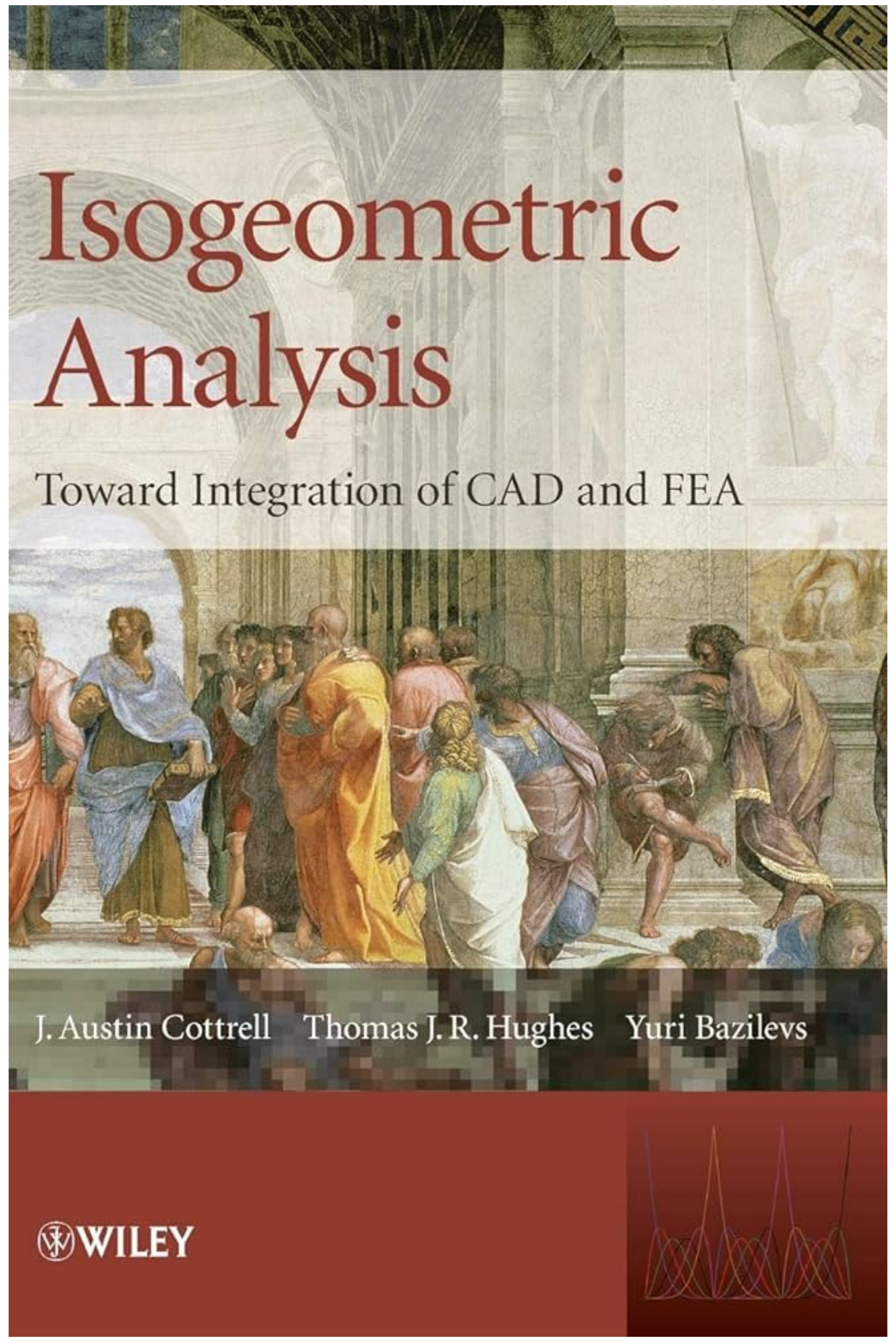
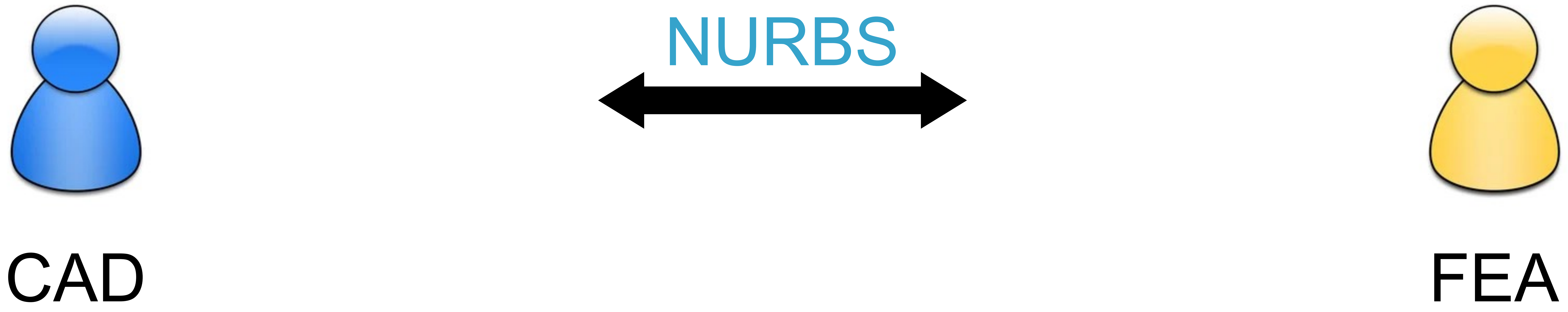
Bridging the gap between isogeometric analysis and deep operator learning

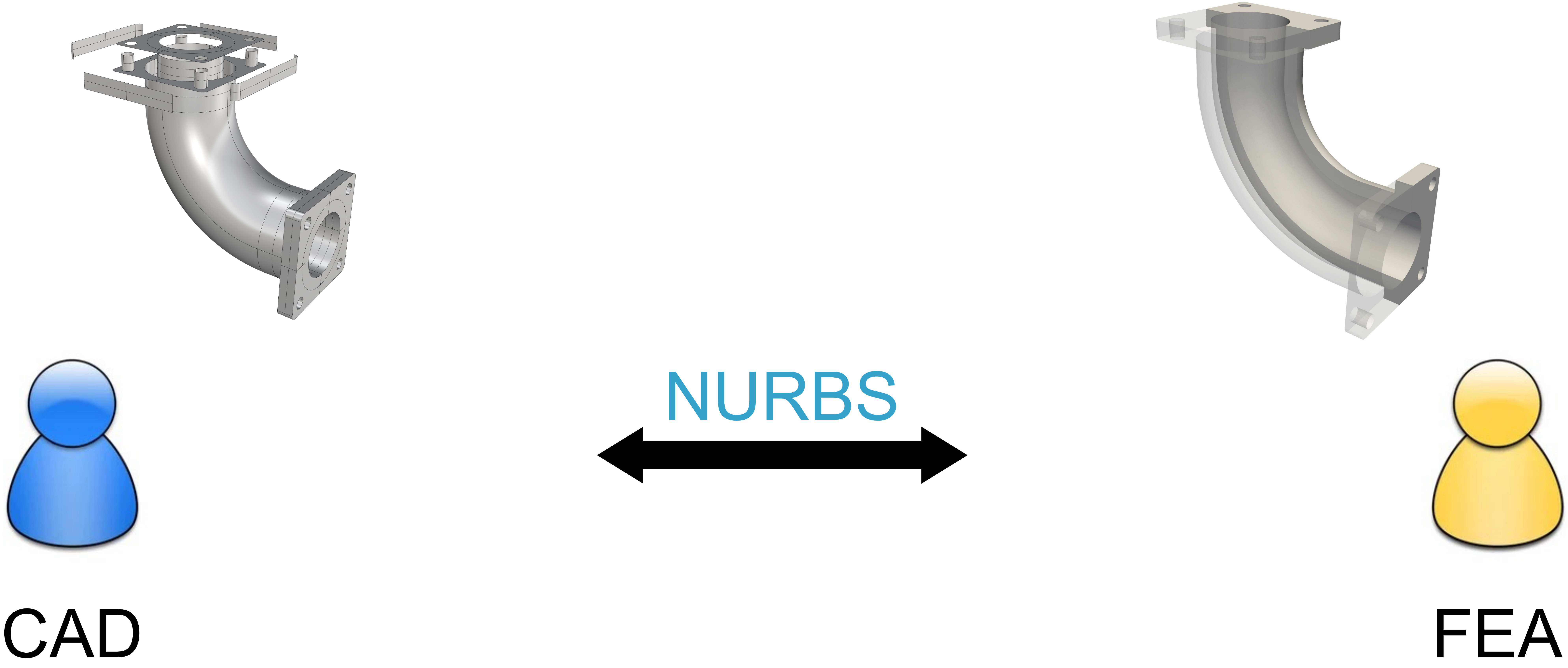
Matthias Möller
Department of Applied Mathematics
Delft University of Technology, The Netherlands

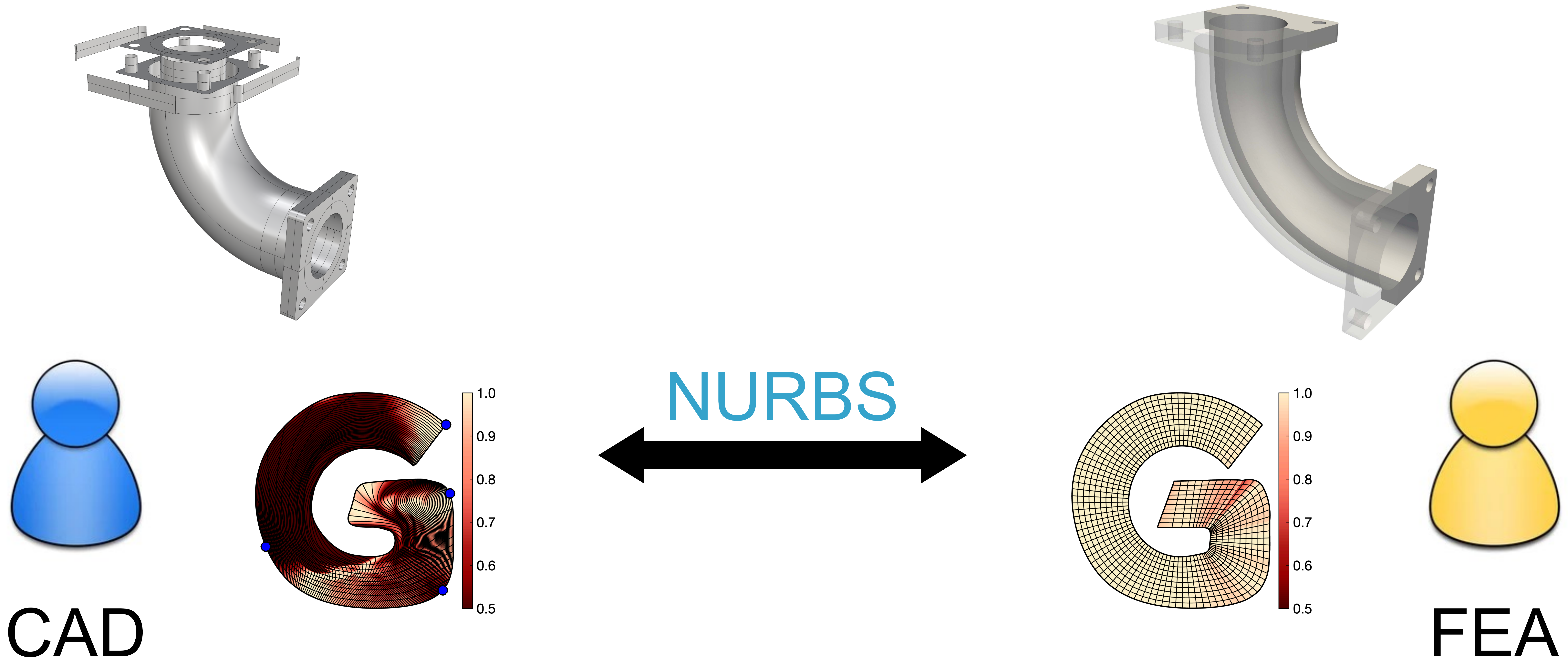
Acknowledgements: Ye Ji, Hugo Verhelst, Mengyun Wang (TU Delft), Jochen Hinz, Merle Backmeyer (ETH Zurich), Casper van Leeuwen (SURF), Jaewook Lee (TU Vienna), Veronika Trávníková (RWTH Aachen), …

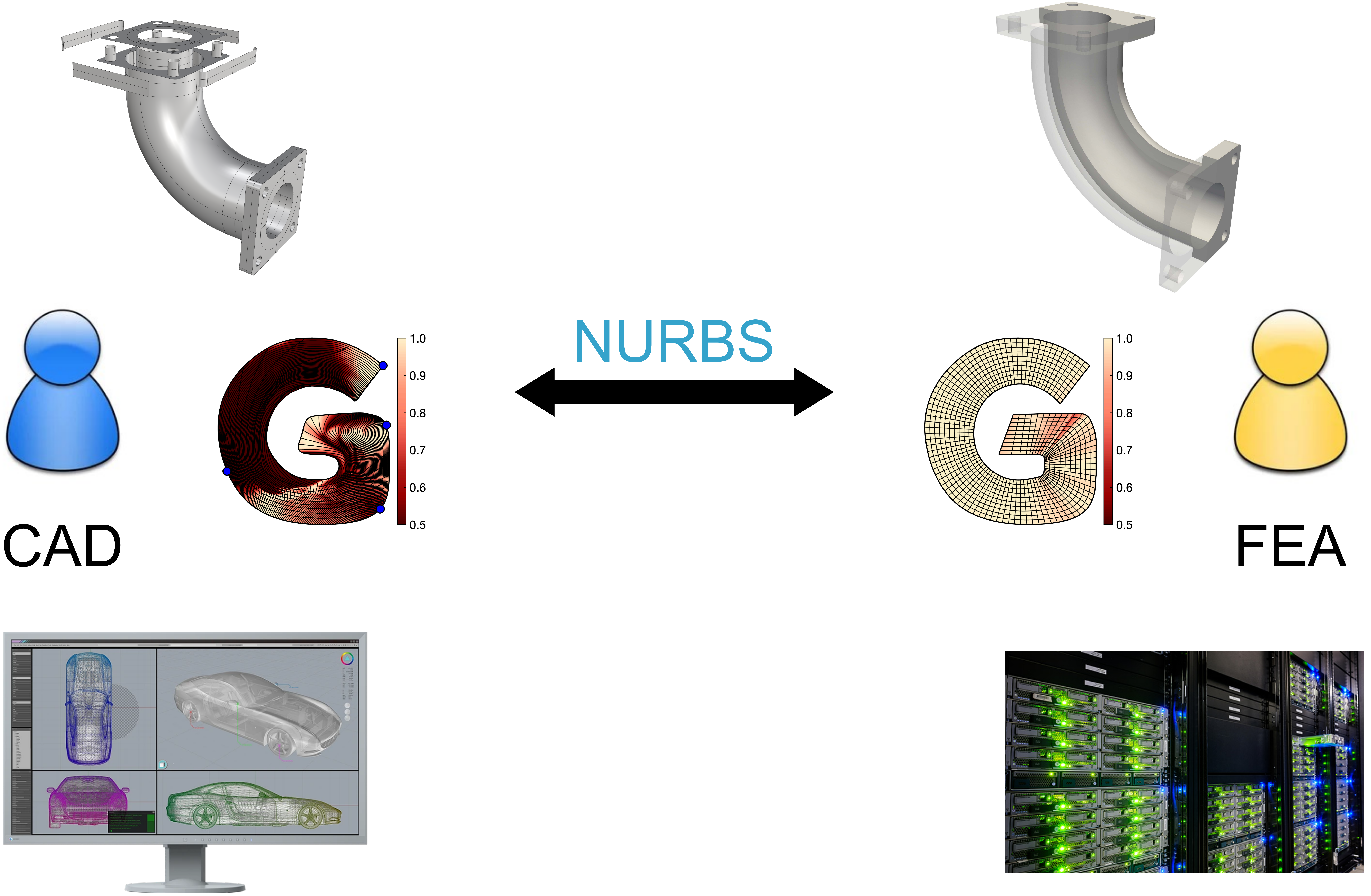
Bridging the gap between CAD and FEA by using B-spline or NURBS basis functions for design <u>and</u> analysis . . .




