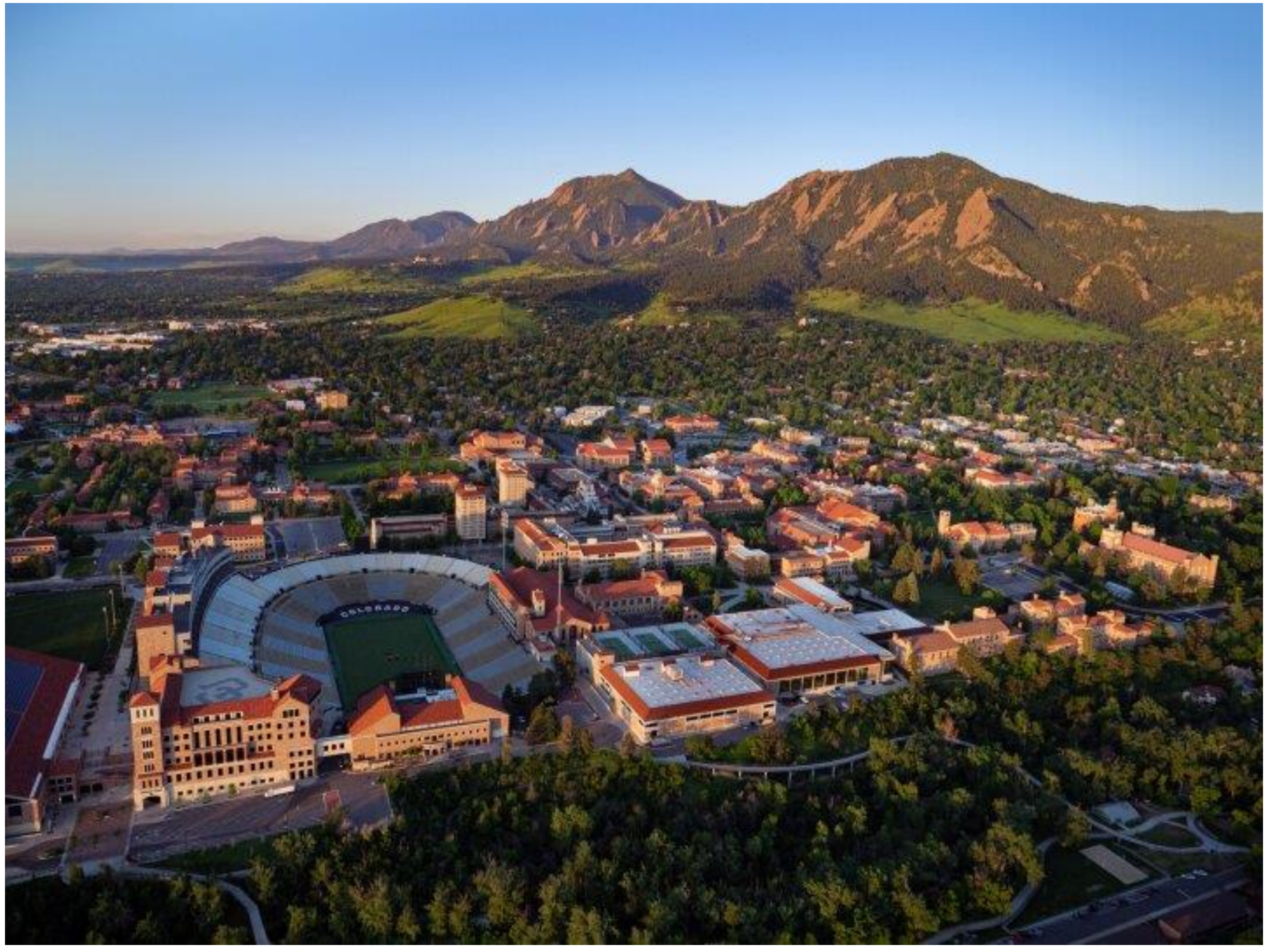


Jennifer Fromm

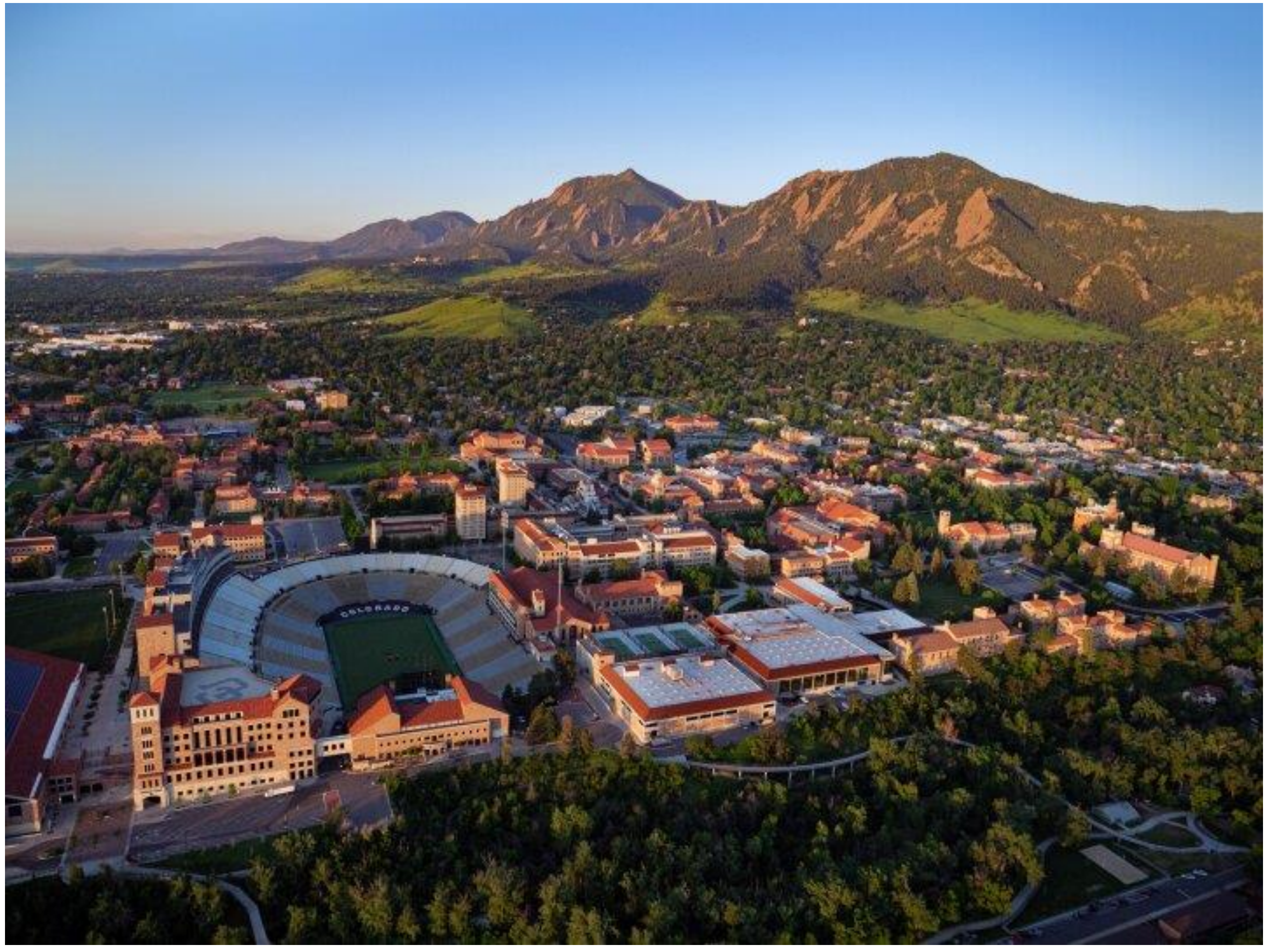
Ru Xiang

Han Zhao

Smead Aerospace Engineering Sciences at CU Boulder



Smead Aerospace Engineering Sciences at CU Boulder



Part 1:

Motivation

Motivation

Classical finite element analysis requires the generation of a high quality body-fitted finite element mesh which can require upwards of 80% of the time required in a design-through-analysis cycle:

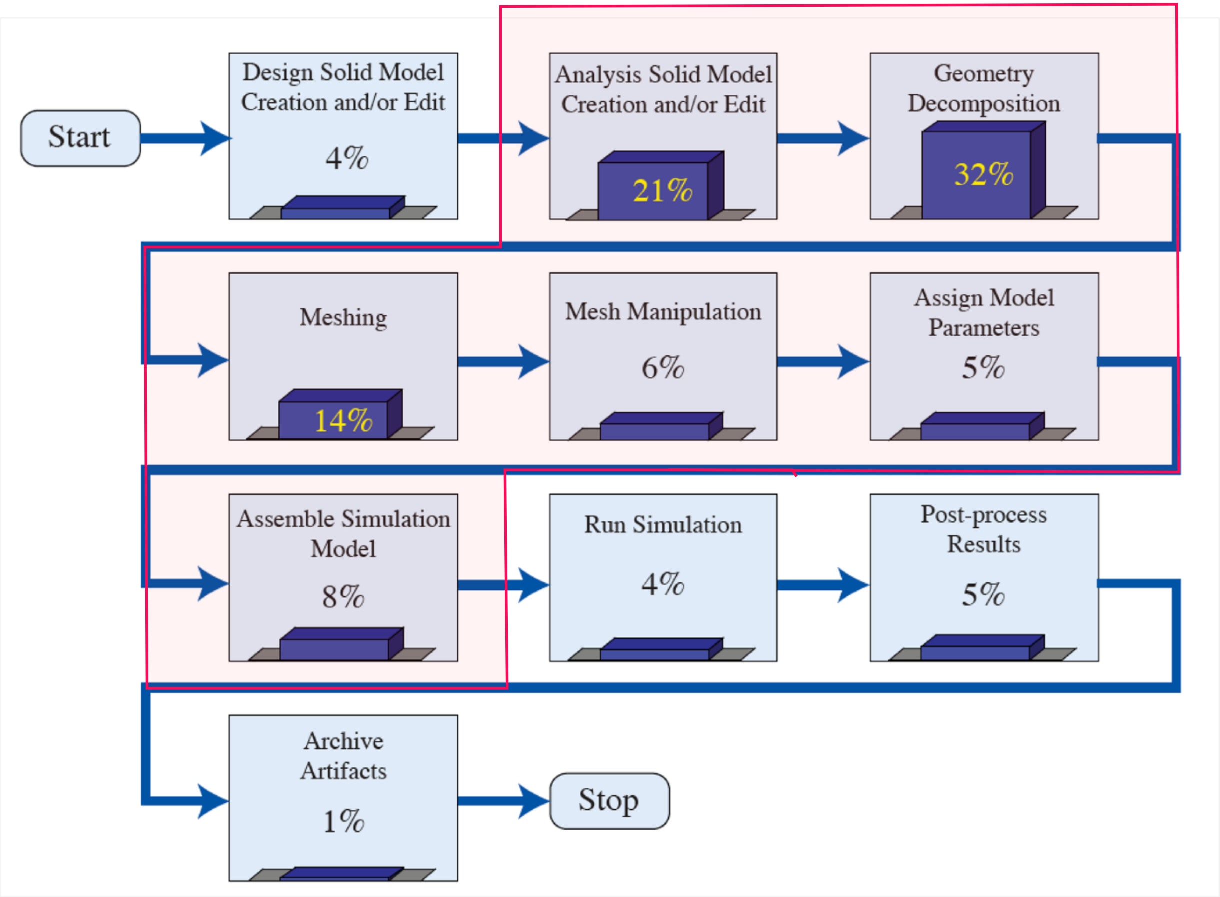
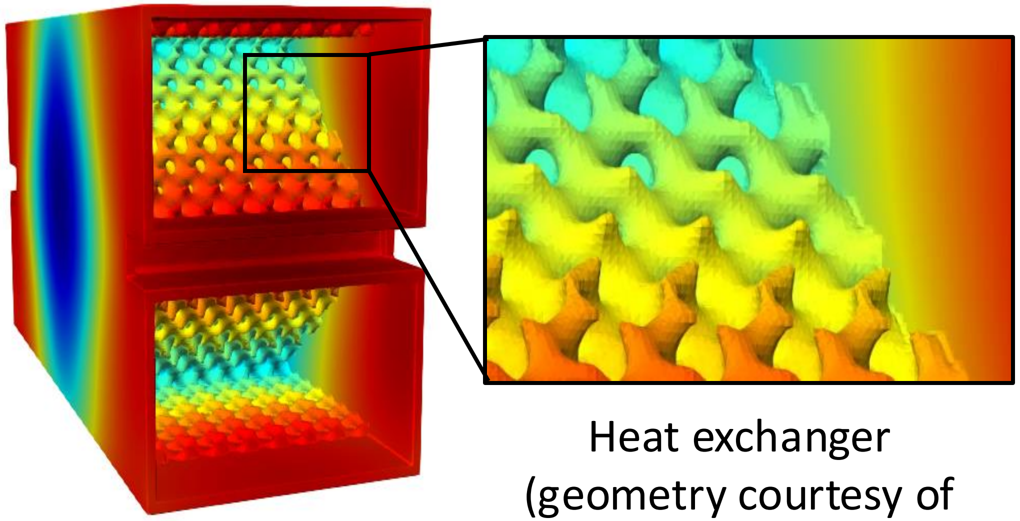


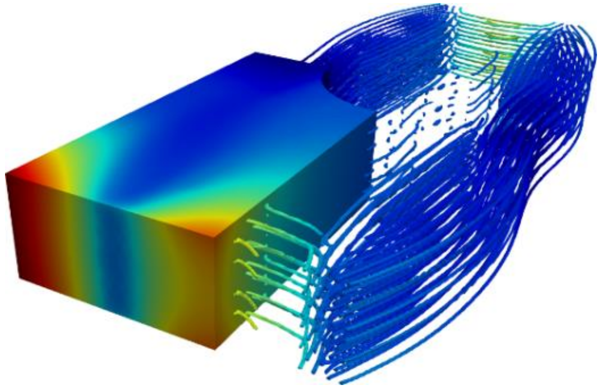
Figure courtesy of Sandia National Laboratories

Motivation

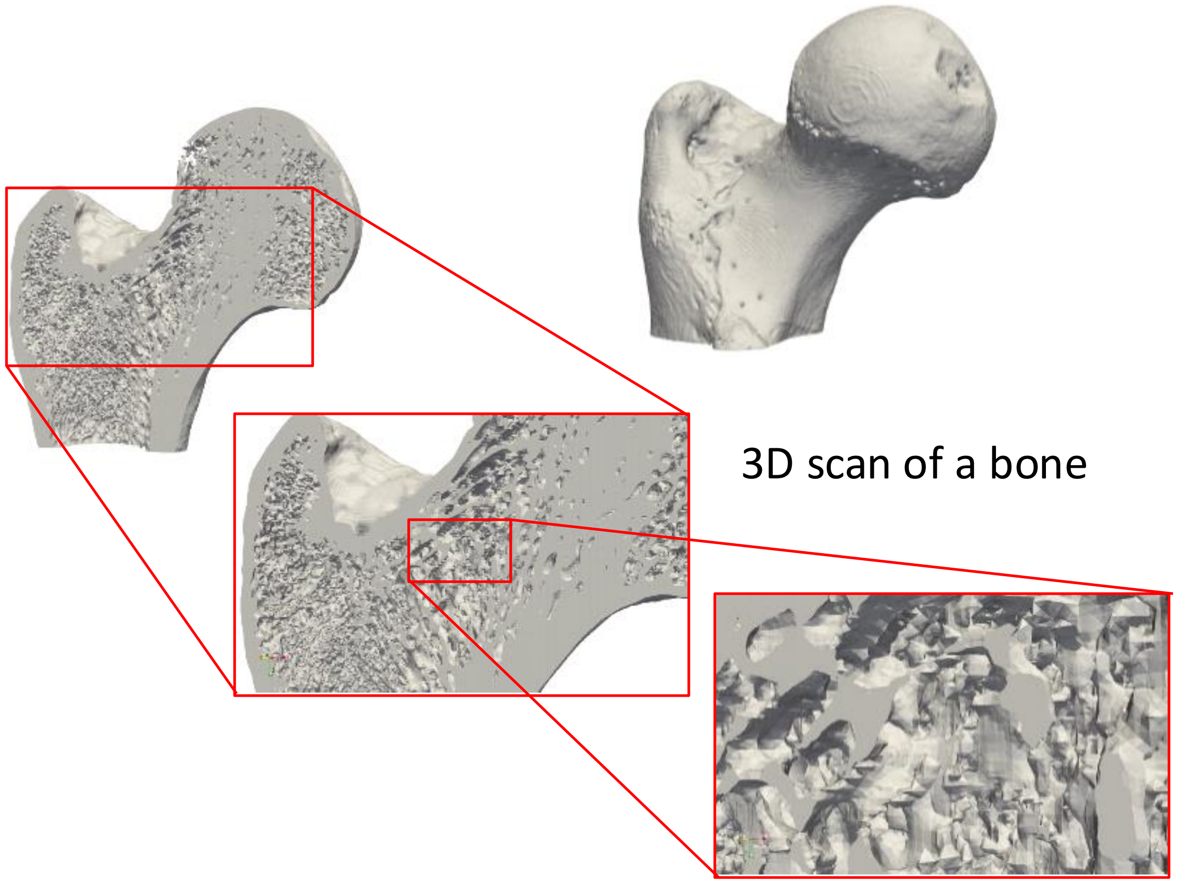
This is especially true for geometrically complex, multi-material, multi-physics problems:



Heat exchanger (geometry courtesy of nTopology)



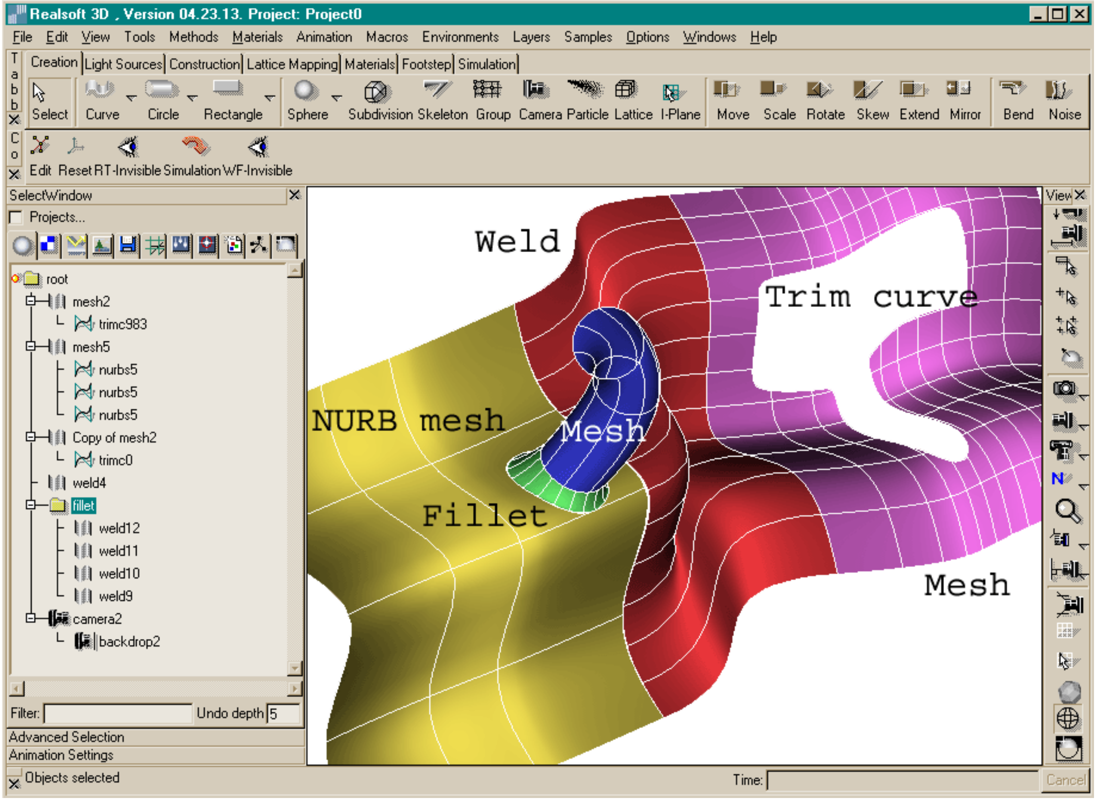
Fiber composite



3D scan of a bone

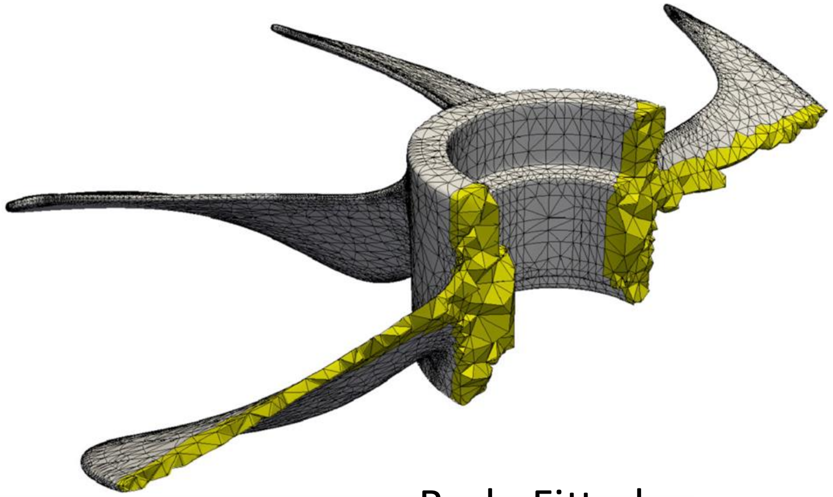
Motivation

Isogeometric analysis directly employs the basis employed in CAD for CAE and thus bypasses the need for the generation of a high quality body-fitted finite element mesh. However, isogeometric analysis requires an explicit parameterization of the geometric domain, and thus it cannot be directly applied to trimmed CAD geometries:

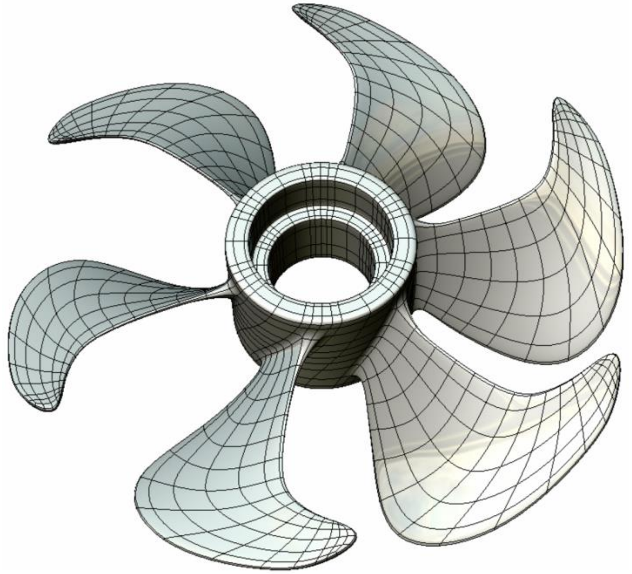


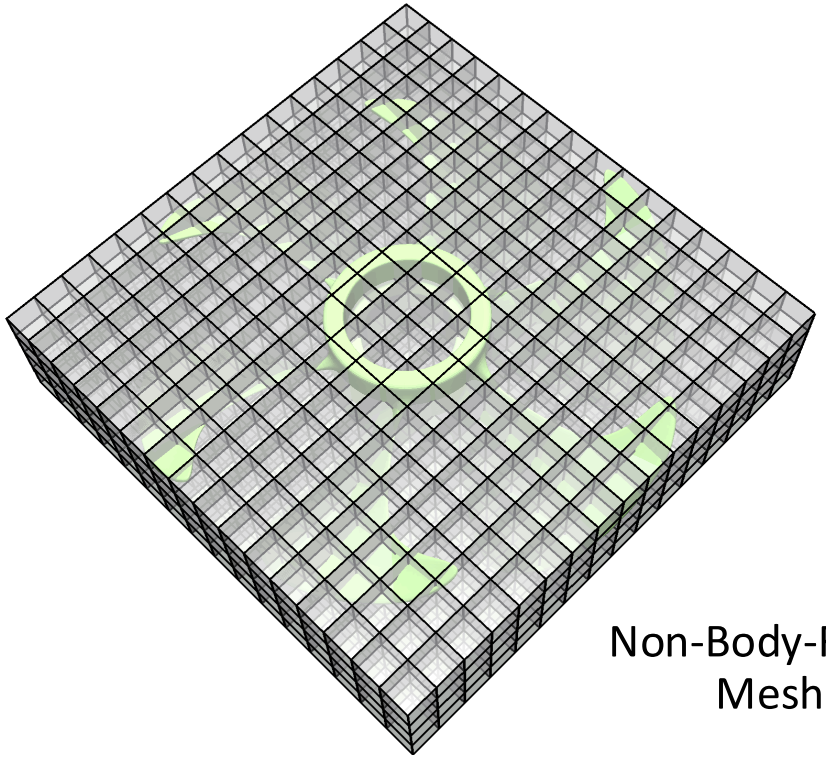
Motivation

Immersed finite element (and isogeometric) analysis is another approach that bypasses the need for the generation of a high quality body-fitted finite element mesh. In this approach, basis functions defined on a non-body-fitted background mesh are instead used for CAE, so it can be directly applied to trimmed CAD geometries:



Body-Fitted Mesh



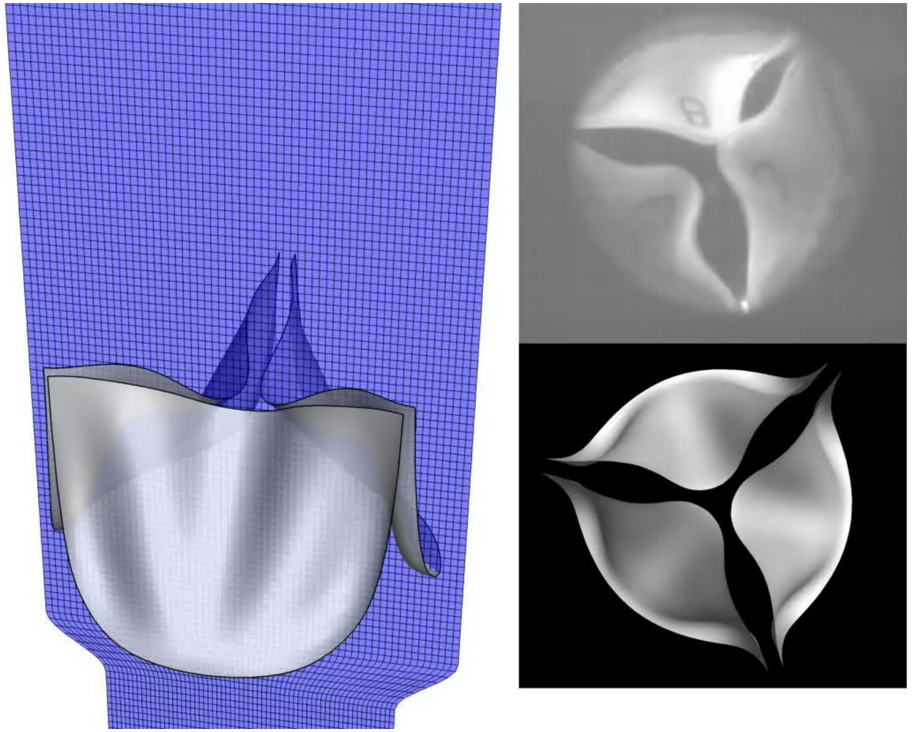


Non-Body-Fitted Mesh

Schillinger, D., et al. "An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces." *CMAME* 249 (2012): 116-150.