NURBS
CAD
FEA

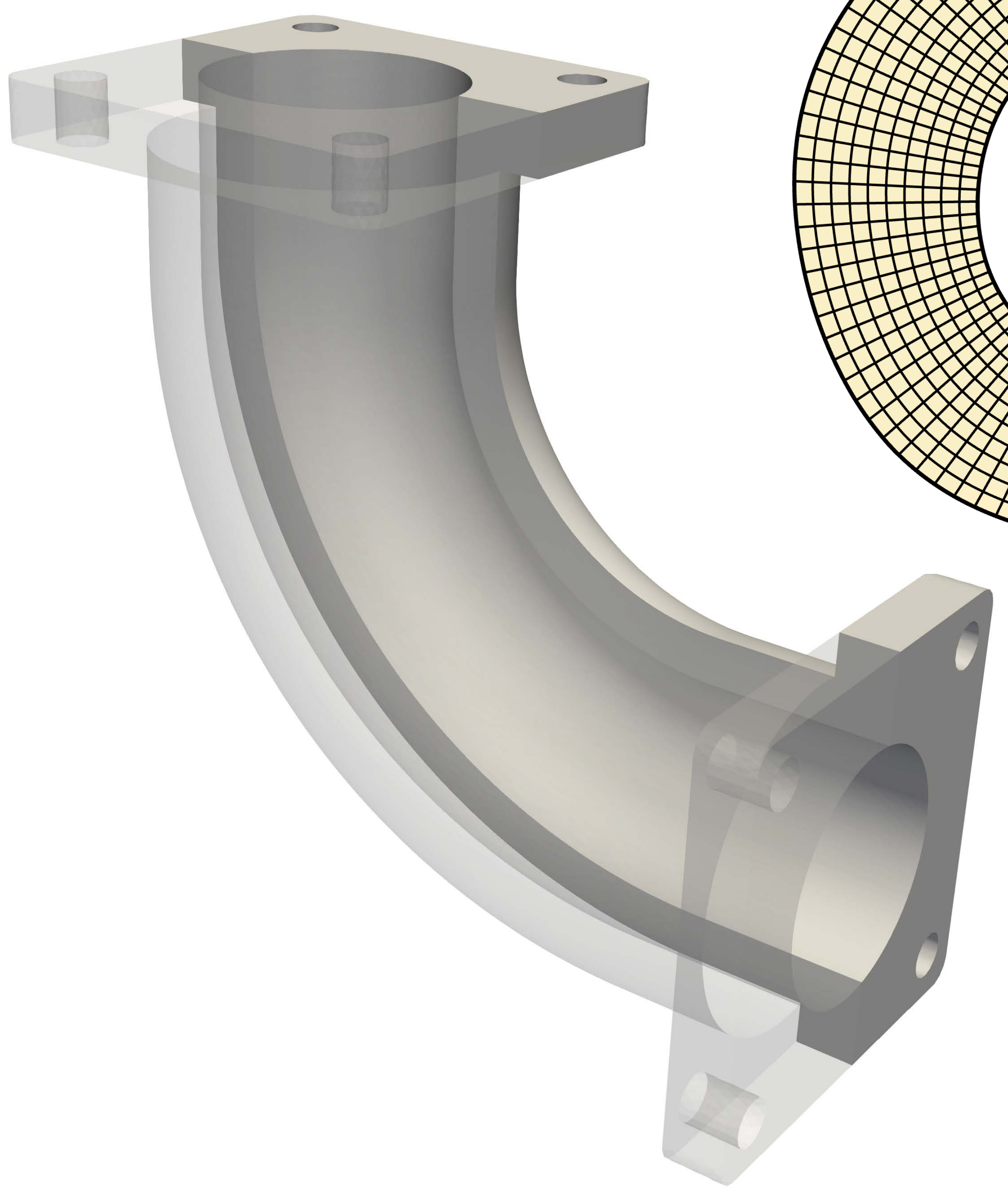

NURBS
CAD
FEA

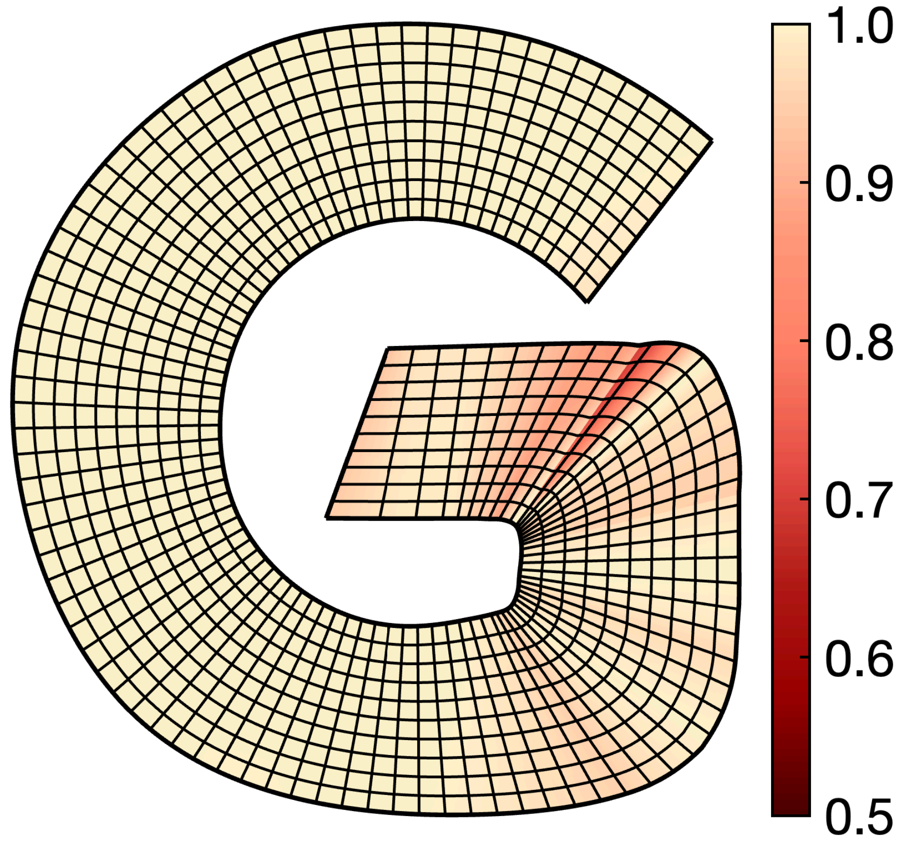

NURBS
CAD
FEA
1.0
0.9
0.8
0.7
0.6
0.5



NURBS
CAD
FEA
1.0
0.9
0.8
0.7
0.6
0.5

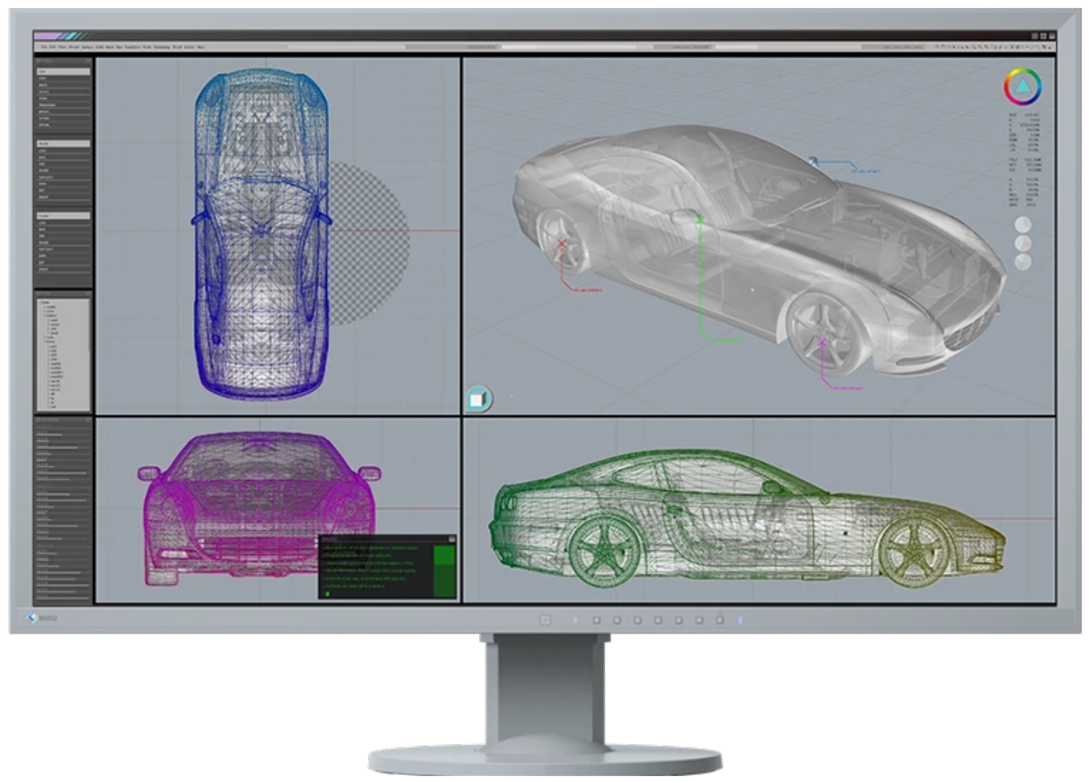
Vision





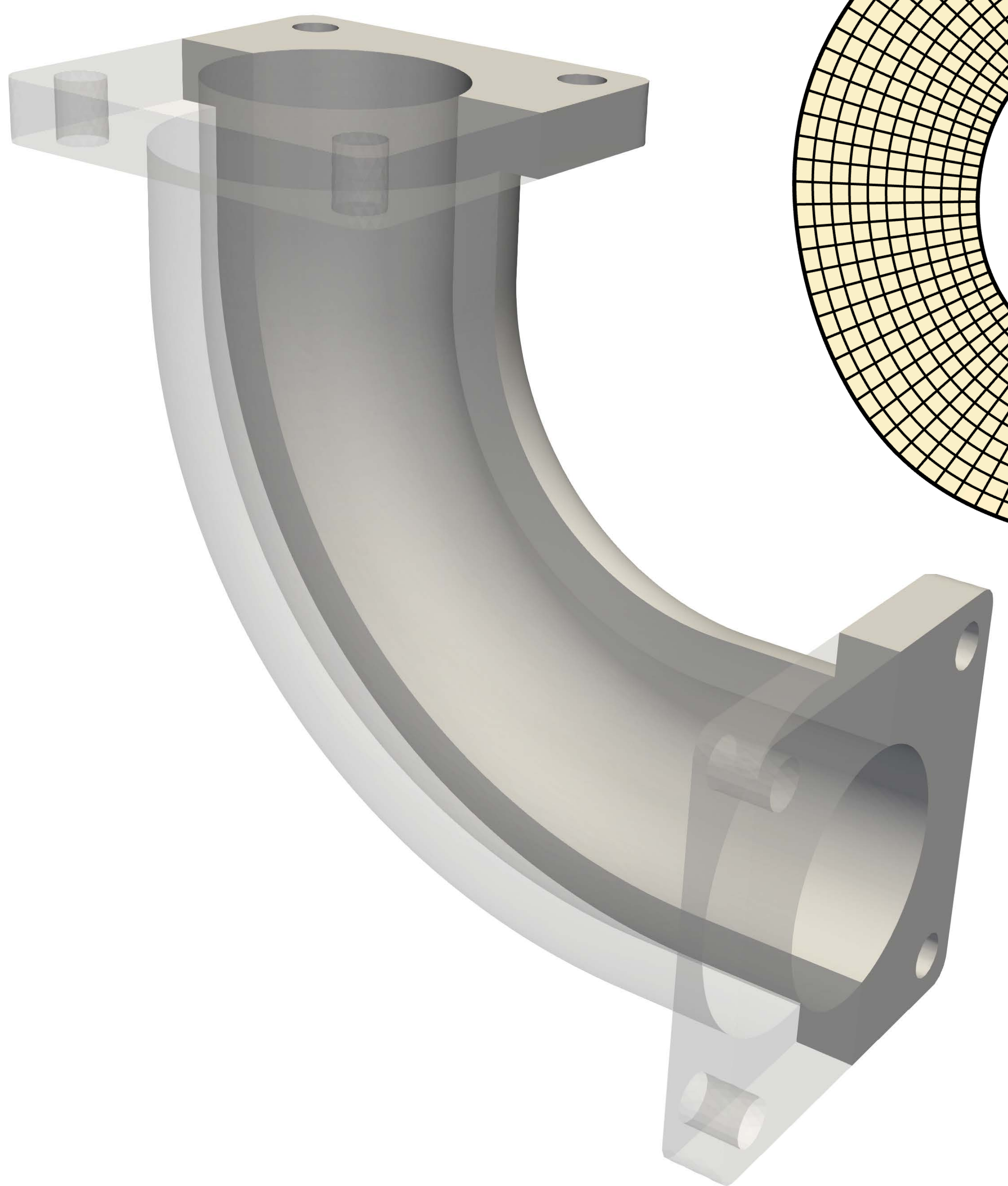
CAD

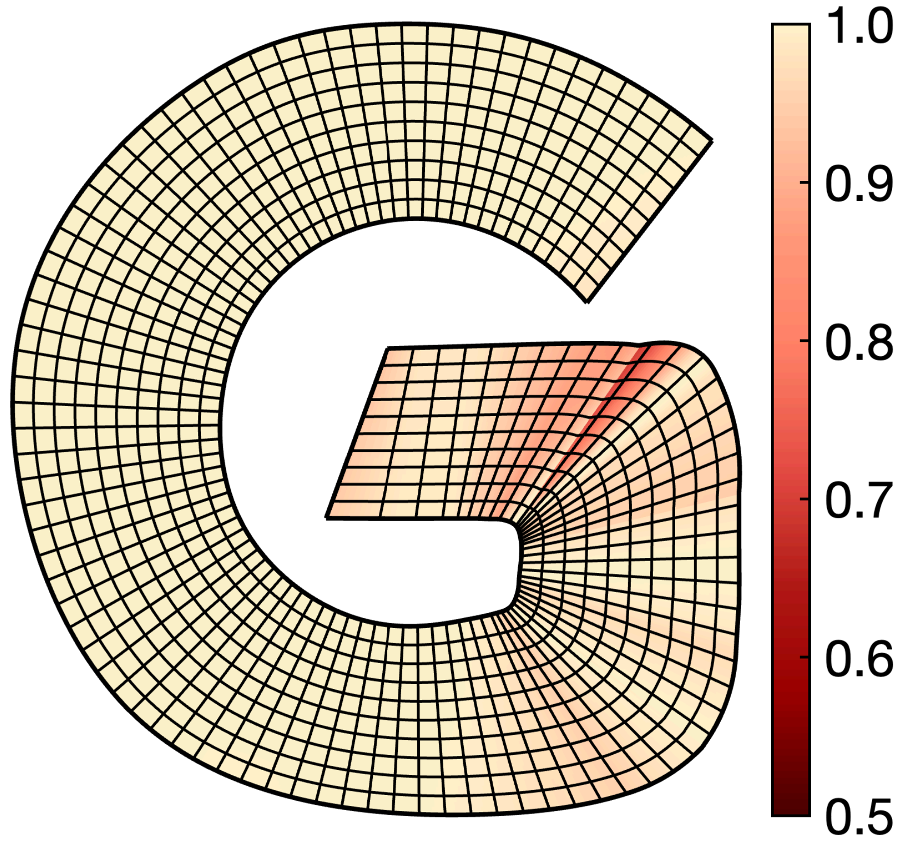
FEA



?

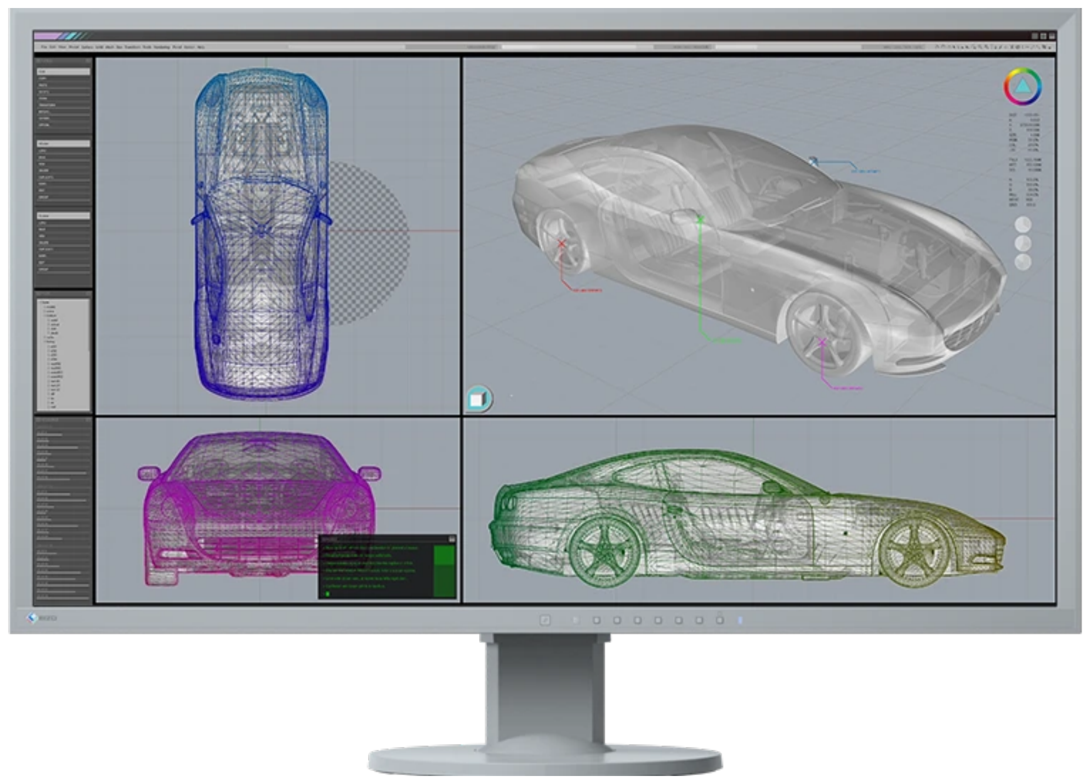
Vision

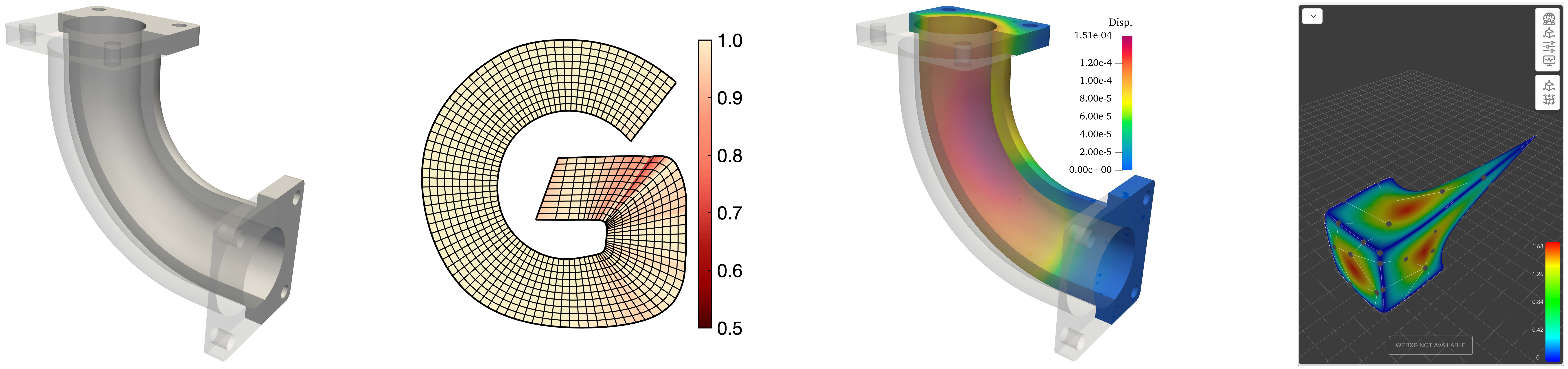


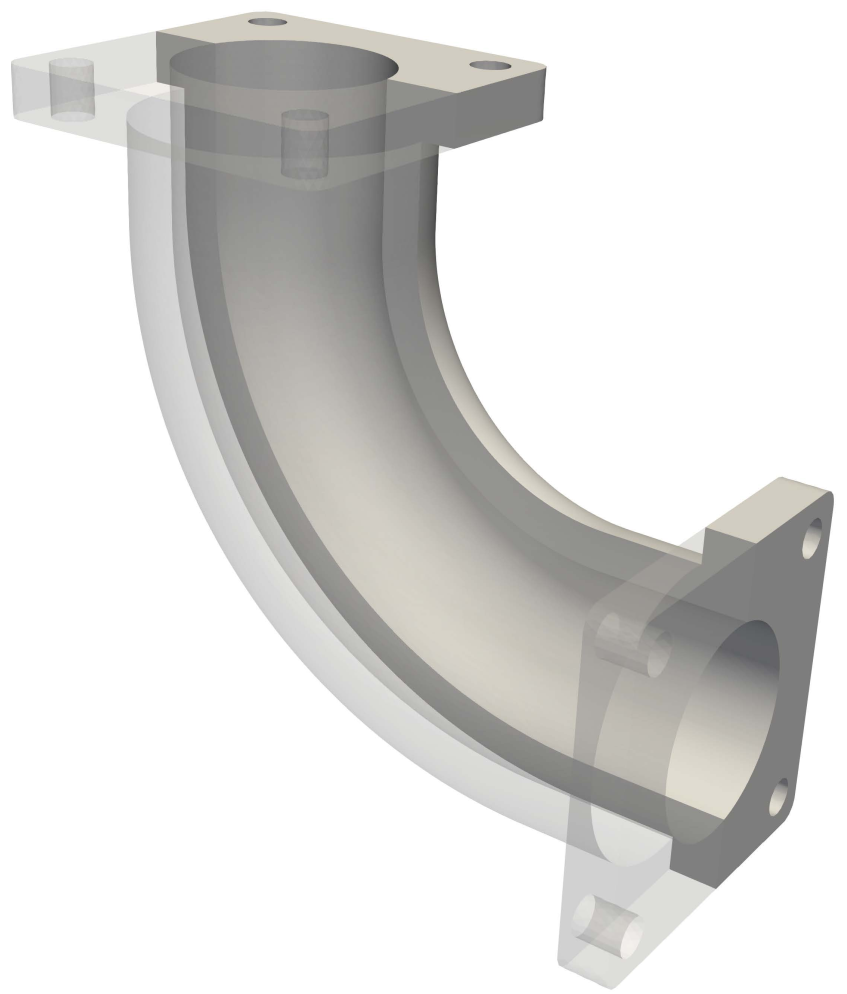


CAD

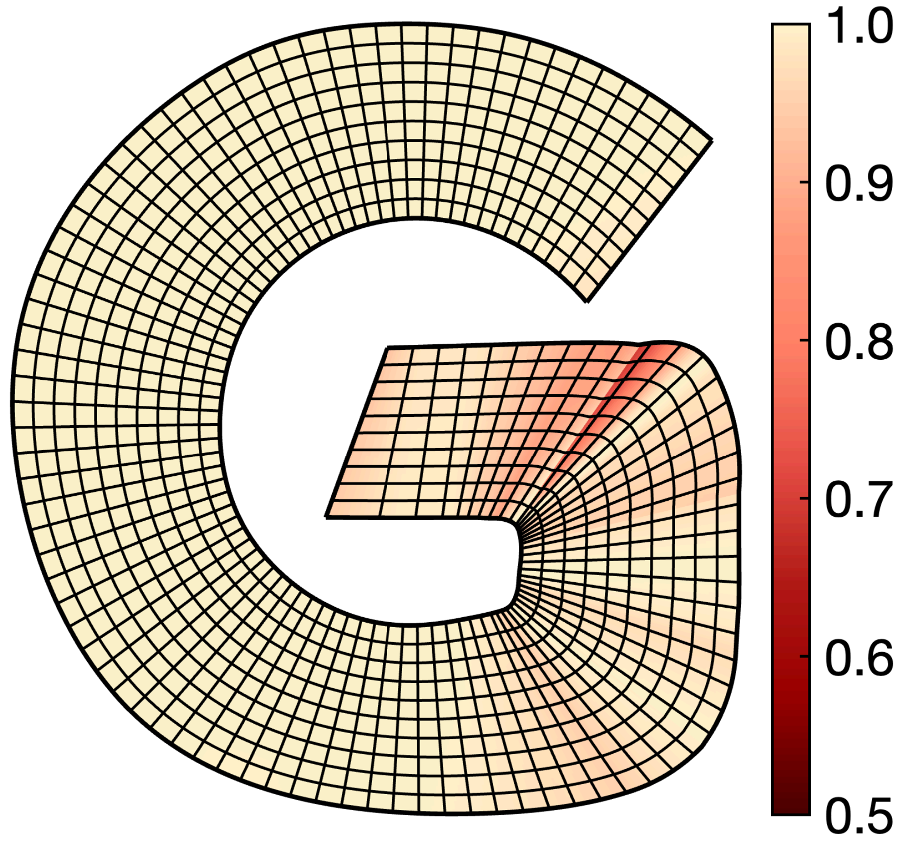
FEA



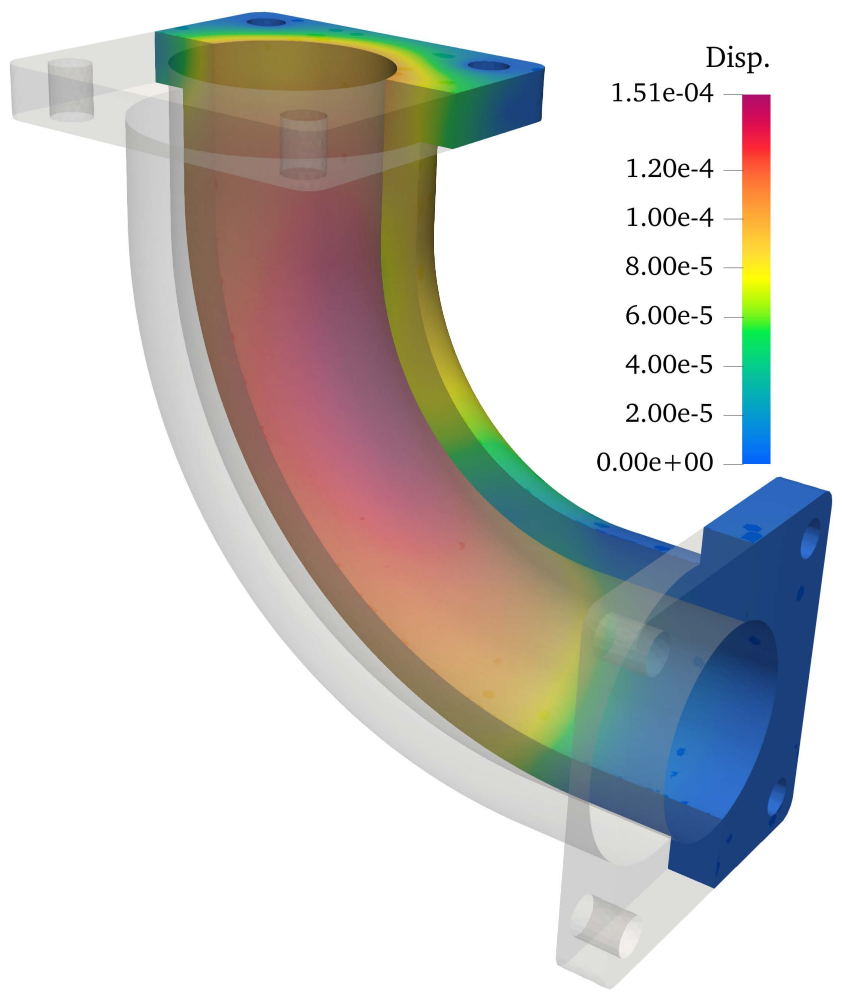

1.0
0.9
0.8
0.7
0.6
0.5
Disp.
1.51e-04
1.20e-4
1.00e-4
8.00e-5
6.00e-5
4.00e-5
2.00e-5
0.00e+00
WEBXR NOT AVAILABLE
1.68
1.26
0.84
0.42
0



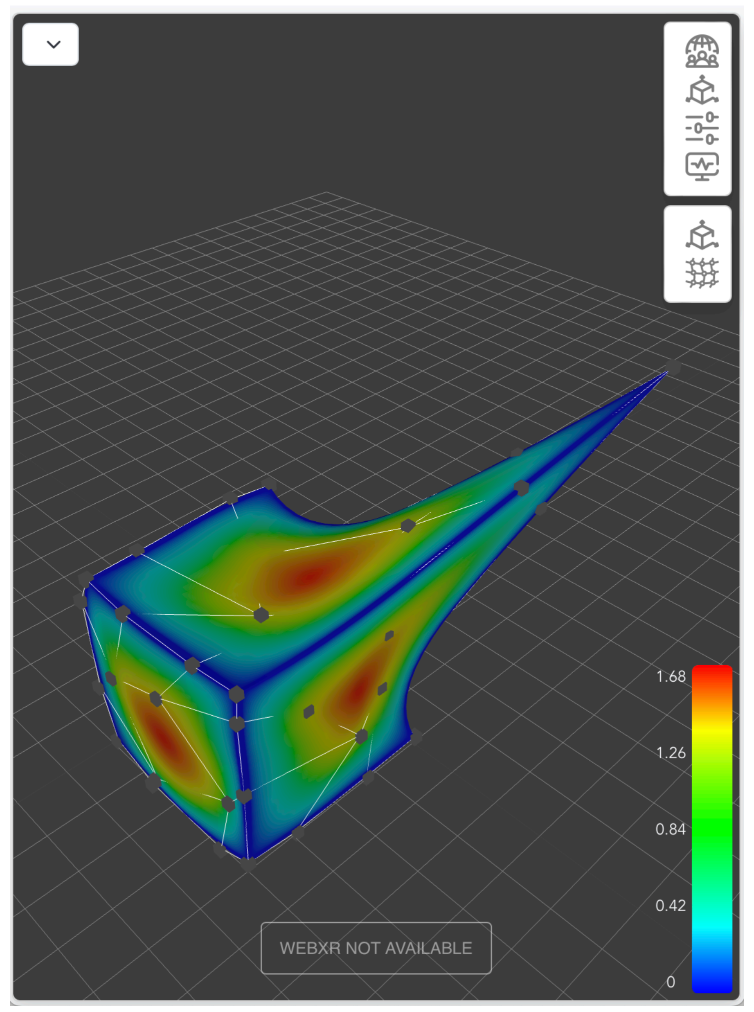
1. Creation of VReps from BReps



2. Reparameterization techniques

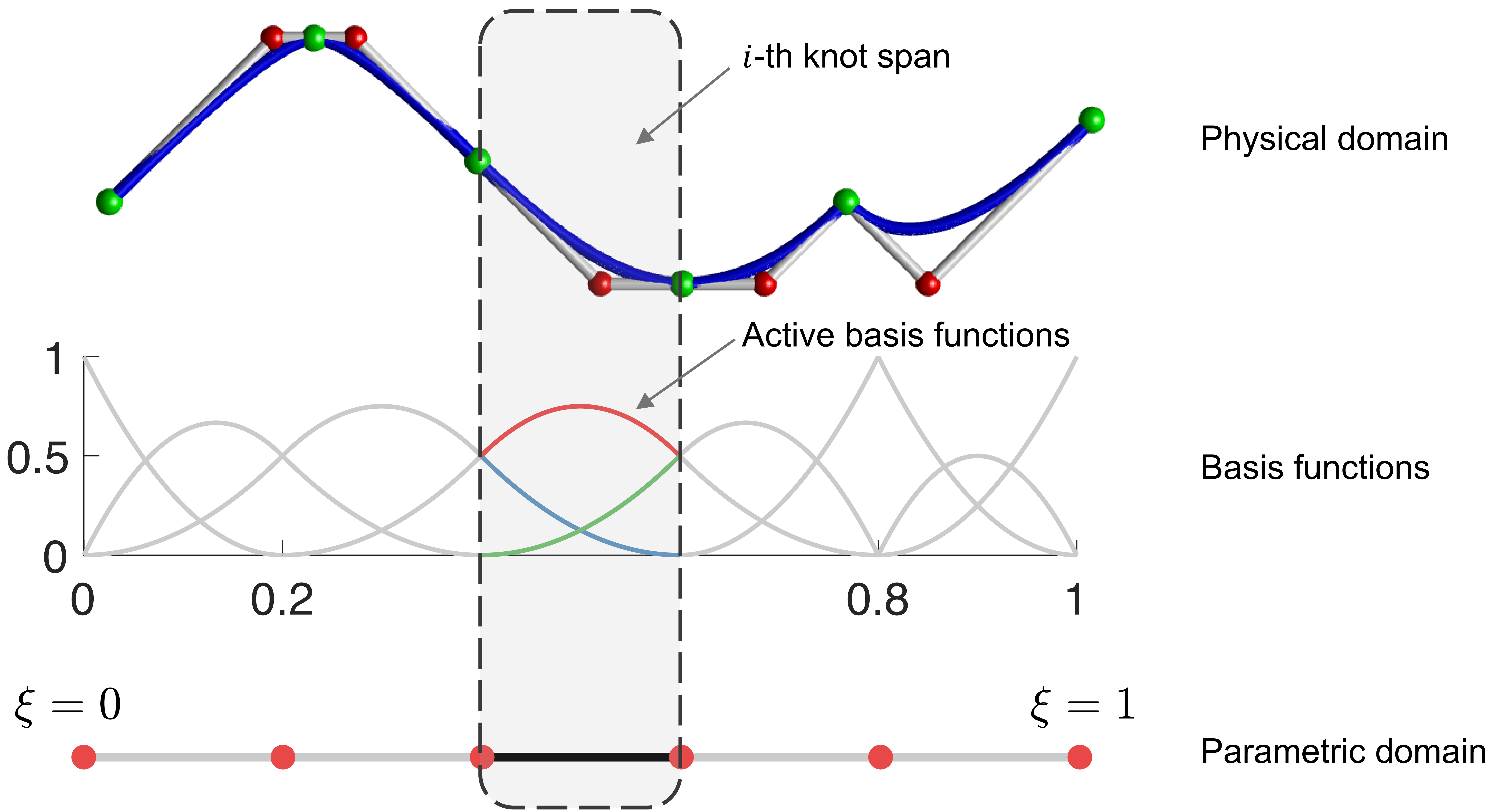


3. Isogeometric Analysis and IgANets

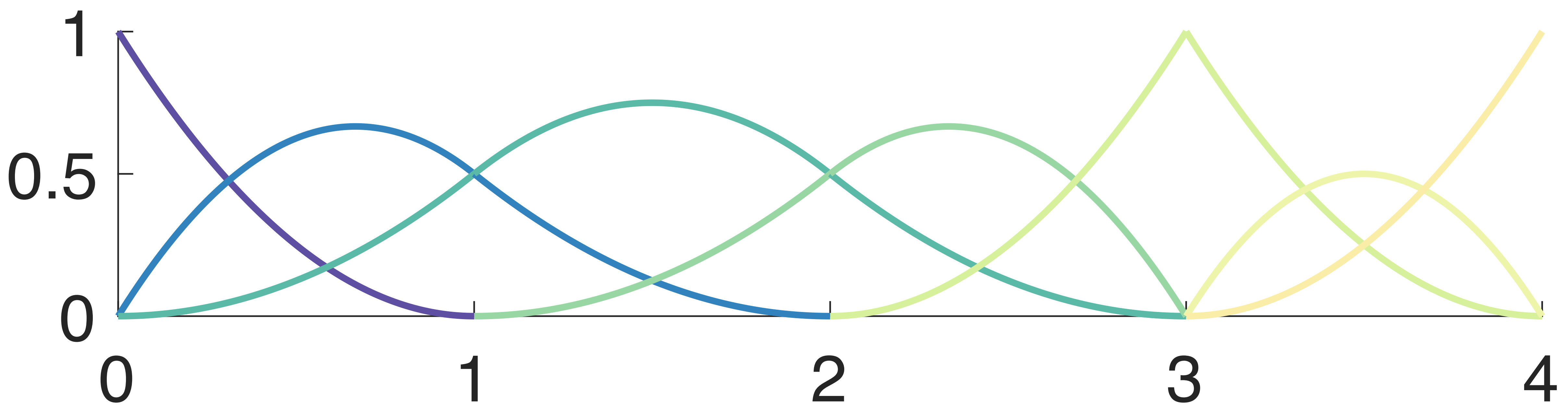


4. Interactive Design-through-Analysis

Spline curves



Univariate B-splines



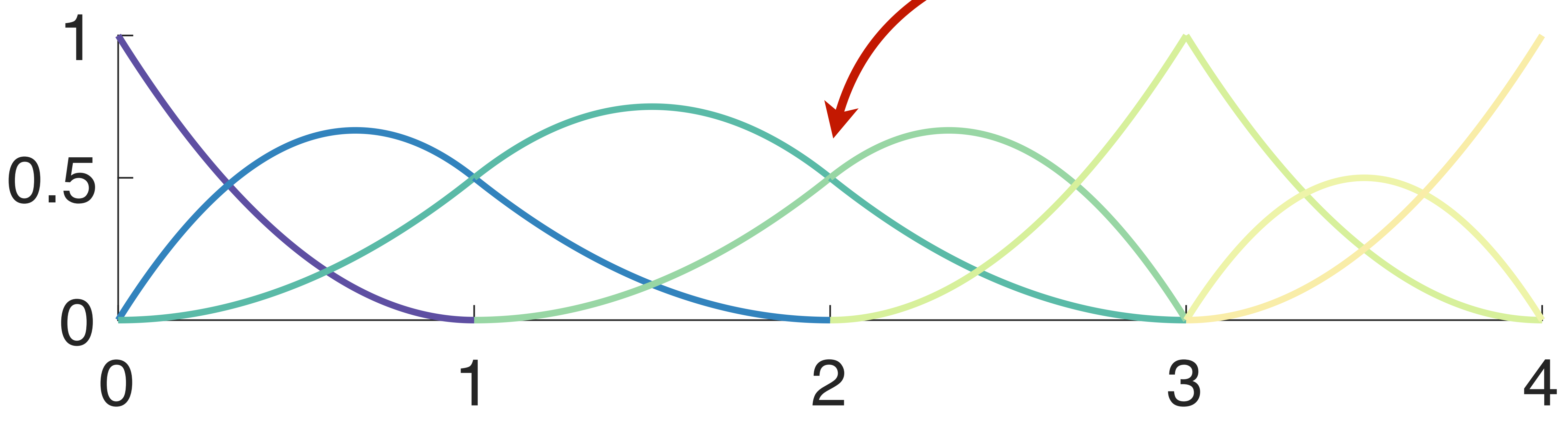
$$\Xi = \{0,0,0,1,2,3,3,4,4,4\}, \quad n = 7, \quad d = 2$$

B-spline basis functions [de Boor, 1971]

$$b^0_{i;\Xi}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$b^d_{i;\Xi}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i} b^{d-1}_{i;\Xi}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}} b^{d-1}_{i+1;\Xi}(\xi) \qquad ''\frac{0}{0}'' := 0$$

Univariate B-splines



C^{d-1}

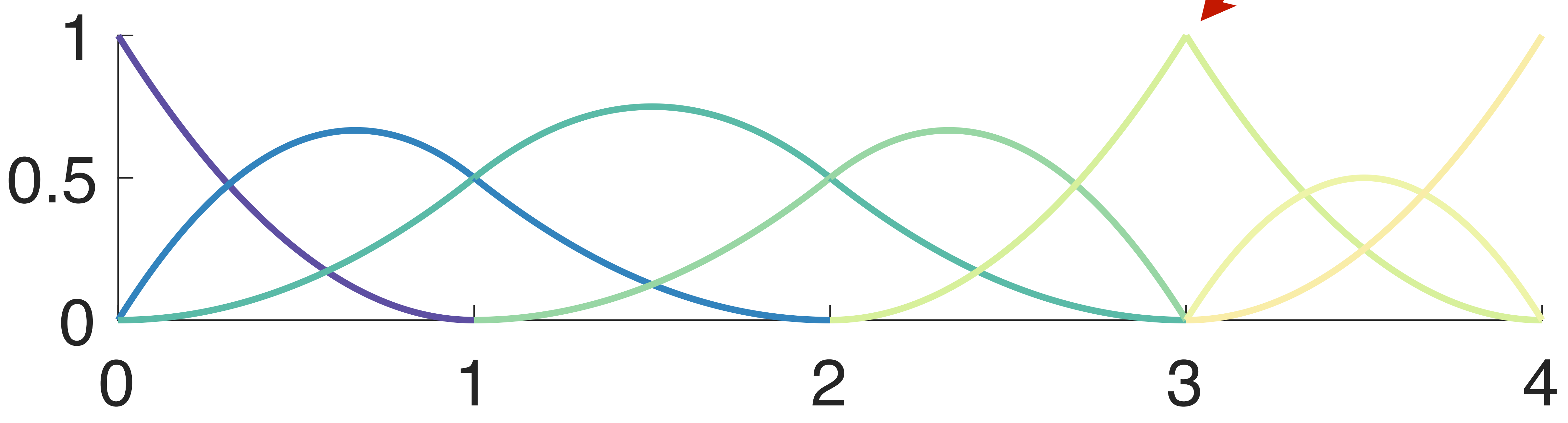
$$\Xi = \{0,0,0,1,\boxed{2},3,3,4,4,4\}, \quad n = 7, \quad d = 2$$

B-spline basis functions [de Boor, 1971]

$$b^0_{i;\Xi}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$b^d_{i;\Xi}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i} b^{d-1}_{i;\Xi}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}} b^{d-1}_{i+1;\Xi}(\xi) \qquad ''\frac{0}{0}'' := 0$$

Univariate B-splines



C^{d-2}

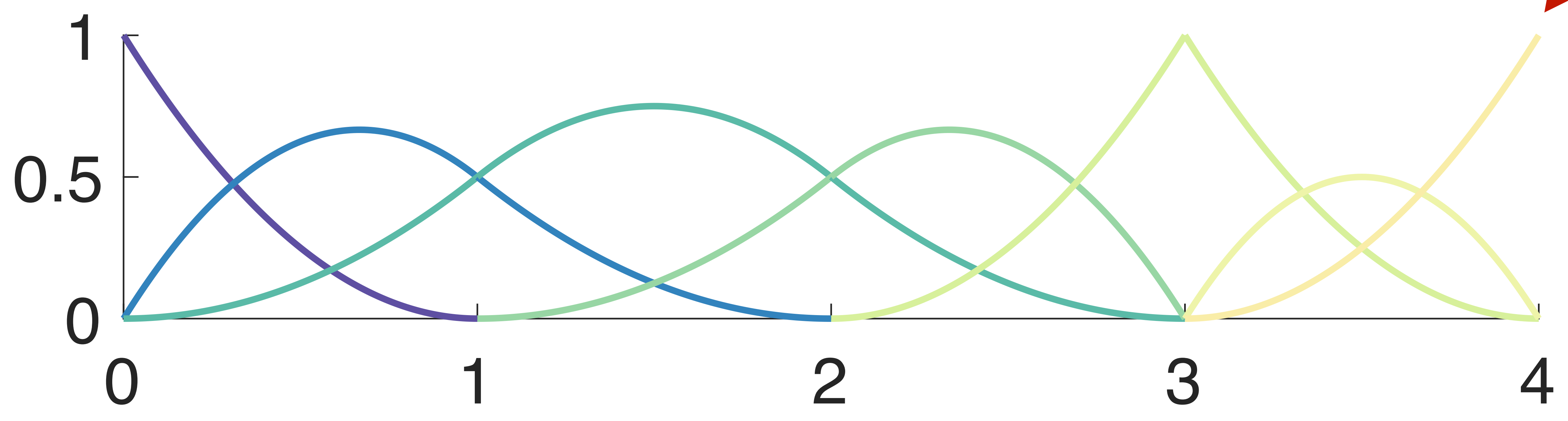
$$\Xi = \{0,0,0,1,2,\boxed{3,3},4,4,4\}, \quad n = 7, \quad d = 2$$

B-spline basis functions [de Boor, 1971]

$$b^0_{i;\Xi}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$b^d_{i;\Xi}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i} b^{d-1}_{i;\Xi}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}} b^{d-1}_{i+1;\Xi}(\xi) \qquad ''\frac{0}{0}'' := 0$$

Univariate B-splines



C^{d-3}

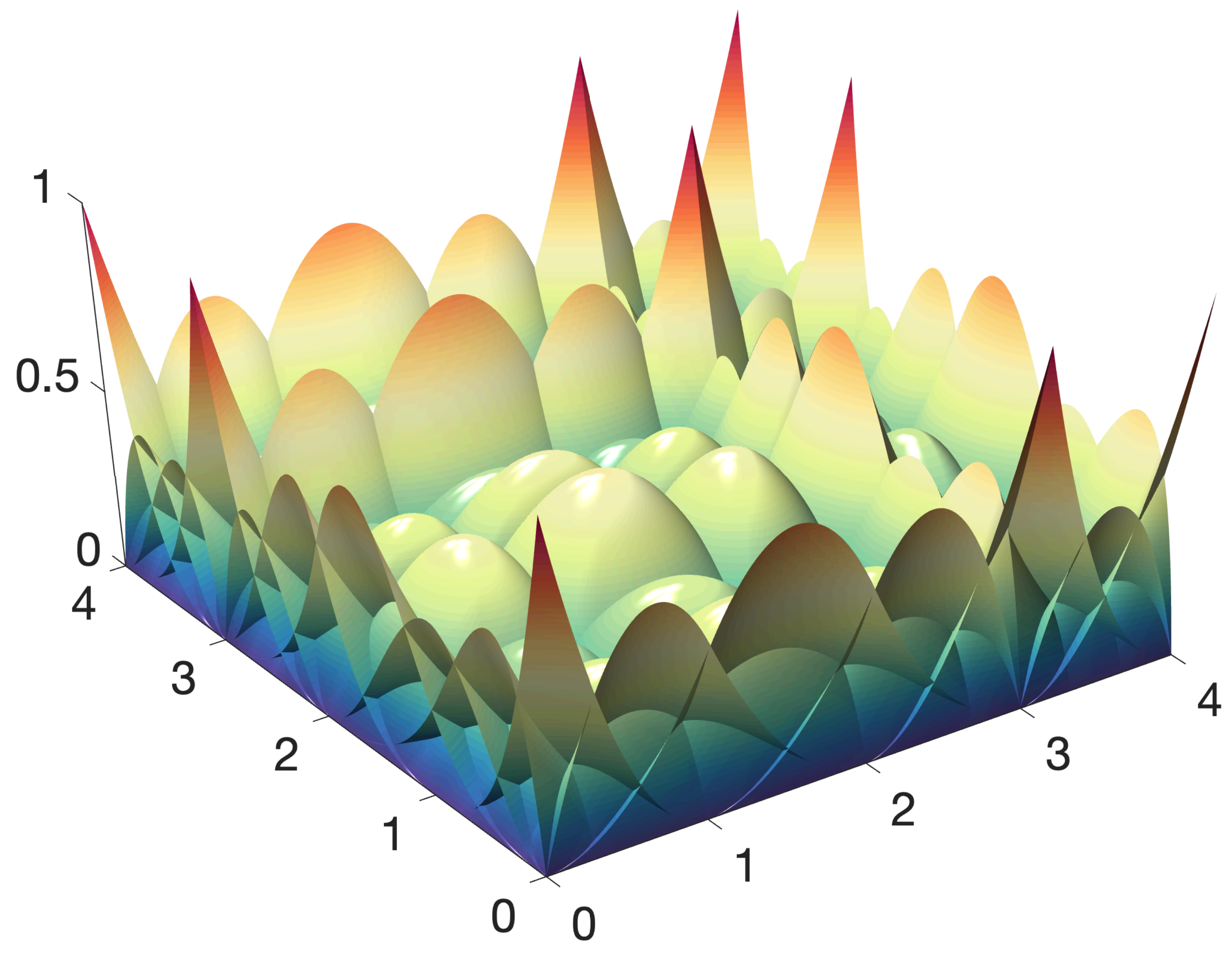
$$\Xi = \{0,0,0,1,2,3,3,\boxed{4,4,4}\}, \quad n = 7, \quad d = 2$$

B-spline basis functions [de Boor, 1971]

$$b^0_{i;\Xi}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$b^d_{i;\Xi}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i} b^{d-1}_{i;\Xi}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}} b^{d-1}_{i+1;\Xi}(\xi) \qquad ``\frac{0}{0}" := 0$$

Multivariate B-splines



Tensor-product basis functions

$$b^{\mathbf{d}}_{\mathbf{i};\boldsymbol{\Xi}}(\boldsymbol{\xi}) = \prod_{k=1}^{p} b^{d_k}_{i_k;\Xi_k}(\xi_k)$$

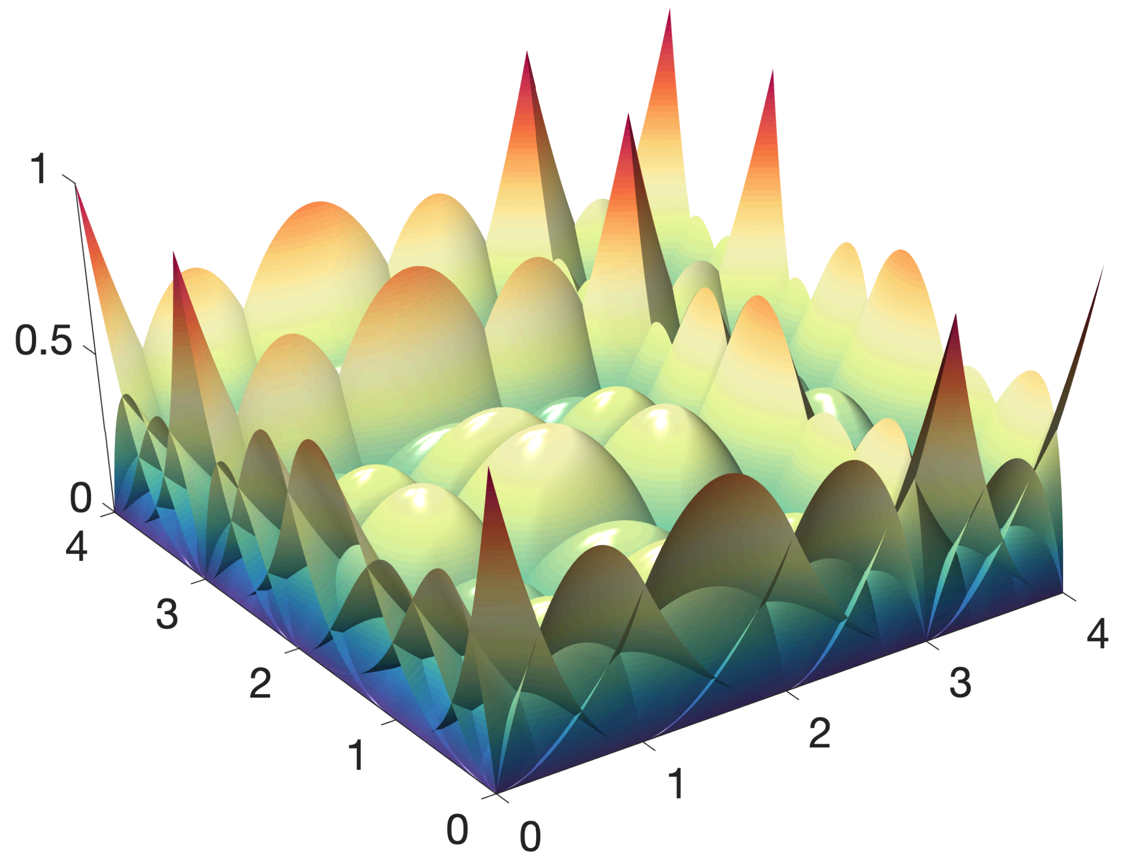
with multi-indices $\mathbf{i}$ and $\mathbf{d}$, $\boldsymbol{\Xi} = \left(\Xi_1, \ldots, \Xi_p\right)$ and parametric domain

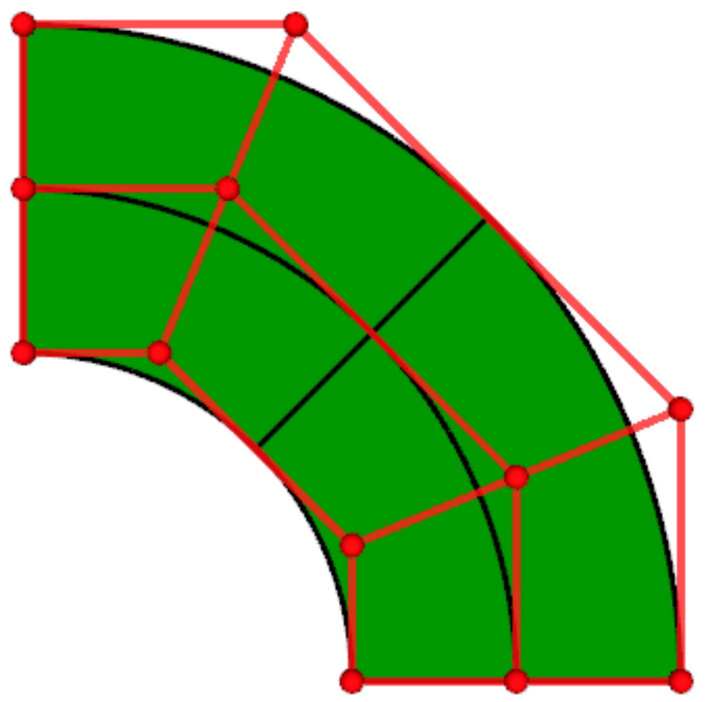
$$\hat{\Omega}_{\boldsymbol{\Xi}} = \bigotimes_{k=1}^{p} [\xi_{k,d_k+1}, \xi_{k,n_k})$$

Spline space

$$\mathbb{S}^{\mathbf{d},s}_{\boldsymbol{\Xi}} = \text{span}\left\{b^{\mathbf{d}}_{\mathbf{1};\boldsymbol{\Xi}}, \ldots, b^{\mathbf{d}}_{\mathbf{n};\boldsymbol{\Xi}}\right\} = \left\{\sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} c_{\mathbf{i}}\, b^{\mathbf{d}}_{\mathbf{i};\boldsymbol{\Xi}}(\boldsymbol{\xi}) \ : \ c_{\mathbf{i}} \in \mathbb{R}^s, \text{for } \mathbf{1} \leq \mathbf{i} \leq \mathbf{n}, \boldsymbol{\xi} \in \hat{\Omega}_{\boldsymbol{\Xi}}\right\}$$