Motivation

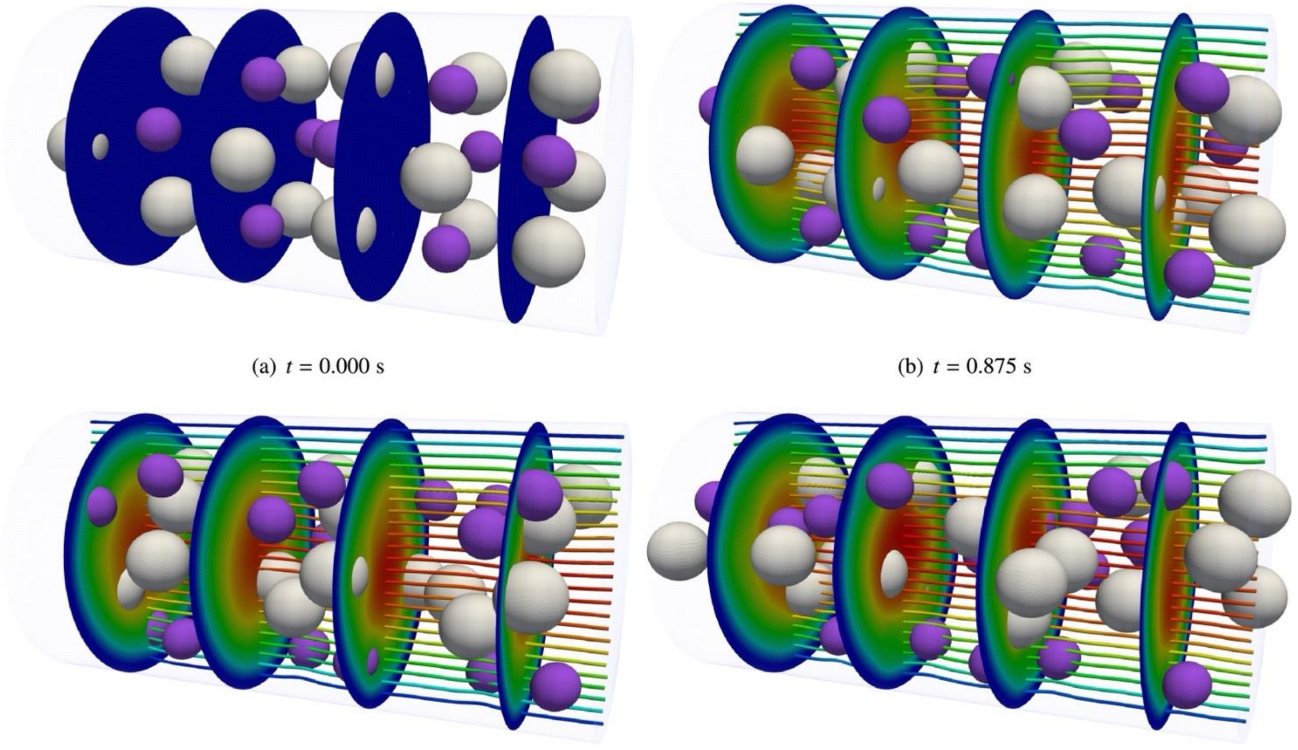
Immersed finite element analysis can also be applied to problems out of reach by both classical finite element analysis and isogeometric analysis, such as problems exhibiting a change in domain topology:



Kamensky, D., et al. "Immersogeometric cardiovascular fluid–structure interaction analysis with divergence-conforming B-splines." *CMAME* 314 (2017): 408-472.

Motivation

Immersed finite element analysis can also be applied to problems out of reach by both classical finite element analysis and isogeometric analysis, such as problems exhibiting a change in domain topology:

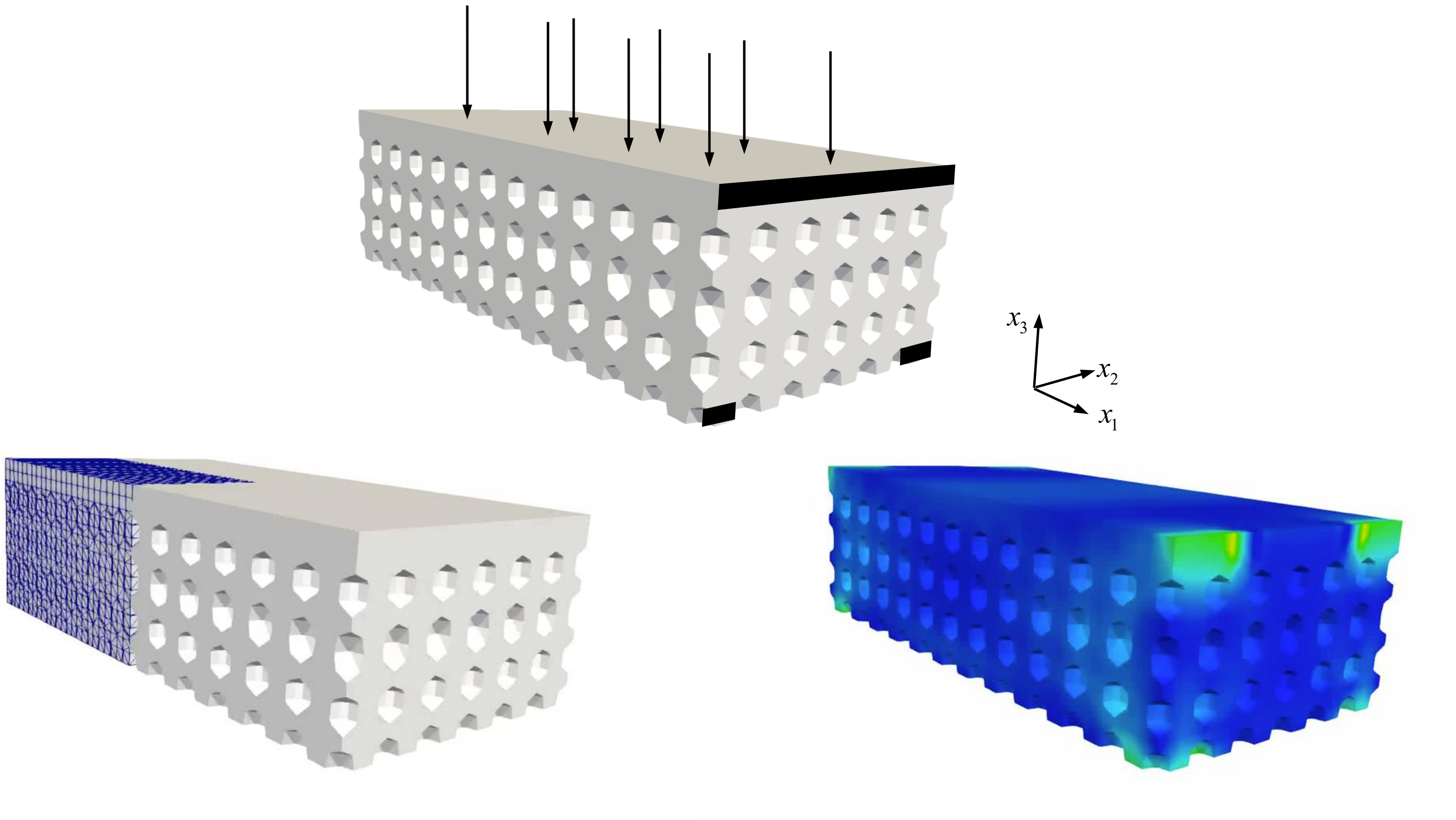


(a) $t = 0.000$ s (b) $t = 0.875$ s (c) $t = 1.375$ s (d) $t = 2.100$ s

Casquero, H.., et al. "The divergence-conforming immersed boundary method: Application to vesicle and capsule dynamics." *JCP* 425 (2021): 109872.

Motivation

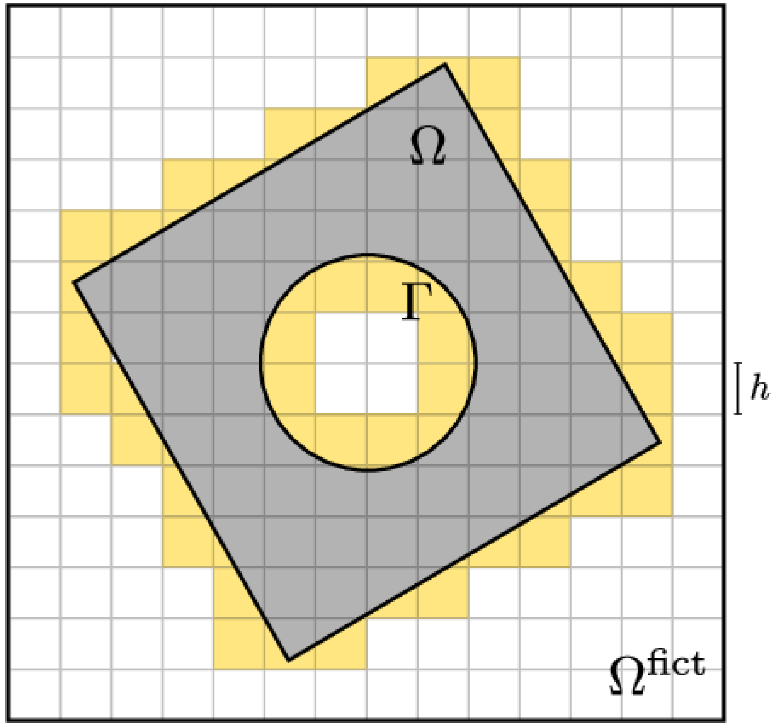
Immersed finite element analysis is also an ideal technology for level set topology optimization as the geometric domain does need to be re-parametrized or re-meshed at every design iteration:

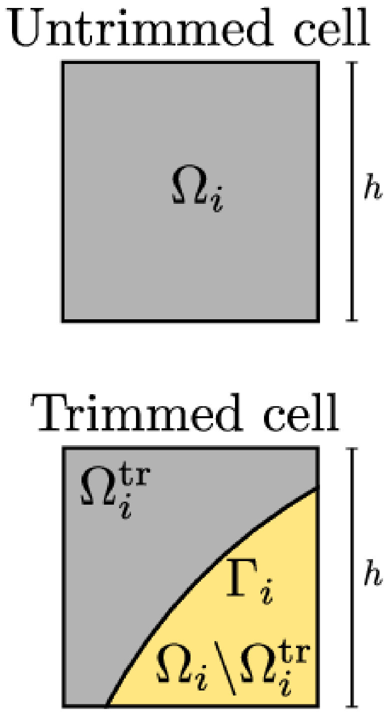


Noël, L., et al. "Adaptive level set topology optimization using hierarchical B-splines."
SAMO 62.4 (2020): 1669-1699.

Motivation

However, implementation of an immersed finite element analysis code is a challenging and time consuming task even for domain experts. The primary challenge is numerical integration over cut cells:

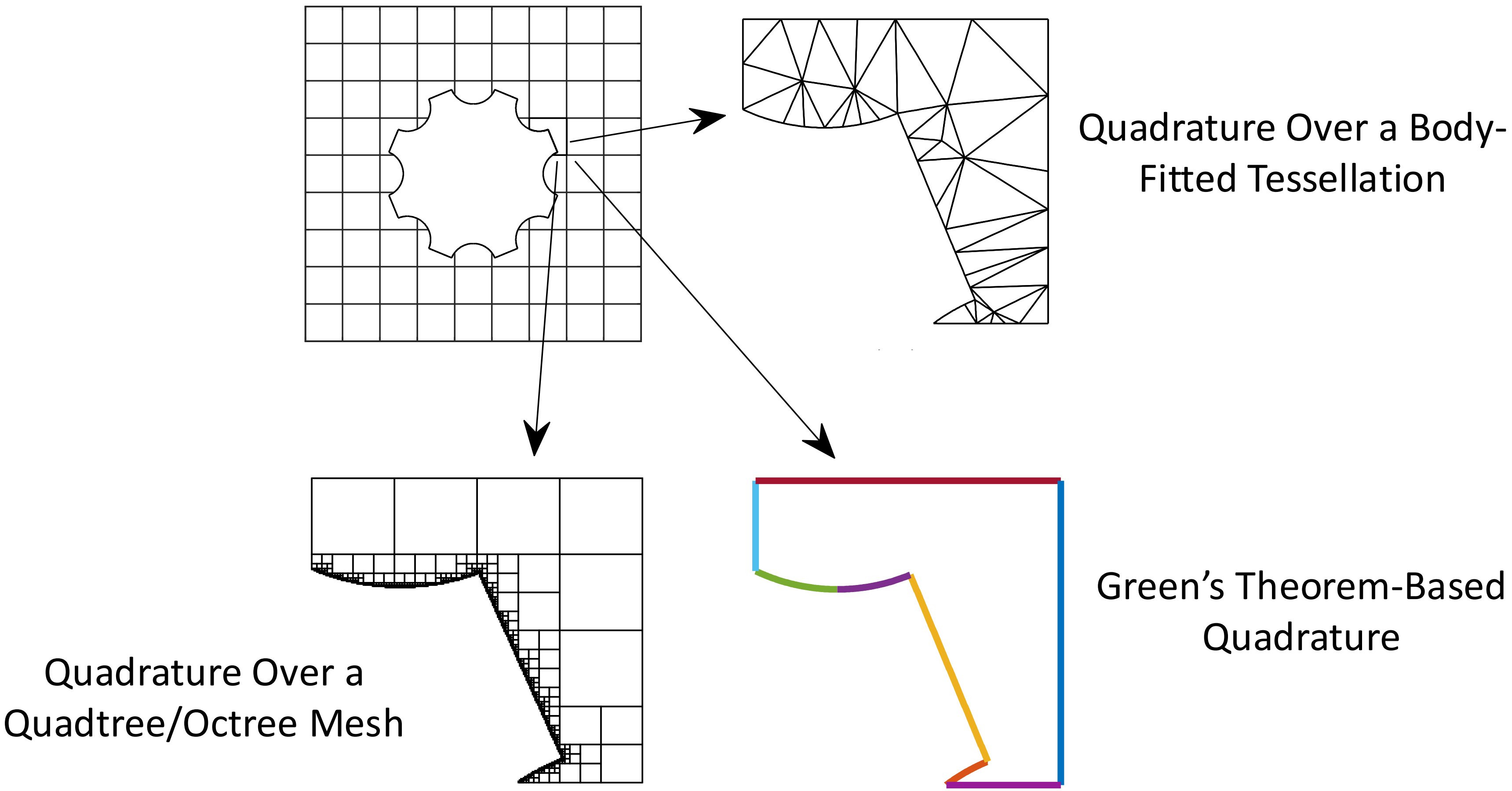




Gunderman, D., K. Weiss, and J.A. Evans. "Spectral mesh-free quadrature for planar regions bounded by rational parametric curves." *CAD* 130 (2021): 102944.

Motivation

Several different strategies have been introduced for this integration, each giving rise to a specialized quadrature rule for every single cut cell:



Motivation

The purpose of the EXHUME project is to ease the burden of implementing an immersed finite element code.

In particular, the EXHUME project enables one to transform classical finite element codes into immersed finite element codes with minimal implementation effort.

The key technology underlying the EXHUME project is the concept of interpolation-based immersed finite element analysis.

Part 2:

Interpolation-Based Immersed Finite Element Analysis

Defining the Method for a Model Problem

- Use **Poisson equation** as model PDE for exposition
- *Strong form:* Find solution u s.t.

$$-\Delta u = f \quad \text{in} \quad \Omega$$

$$u = g \quad \text{on} \quad \partial\Omega$$

- *Weak form:* Find $u \in H_0^1(\Omega) + \ell_g$ s.t. $\forall v \in H_0^1(\Omega)$

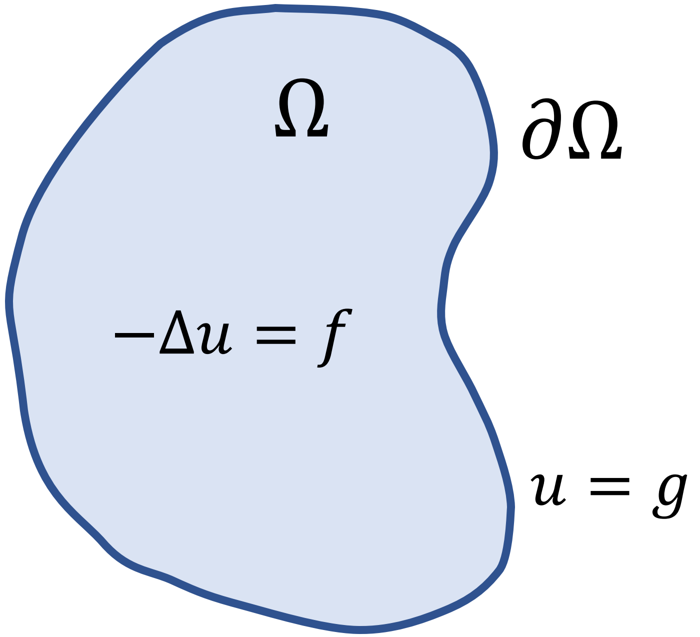
$$\int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega$$

- *Discrete problem:* Find $u^h \in V^h \subset H^1(\Omega)$ s.t. $\forall v^h \in V^h$

$$\int_\Omega \nabla u^h \cdot \nabla v^h \, d\Omega - \int_{\partial\Omega} \nabla u^h \cdot \boldsymbol{n} \, v^h \, d\Gamma$$

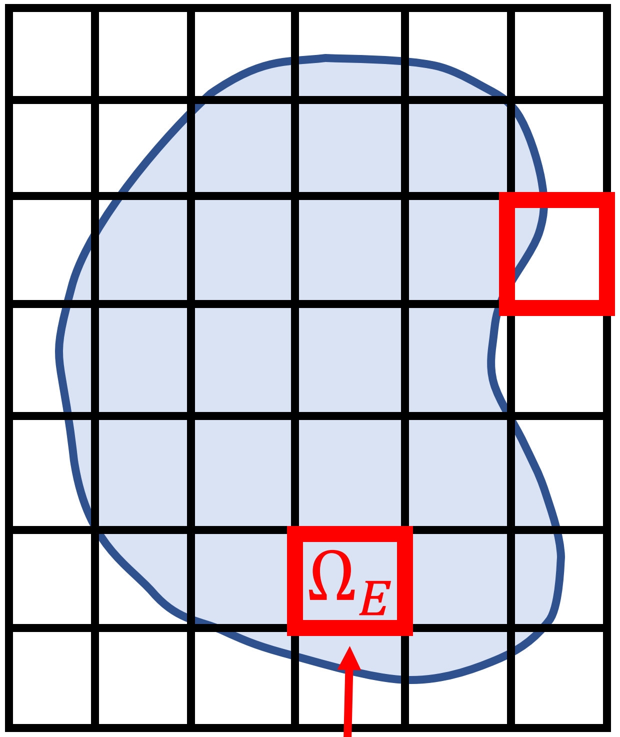
$$\mp \int_{\partial\Omega} \nabla v^h \cdot \boldsymbol{n} \left(u^h - g\right) d\Gamma$$

$$+ \int_{\partial\Omega} \frac{\alpha}{h} \left(u^h - g\right) v^h \, d\Gamma = \int_\Omega f v^h \, d\Omega$$



Using Nitsche's method for weak enforcement of Dirichlet BCs

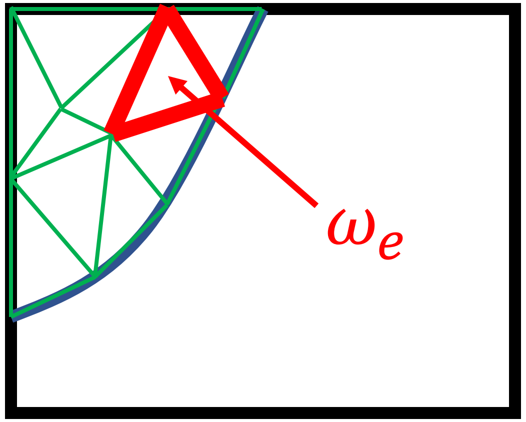
Quadrature-Based Immersed Methods



Define V^h on **background mesh**

Challenge: Integrate accurately over intersections of domain with **cut cells**

Can use standard element Gaussian quadrature on cells contained in Ω



Solution: Tesselate mesh-cell intersections with quadrature elements, forming a **foreground mesh**

Can use standard Gaussian quadrature on elements of foreground mesh

In Mathematical Notation:

Discrete problem is: Find $u^h \in V^h$ s.t. $\forall v^h \in V^h$,

$$a_h\left(u^h, v^h\right) = L_h\left(v^h\right),$$

where

$$a_h(u, v) \coloneqq \sum_{e=1}^{\nu_{el}} \int_{\omega_e} \nabla u \cdot \nabla v \, d\Omega + \sum_{e=1}^{\nu_{el}} \int_{\partial\omega_e \cap \partial\Omega} -v \nabla u \cdot \boldsymbol{n} \mp u \nabla v \cdot \boldsymbol{n} + \left(\frac{\alpha}{h}\right) uv \, d\Gamma,$$

$$L_h(v) \coloneqq \sum_{e=1}^{\nu_{el}} \int_{\omega_e} f v \, d\Omega + \sum_{e=1}^{\nu_{el}} \int_{\partial\omega_e \cap \partial\Omega} \mp g \nabla v \cdot \boldsymbol{n} + \left(\frac{\alpha}{h}\right) gv \, d\Gamma,$$

$\{\omega_e\}_{e=1}^{\nu_{el}}$ are **elements of the foreground mesh**, and the discrete space is

$$V^h \coloneqq \mathrm{span}\{N_i|_\Omega\}_{\mathrm{i}=1}^{\mathrm{n}}$$

with basis functions $\{N_i\}_{i=1}^{n}$ **defined on the background mesh**.

Implementation problem: Quadrature defined on foreground mesh, but basis functions defined on background mesh; cannot use standard FEM data structures and assembly algorithms!

Interpolation-Based Immersed Methods

Want: Basis functions used for assembly defined on the same mesh as the quadrature rule.