Isogeometric analysis in a nutshell

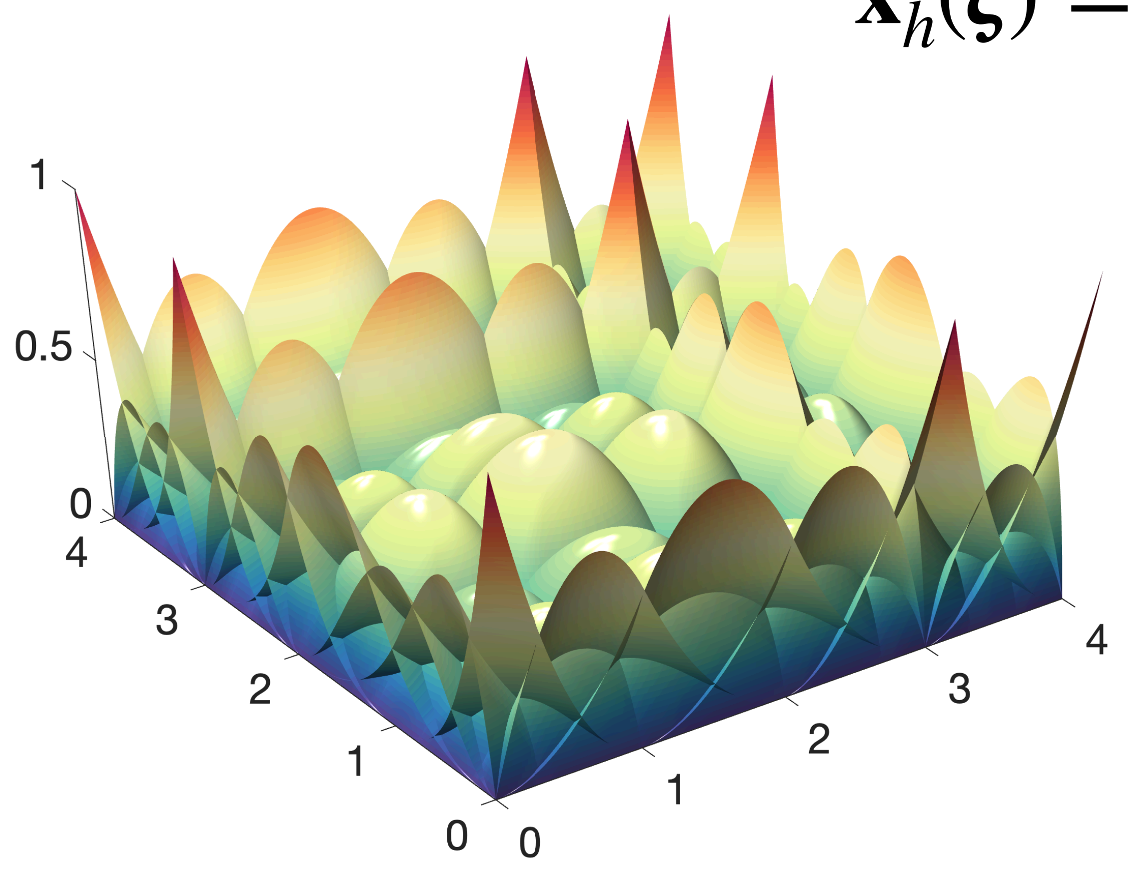


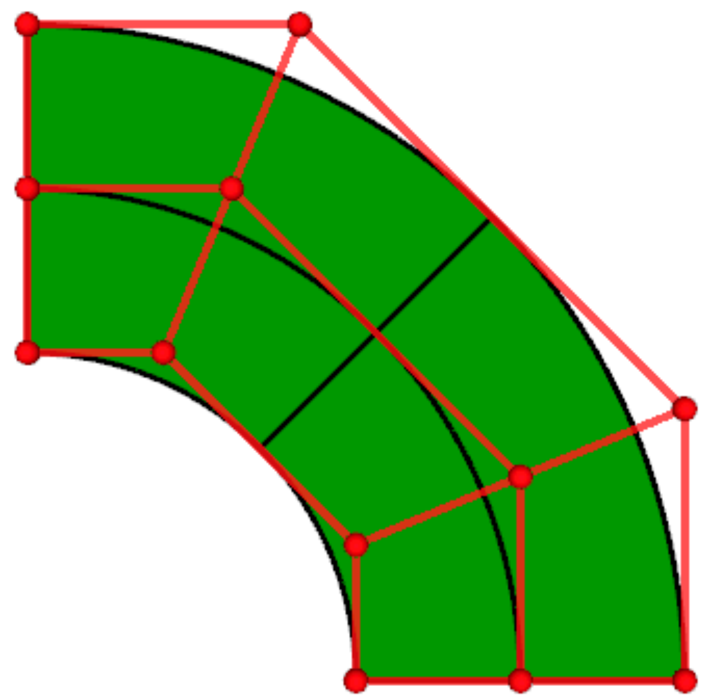


Isogeometric analysis in a nutshell

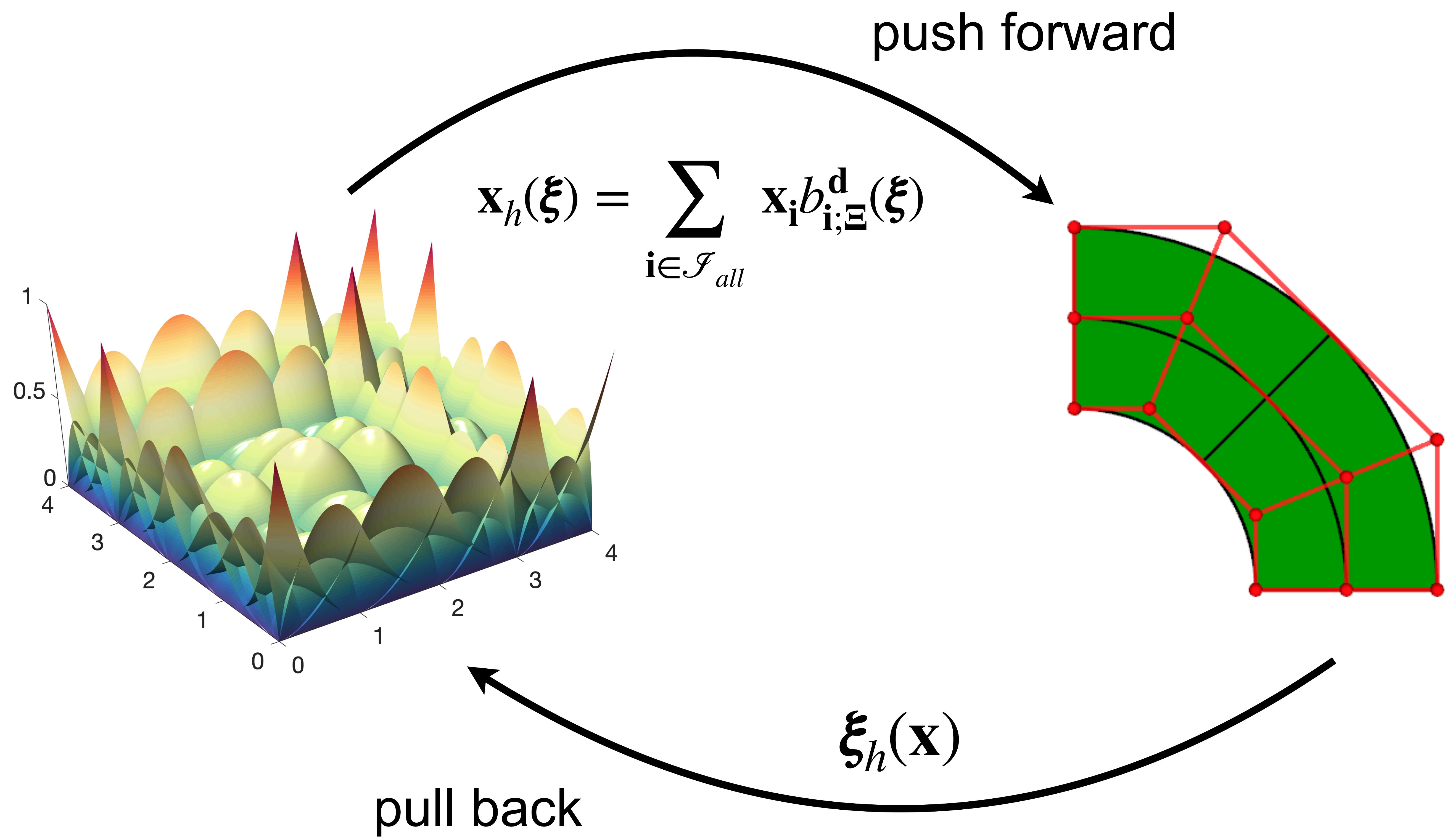
push forward

$$\mathbf{x}_h(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in\mathscr{I}_{all}} \mathbf{x}_{\mathbf{i}} b^{\mathbf{d}}_{\mathbf{i};\Xi}(\boldsymbol{\xi})$$

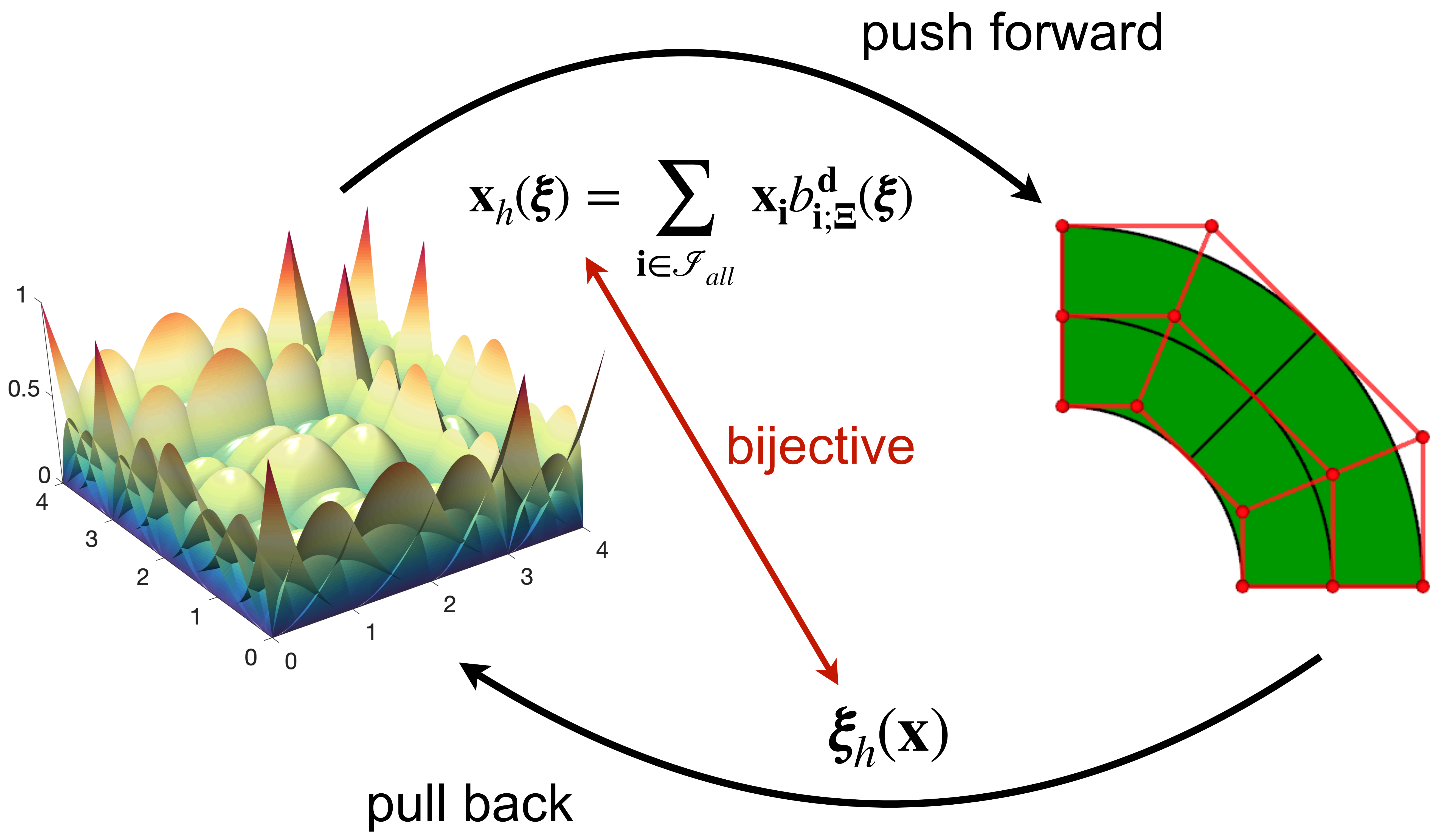




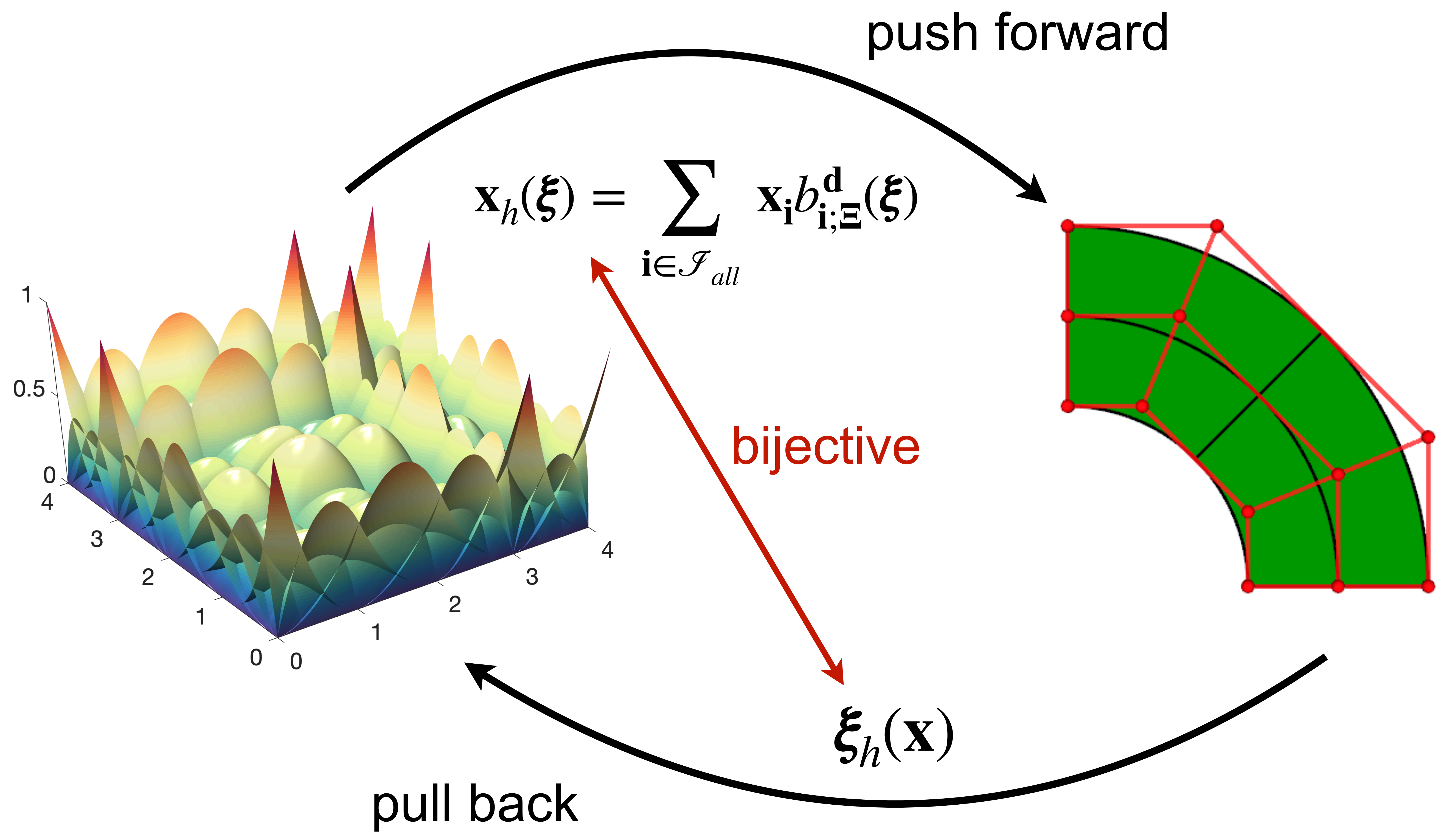
Isogeometric analysis in a nutshell

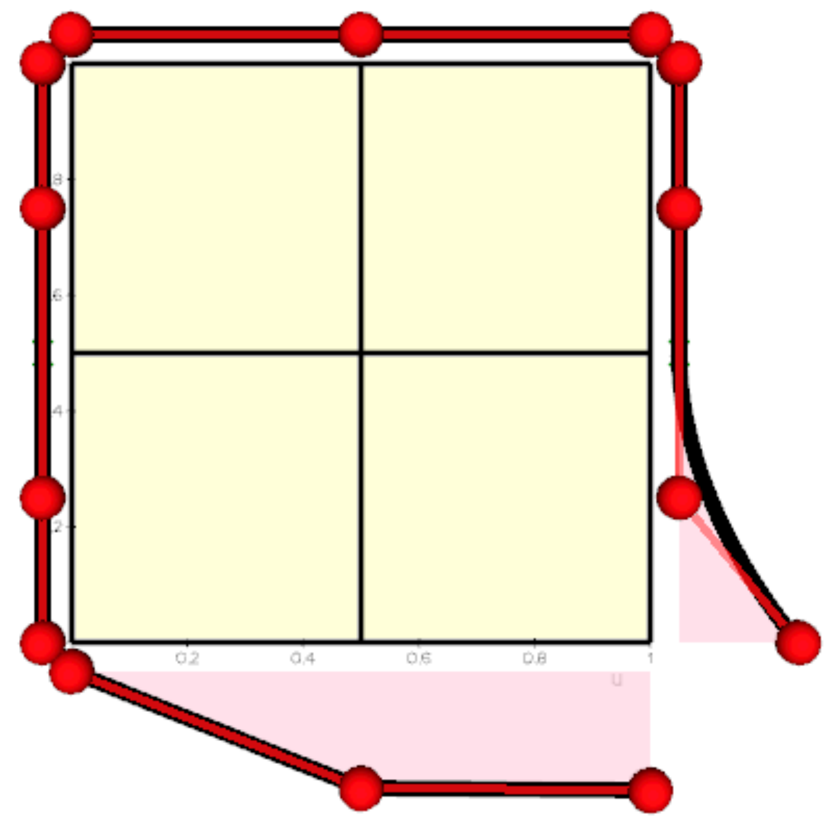


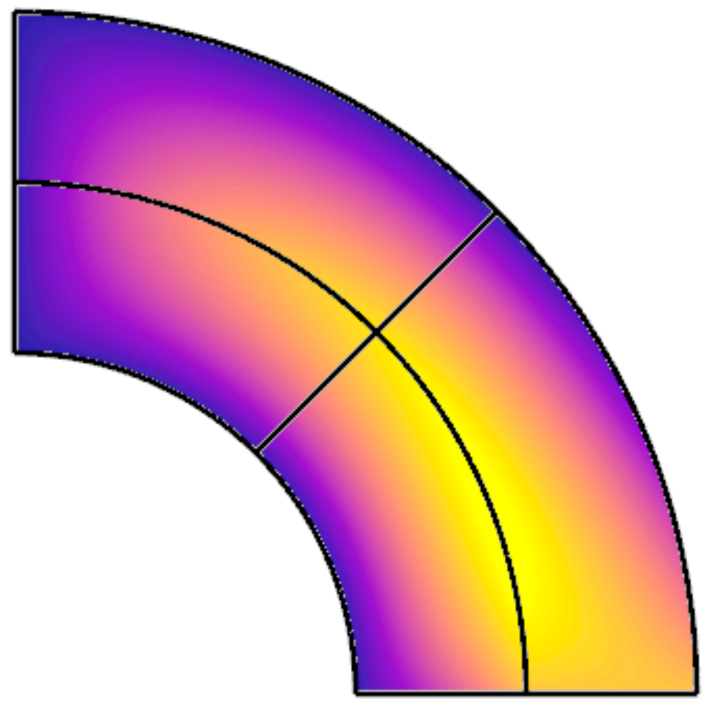
Isogeometric analysis in a nutshell



Isogeometric analysis in a nutshell

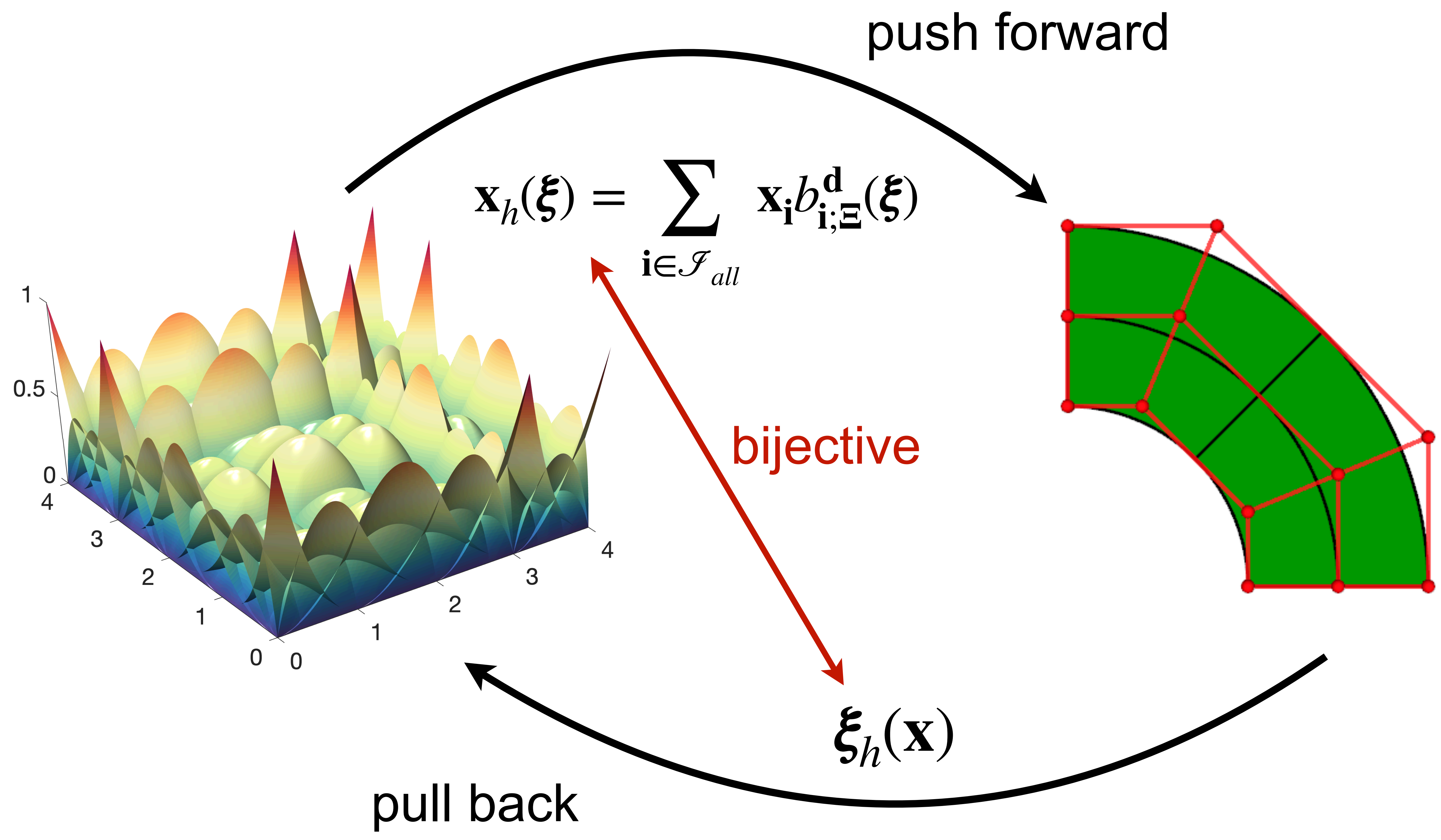


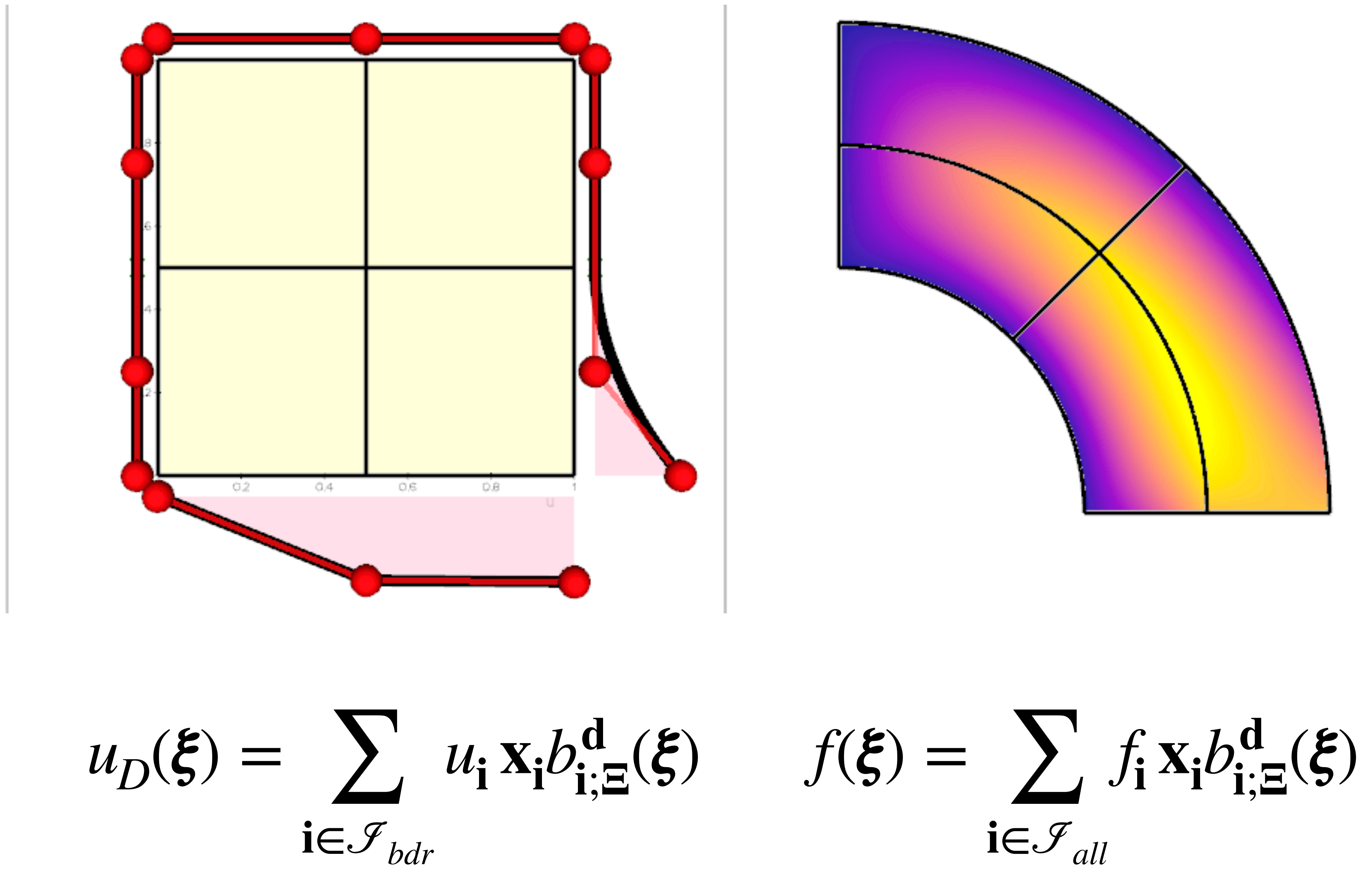




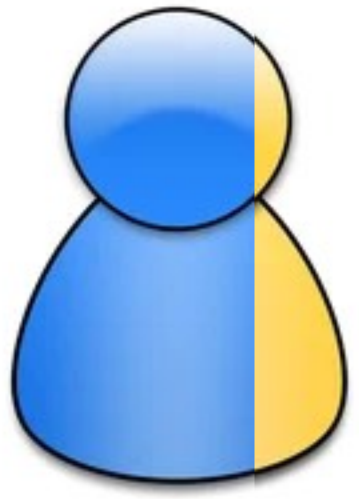
$$u_D(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in\mathcal{I}_{bdr}} u_{\mathbf{i}}\, \mathbf{x}_{\mathbf{i}} b^{\mathbf{d}}_{\mathbf{i};\Xi}(\boldsymbol{\xi}) \qquad f(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in\mathcal{I}_{all}} f_{\mathbf{i}}\, \mathbf{x}_{\mathbf{i}} b^{\mathbf{d}}_{\mathbf{i};\Xi}(\boldsymbol{\xi})$$

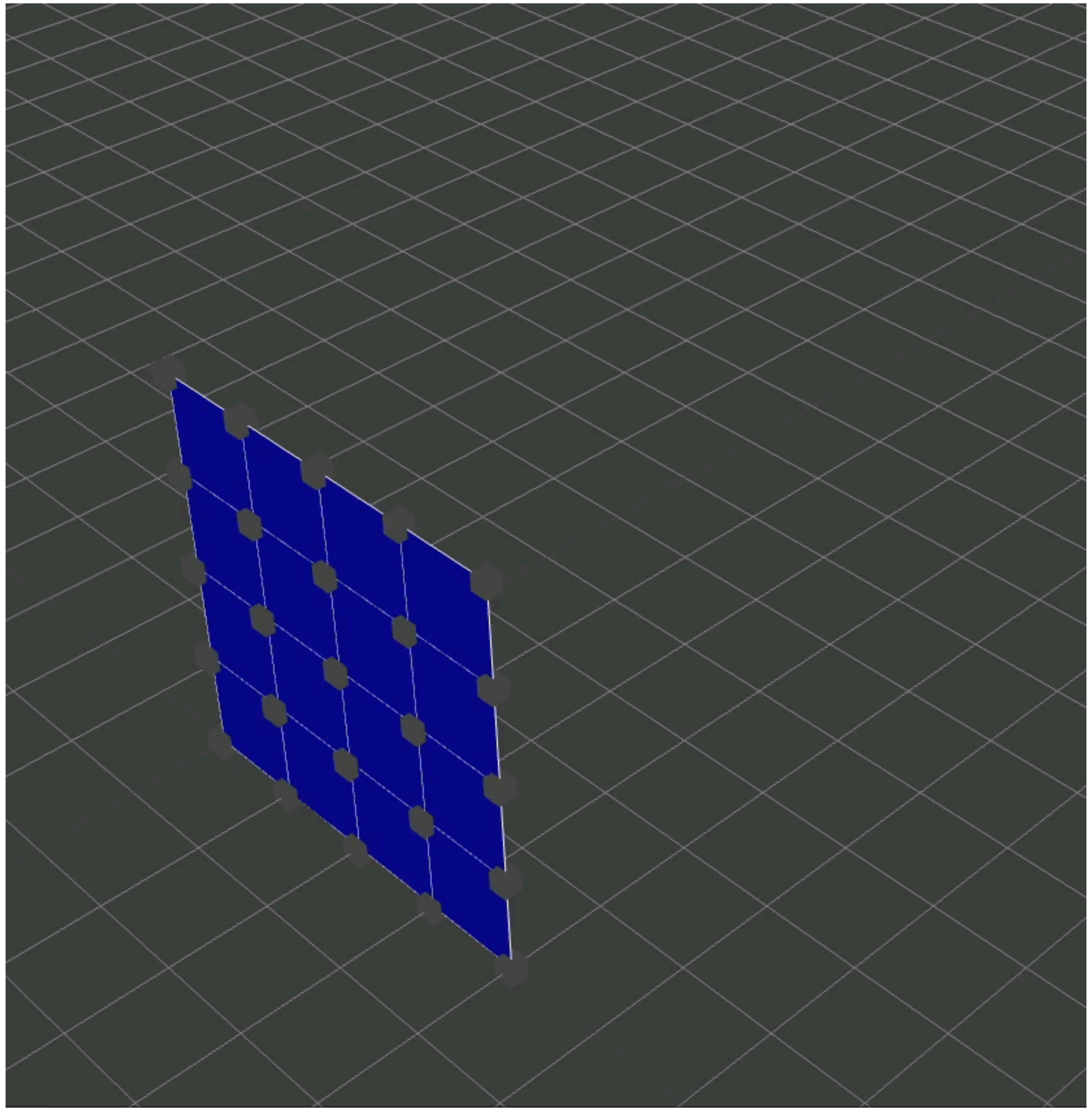
Isogeometric analysis in a nutshell

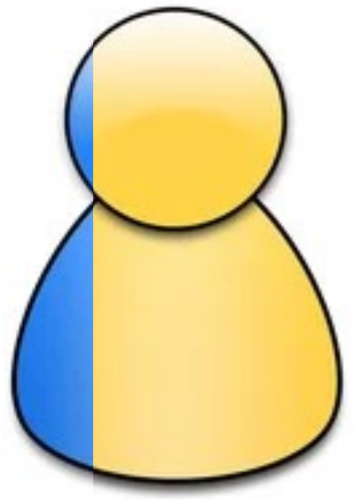




Compute solution $u(\boldsymbol{\xi}) = u_D(\boldsymbol{\xi}) + \sum_{\mathbf{i}\in\mathcal{I}_{int}} u_{\mathbf{i}}\, \mathbf{x}_{\mathbf{i}} b^{\mathbf{d}}_{\mathbf{i};\Xi}(\boldsymbol{\xi})$ using Galerkin or collocation IGA

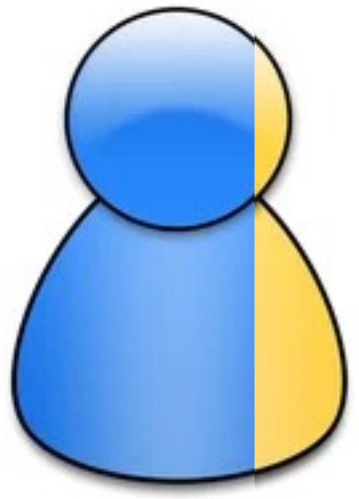




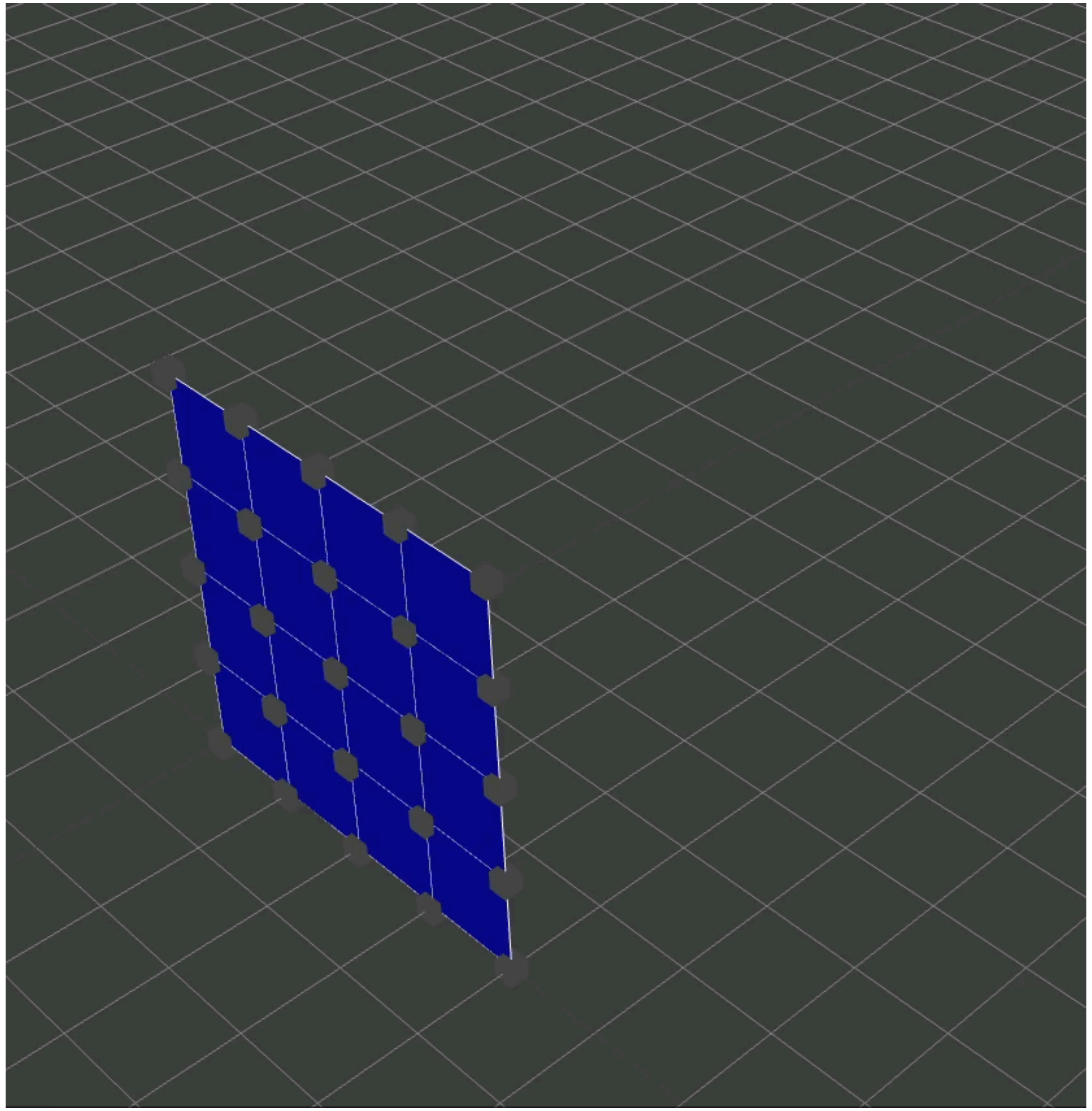


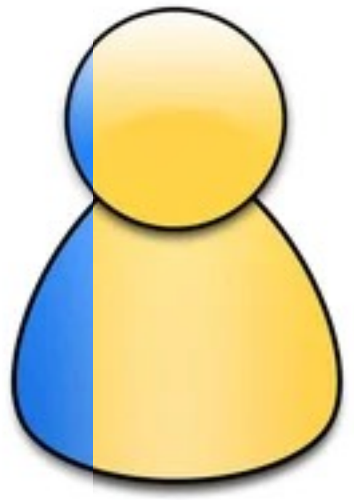
CAD

FEA

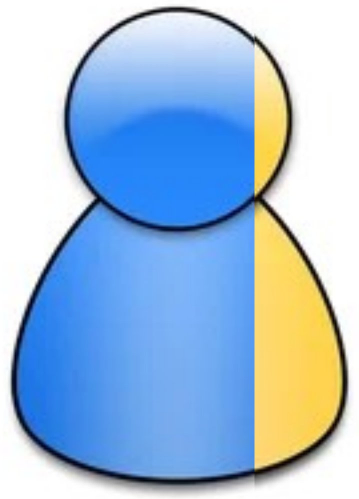


CAD

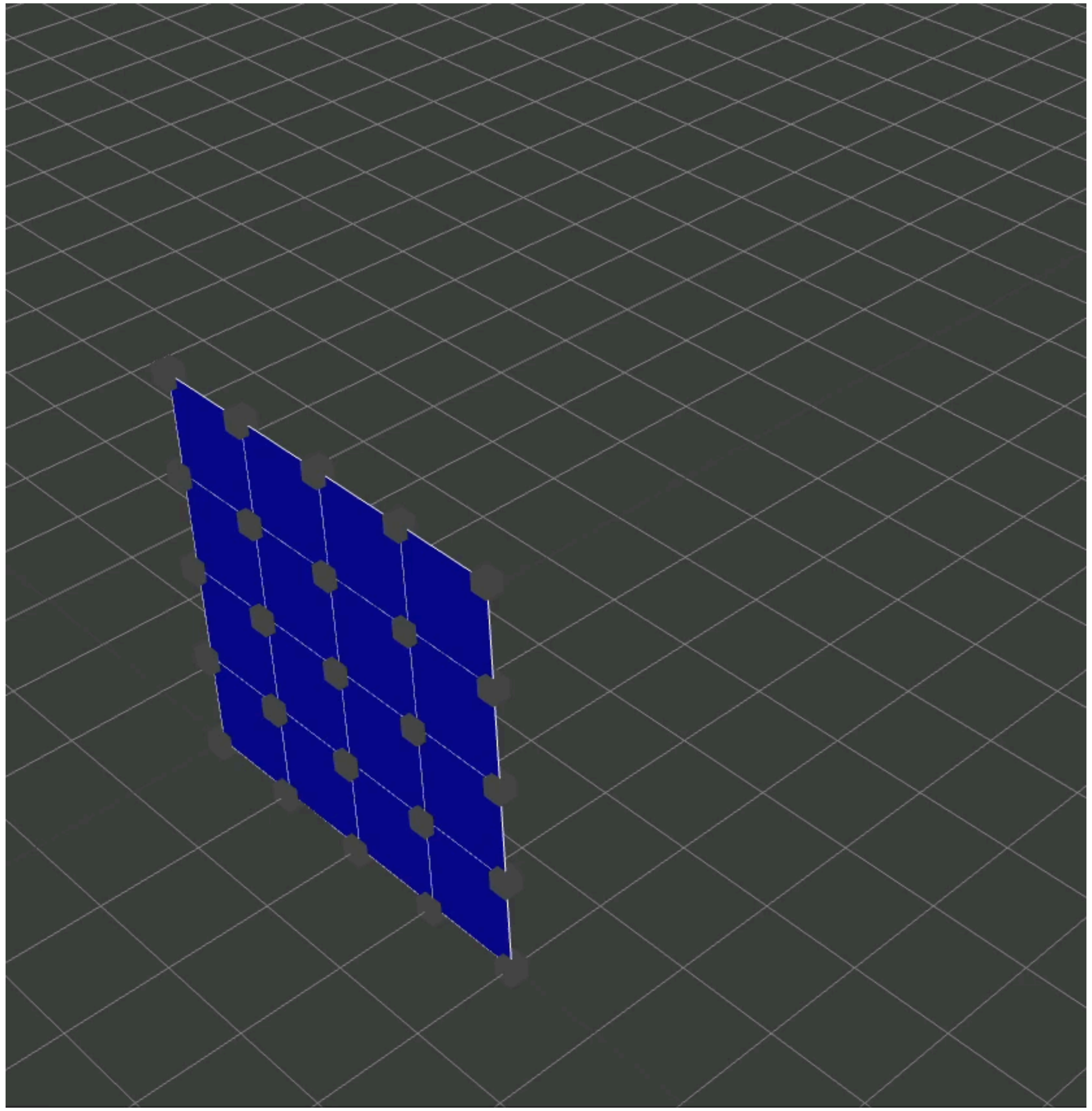




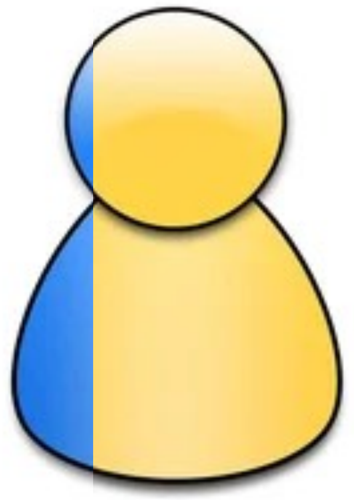
FEA



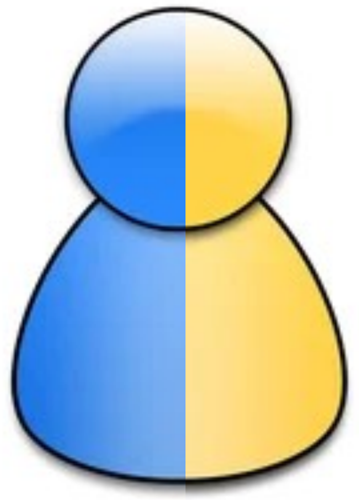
CAD



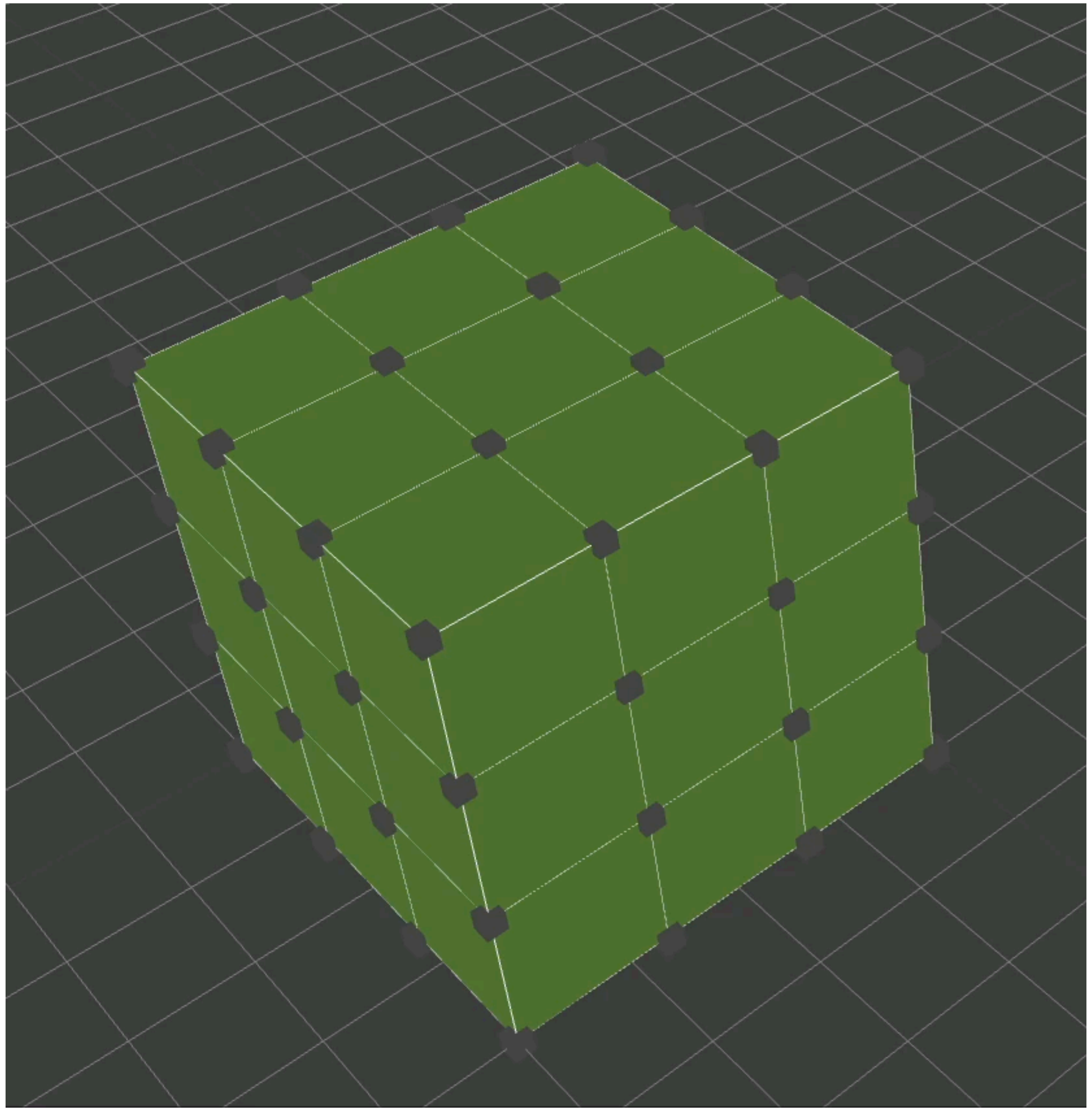
Automatic reparameterization of surfaces!



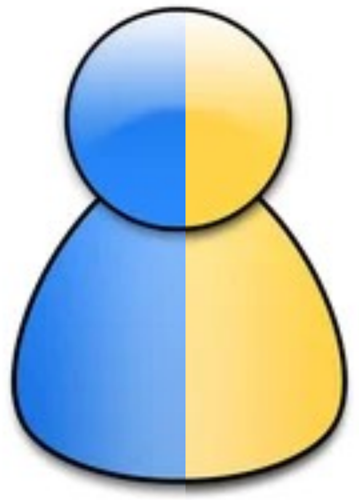
FEA

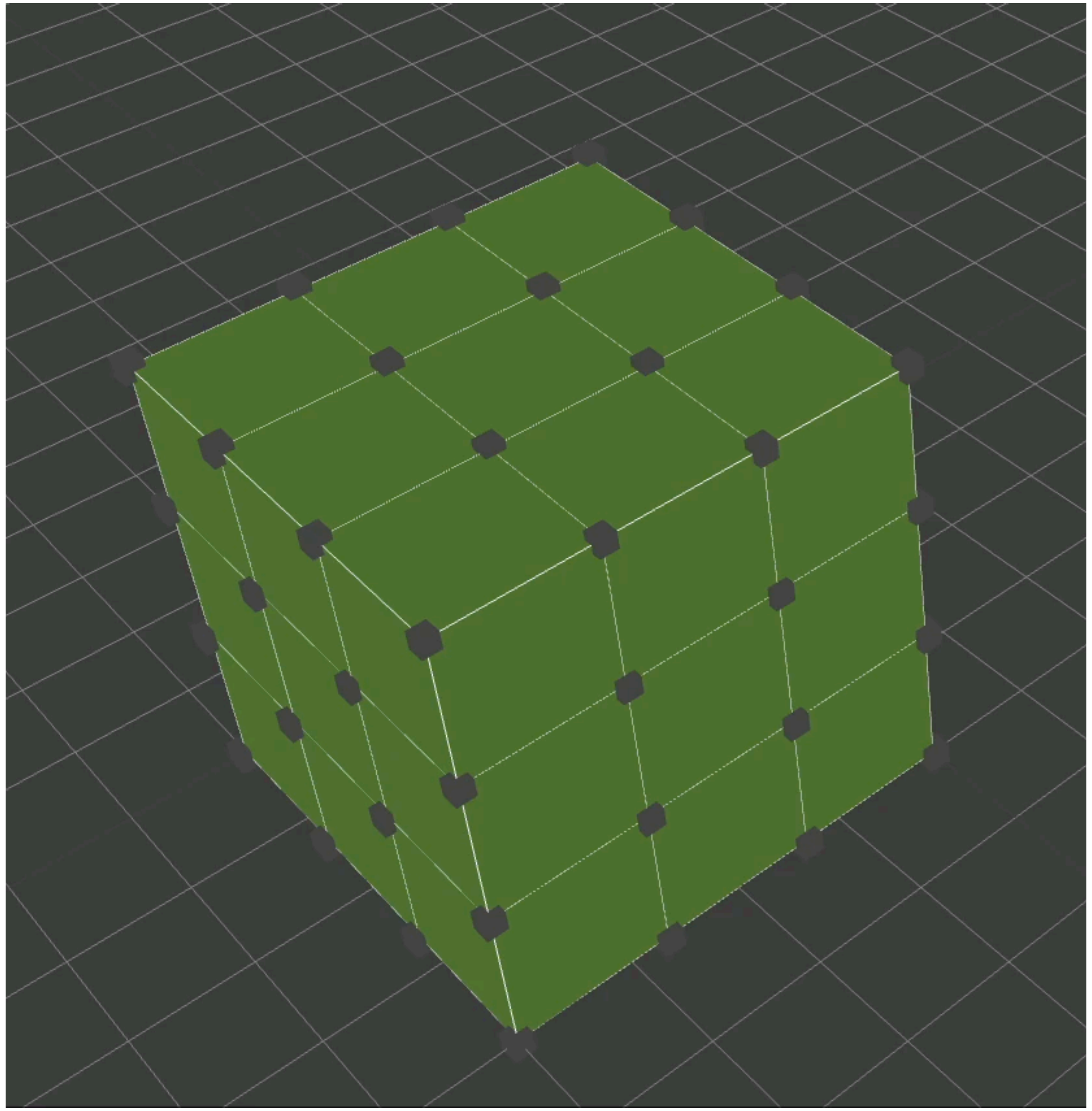


CAD



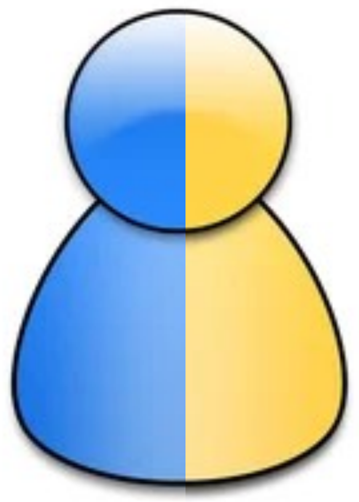
FEA



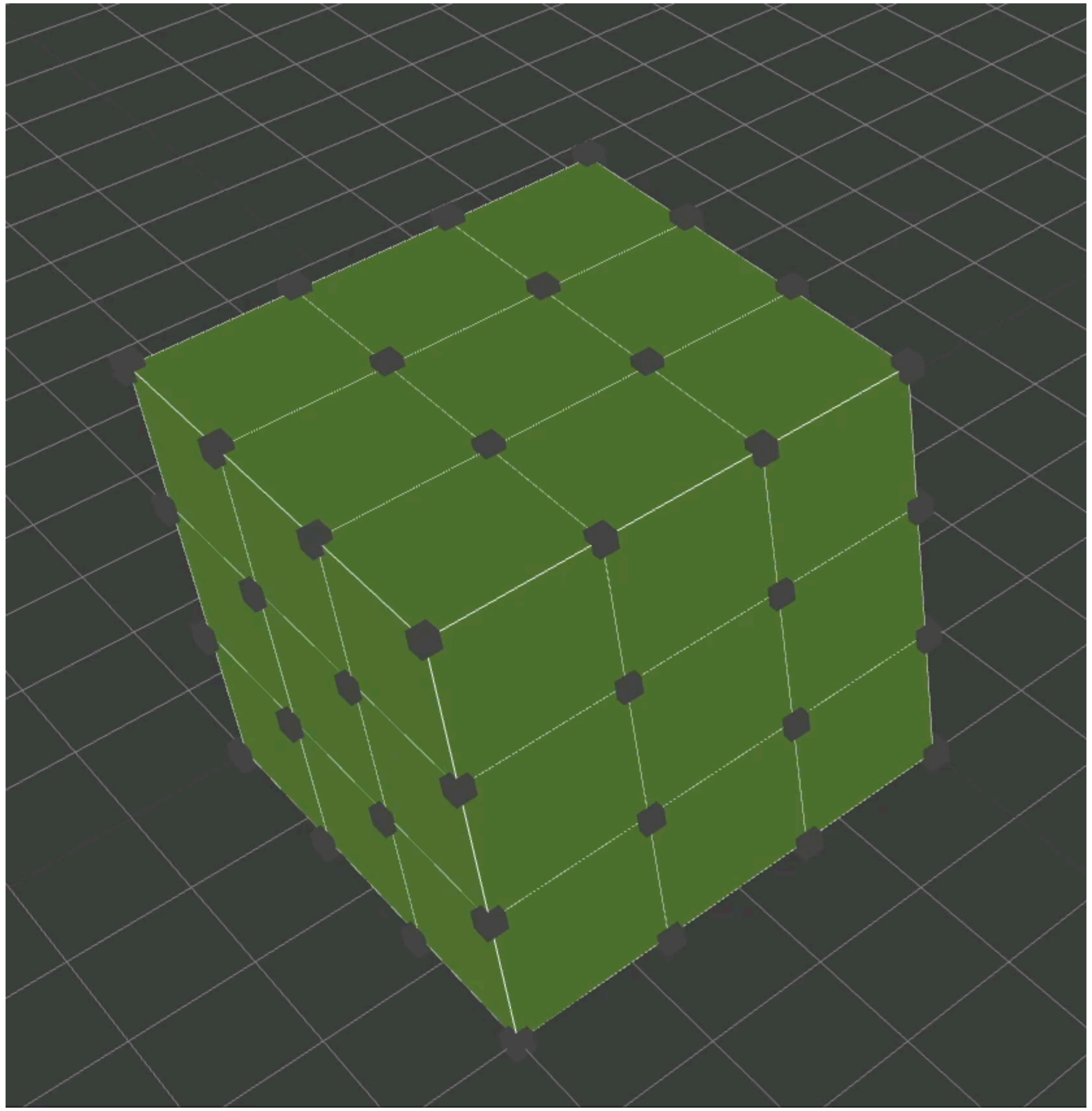


CAD

FEA



CAD



Automatic placement of interior control points!