Approach:

- Define nodal basis on foreground mesh: $\{\phi_j\}_{j=1}^{\nu}$ with nodes $\{\boldsymbol{x}_j\}_{j=1}^{\nu}$
- Interpolate background basis functions in foreground space: $\widehat{N_i} = \sum_{j=1}^{\nu} N_i(\boldsymbol{x}_j)\phi_j$
- Use span of interpolated background basis functions as solution space: $V^h = \mathrm{span}\{\widehat{N_i}\}_{i=1}^{n}$

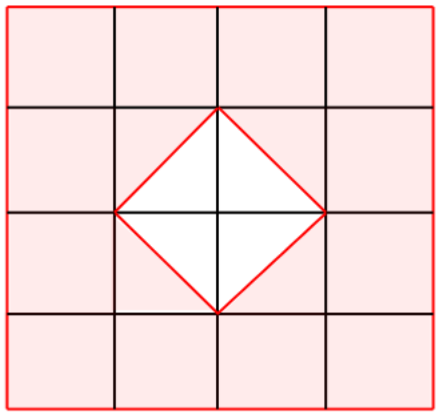
Equivalent to quadrature-based method *if*:

- Foreground mesh fitted to background mesh
- Foreground basis can exactly represent all monomials in background basis functions

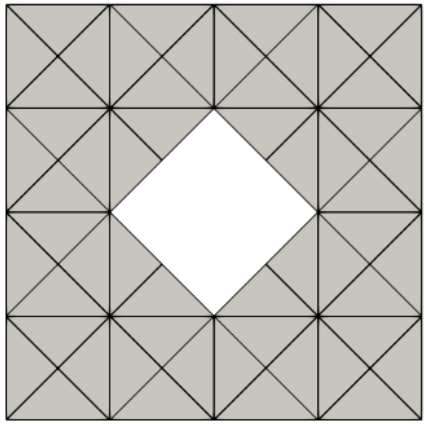
Possible approximations (new numerical methods):

- Reduced polynomial degree of foreground basis $\Rightarrow$ More efficient assembly
- Background-unfitted foreground mesh $\Rightarrow$ More flexibility in mesh generation

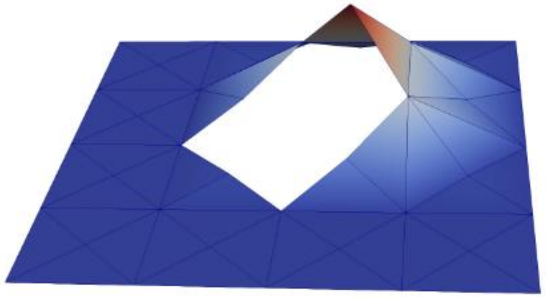
Interpolated Basis Functions



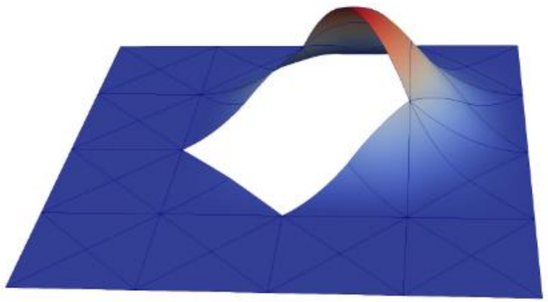
Domain in red,
background mesh in black



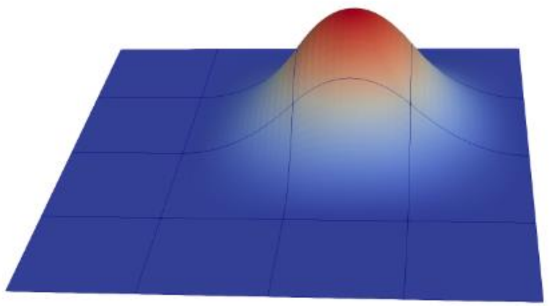
"Background-fitted"
foreground mesh



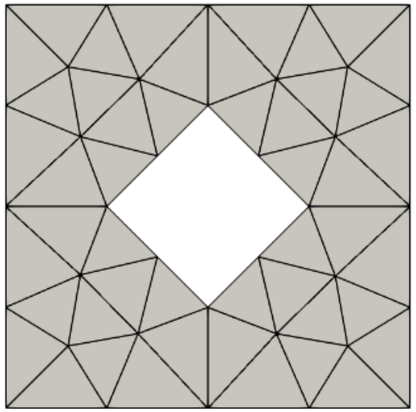
$\widehat{N_i}$ for piecewise-linear Lagrange
foreground basis $\{\phi_j\}$



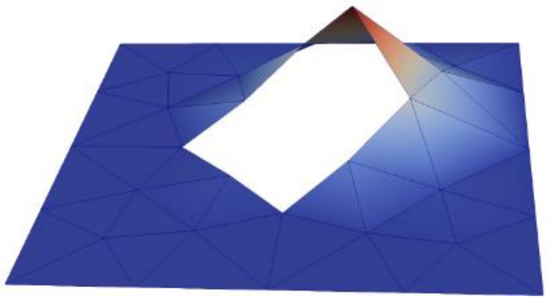
$\widehat{N_i}$ for piecewise-quadratic
Lagrange foreground basis $\{\phi_j\}$



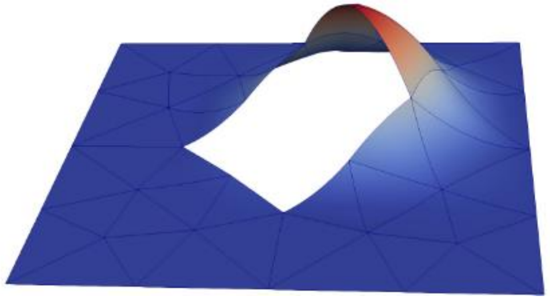
C^1 quadratic B-spline
background basis function N_i



"Background-*un*fitted"
foreground mesh



$\widehat{N_i}$ for piecewise-linear Lagrange
foreground basis $\{\phi_j\}$



$\widehat{N_i}$ for piecewise-quadratic
Lagrange foreground basis $\{\phi_j\}$

Implementation Using an Existing FE Code

Matrix notation for discrete problem we want to solve:

$$\mathbf{Kd} = \mathbf{F}\text{ , where } K_{ij} = a_h\left(\widehat{N_j}, \widehat{N_i}\right) \text{ and } F_i = L_h\left(\widehat{N_i}\right)\ .$$

What is easy to assemble over the foreground mesh with an <u>existing FE code</u>:

$$A_{ij} = a_h\left(\phi_j, \phi_i\right) \ \text{ and } \ B_i = L_h(\phi_i)\ .$$

Transformation to desired problem, using standard linear algebra routines (e.g., backend library):

$$\mathbf{K} = \mathbf{M}^T\mathbf{A}\mathbf{M} \ \text{ and } \ \mathbf{F} = \mathbf{M}^T\mathbf{B}\ \ ,$$
$$\text{where } M_{ij} = N_j(\boldsymbol{x}_i)\ .$$

Only non-standard assumption: Existing FE code can implement a_h and L_h for **<u>Nitsche's method</u>**.

Interpolation Error Bound

For background-fitted foreground meshes, and under various technical assumptions, we have the interpolation bound

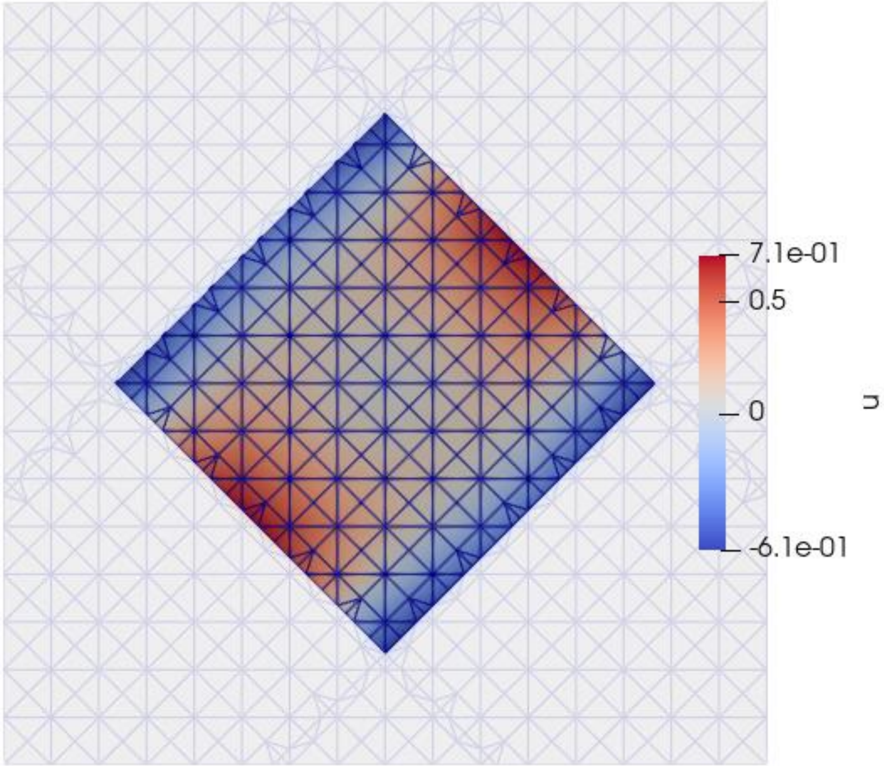
$$\inf_{v^h \in \operatorname{span}\{\widehat{N_i}\}} \left\| u - v^h \right\|_{H^1(\Omega)} \leq C h^{\hat{k}} \|u\|_{H^{\hat{k}+1}(\Omega)}$$

where $\hat{k} = \min\{k, \kappa\}$ is the minimum degree of polynomial completeness between the background (k) and foreground (κ) function spaces.

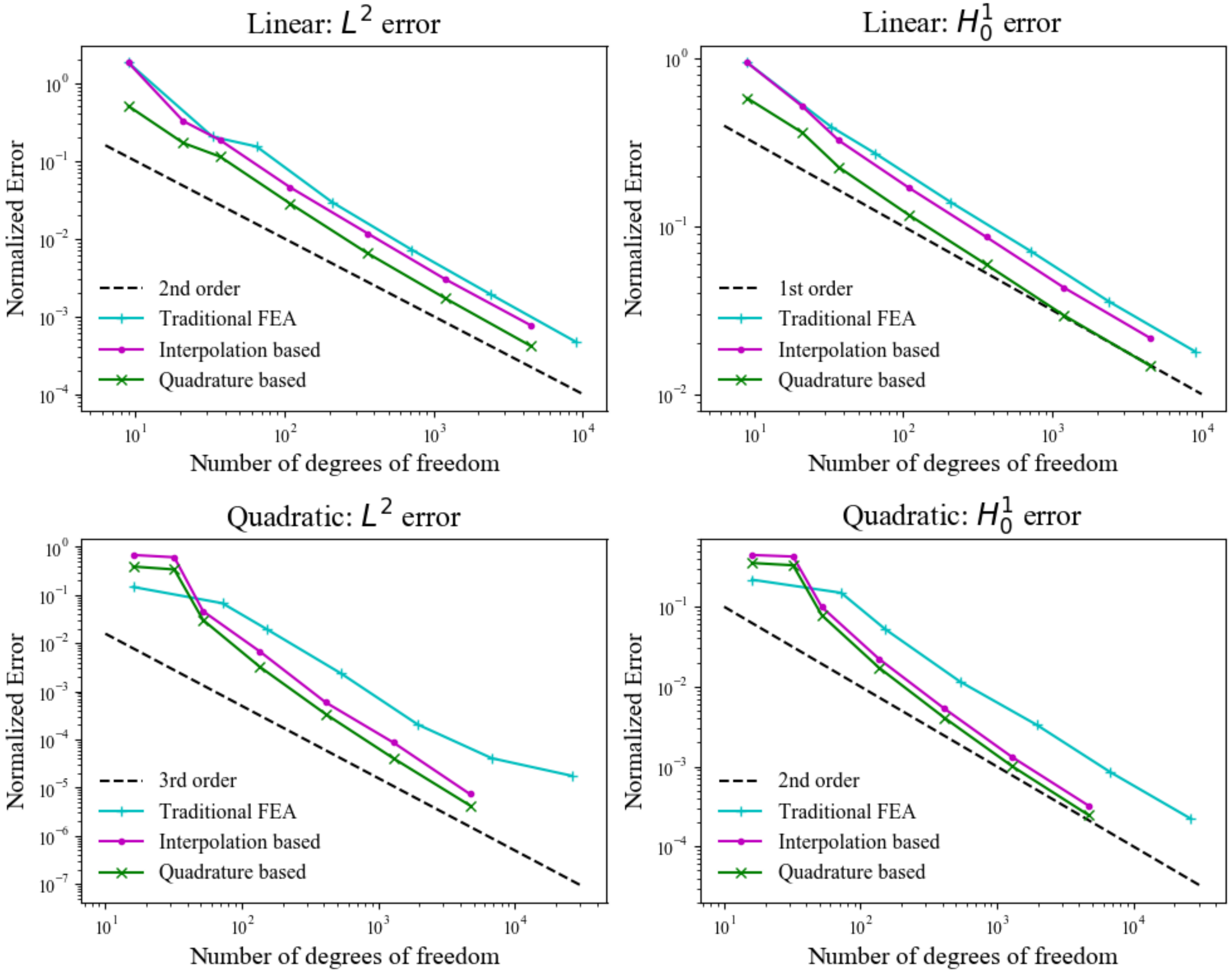
Note: Background space may have some monomials with degree $\geq k$; these do not need to be captured in foreground space for optimal-order interpolation error.

Numerical results suggest that this holds under significantly more general conditions.

Results: 2D Poisson with Background-*Fitted* Foreground Mesh

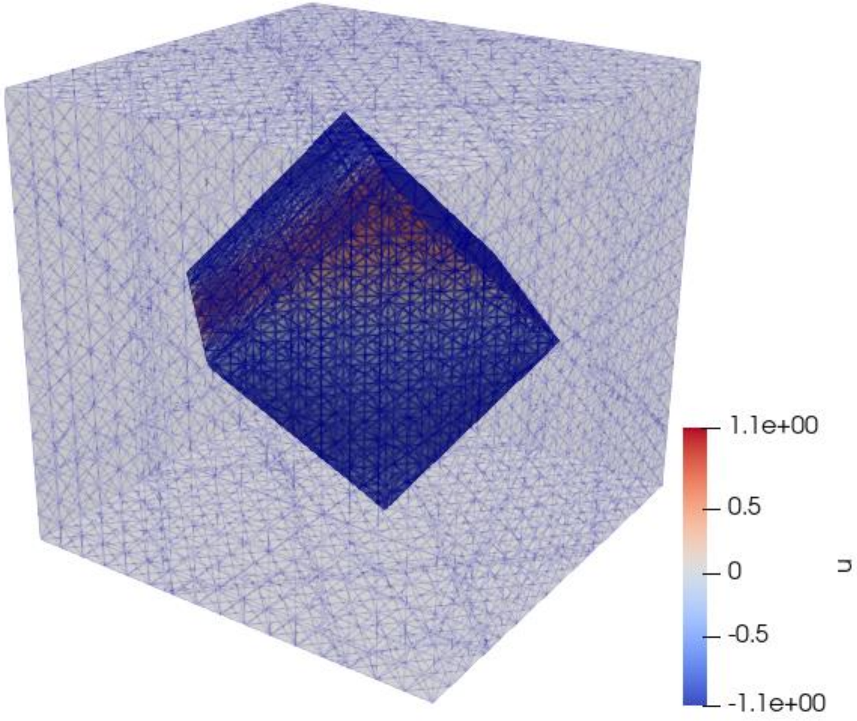


Convergence testing using a manufactured solution to Poisson's equation

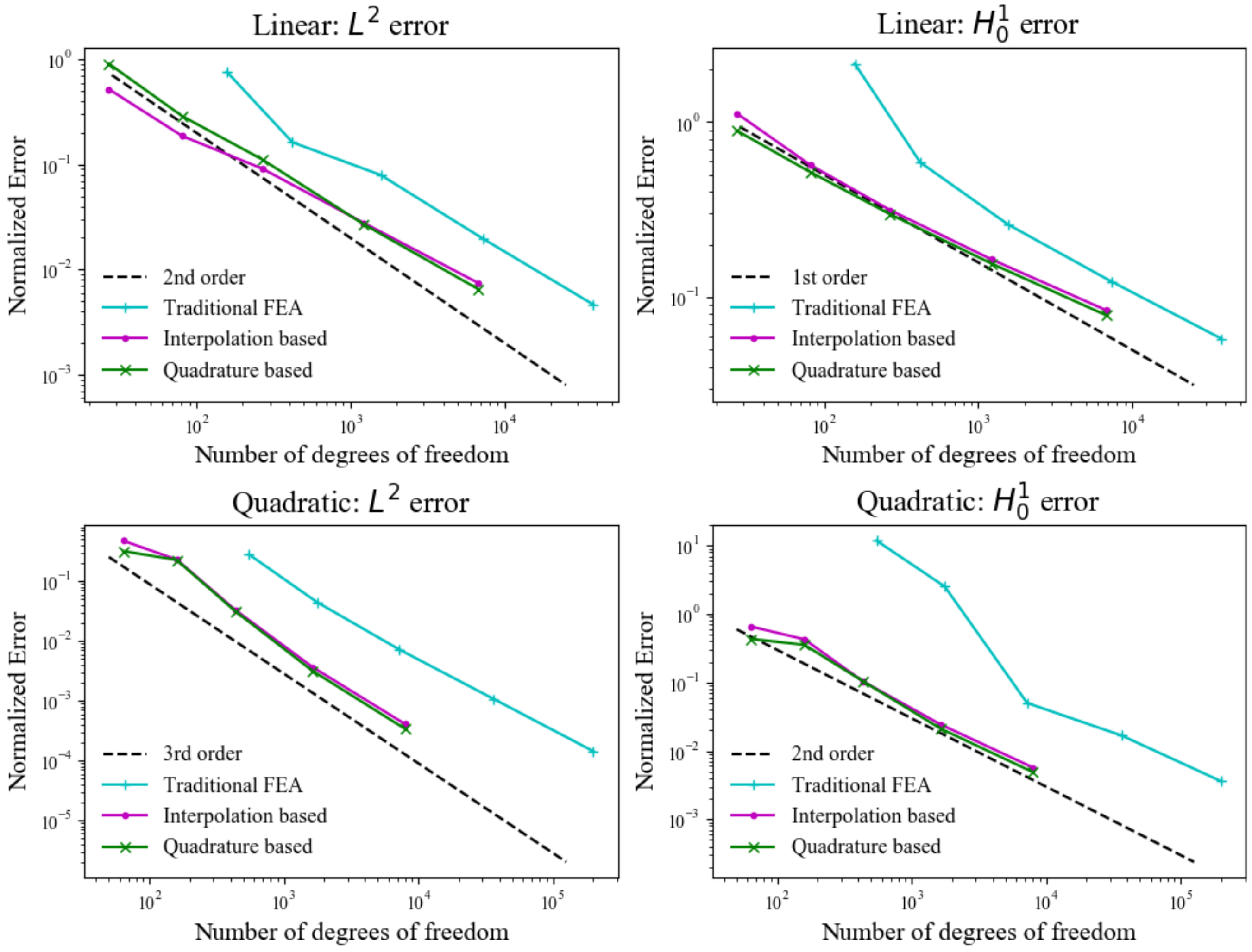


Convergence at optimal rates

Results: 3D Poisson with Background-*Fitted* Foreground Mesh

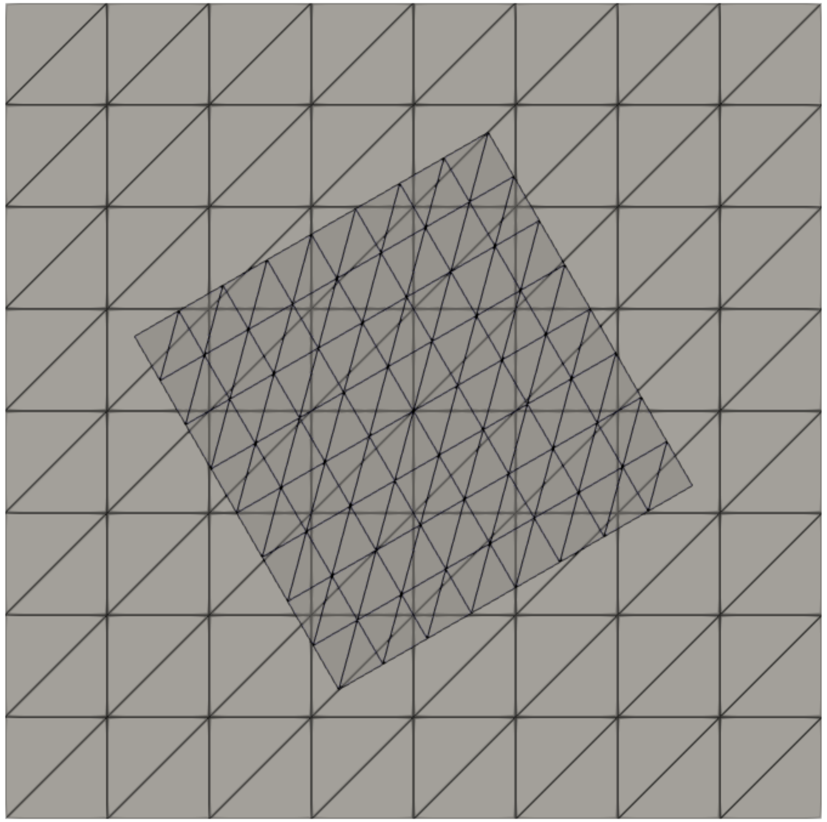


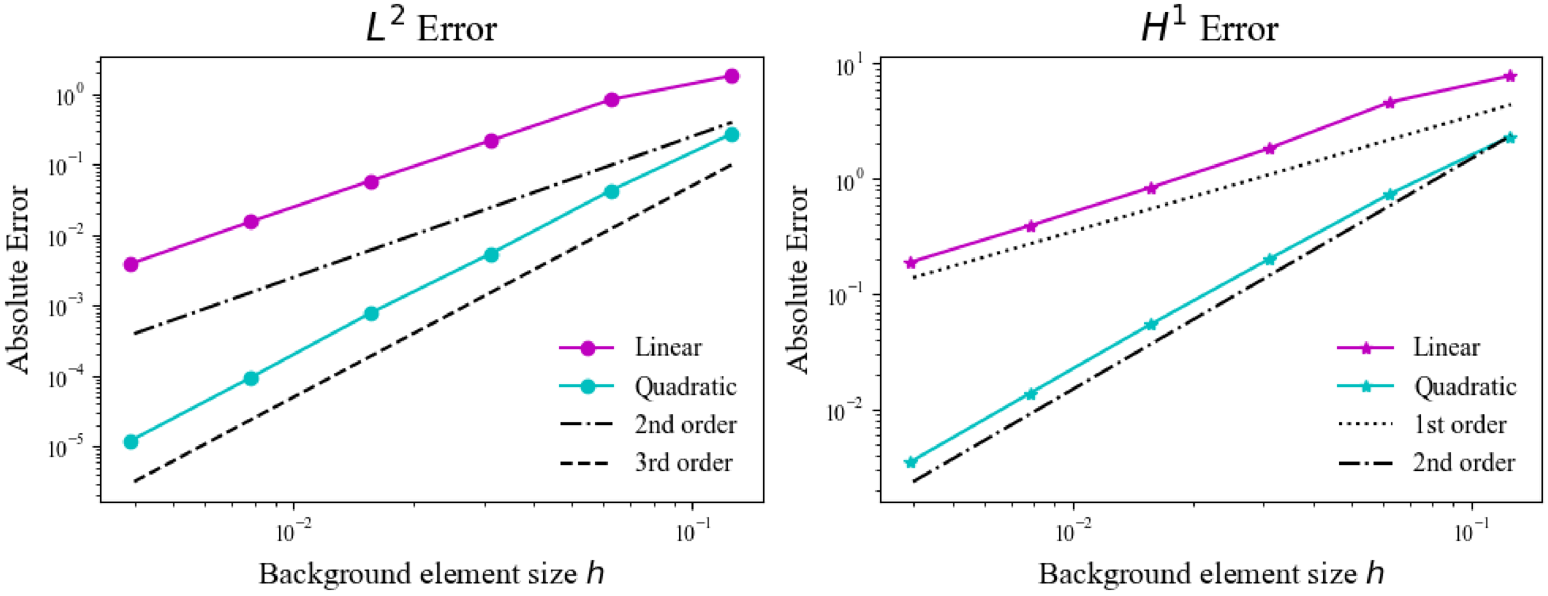
Convergence testing using a manufactured solution to Poisson's equation



Convergence at optimal rates

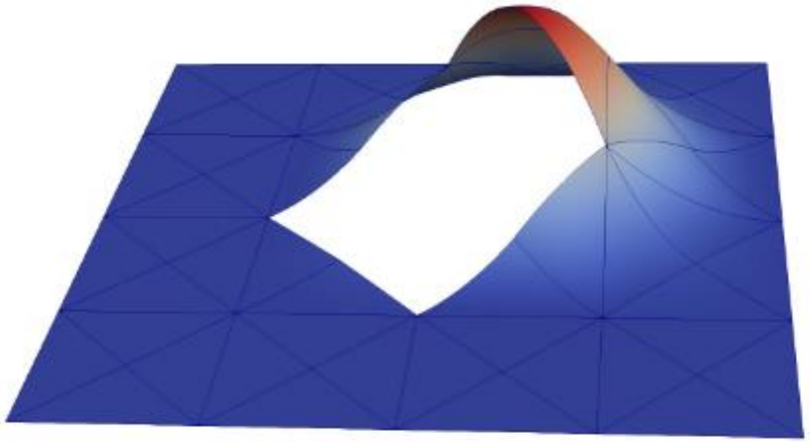
Results: 2D Poisson with Background-*Unfitted* Foreground Mesh



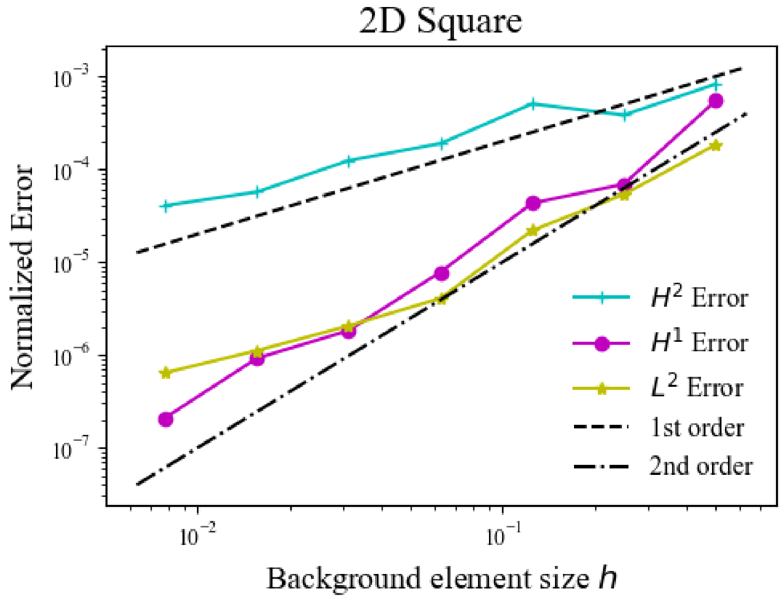


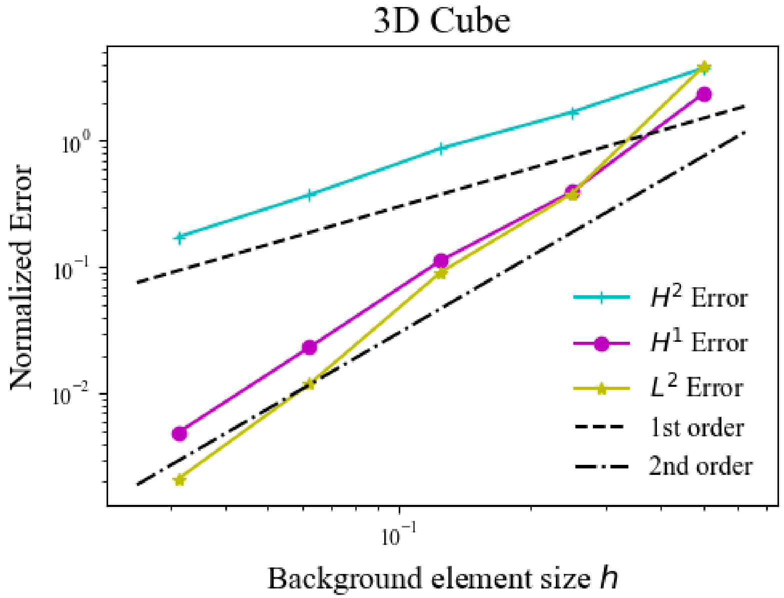
Still converging at optimal rates, but outside the scope of our initial interpolation error analysis.