FEA

Creation of analysis-suitable parameterizations

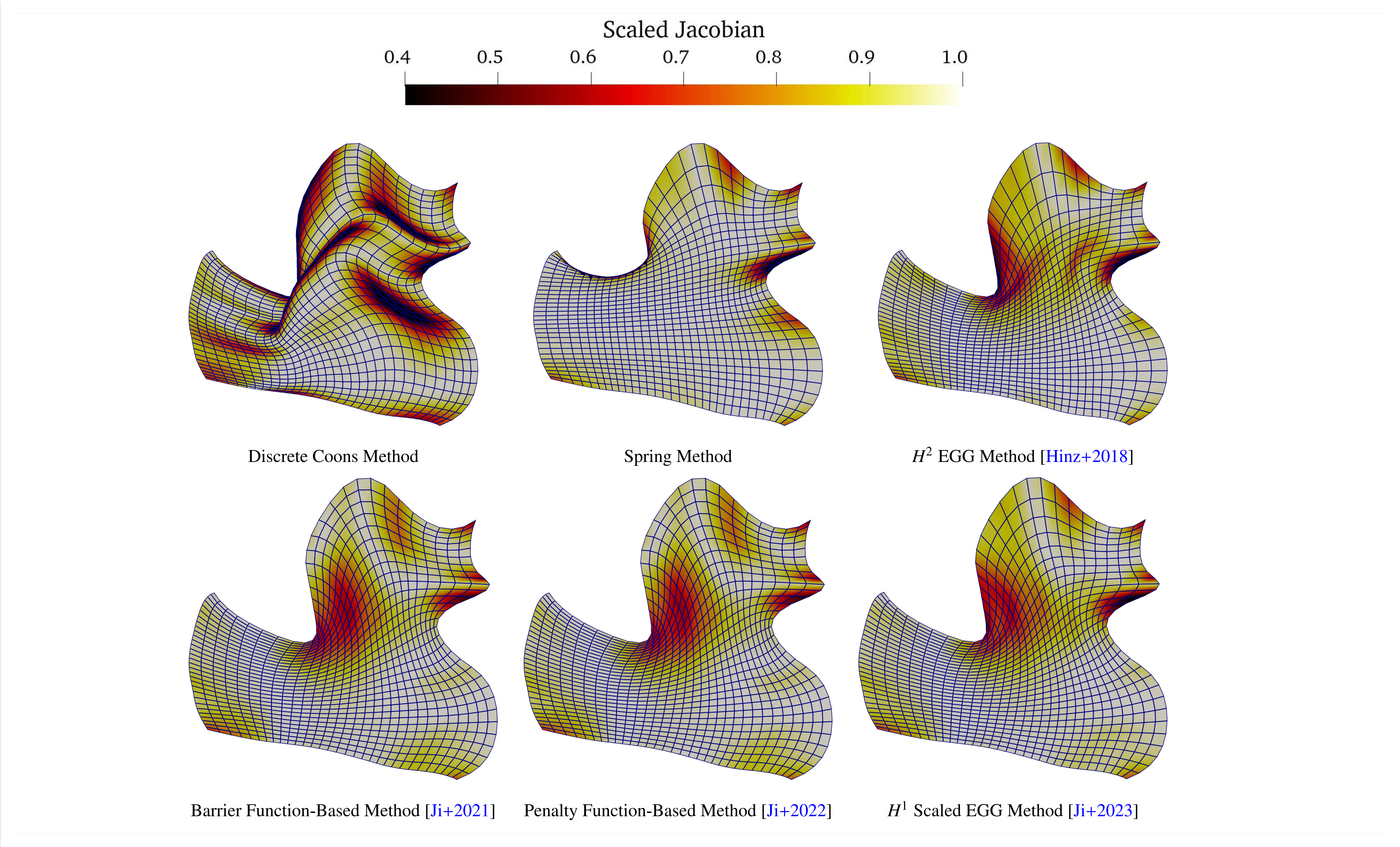
Algebraic methods

- Coons patch [Farin et al. 1999]
- spring patch [Gravesen et al. 2012], …

Nonlinear methods

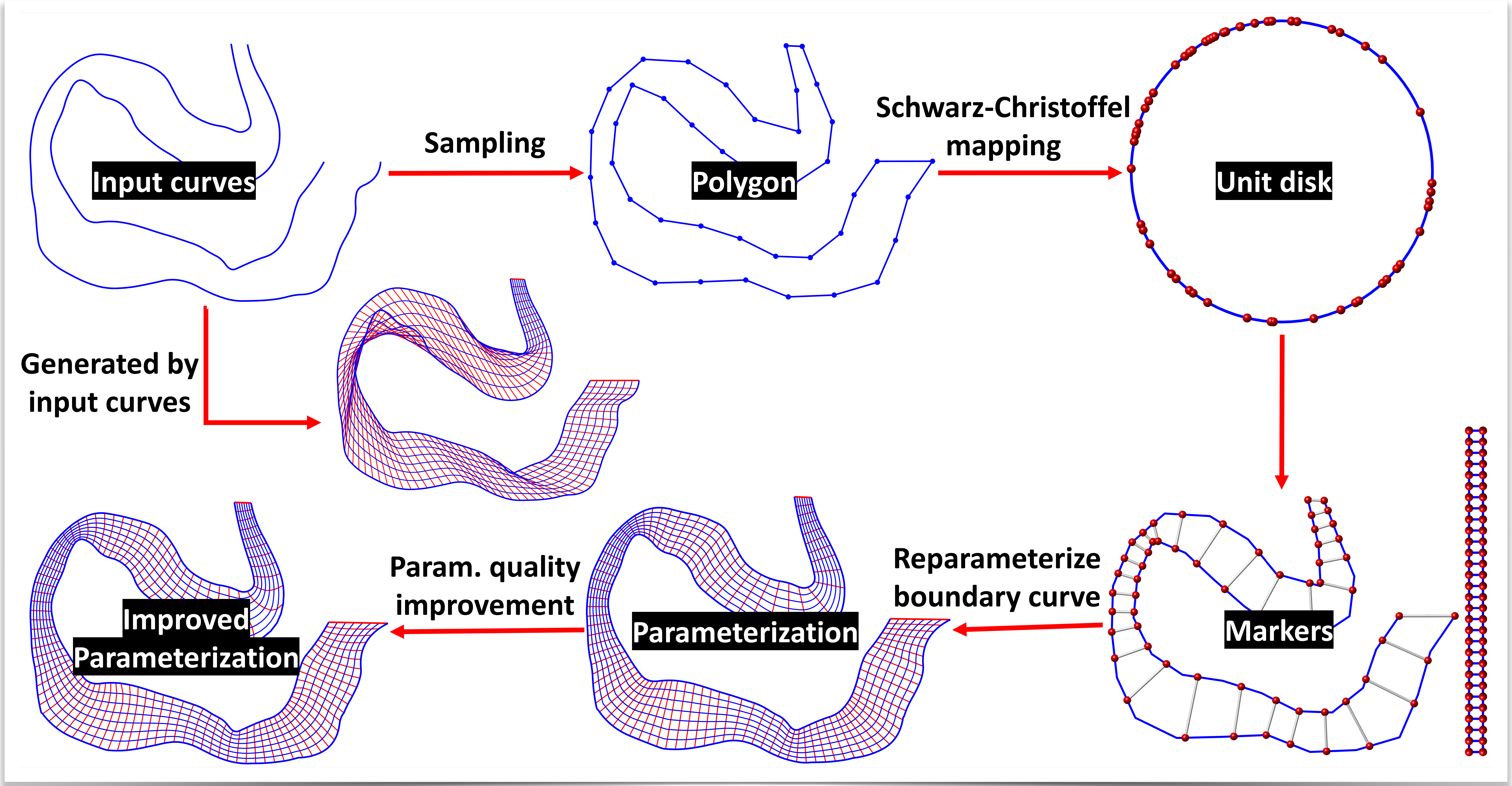
- constrained optimization [Wang et al. 2014, Pan et al. 2020],
- unconstrained optimization: barrier functions [Ji et al. 2021], penalty functions [Wang 2021, Ji 2022]
- Teichmüller mappings [Chen et al. 2016], low-rank quasi-conformal mappings [Pan et al. 2024], harmonic mappings [Martin et al. 2009, Nguyen et al. 2010, Xu et al. 2013, Falini et al. 2015]
- PDE-based methods [Hinz et al. 2018 & 2020, Ji et al. 2023]
- …

Comparison of methods



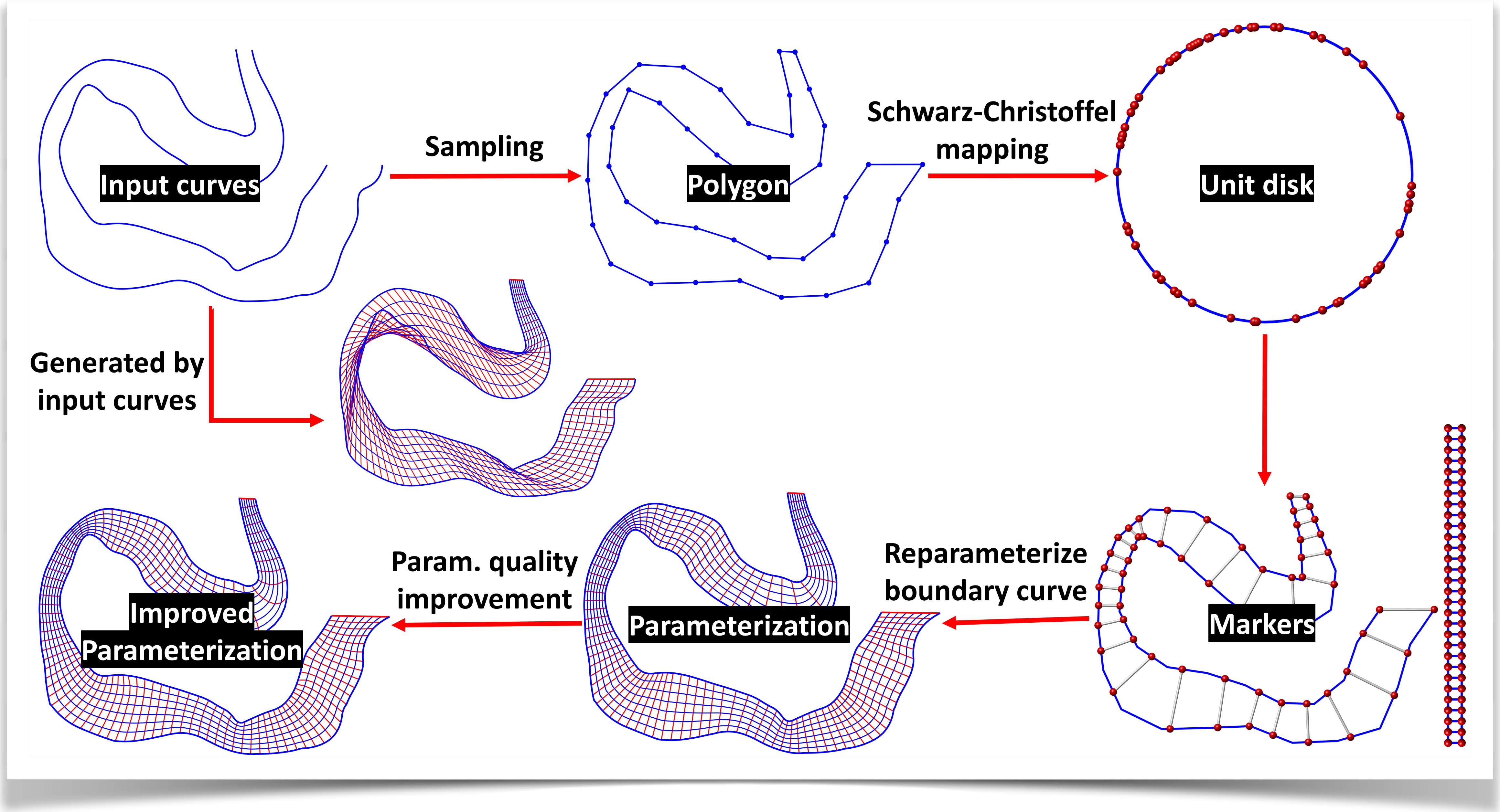
Ye Ji

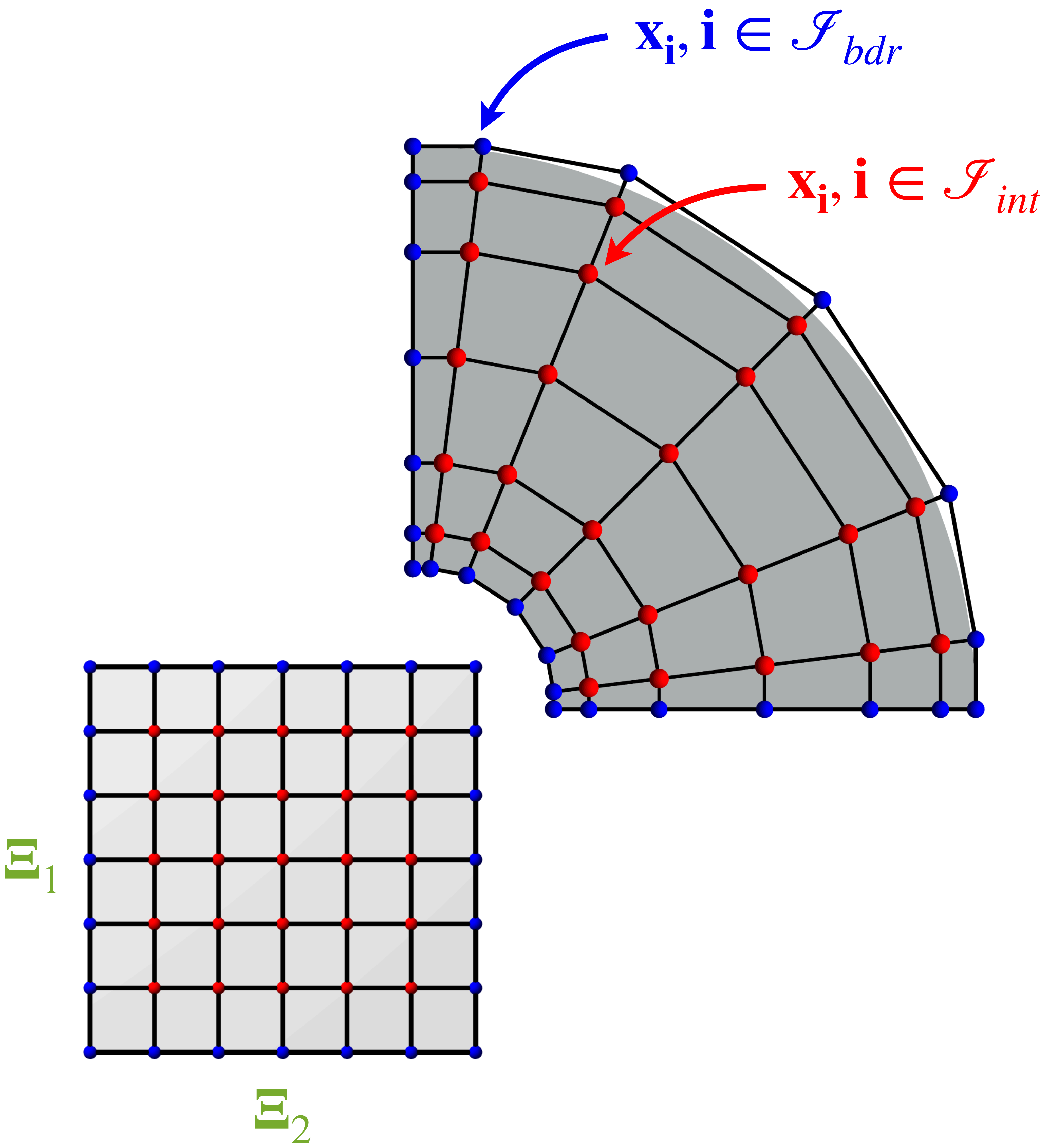
Workflow for planar domains



Ye Ji, Wed 10:30 - 12:30 in #MS012A

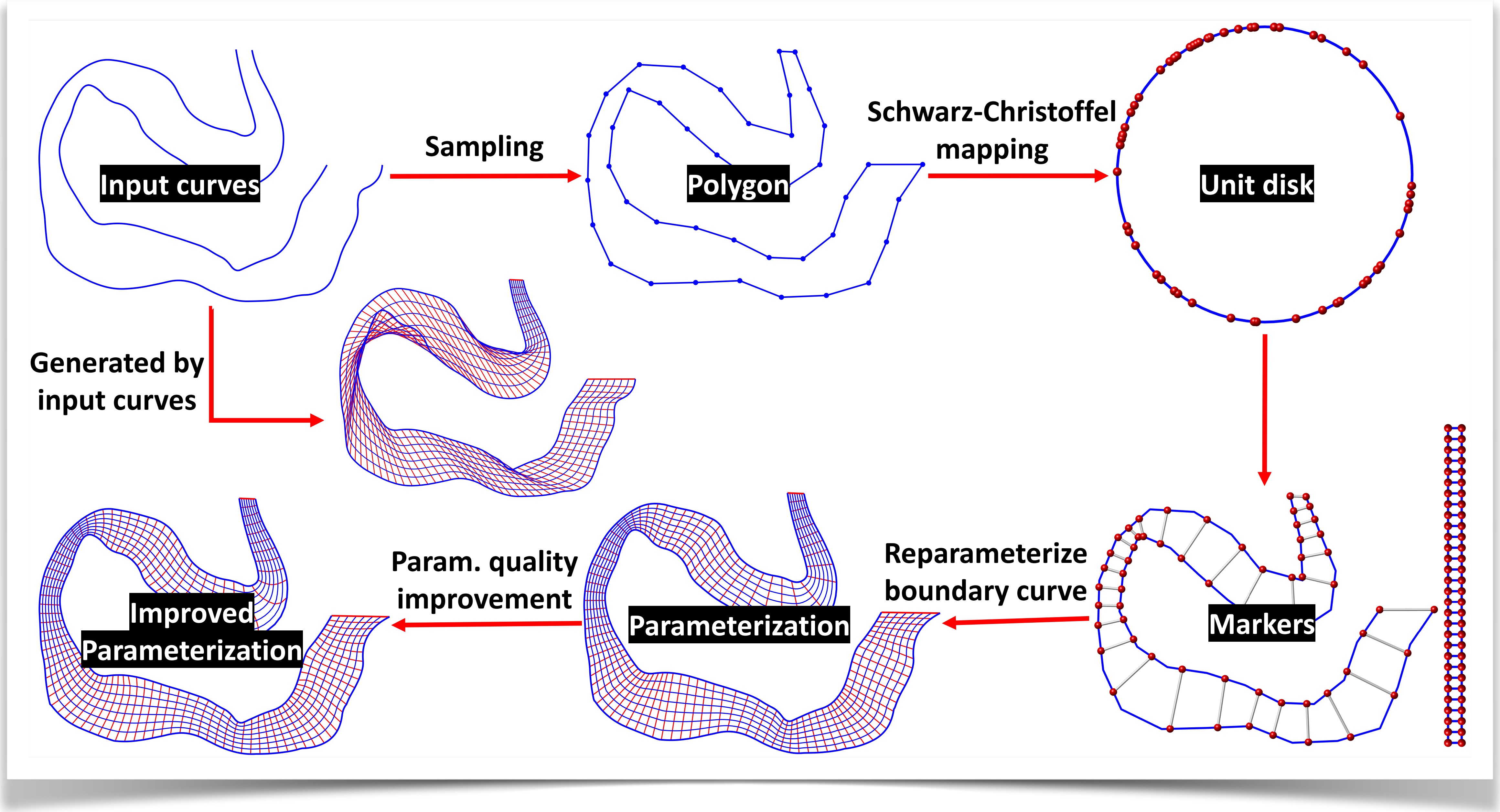
Workflow for planar domains

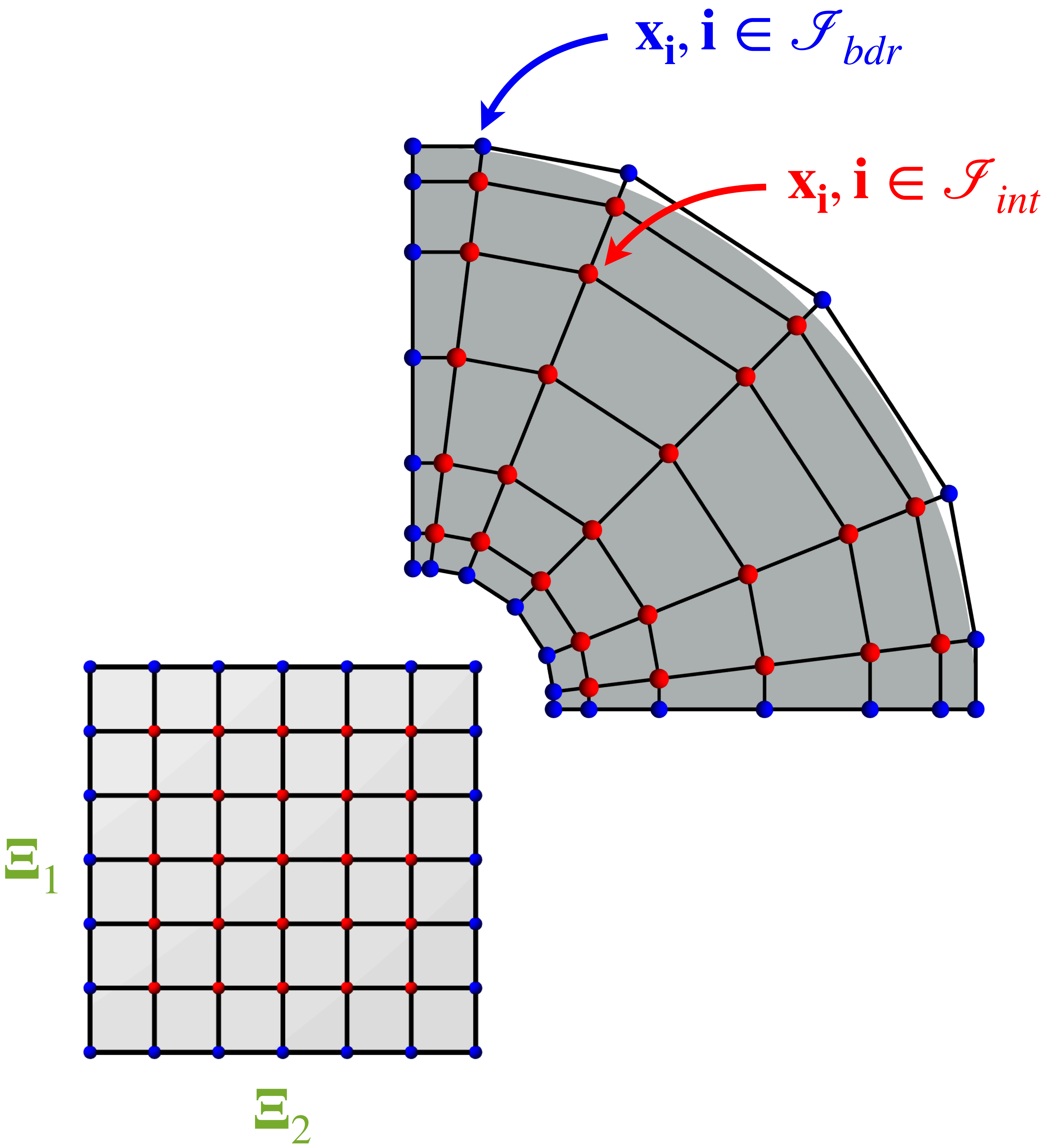




Ye Ji, Wed 10:30 - 12:30 in #MS012A

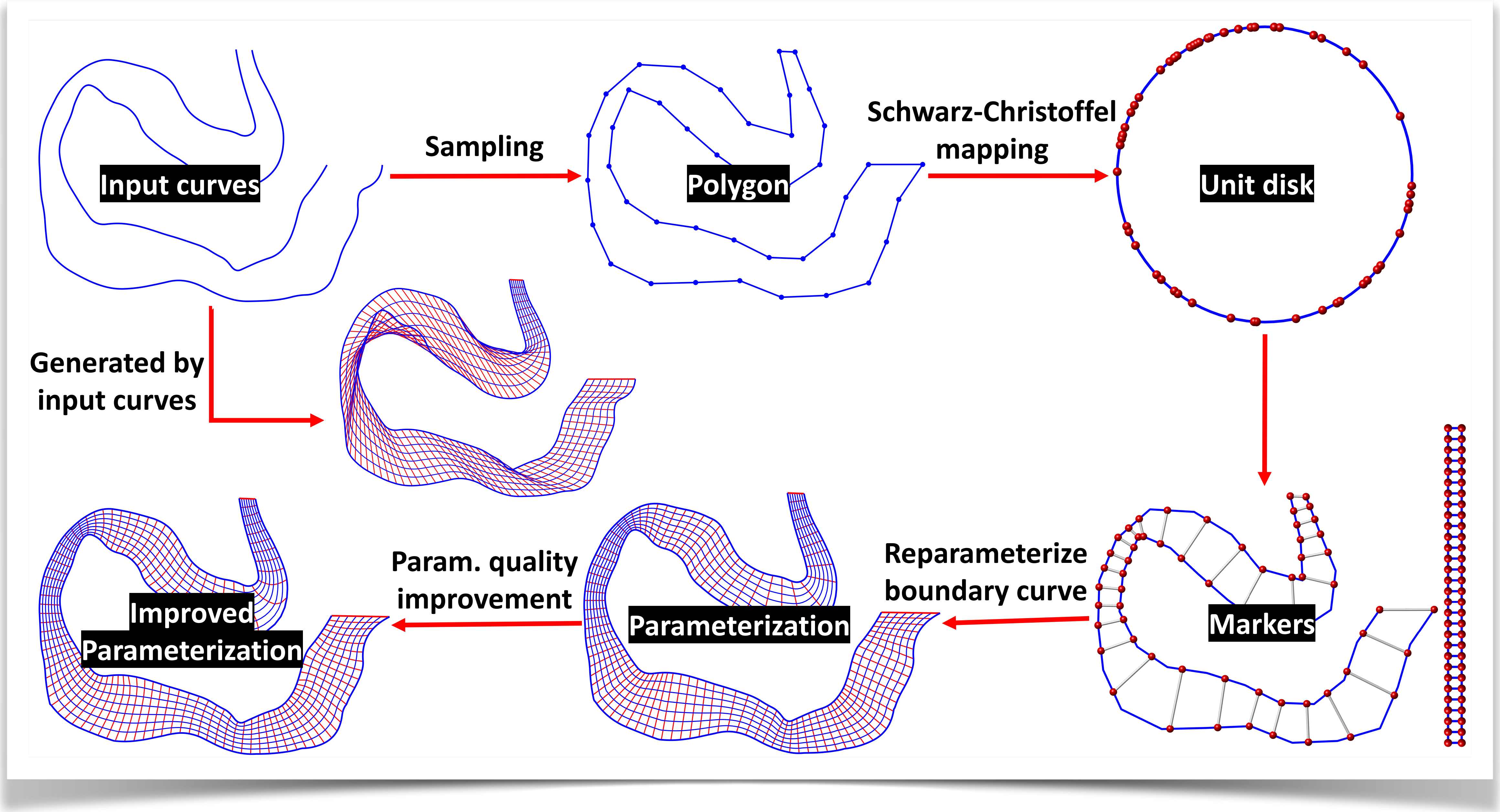
Workflow for planar domains

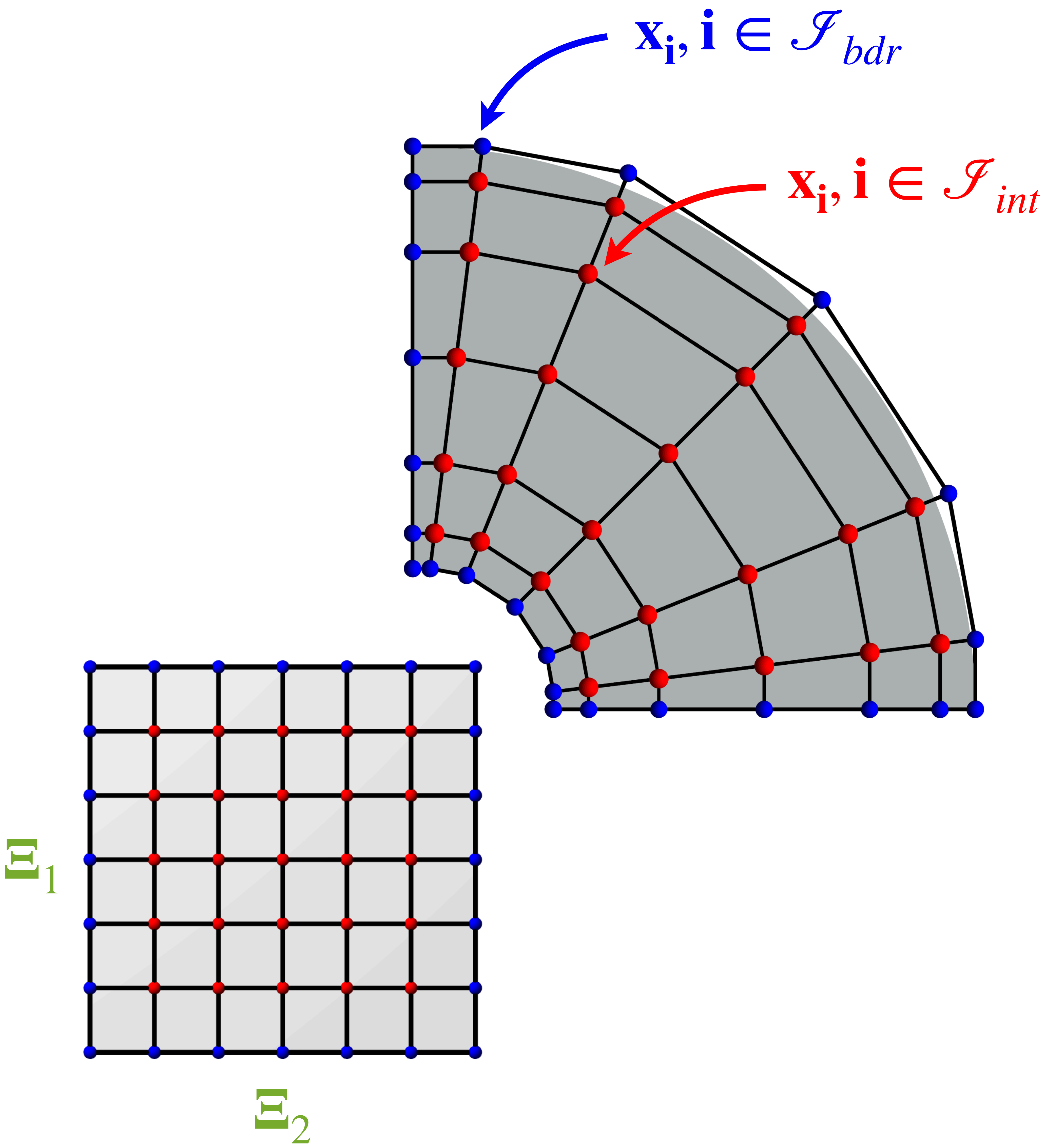




Ye Ji, Wed 10:30 - 12:30 in #MS012A

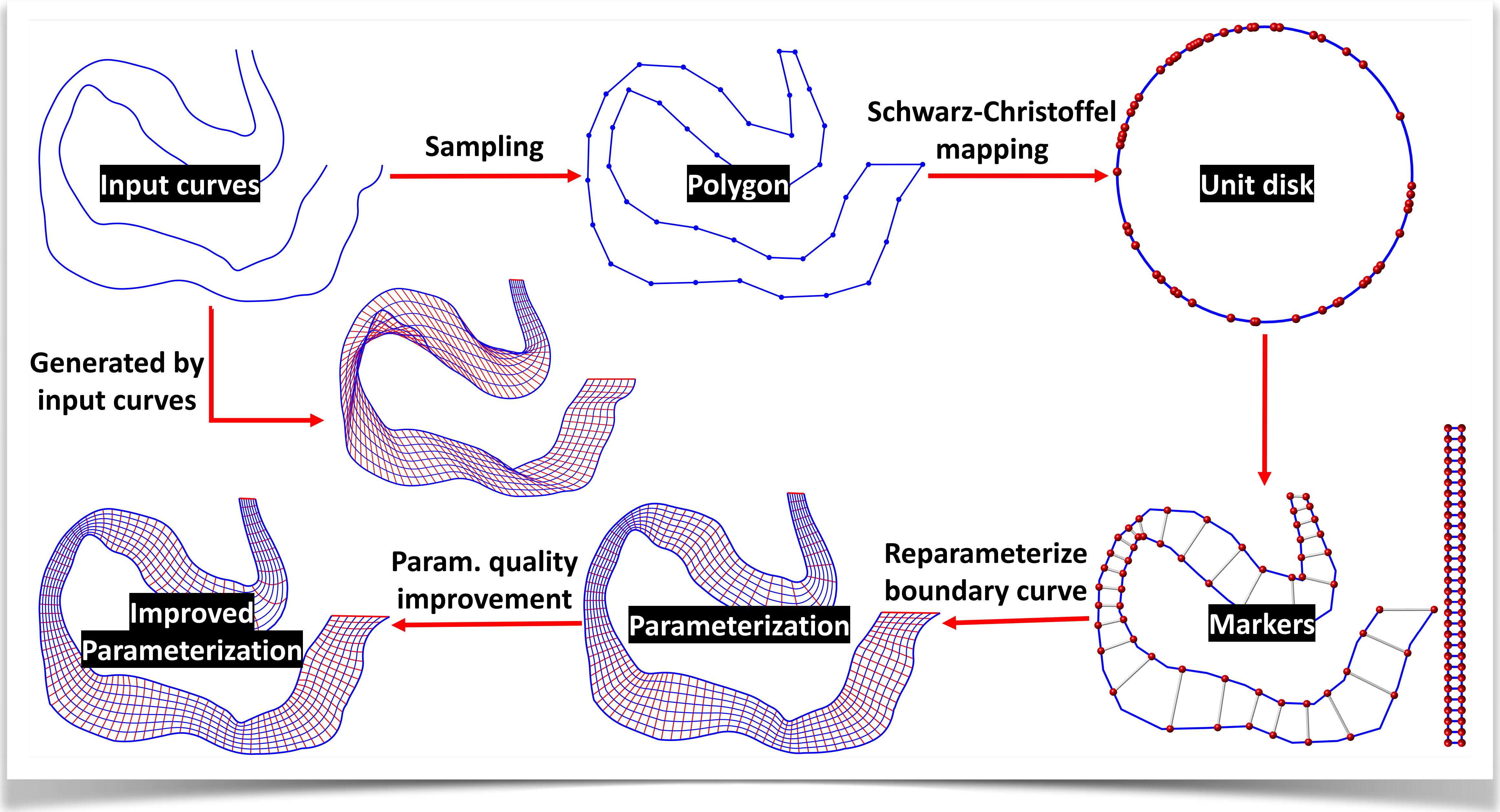
Workflow for planar domains

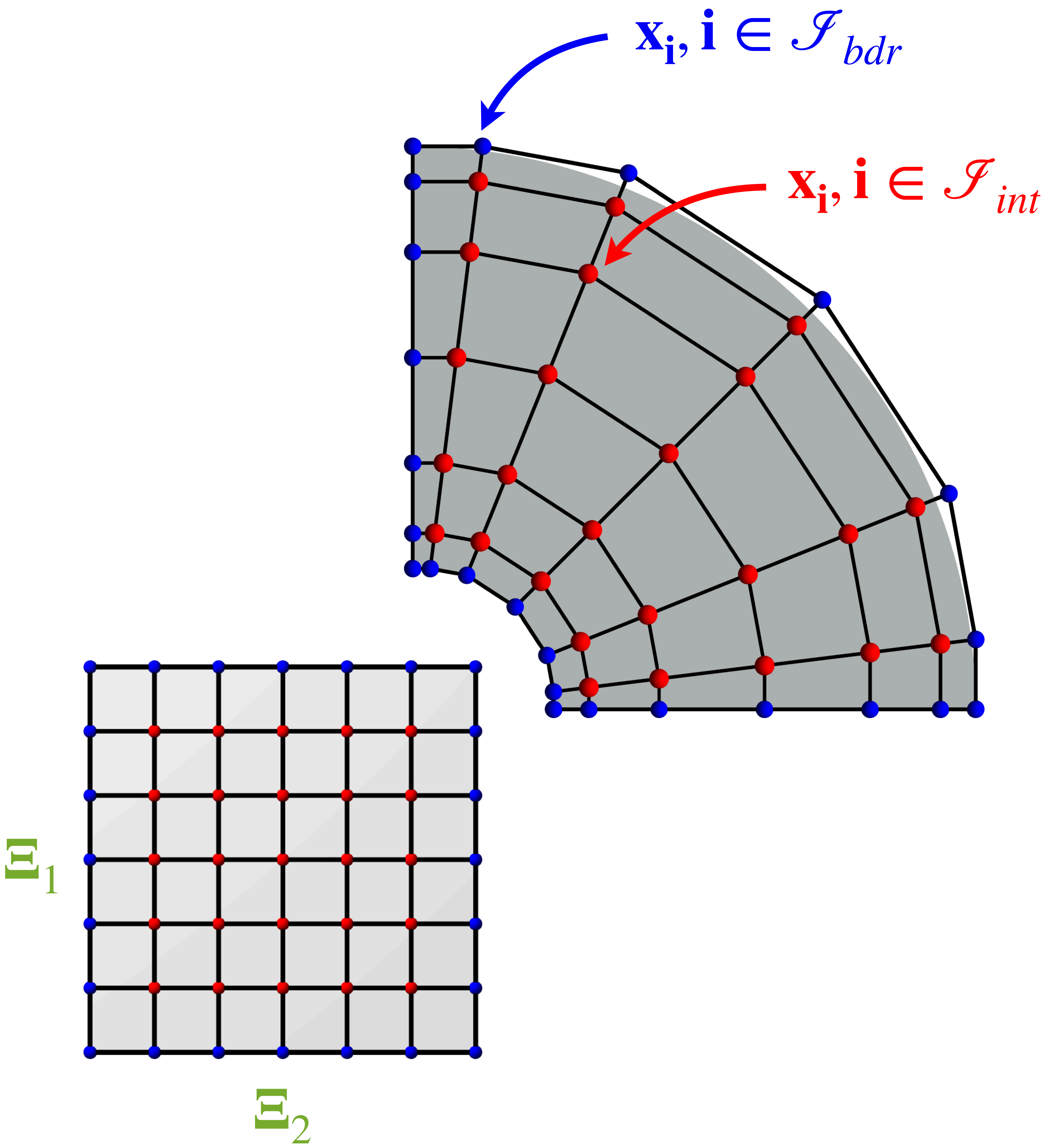




Ye Ji, Wed 10:30 - 12:30 in #MS012A

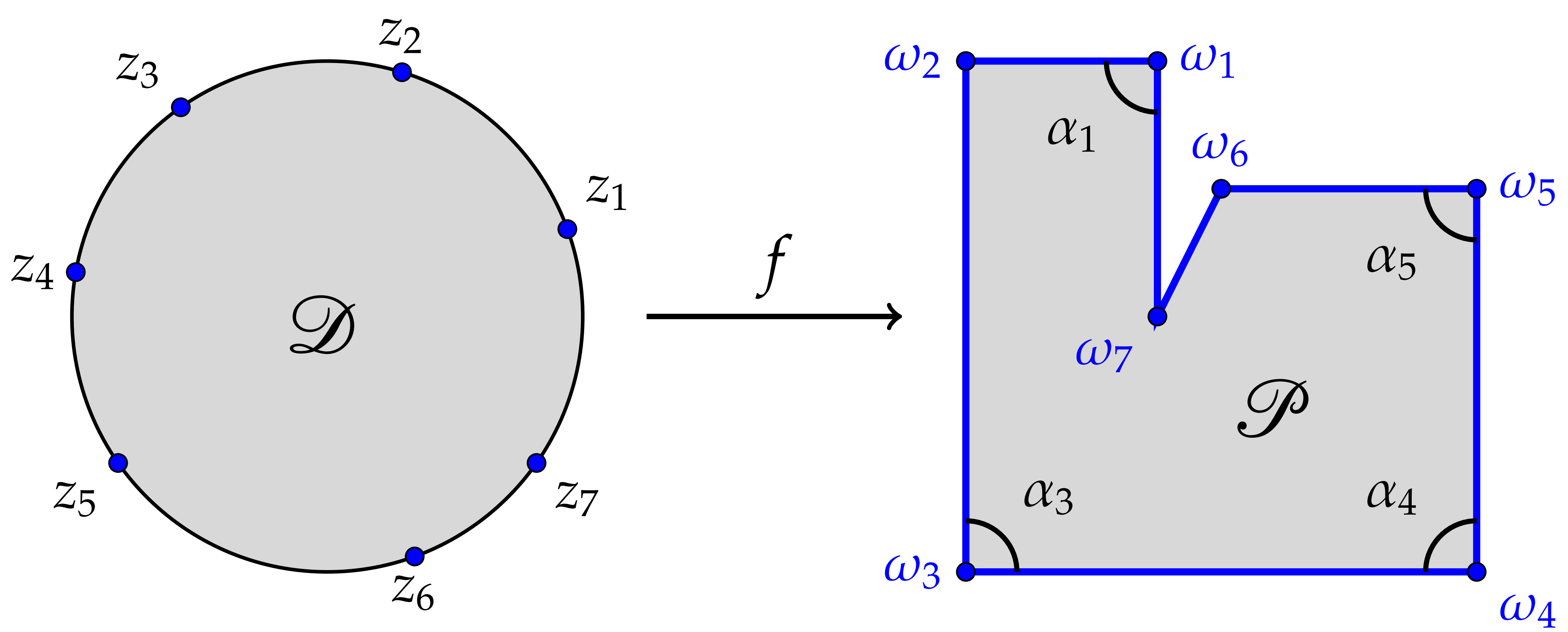
Workflow for planar domains





Ye Ji, Wed 10:30 - 12:30 in #MS012A

Schwarz-Christoffel mapping

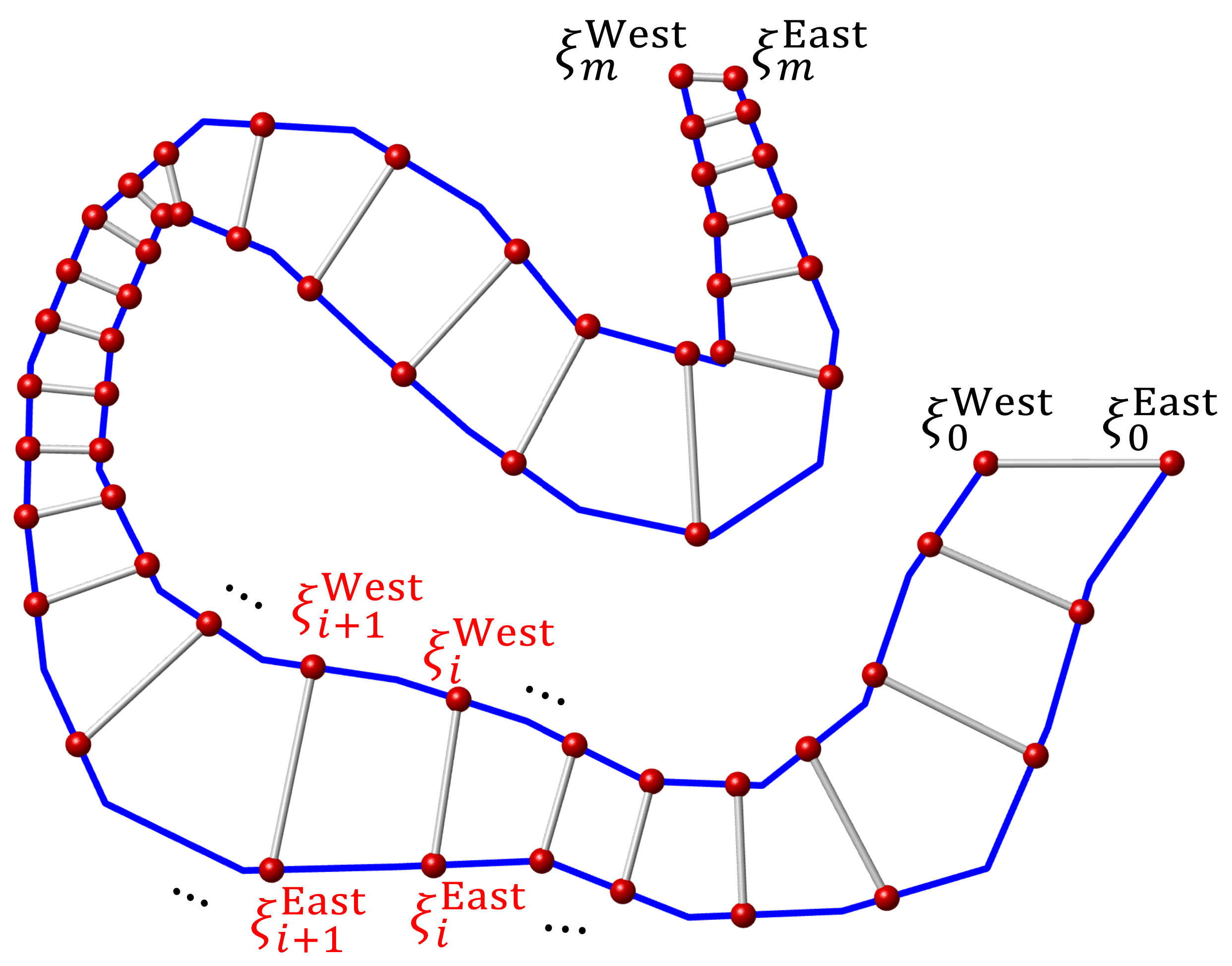


Riemann mapping theorem: $\exists$ analytic function f with non-zero derivative such that $f(\mathscr{D}) = \mathscr{P}$.