Results: Biharmonic (non-conforming!)



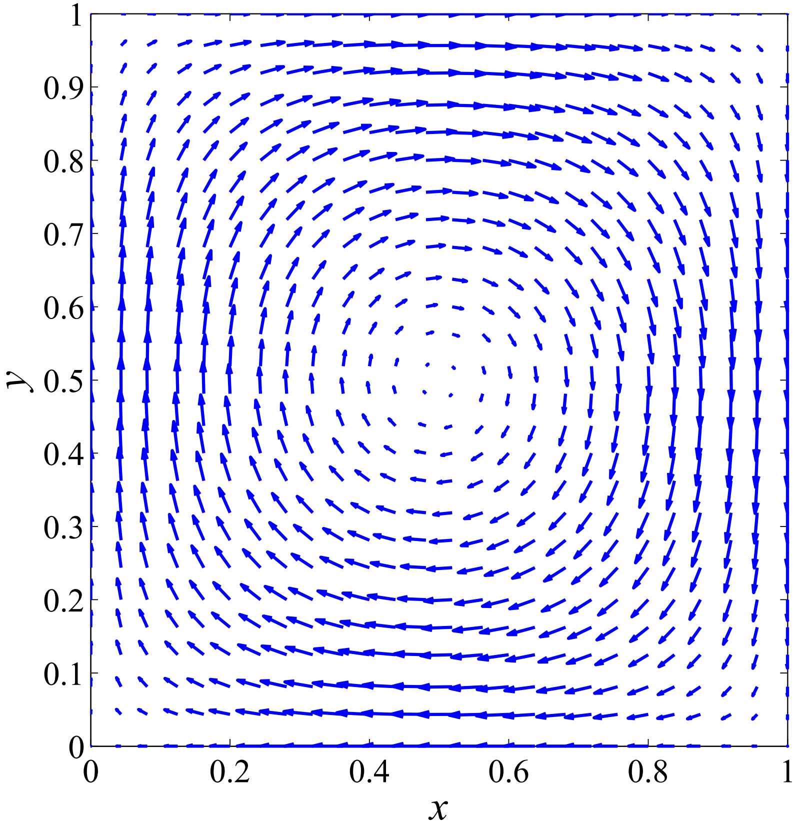
Note: Quadratic Lagrange interpolation of a C^1 quadratic B-spline function is not exactly C^1, and thus not $\in H^2(\Omega)$!





Still obtain convergence rates expected from theory for conforming formulation.

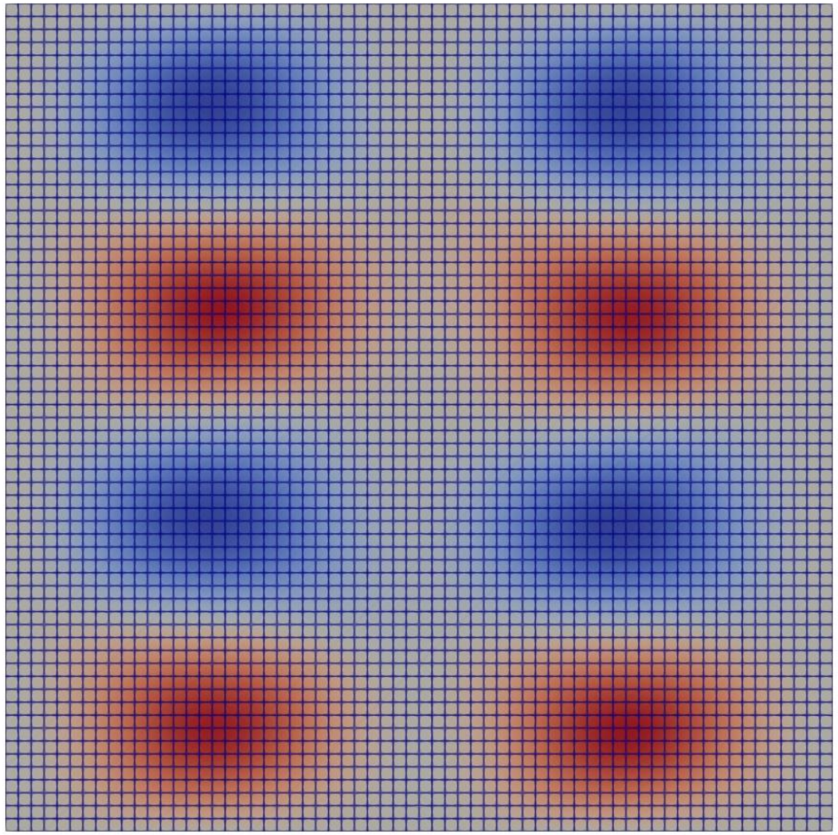
Interpolation-Based Isogeometric Analysis of Navier-Stokes Flow Using Equal-Order Elements and VMS Stabilization



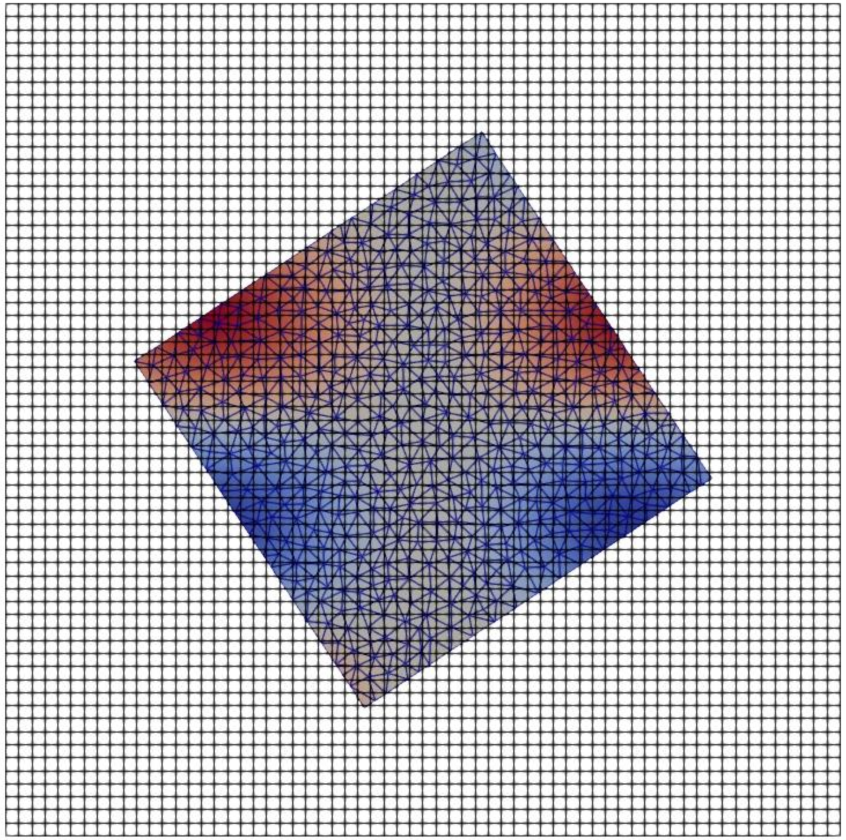
Taylor-Green Vortex at *Re* = 100

$$
\begin{aligned}
&\int_{\Omega} \left(\rho\left(\partial_t \boldsymbol{u}^h + \boldsymbol{u}^h \cdot \nabla \boldsymbol{u}^h\right) \cdot \boldsymbol{v}^h + \boldsymbol{\sigma}(\boldsymbol{u}^h, p^h) : \nabla \boldsymbol{v}^h + \nabla \cdot \boldsymbol{u}^h q^h\right) d\Omega \\
&+ \int_{\partial\Omega} \left(-(\boldsymbol{\sigma}(\boldsymbol{u}^h, p^h)\boldsymbol{n}) \cdot \boldsymbol{v}^h + (\boldsymbol{\sigma}(\boldsymbol{v}^h, q^h) \cdot \boldsymbol{n}) \cdot (\boldsymbol{u}^h - \boldsymbol{g})\right) d\Gamma \\
&- \int_{\partial\Omega} \left(\rho \min\{\boldsymbol{u}^h \cdot \boldsymbol{n}, 0\}(\boldsymbol{u}^h - \boldsymbol{g}) \cdot \boldsymbol{v}^h\right) d\Gamma \\
&+ \int_{\Omega} \tau_{\mathrm{M}} \left(\left(\boldsymbol{u}^h \cdot \nabla \boldsymbol{v}^h + \frac{1}{\rho}\nabla q^h\right) \cdot \boldsymbol{r}_{\mathrm{M}} + \boldsymbol{v}^h \cdot \boldsymbol{r}_{\mathrm{M}} \cdot \nabla \boldsymbol{u}^h\right) d\Omega \\
&- \int_{\Omega} \frac{\tau_{\mathrm{M}}^2}{\rho} (\nabla \boldsymbol{v}^h) : (\boldsymbol{r}_{\mathrm{M}} \otimes \boldsymbol{r}_{\mathrm{M}})\, d\Omega \\
&+ \int_{\Omega} \tau_{\mathrm{C}} \left(r_{\mathrm{C}} \nabla \cdot \boldsymbol{v}^h\right) d\Omega = 0
\end{aligned}
$$

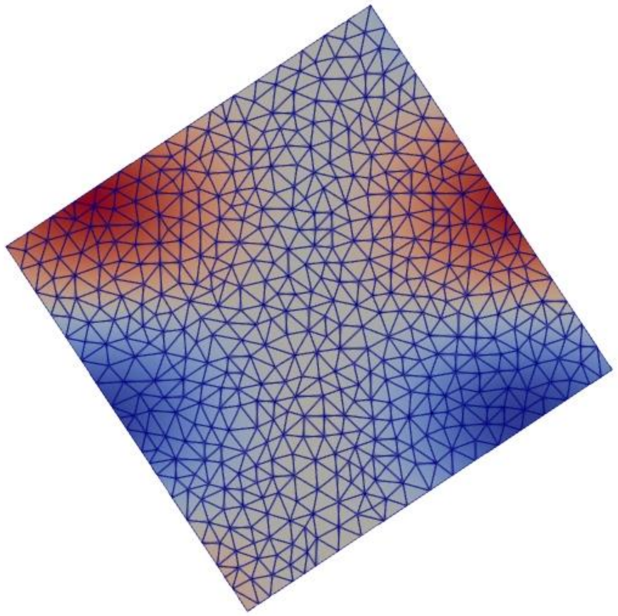
Interpolation-Based Isogeometric Analysis of Navier-Stokes Flow Using Equal-Order Elements and VMS Stabilization



BG Mesh with Exact Solution (Tensor Product Spline Space)



BG Mesh with Overlaying FG Mesh



FG Mesh

Interpolation-Based Isogeometric Analysis of Navier-Stokes Flow Using Equal-Order Elements and VMS Stabilization

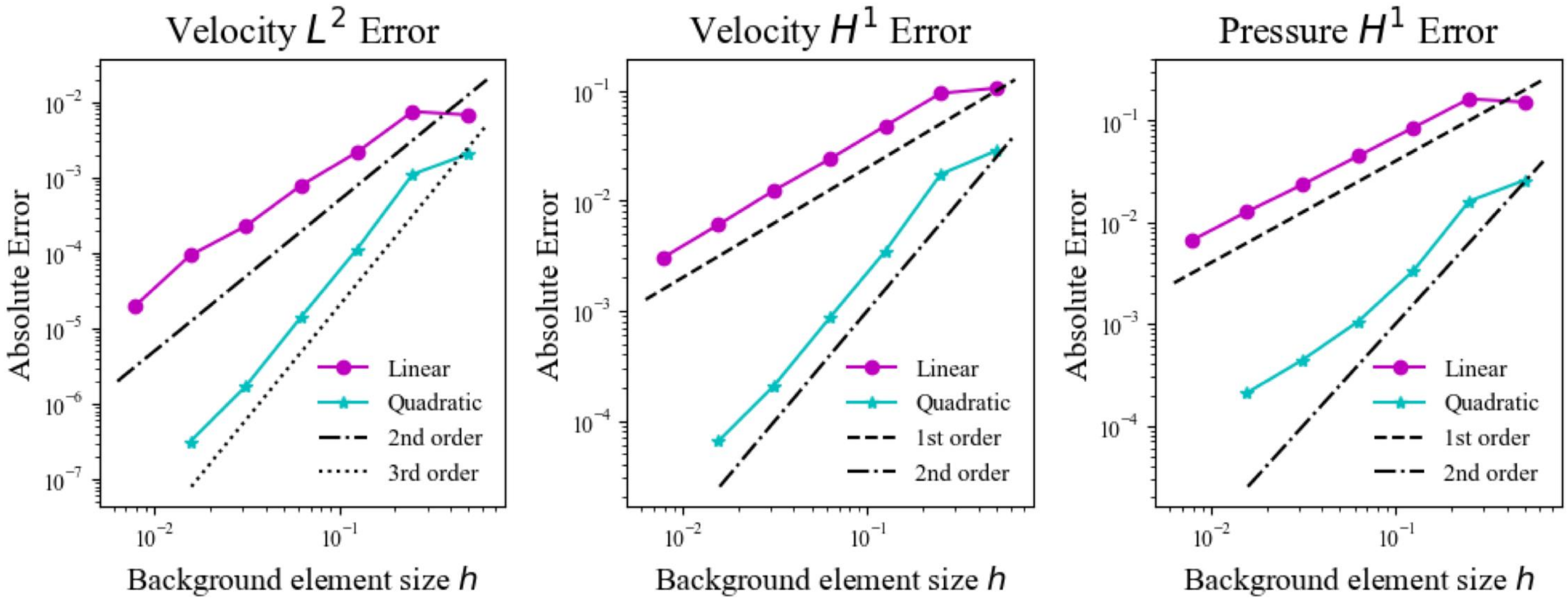


Image-Based Analysis for Composite Materials

Linear elastic analysis of alumina particles embedded in epoxy from Micro-CT image

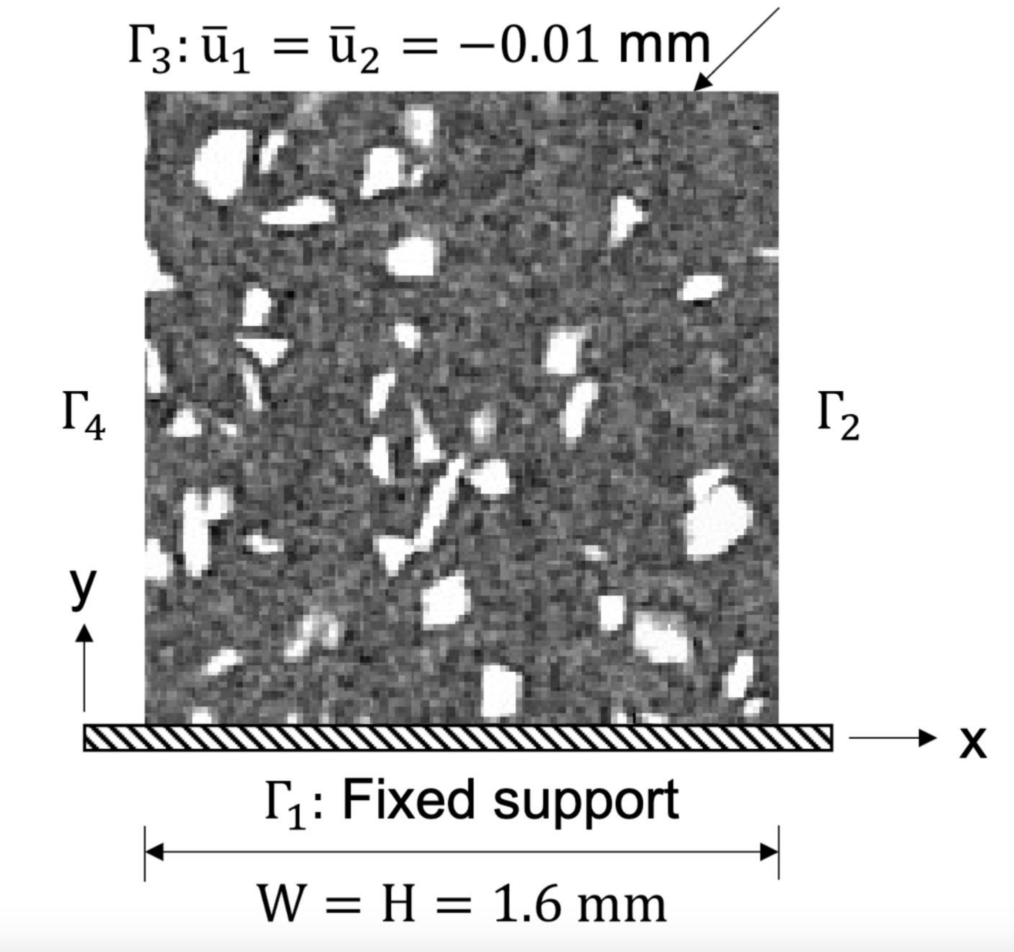
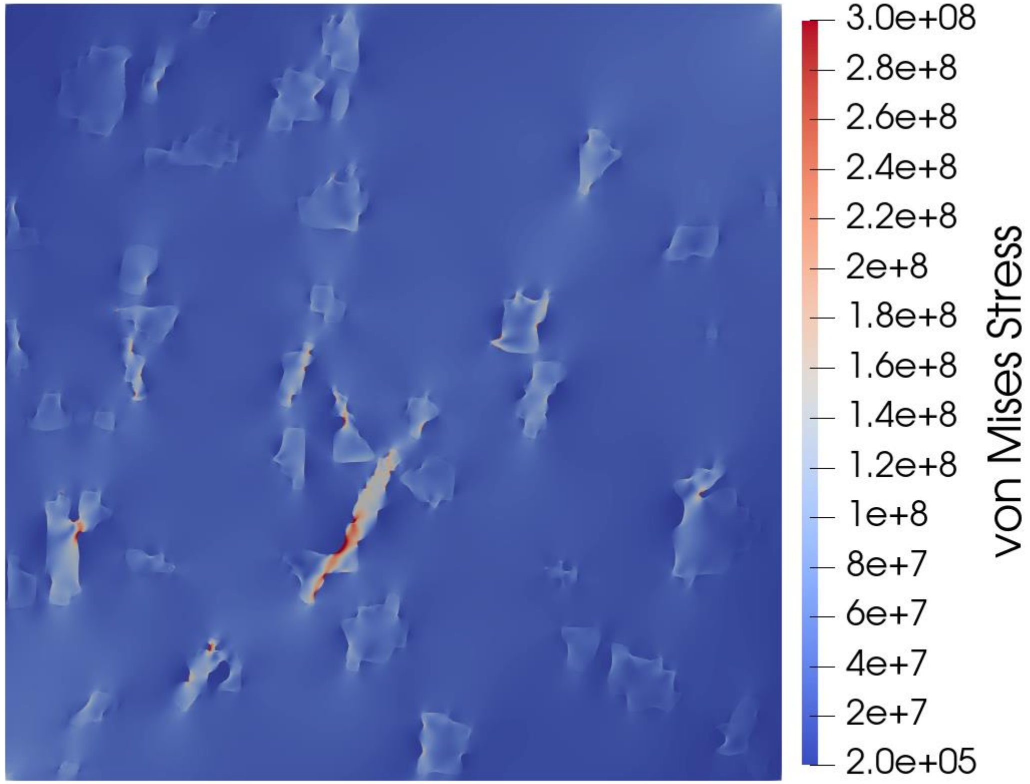
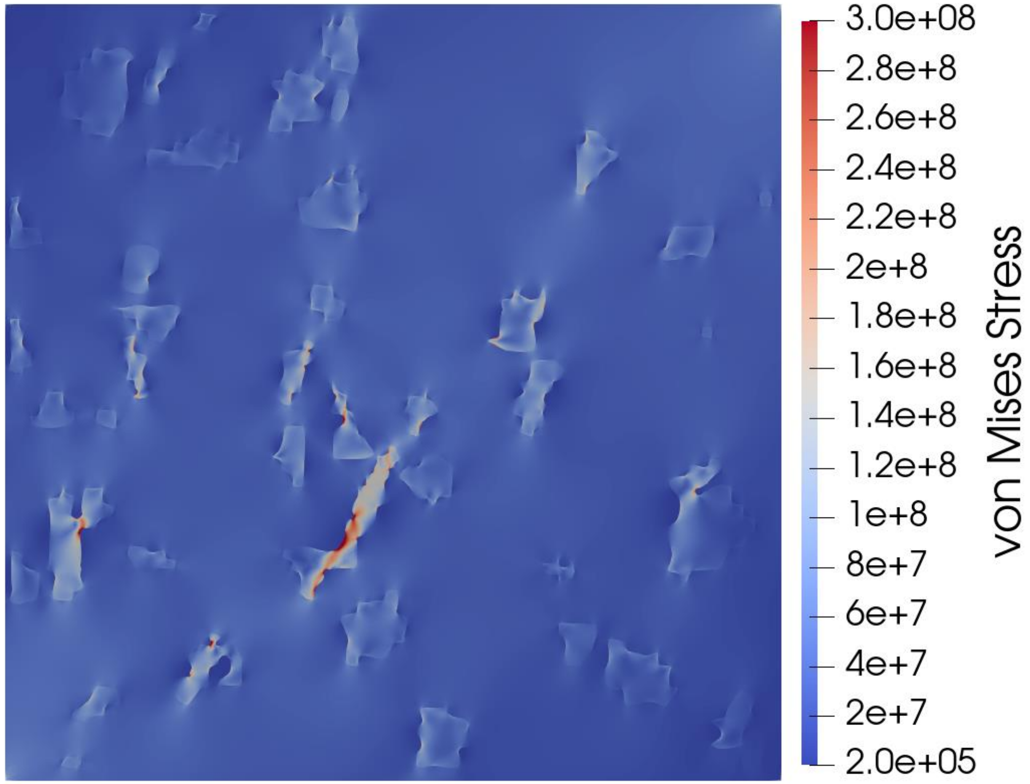