Scharz-Christoffel formula

$$f(z) = f(z_0) + C \int_{z_0}^{z} \prod_{j=1}^{n} \left(1 - \frac{\zeta}{z_j}\right)^{\alpha_j/\pi - 1} d\zeta$$

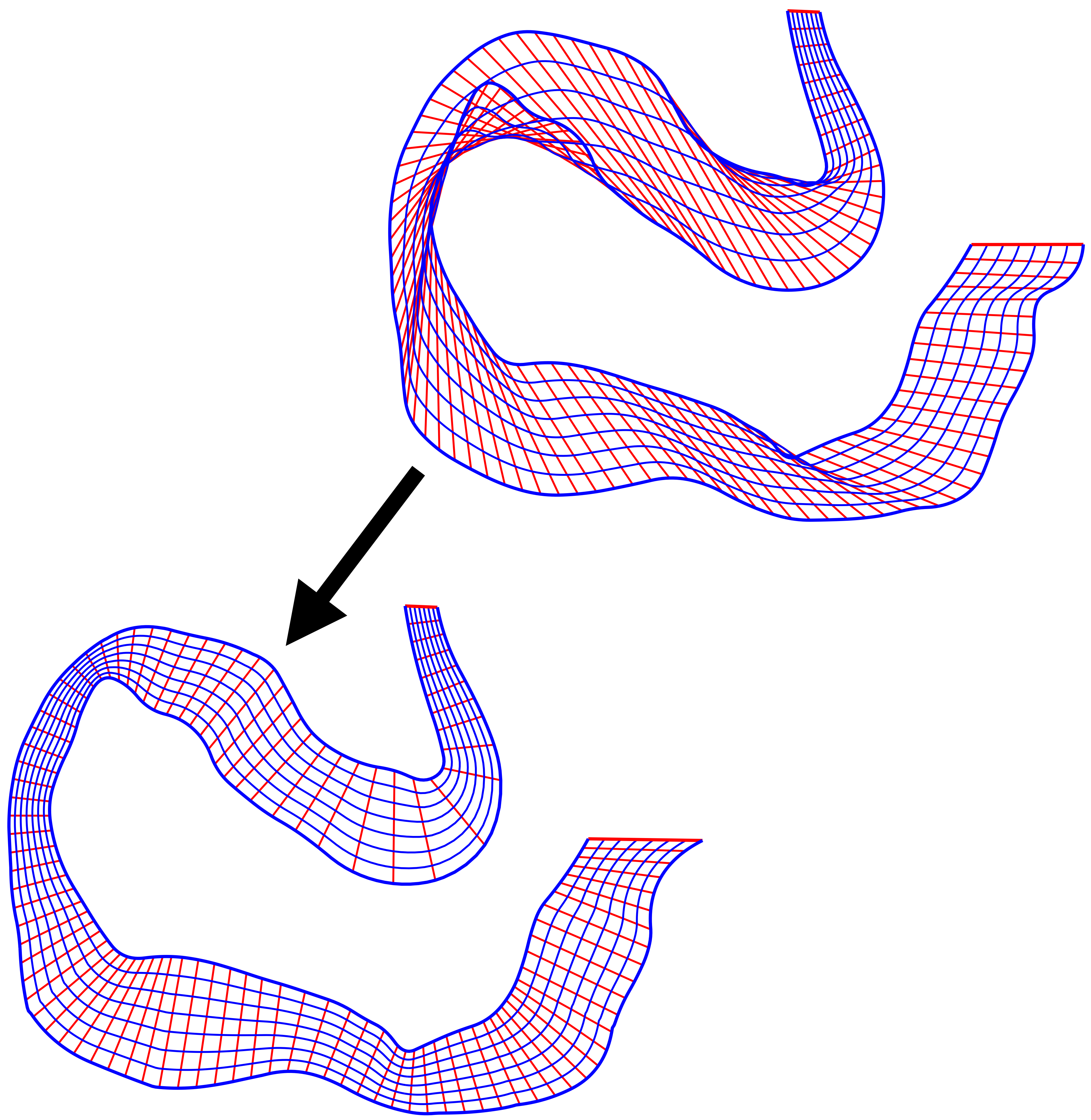
Solving the Schwarz-Christoffel parameter problem for $\{z_j\}$ numerically allows us to calculate sets of markers $\{\mathbf{P}_i^{West}\}$ and $\{\mathbf{P}_i^{East}\}$ on the two opposite curves C^{West} and C^{East} that can be used to reparameterize one curve w.r.t. the other



Y. Ji, MM, Y. Yu, Ch. Zhu, arXiv: 2403.10284

Boundary reparameterization

Theorem [Ji et al. 2024]: B-spline and NURBS basis functions are invariant under scaling and translation of the knot vector, i.e. $b^d_{i,\Xi}(\xi) = b^d_{i,\hat{\Xi}}(s\xi + t)$ with $\hat{\Xi} = s\Xi + t, \quad s > 0$



Sketch of the reparameterization algorithm

- For each pair of markers $(\mathbf{P}_i^{East}, \mathbf{P}_i^{West})$ identify the pair of parameters $(\xi_i^{East}, \xi_i^{West})$ by solving the nonlinear equation $\left(C^*(\xi) - \mathbf{P}_i^*\right) \cdot \partial C^*(\xi)/\partial\xi = 0$ using Newton's method

- Without loss of generality, align the segment of curve C^{East} defined over the parameter interval $[\xi_i^{East}, \xi_{i+1}^{East}]$ with C^{West} by applying an affine transformation

Y. Ji, MM, Y. Yu, Ch. Zhu, arXiv: 2403.10284

From boundary to planar domain parameterizations

Given: $\Gamma = C^{North} \cup C^{South} \cup C^{East} \cup C^{West}$ as push-forward from $\hat{\Gamma}$

Harmonic mapping: compute $\mathbf{x} : \hat{\Omega} \to \Omega$ by solving Laplace problems for $\mathbf{x}^{-1} = \boldsymbol{\xi} : \Omega \to \hat{\Omega}$

$$\begin{aligned} \nabla \cdot \left[\mathbb{A} \, \nabla \xi_1(x_1, x_2) \right] &= 0 \\ \nabla \cdot \left[\mathbb{A} \, \nabla \xi_2(x_1, x_2) \right] &= 0 \end{aligned} \qquad \text{s.t. } \mathbf{x}^{-1} \Big|_{\Gamma} = \hat{\Gamma}$$

with $\mathbb{A} = 1$ [Hinz et al. 2018] or $\mathbb{A} = \mathrm{diag}(1/|\mathbf{J}|)$, Jacobian $\mathbf{J}(\xi_1, \xi_2)$ [Ji et al. 2023]

Radó-Kneser-Choquet theorem: convexity of $\hat{\Omega}$ ensures one-to-one mapping between $\hat{\Omega}$ and Ω

Y. Ji, MM, Proc. 13th Int. Conf. On Compressors and Their Systems 2023

Computation of planar parameterization

Weak form in H^2 [Hinz et al. 2018]

$$\int_{\hat{\Omega}} \mathbf{w} \tilde{\mathscr{L}} \xi_1 \, \mathrm{d}\hat{\Omega} = \mathbf{0}$$
$$\int_{\hat{\Omega}} \mathbf{w} \tilde{\mathscr{L}} \xi_2 \, \mathrm{d}\hat{\Omega} = \mathbf{0}$$

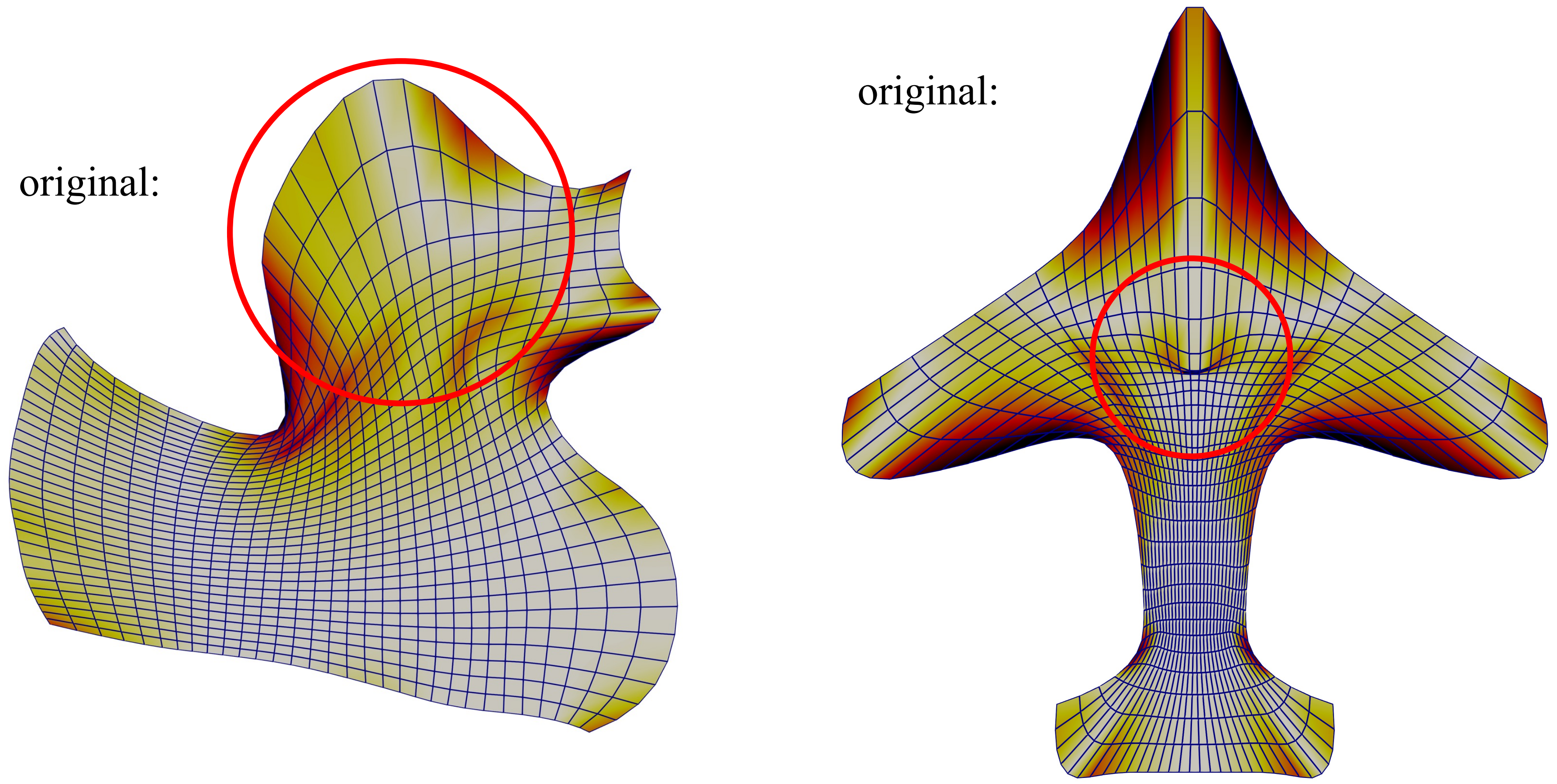
s.t. $\mathbf{x}^{-1}|_{\Gamma} = \hat{\Gamma}$

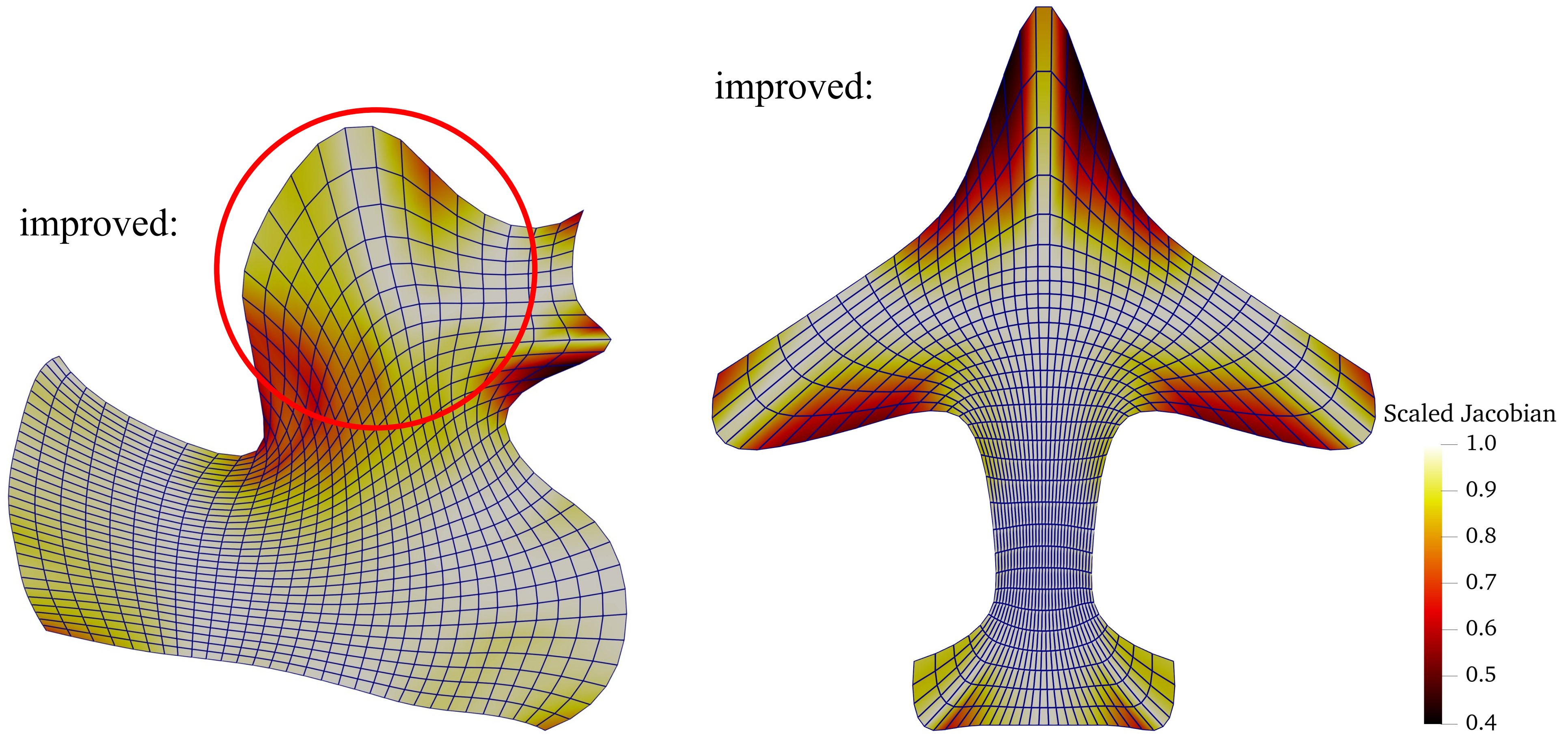
where $\tilde{\mathscr{L}} = \left(g_{22} \frac{\partial^2}{\partial \xi_1^2} - 2 g_{12} \frac{\partial^2}{\partial \xi_1 \partial \xi_2} + g_{11} \frac{\partial^2}{\partial \xi_2^2} \right) / (g_{11} + g_{22})$

Weak form in H^1 [Ji et al. 2023]

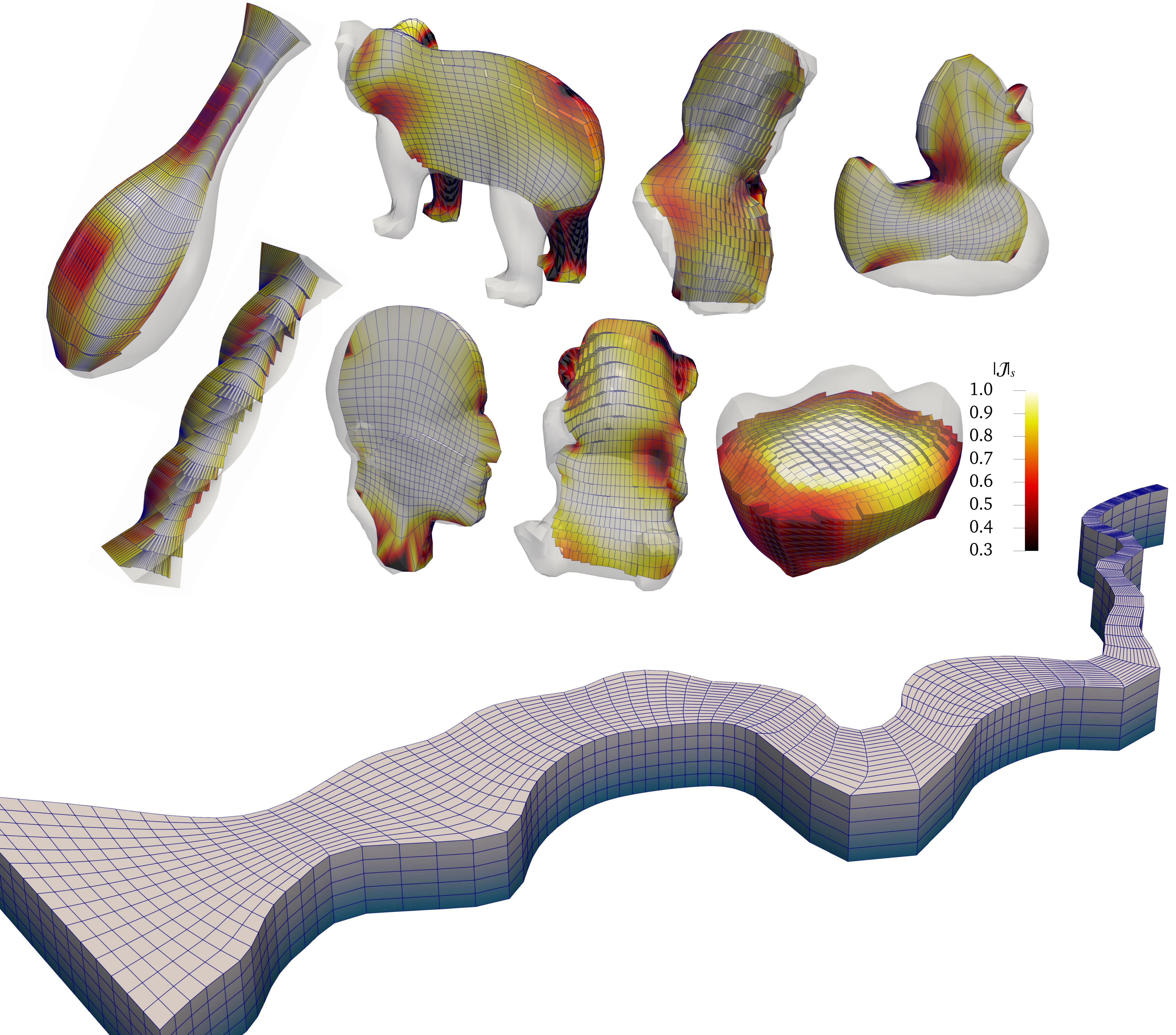
$$\int_{\hat{\Omega}} \nabla \mathbf{w} \cdot \mathbb{A} \nabla \xi_1 \, \mathrm{d}\hat{\Omega} = \mathbf{0}$$
$$\int_{\hat{\Omega}} \nabla \mathbf{w} \cdot \mathbb{A} \nabla \xi_2 \, \mathrm{d}\hat{\Omega} = \mathbf{0}$$

s.t. $\mathbf{x}^{-1}|_{\Gamma} = \hat{\Gamma}$

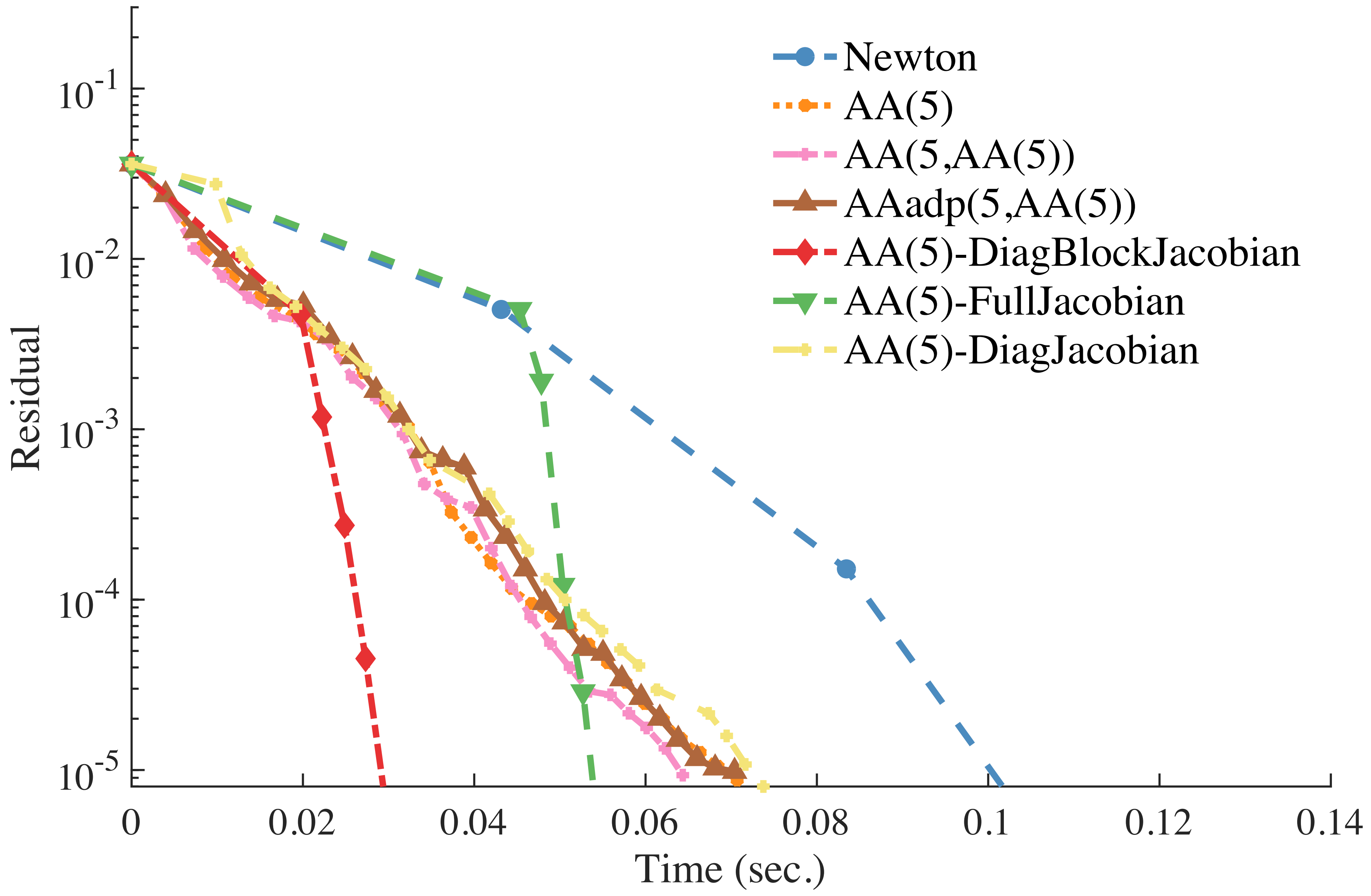




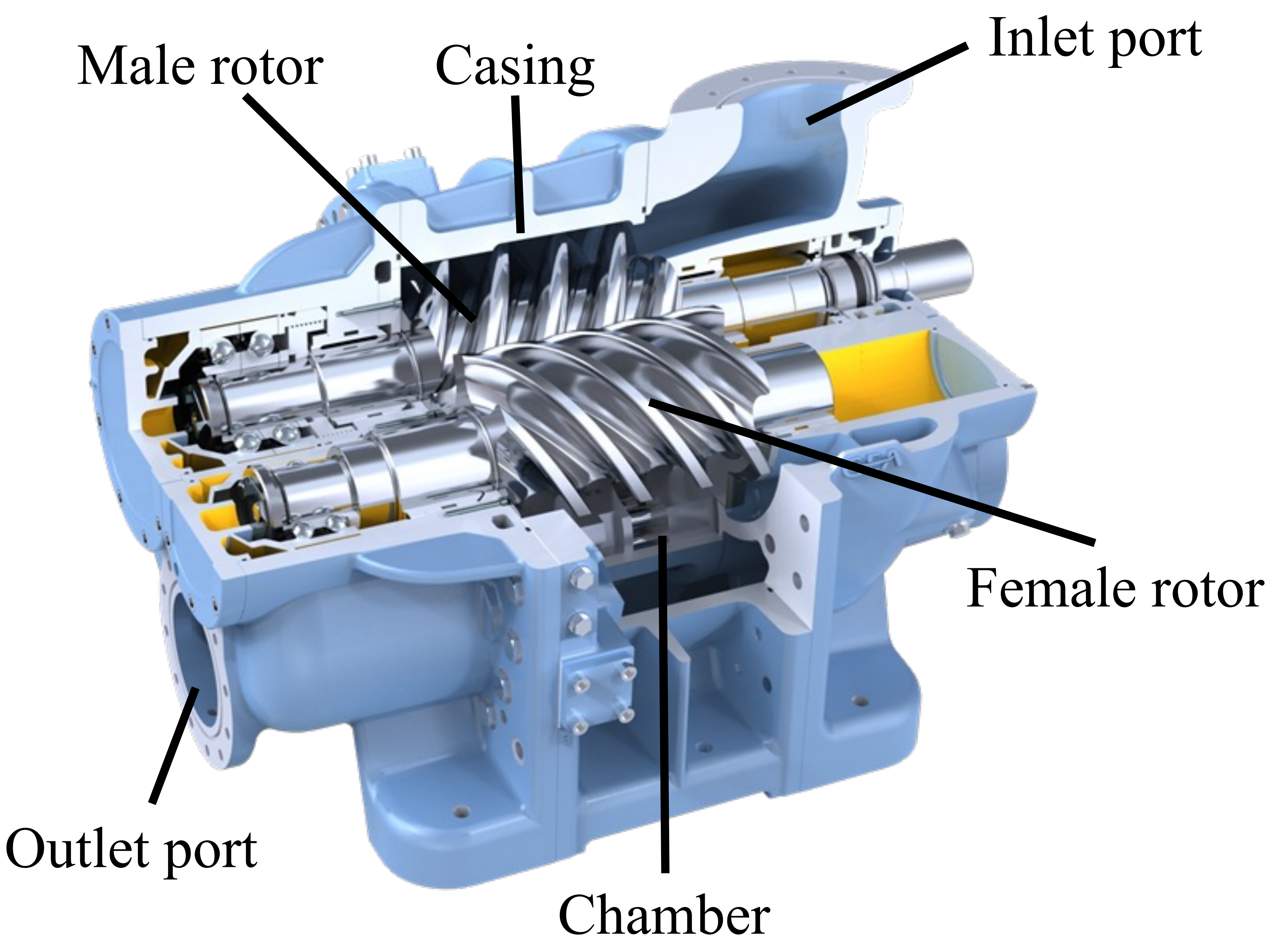
Extension to trivariate parameterizations



Existence of one-to-one mapping is no longer ensured by RKC theorem for non-planer domains but our **approach is efficient and robust in practice** when the nonlinear problems are solved with a preconditioned Anderson accelerator