Image-Based Analysis for Composite Materials

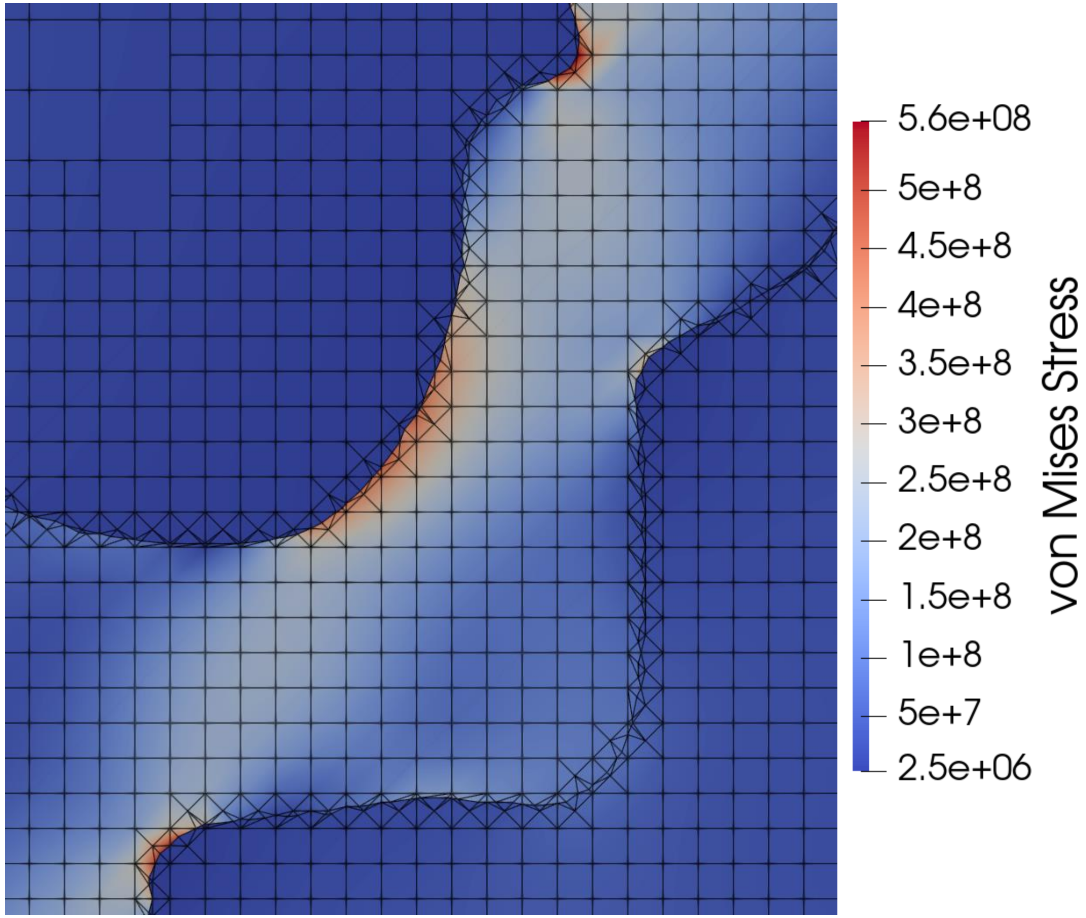


Immersed FEA

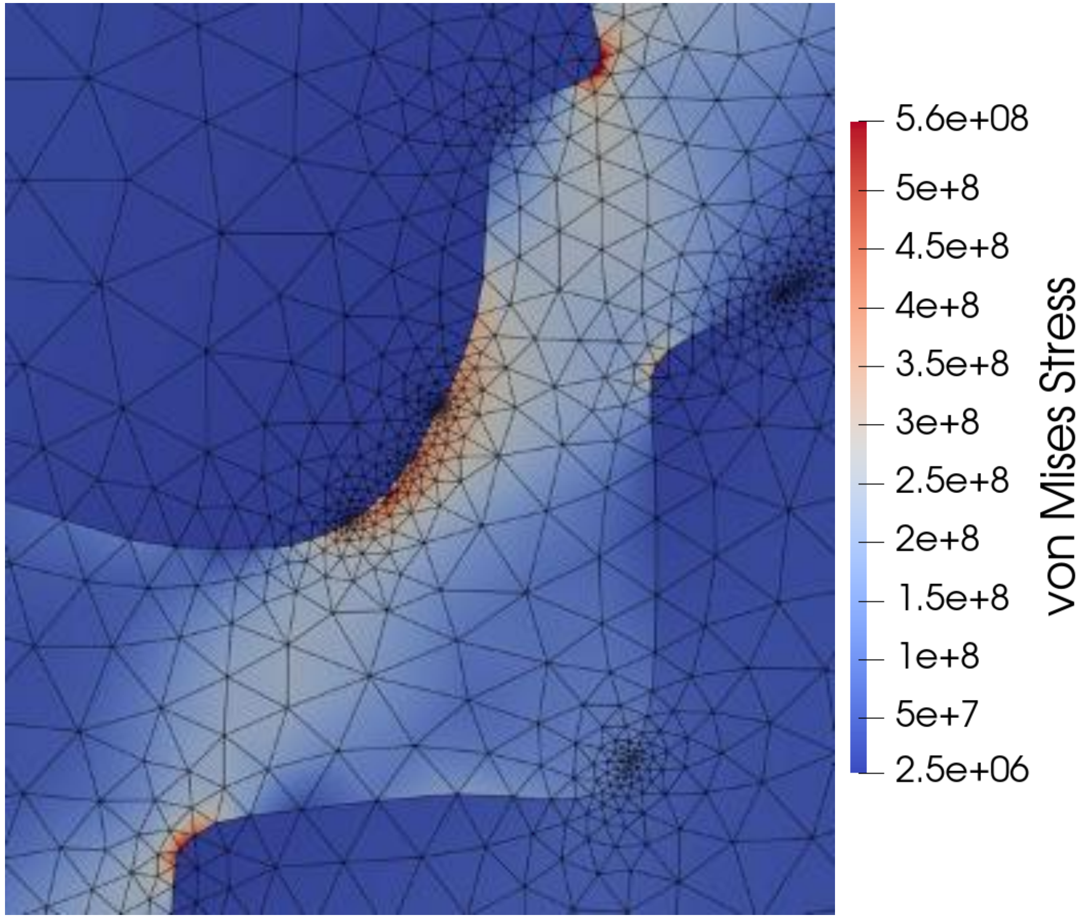


Body-Fitted FEA

Image-Based Analysis for Composite Materials



Immersed FEA



Body-Fitted FEA

Application to Thermo-Elasticity: Heating of Plate with Inclusion

Subdomain A:

$E = 1.0$

$\nu = 0.3$

$\alpha = 1.0 \times 10^{-5}$

Subdomain B:

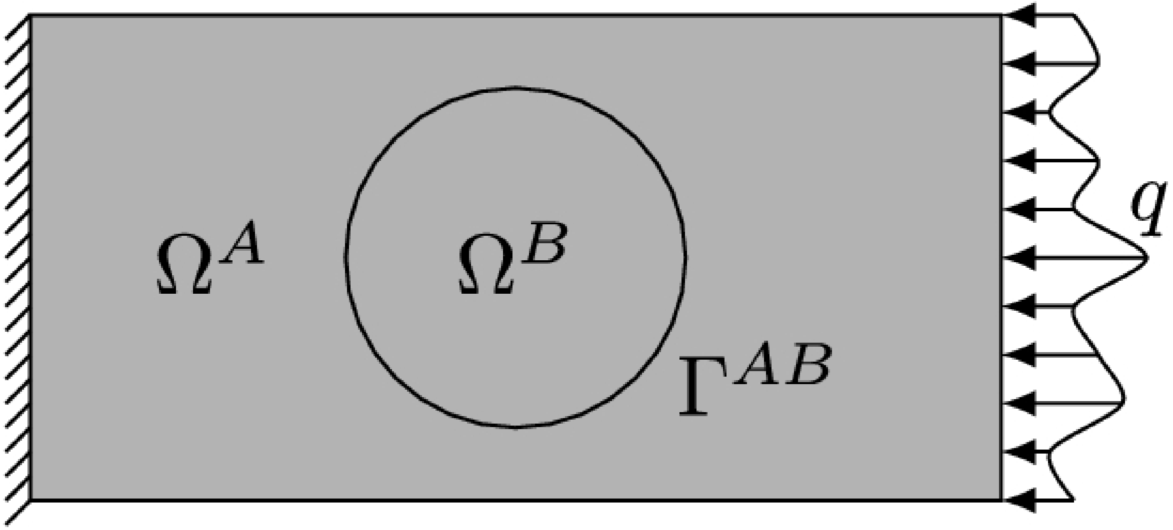
$E = 1.0$

$\nu = 0.3$

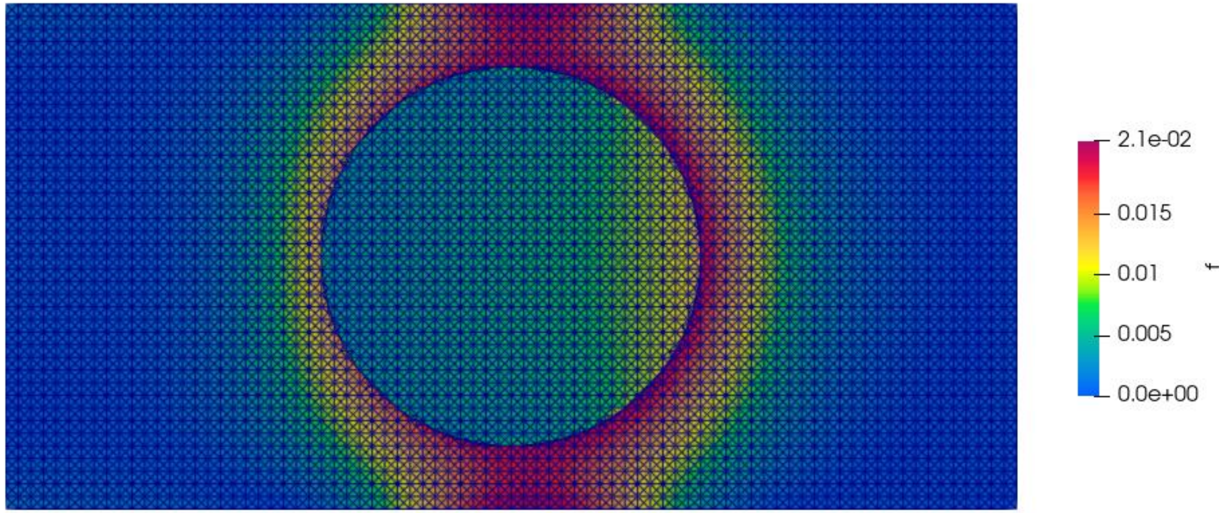
$\alpha = 10.0 \times 10^{-5}$

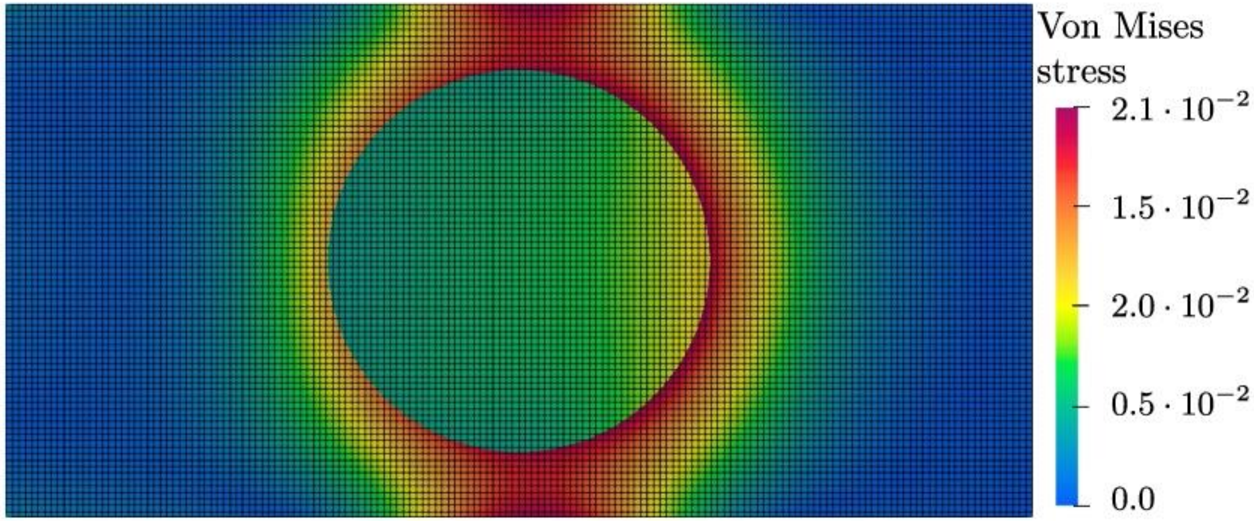
$\boldsymbol{u} = \boldsymbol{0}$

$T = 0$



Application to Thermo-Elasticity: Heating of Plate with Inclusion

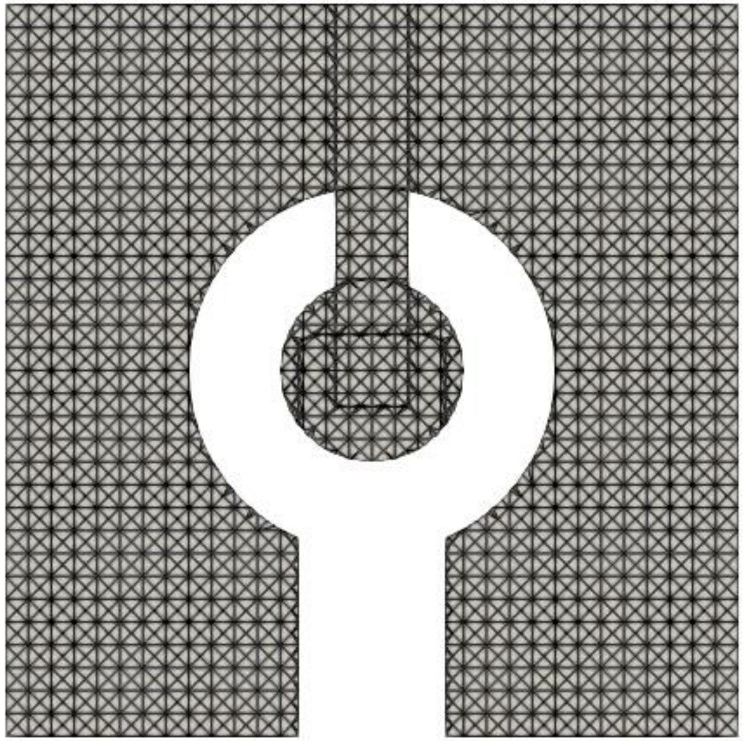




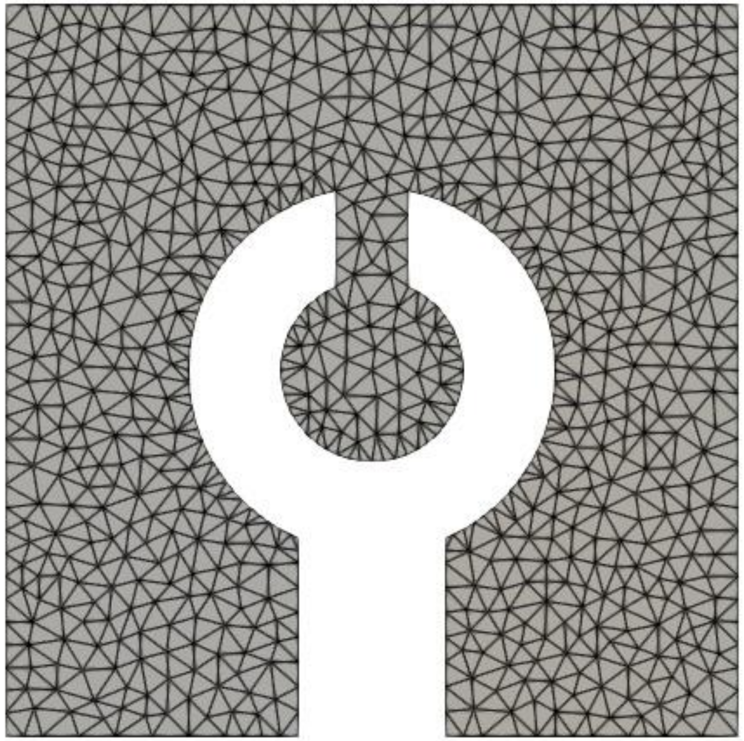
Interpolation-Based Immersed Isogeometric Analysis $k_1 = k_1 = 2$

Quadrature-Based Immersed Isogeometric Analysis $k_1 = k_1 = 2$

Interpolation-Based Trimmed Isogeometric Shell Analysis



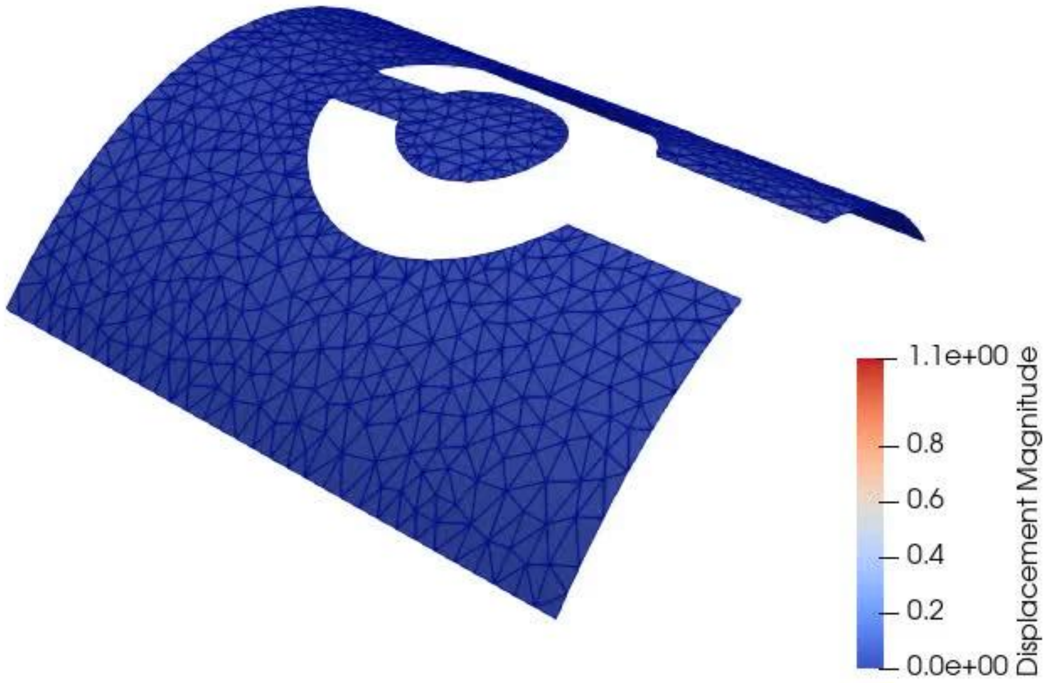
Background-fitted foreground mesh



Background-unfitted foreground mesh

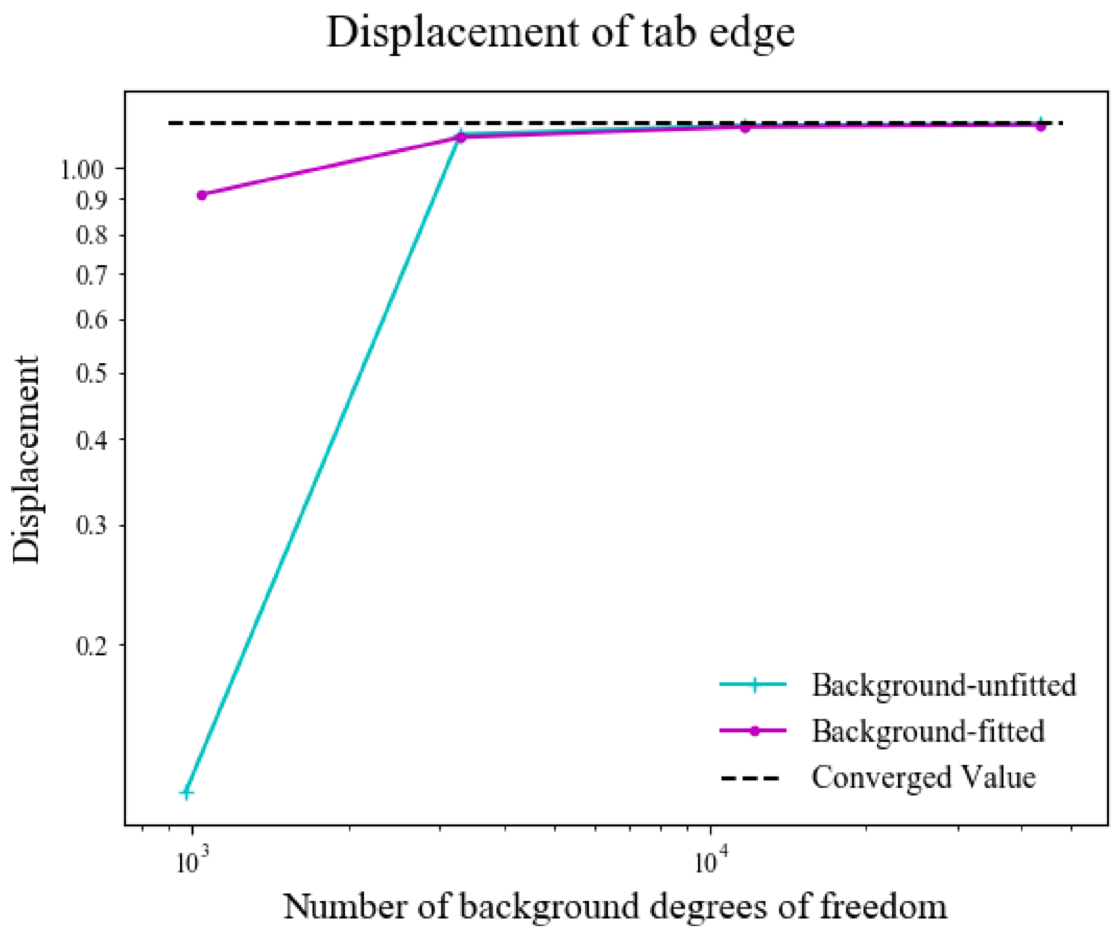
Interpolation-Based Trimmed Isogeometric Shell Analysis

Background-fitted foreground mesh



Background-unfitted foreground mesh

Interpolation-Based Trimmed Isogeometric Shell Analysis

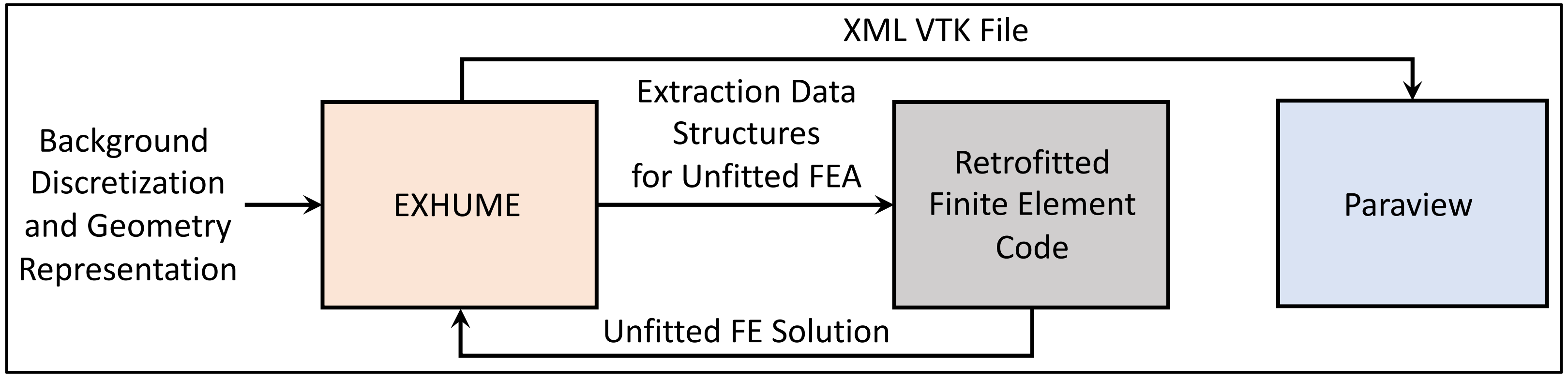


Part 3:

Transforming Classical Codes Into Immersed Codes with EXHUME

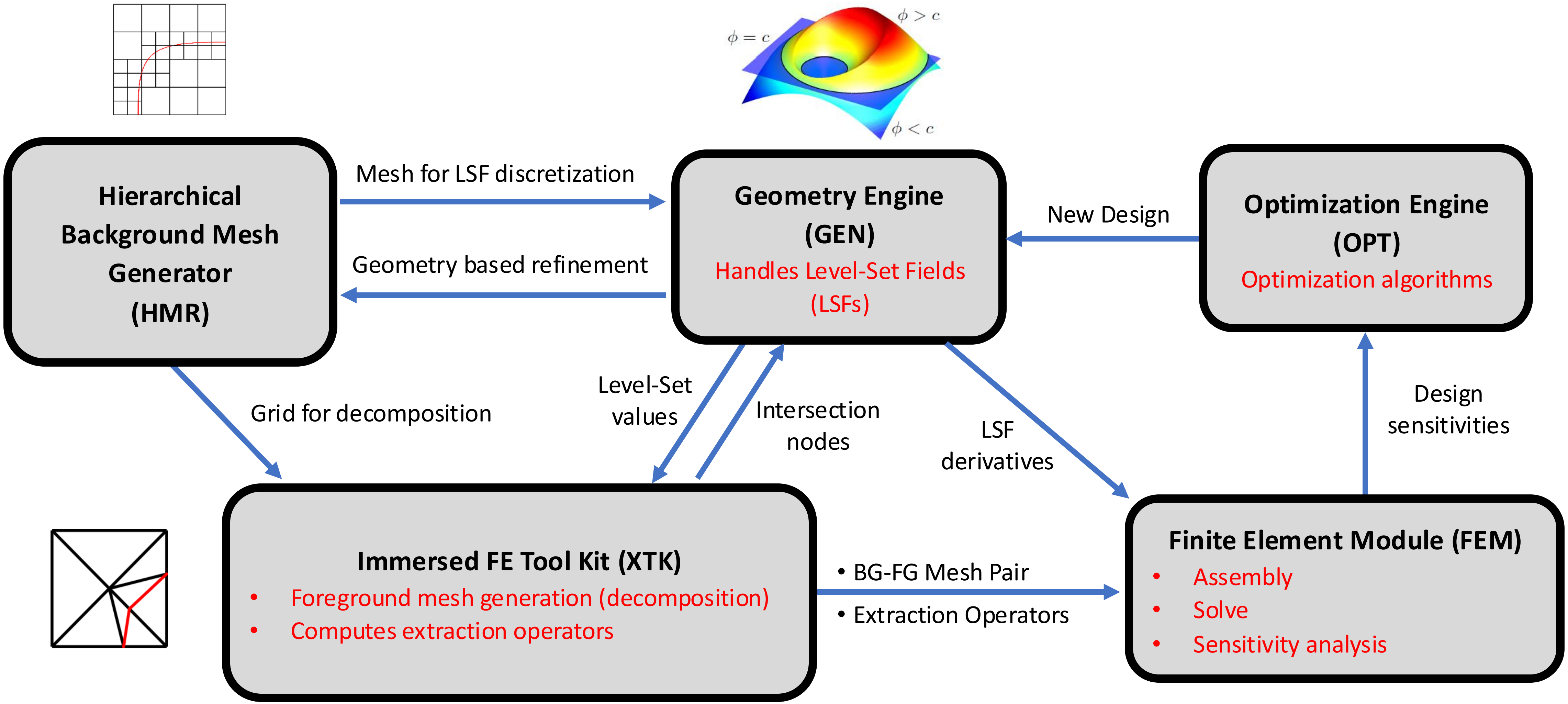
Retrofitting Classical Finite Element Codes for Interpolation-Based Immersed Analysis

Our software library EXHUME creates foreground meshes, extraction operators, and connectivity arrays outside of the confines of any particular finite element code:

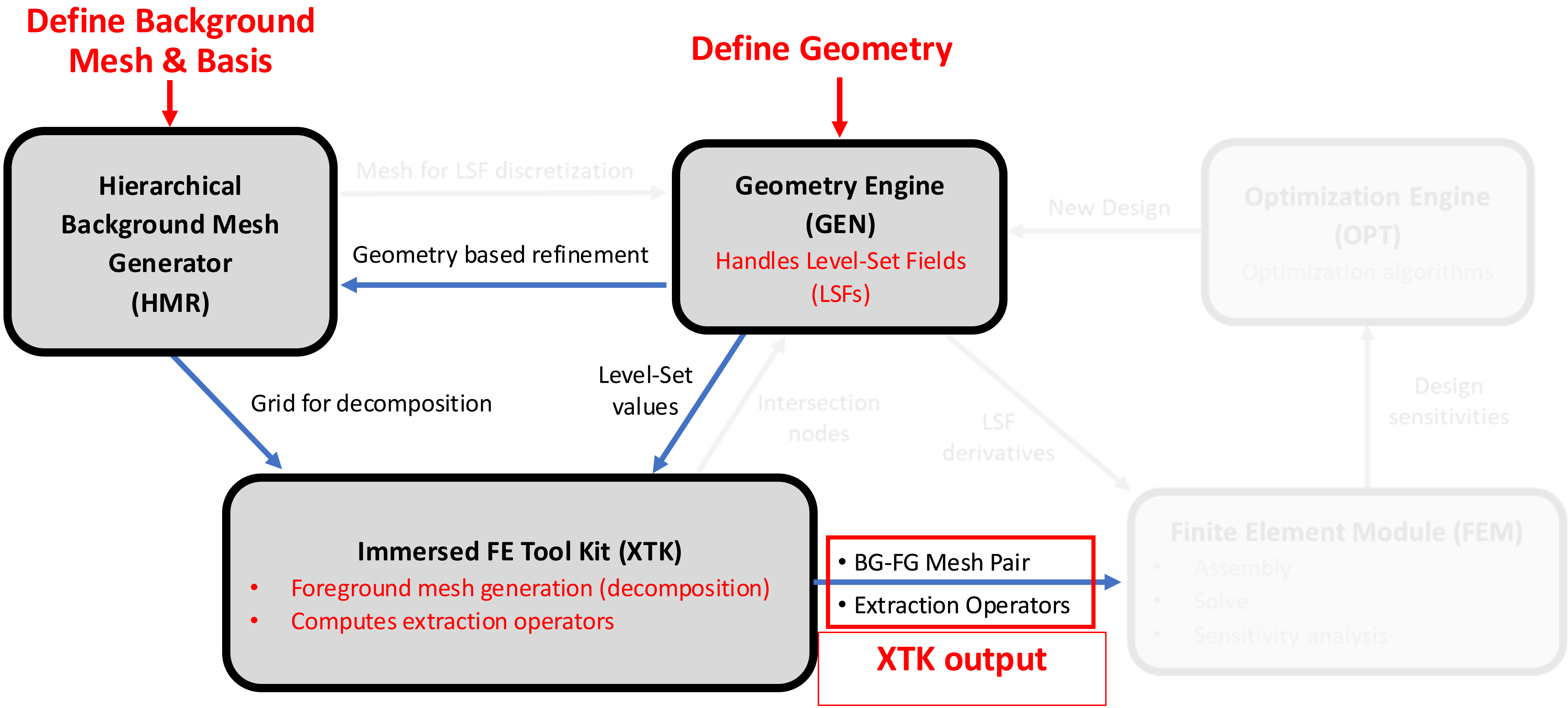


As demonstrated later, we have leveraged these extraction data structures to transform the popular open-source classical finite element code FEniCS into an interpolation-based immersed finite element code.

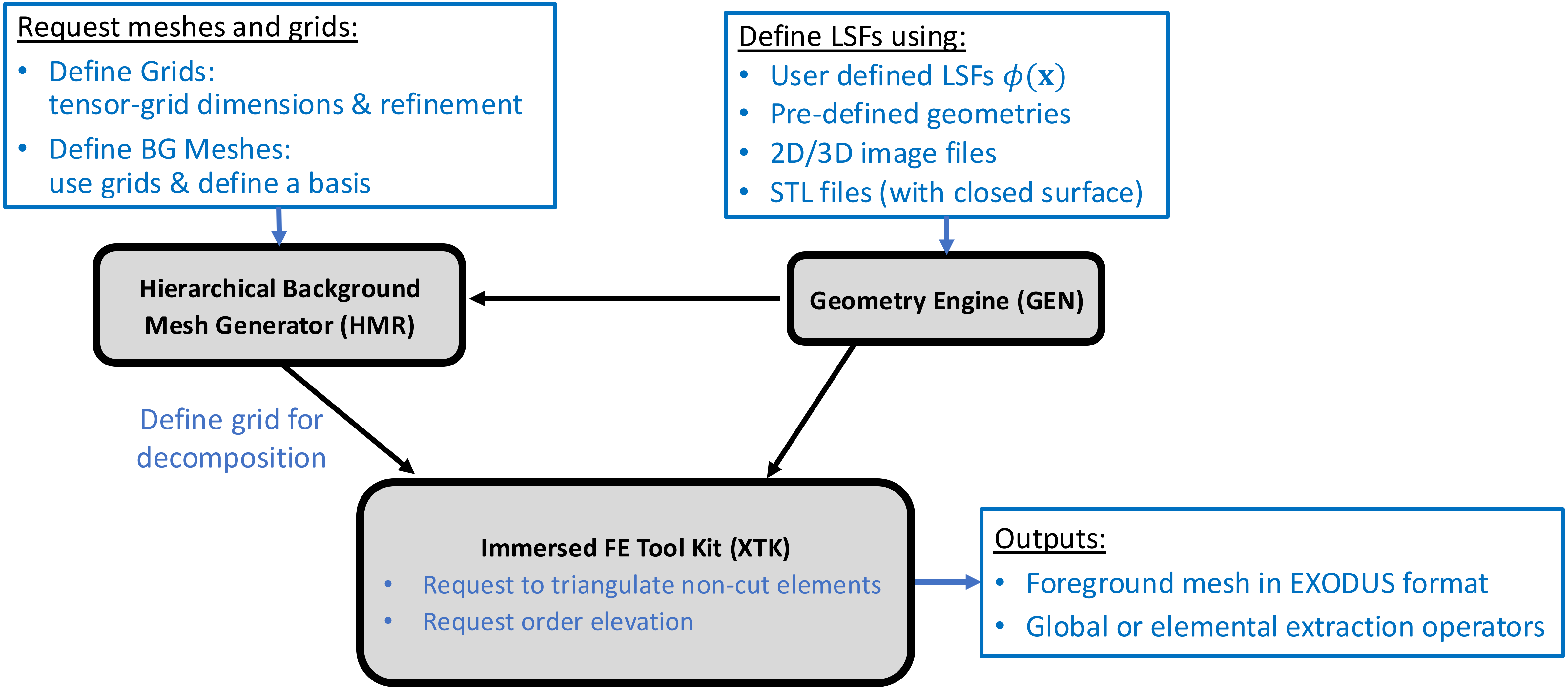
The MORIS Software Library for XFEM-Based Level Set Topology Optimization



The MORIS Software Library for XFEM-Based Level Set Topology Optimization



The EXHUME Software Library



Part 4:

Enabling Interpolation-Based Immersed FEA in FEniCS

Leveraging Code Generation: The FEniCS Project

Variational forms

$a(u,v) =$
$\int \nabla u \cdot \nabla v \, dx$

→

UFL (Python)

*a = dot(grad(u), grad(v))*dx*

FFC →

Optimized C++ code

…….

DOLFIN ↓

System of algebraic equations

$K_{ij} = a(\widehat{N}_j, \widehat{N}_i)$

- FEniCS is an opensource software toolchain (UFL, FFC, DOLFIN,…) that automates the compilation of FE code
- Assembles a system of algebraic equations from the variational forms and discretizations, for numerous finite elements

Interpolation-Based Immersed Finite Element Analysis Workflow

Generate foreground boundary-fitted mesh, and its extraction operator $M_{ij} = N_i(\boldsymbol{x}_j)$

Assemble $A_{ij} = a(\phi_j, \phi_i)$ and $B_i = L(\phi_i)$ using foreground basis ϕ_i

Compute $\mathbf{K} = \mathbf{M}^{\mathrm{T}}\mathbf{A}\mathbf{M}$ and $\mathbf{F} = \mathbf{M}^{\mathrm{T}}\mathbf{B}$

Solve $\mathbf{K}\mathbf{d} = \mathbf{F}$

Project solution onto foreground basis for visualization and analysis: $\mathbf{u} = \mathbf{M}^{\mathrm{T}}\mathbf{d}$

FEniCS: Solving the Poisson Problem with Classical FEA

Load mesh and define subdomains

```
from dolfin import *

#load mesh, and define geometry subdomains
mesh_f = Mesh()
file = XDMFFile(mesh_f.mpi_comm(),'mesh.xdmf')
file.read(mesh_f)
sub_domains=MeshFunction('size_t',mesh_f,mesh_f.geometry().dim())
file.read(sub_domains,'material')
outside_ID = 1; block_ID = 2; surf_ID = 3
sub_domains_surf = MeshFunction("size_t",mesh_f,mesh_f.geometry().dim()-1)
for facet in facets(mesh_f):…
dx_custom = Measure('dx', subdomain_data=sub_domains,subdomain_id=block_ID, metadata={'quadrature_degree': 2*k})
ds_custom = Measure('dS', subdomain_data=sub_domains_surf,subdomain_id=surf_ID, metadata={'quadrature_degree': 2*k})
```

Define classic FE function space

```
#Define a foreground function space on the mesh, with continuous Galerkin elements, with polynomial order k = 2
k = 2
V_f = FunctionSpace(mesh_f, 'CG', k)
u_f = Function(V_f)
v_f = TestFunction(V_f)
int_constant = Constant(0.0)*v_f*dx(domain=mesh_f, subdomain_data=sub_domains)
```

Define variational problem

```
# define the exact solution, to be used with the method of manufactured solutions
x = SpatialCoordinate(mesh)
u_ex = sin(x[0])*cos(x[1])
f = -div(grad(u_ex))
a = inner(grad(u_f),grad(v_f))*dx_custom
L = inner(f,v_f)*dx_custom
```

FEniCS: Solving the Poisson Problem with Classical FEA

Enforce boundary conditions

Linear solve

```
#solve with classic FE

# strongly enforce boundary conditions
def boundary(x,on_boundary):
    return on_boundary
bc = DirichletBC(V_f, u_ex, sub_domains_surf, surf_ID)

#Compute the solution on the foreground mesh
u = Function(V_f)
solve(a==L, u, bc)
```

FEniCS: Solving the Poisson Problem with Interpolation-Based Immersed FEA

Load EXHUME module

```
# solve using the interpolation-based immersed boundary method
# import EXHUME module
from EXHUME.common import * ;
from EXHUME.la_utils import * ;
```

Enforce boundary conditions with Nitsche's method

```
# weakly enforce boundary conditions using Nitsche's method
n = FacetNormal(mesh_f)
h_E = CellDiameter(mesh_f)("+")
beta = Constant(10) # user specified penalty parameter
nitsche_form = -inner(dot(grad(u_f("+")), n("+")), v_f("+"))*ds_custom + int_constant
nitsche_consistency = inner(u_ex("+")-u_f("+"), dot(grad(v_f("+")), n("+")))*ds_custom + int_constant
nitsche_penalty = beta*h_E**(-1)*inner(u_f("+")-u_ex("+"), v_f("+"))*ds_custom + int_constant

# define the residual and it's jacobian
res = a - L + nitsche_form + nitsche_consistency + nitsche_penalty
J_f = derivative(res, u_f)
res_f_P = as_backend_type(assemble(res)).vec() # convert to PETSc vector
J_f_P = as_backend_type(assemble(J_f)).mat() # convert to PETSc matrix
```

Transform matrix system

```
# Read in Extraction Operator, and interpolate
local_size = v2p(assemble(res)).getLocalSize() #determine local matrix size
M = readFromCSV_Notebook(['ExOp_Cons.csv'],V_f,mesh_f,local_size,nodeFileNames='cell_nodes.csv',k=k);
M.assemble()
J_b = AT_R_A(M,J_f_P)
res_b = AT_x(M,res_f_P)
u_b = J_b.createVecLeft()
```

Linear solve

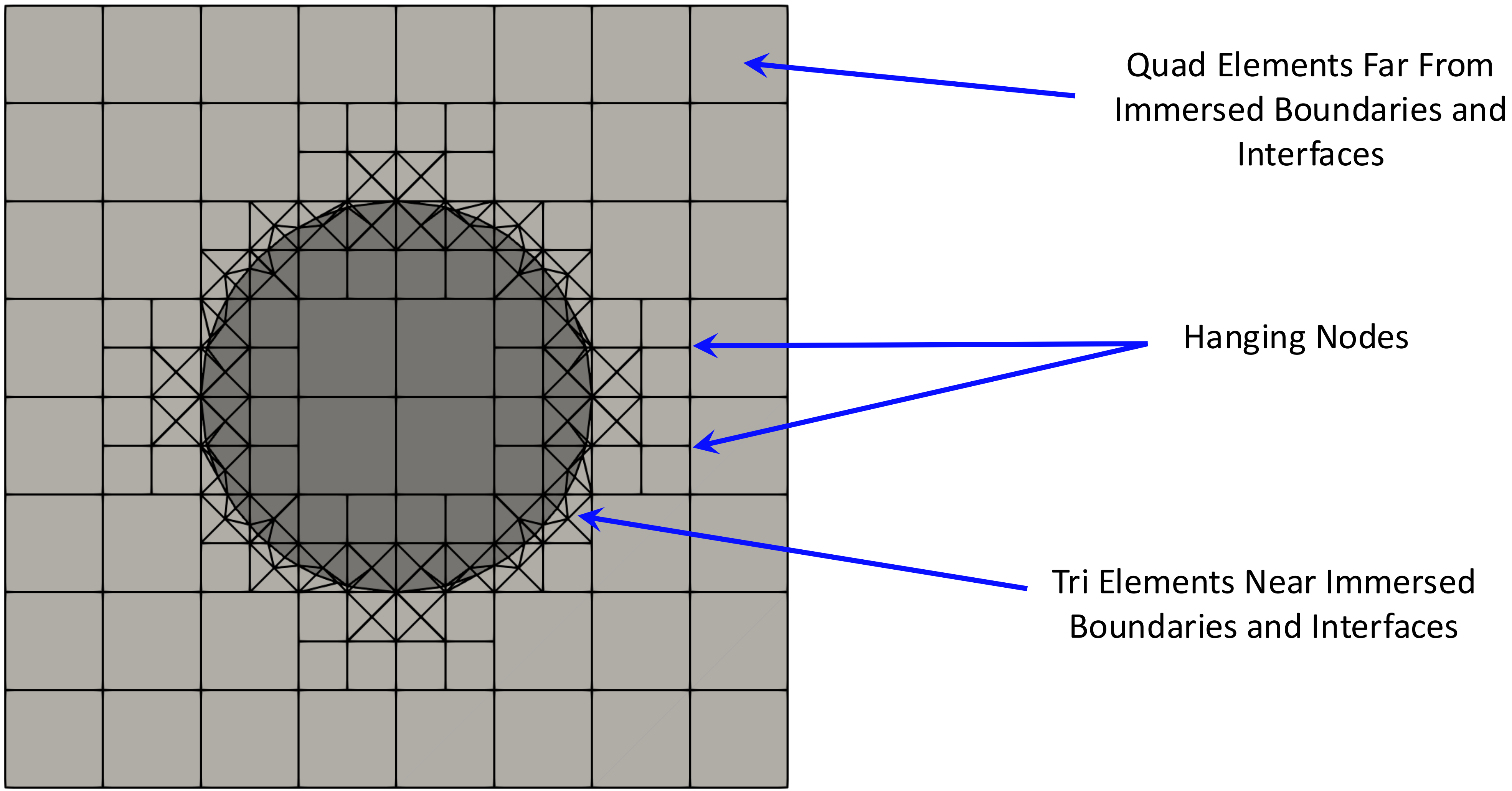
```
solveKSP(J_b,-res_b,u_b, method='gmres', PC='jacobi',monitor=False)

# project onto the foreground basis for analysis
u_f_p = as_backend_type(u_f.vector()).vec()
M.mult(u_b,u_f_p)
updateU(u_f)
```

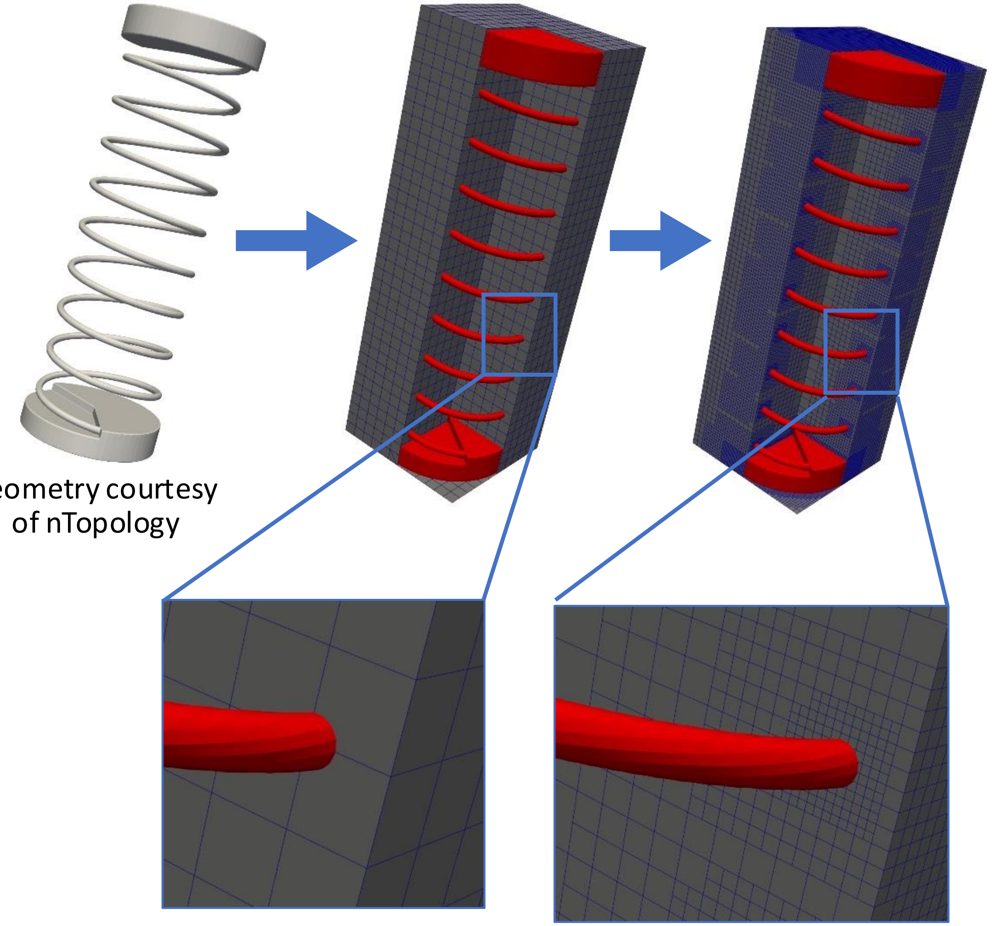
Part 5:

Current Status and Future Plans

Capability #1: Support for Locally Refined Mixed Foreground Meshes of Tris/Quads (in 2D) and Tets/Hexes (in 3D) with Hanging Nodes

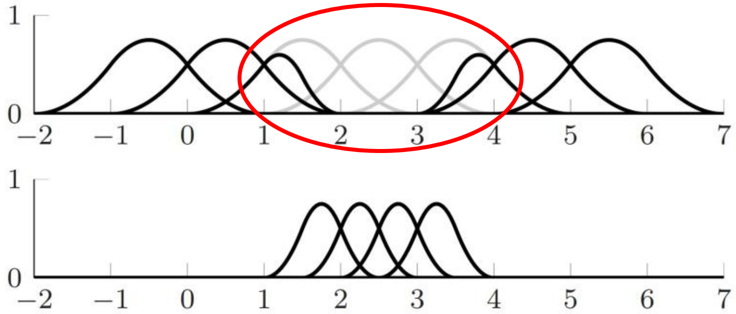


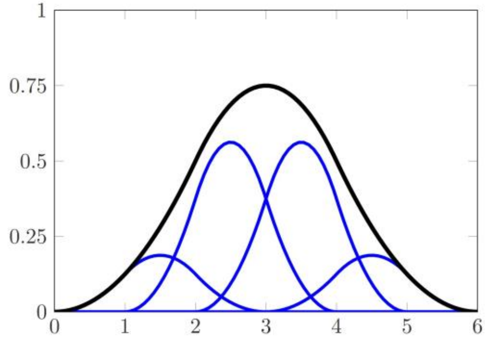
Capability #2: Support for Locally Refined Background Discretizations of Hierarchical B-splines



geometry courtesy of nTopology

Hierarchical basis enables refinement



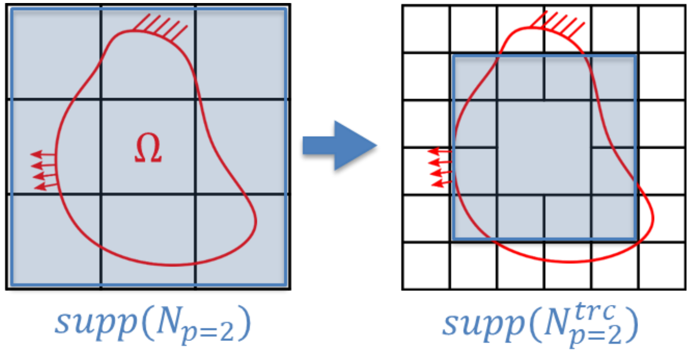


Basis function is sum of scaled and translated versions of itself

- Repeated refinement where needed

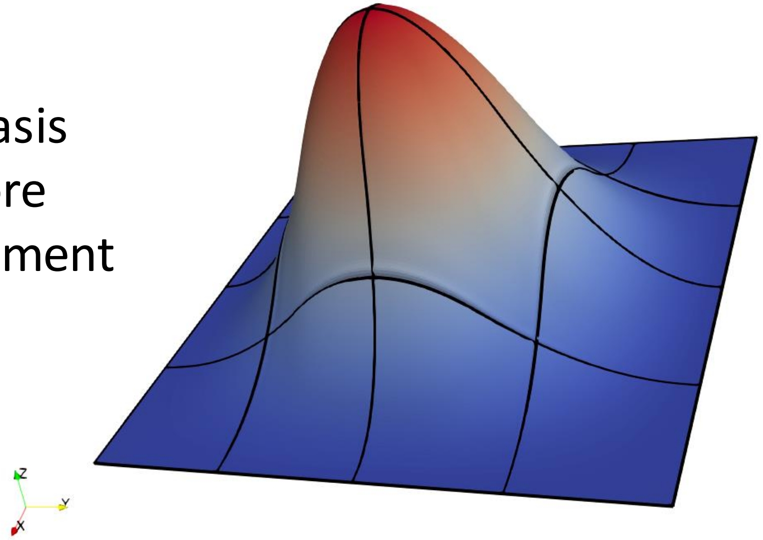
Truncation leads to:

- Reduced support size
- Retain partition of unity

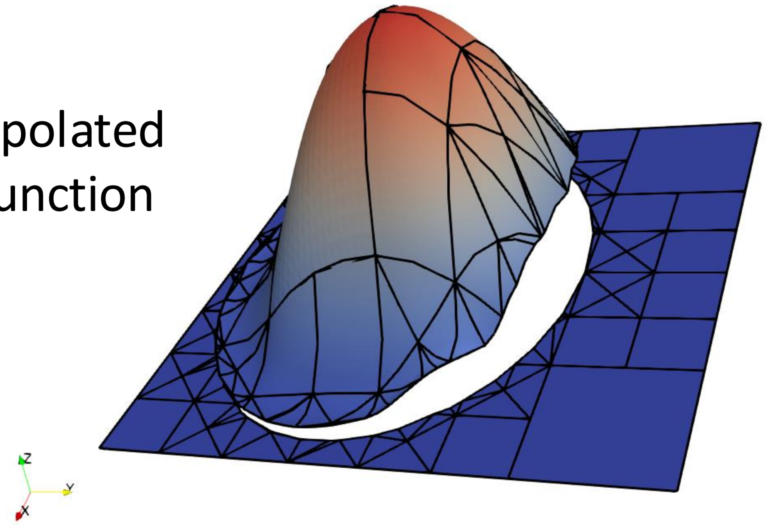


Capability #3: Support for Enriched Background Discretizations for Multi-Material Problems

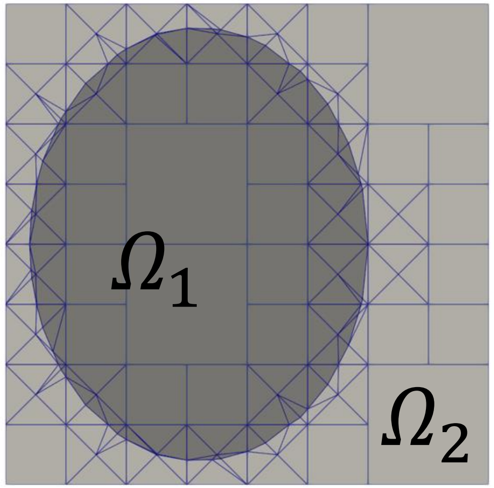
Background basis function before Heaviside enrichment



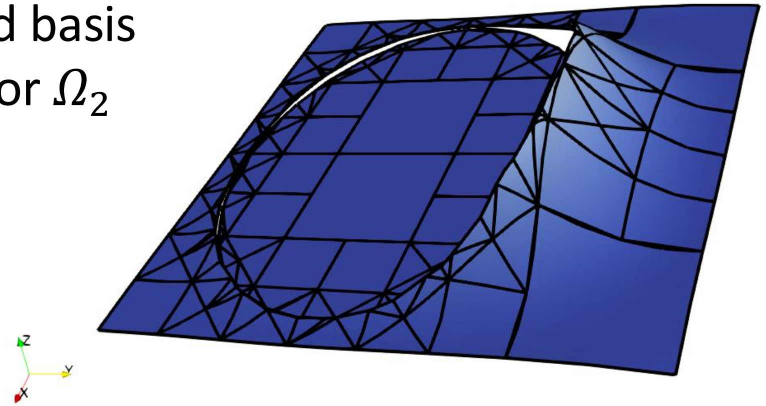
Enriched interpolated background function for Ω_1



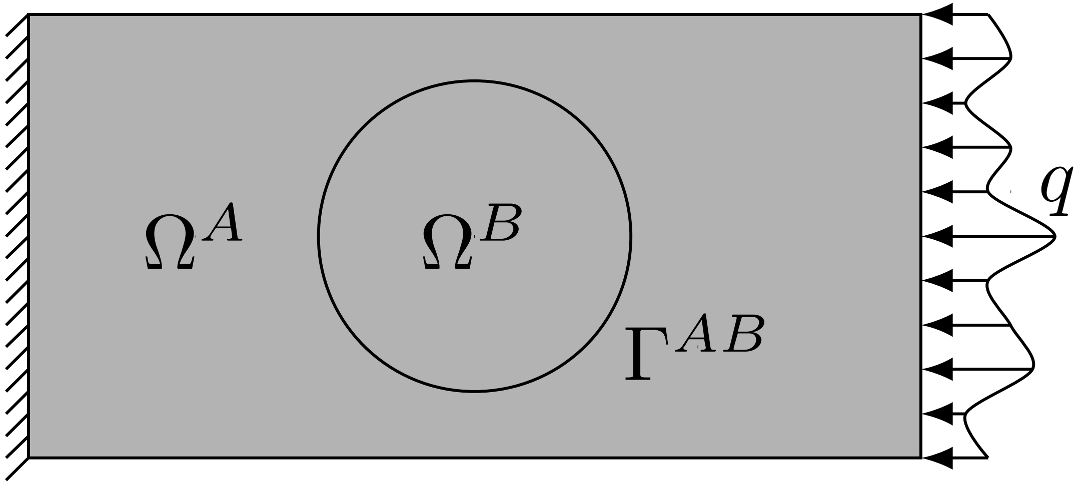
Foreground mesh with local refinement



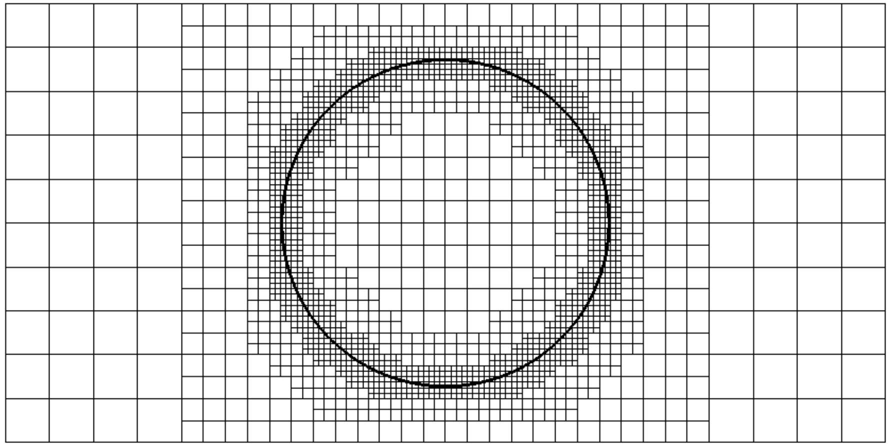
Enriched interpolated background basis function for Ω_2



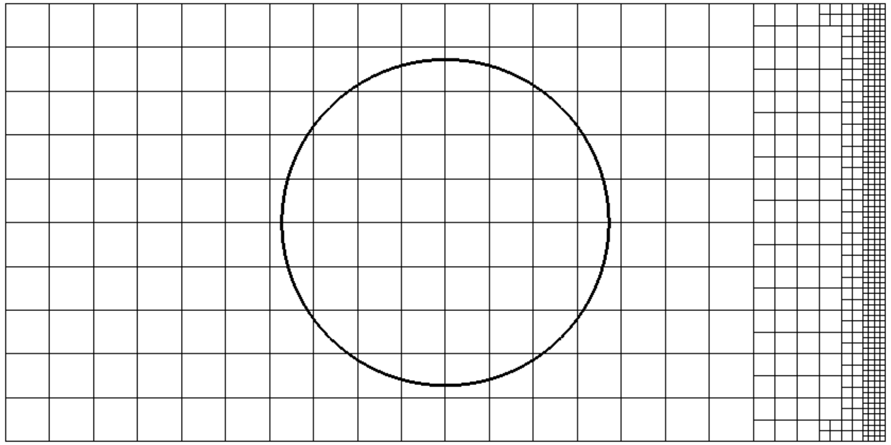
Capability #4: Support for Separate Background Discretizations for Multi-Physics Problems



Thermo-elastic multi-material problem subject to a spatially varying heat load



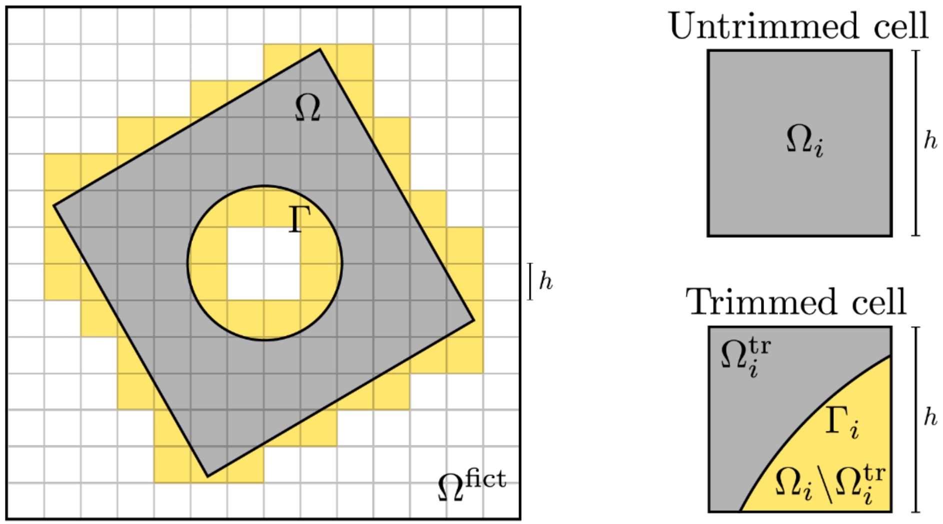
Background mesh for displacement field



Background mesh for temperature field

Capability #5: Basis Stabilization in Presence of Small Cut Cells

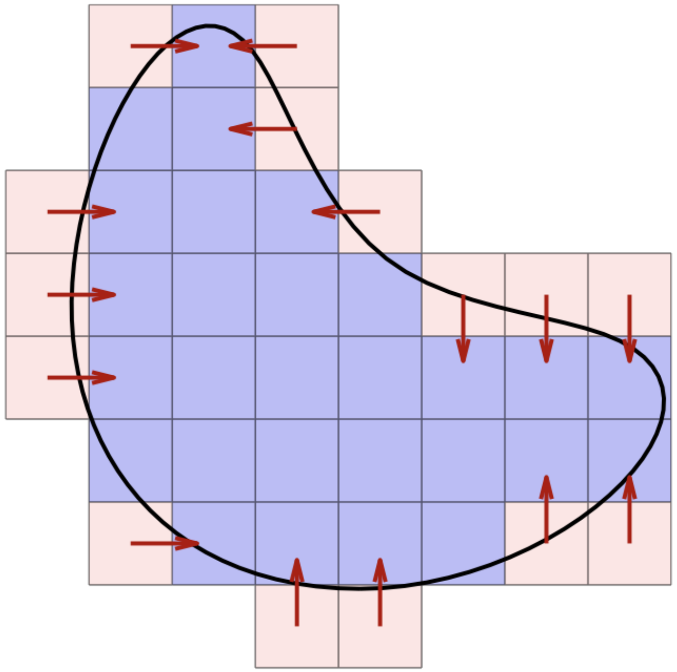
Issue: Stability and condition number deteriorate in the presence of small cut cells



$$\kappa \sim \eta^{-\left(2p+1-\frac{2}{d}\right)} \quad \text{where} \quad \eta = \min_i \frac{|\Omega_i^{tr}|}{|\Omega_i|}$$

Solution: Remove basis functions with small support and modify remaining basis functions to maintain:

(i) polynomial completeness
(ii) partition of unity



Step 1: For each bad element, extend basis functions from neighboring good element

Step 2: Project back into original spline space

The end-user does not need to modify their analysis code or workflow to include stabilization – it only affects the extraction operators and connectivity!

Burman, E., et al. "Extension operators for trimmed spline spaces." *CMAME* 403 (2023): 1

Planned Capabilities

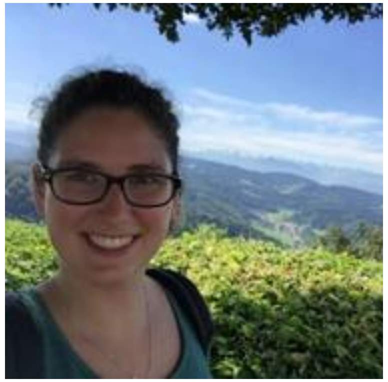
Support for foreground meshes of curved integration elements is currently being implemented.

Support for Boundary Representation (B-Rep) geometric descriptions is currently being implemented.

We are also equipping EXHUME with the capability to generate and output cut-cell quadrature rules.

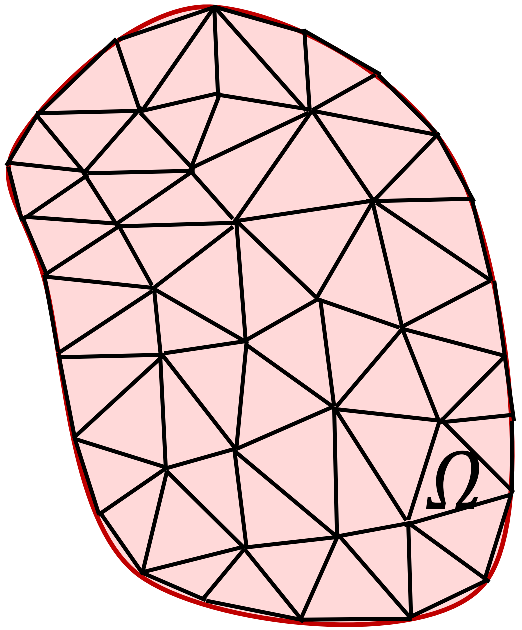
Part 6:

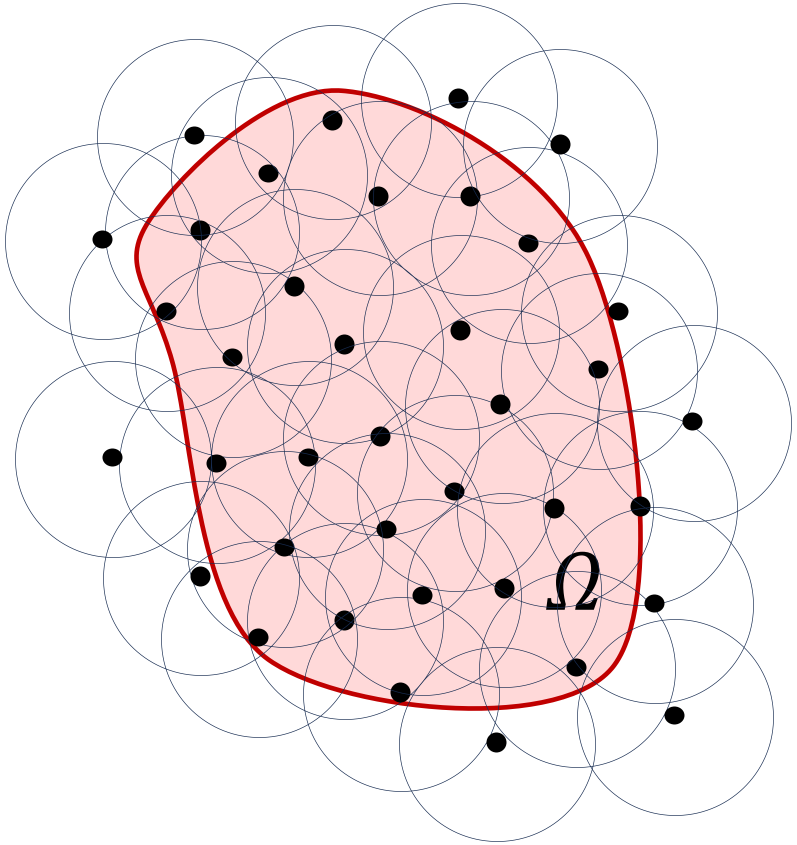
Can Interpolation Be Used To Enable Other Capabilities in FEniCS?



Jennifer Fromm

A (brief) introduction to meshfree methods





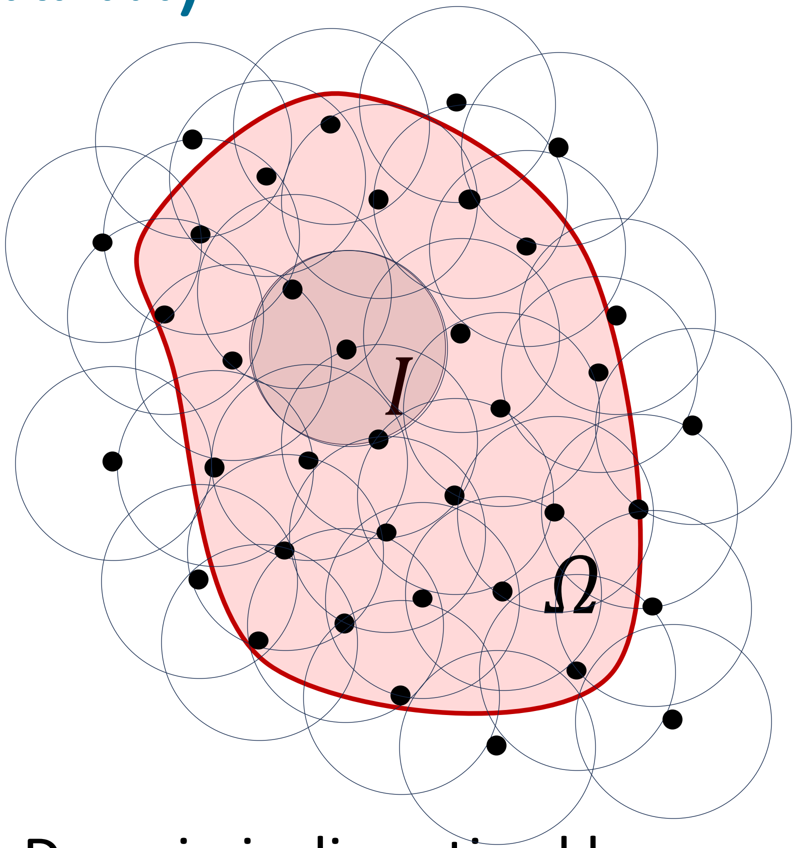
Classic Finite Element (FE):

Domain is discretized by a boundary fitted mesh

Meshfree methods:

Domain is discretized by a point cloud

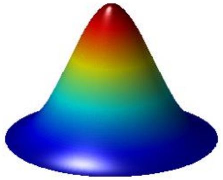
Meshfree methods: the reproducing kernel particle method (RKPM)



Domain is discretized by a point cloud

$$\Psi_I(\boldsymbol{x}) = C(\boldsymbol{x};\boldsymbol{x}-\boldsymbol{x}_I)\phi_a(\boldsymbol{x}-\boldsymbol{x}_I)$$

Shape Function | Correction Function | Kernel Function



Typically B-splines

$$C(\boldsymbol{x};\boldsymbol{x}-\boldsymbol{x}_I) = \boldsymbol{H}^T(0)\boldsymbol{M}^{-1}(\boldsymbol{x})\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}_I)$$

Basis Vector:

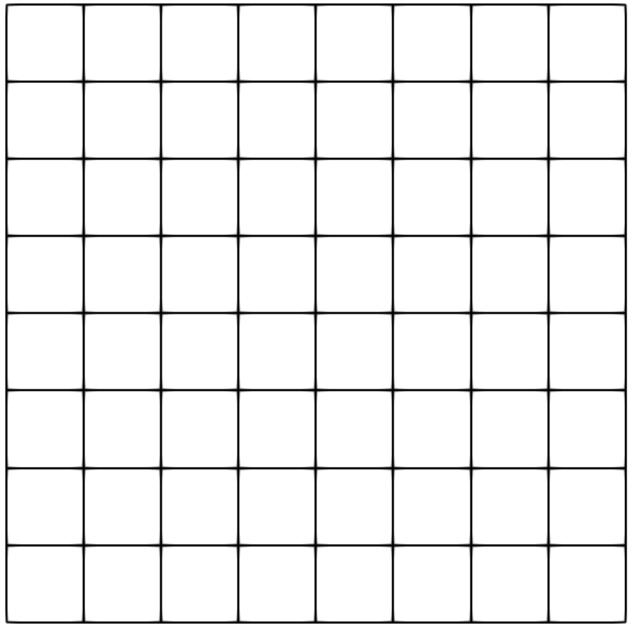
$$\boldsymbol{H}(\boldsymbol{x}) = [1, x, y, x^2, xy, y^2, \ldots xy^{n-1}, y^n]$$

Moment Matrix:

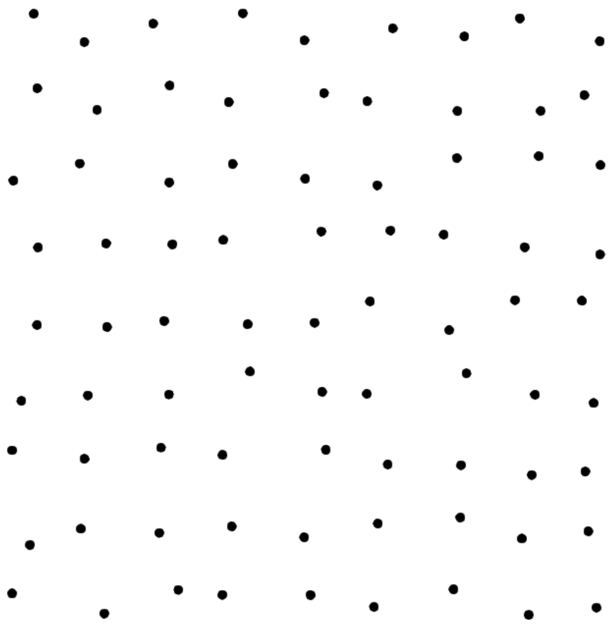
$$\boldsymbol{M}(\boldsymbol{x}) = \sum_I \boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}_I)\boldsymbol{H}^T(\boldsymbol{x}-\boldsymbol{x}_I)\phi_a(\boldsymbol{x}-\boldsymbol{x}_I)$$

Chen, J. S., Pan, C., Wu, C. T., and Liu, W. K., "Reproducing Kernel Particle Methods for Large Deformation Analysis of Nonlinear Structures," *Computer Methods in Applied Mechanics and Engineering*, Vol. 139, pp. 195-227, 1996.

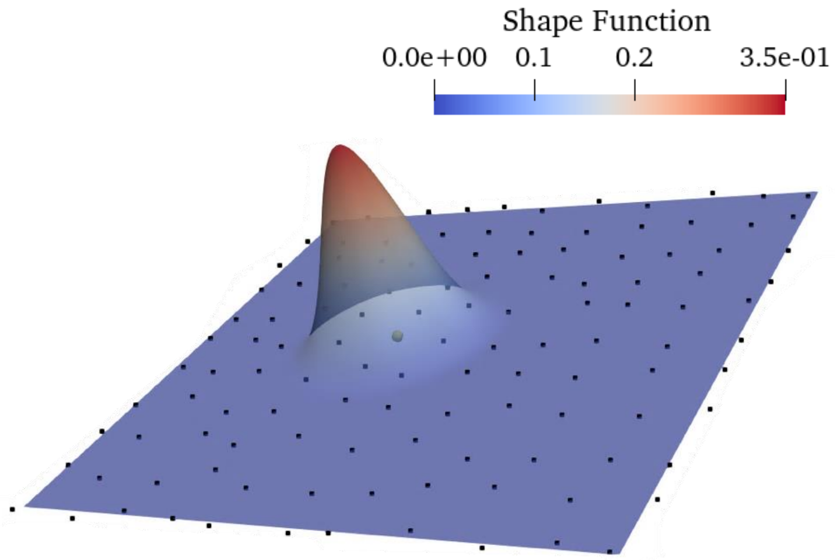
Interpolated-RKPM: introducing a foreground mesh



Foreground integration mesh, with Lagrange polynomial basis $\{N_i\}_{i=0}^{\nu}$



RKPM point cloud



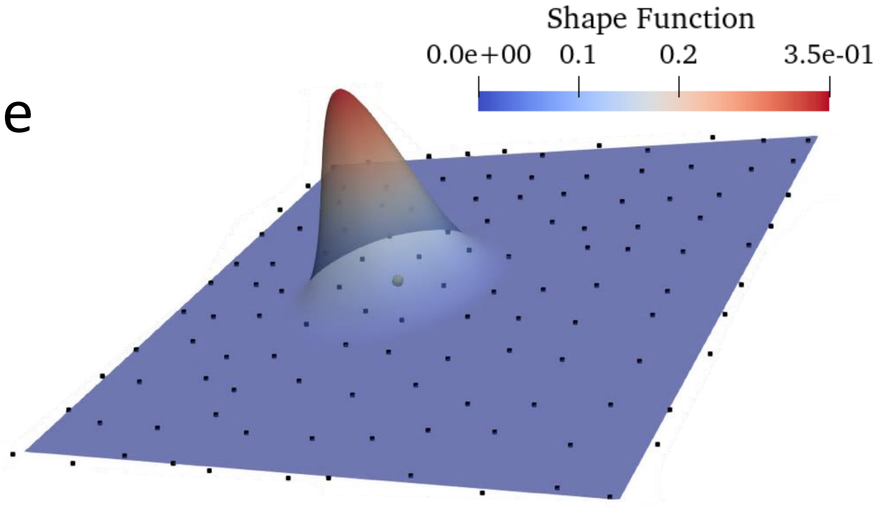
Linear RKPM shape function $\Psi_I(\boldsymbol{x})$

Each shape function can be interpolated: $\widehat{\Psi}_I(\boldsymbol{x}) = \sum_{i=0}^{\nu} M_{Ij}\, N_j(\boldsymbol{x})$

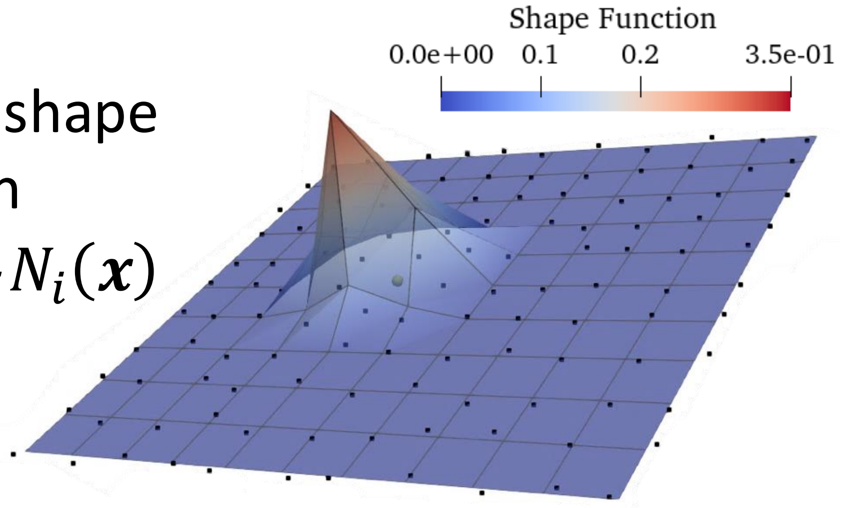
With extraction operator $M_{Ij} = \Psi_I(\boldsymbol{x}_j)$

Int-RKPM shape functions: equal element size and polynomial order

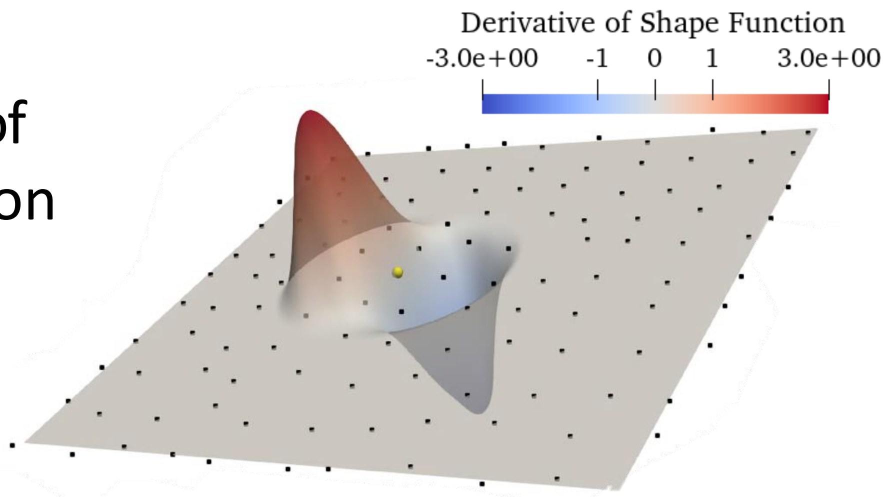
Linear shape function $\Psi_I(\boldsymbol{x})$



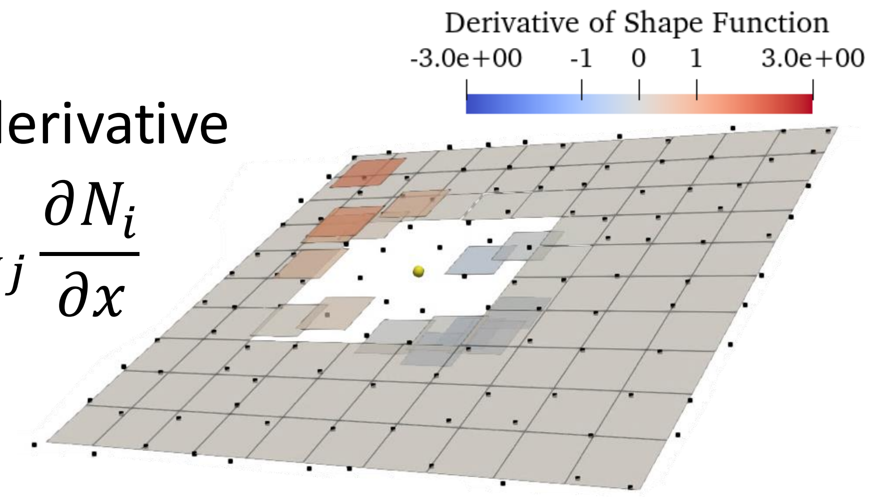
Interpolated shape function $\widehat{\Psi}_I(\boldsymbol{x}) = \boldsymbol{M}_{Ij} N_i(\boldsymbol{x})$



Derivative of shape function $\dfrac{\partial \Psi_I}{\partial x}$

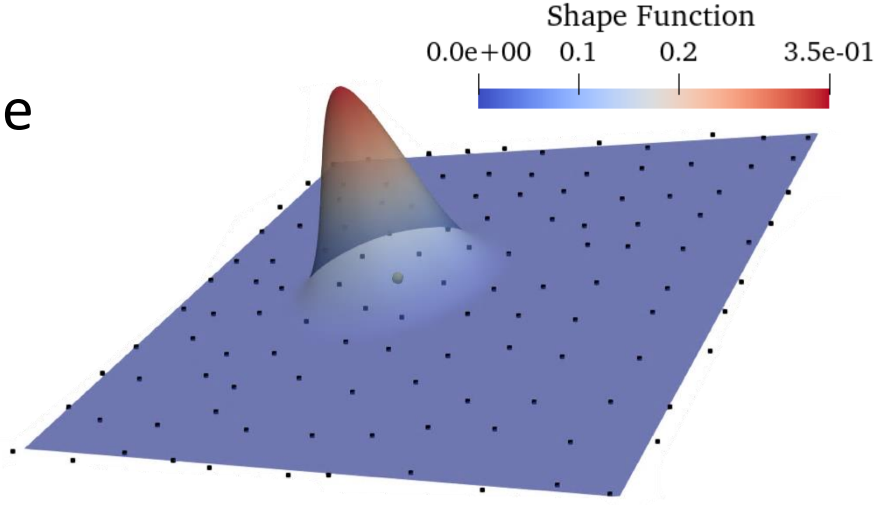


Interpolated derivative $\dfrac{\partial \widehat{\Psi}_I}{\partial x} = \boldsymbol{M}_{Ij} \dfrac{\partial N_i}{\partial x}$

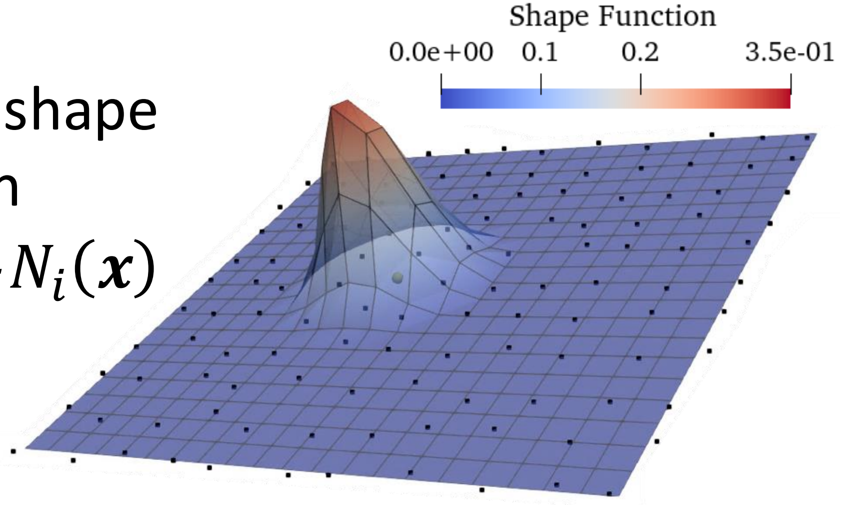


Int-RKPM shape functions: foreground h-refinement

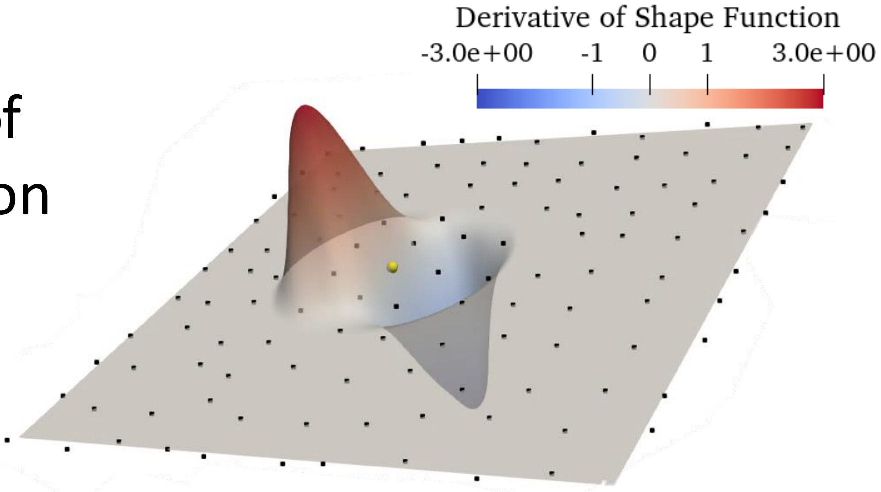
Linear shape function
$\Psi_I(\boldsymbol{x})$



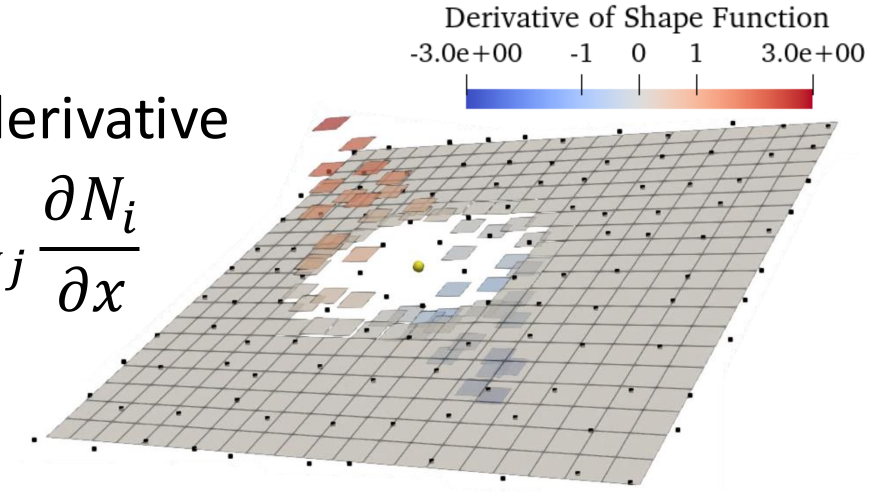
Interpolated shape function
$\widehat{\Psi}_I(\boldsymbol{x}) = \boldsymbol{M}_{Ij} N_i(\boldsymbol{x})$



Derivative of shape function
$\frac{\partial \Psi_I}{\partial x}$

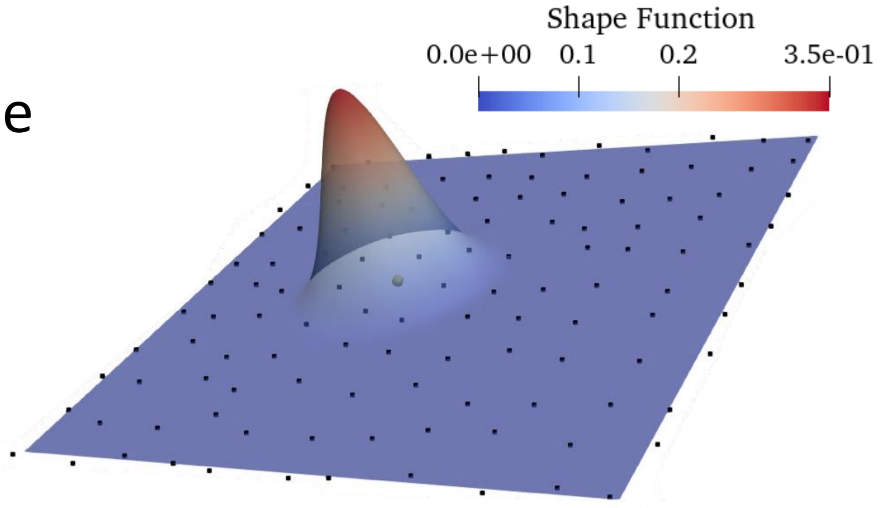


Interpolated derivative
$\frac{\partial \widehat{\Psi}_I}{\partial x} = \boldsymbol{M}_{Ij} \frac{\partial N_i}{\partial x}$

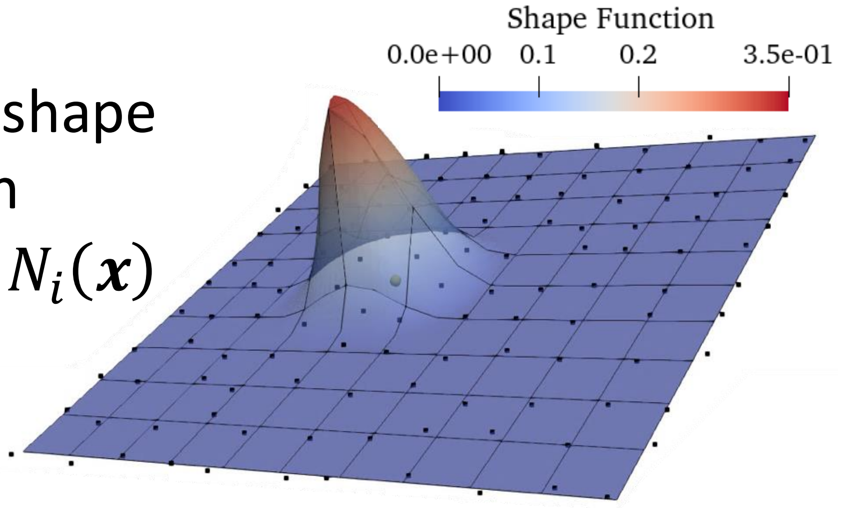


Int-RKPM shape functions: foreground p-refinement

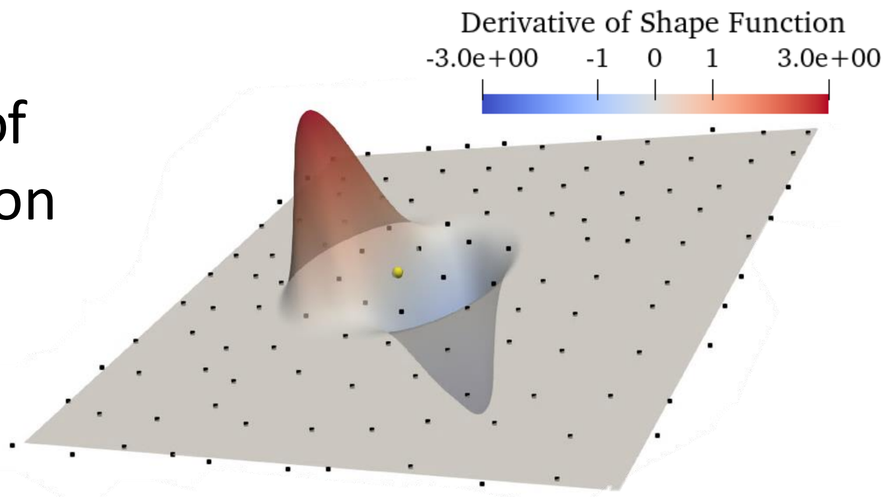
Linear shape function
$\Psi_I(\boldsymbol{x})$



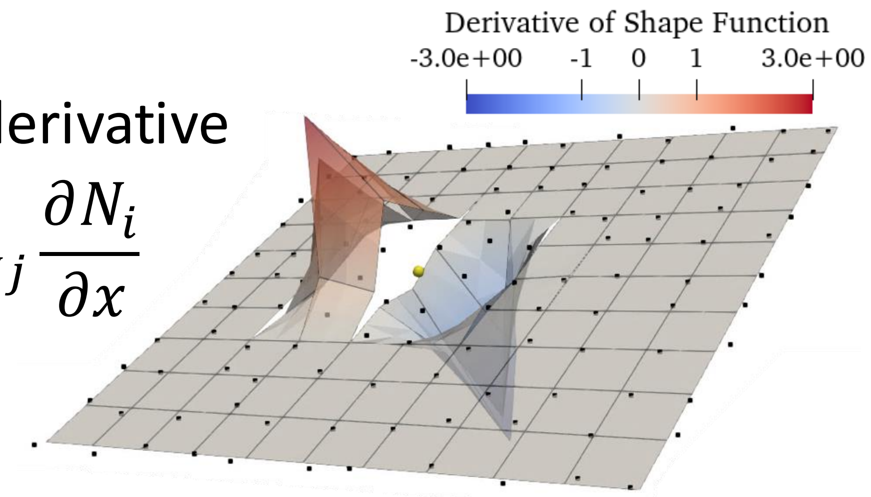
Interpolated shape function
$\widehat{\Psi}_I(\boldsymbol{x}) = \boldsymbol{M}_{Ij} N_i(\boldsymbol{x})$



Derivative of shape function
$\frac{\partial \Psi_I}{\partial x}$

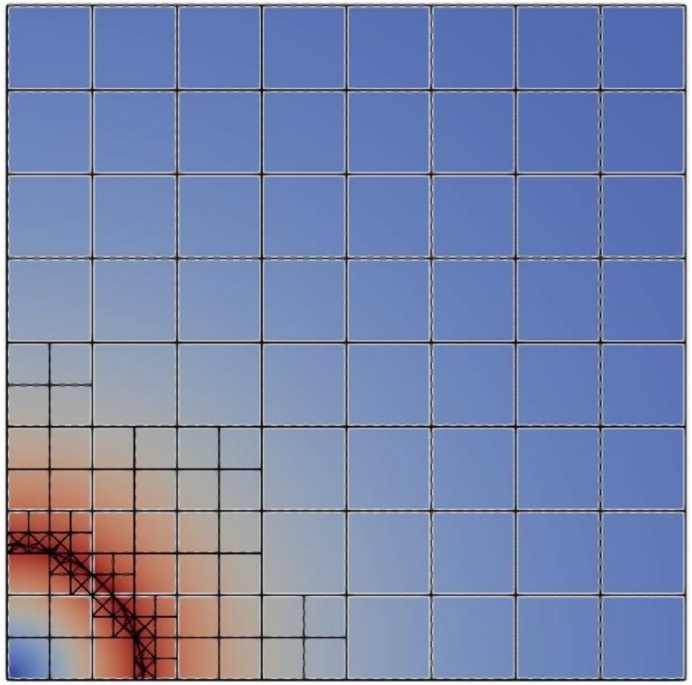


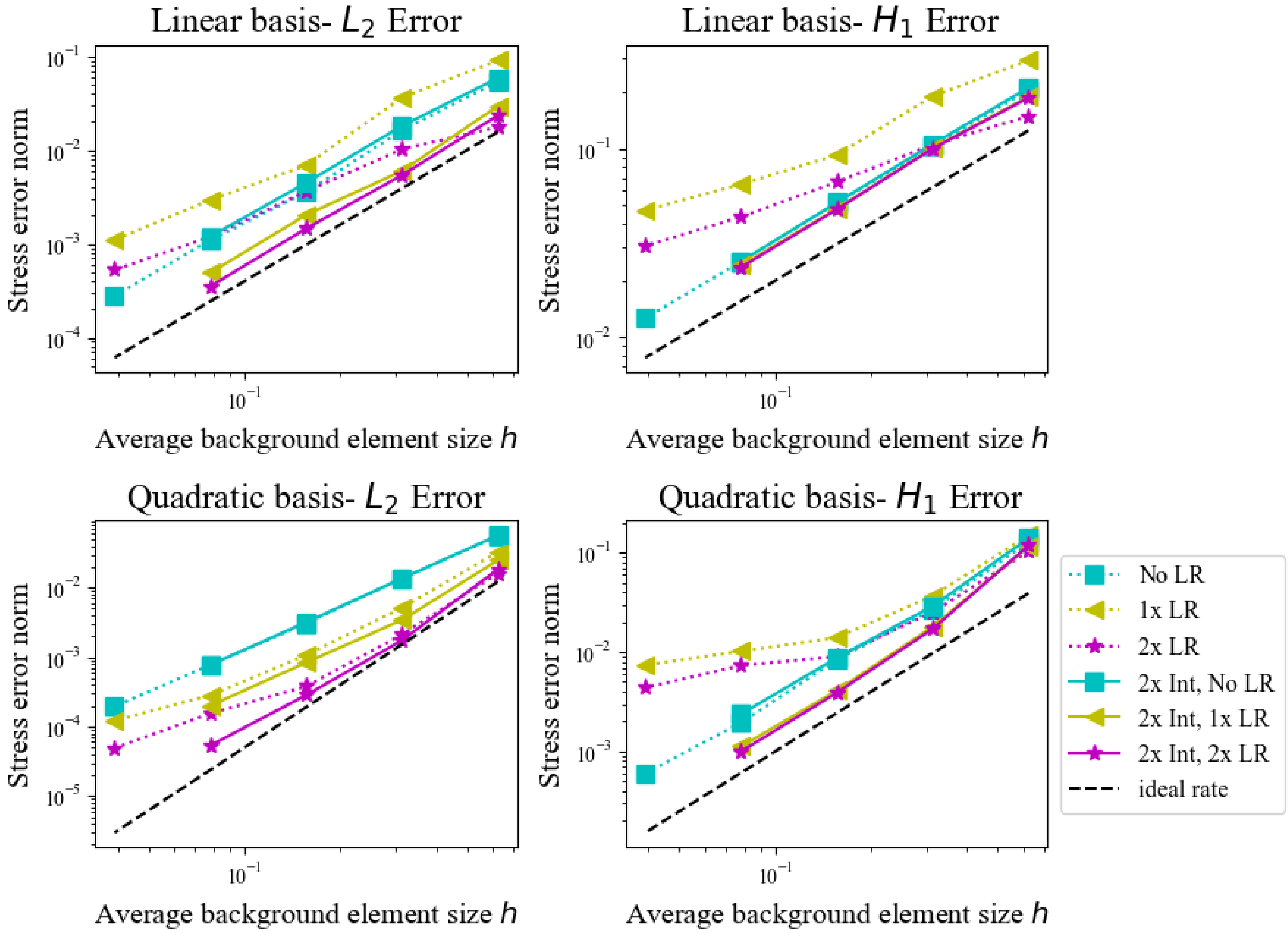
Interpolated derivative
$\frac{\partial \widehat{\Psi}_I}{\partial x} = \boldsymbol{M}_{Ij} \frac{\partial N_i}{\partial x}$



Int-RKPM analysis of multi-material problems with enrichment and local refinement

Linear-elasticity problem with a circular inclusion and induced eigenstrain





Part 7:

Useful Links

Useful Links

- EXHUME (MORIS) software library

https://github.com/kkmaute/moris

- FEniCS implementation of interpolation-based immersed analysis:

https://github.com/jefromm/interpolation-based-immersed-fea

- First paper on interpolation-based immersed analysis:

J.E. Fromm, N. Wunsch, R. Xiang, H. Zhao, K. Maute, J.A. Evans, and D. Kamensky. "Interpolation-based immersed finite element and isogeometric analysis." CMAME 405 (2023): 115890.

https://doi.org/10.1016/j.cma.2023.115890

MORIS

FEniCS Implementation

Useful Links

- Slides and videos from Fall 2023 EXHUME Collaborators Workshop

Thank you!

Funding for EXHUME and MORIS provided by: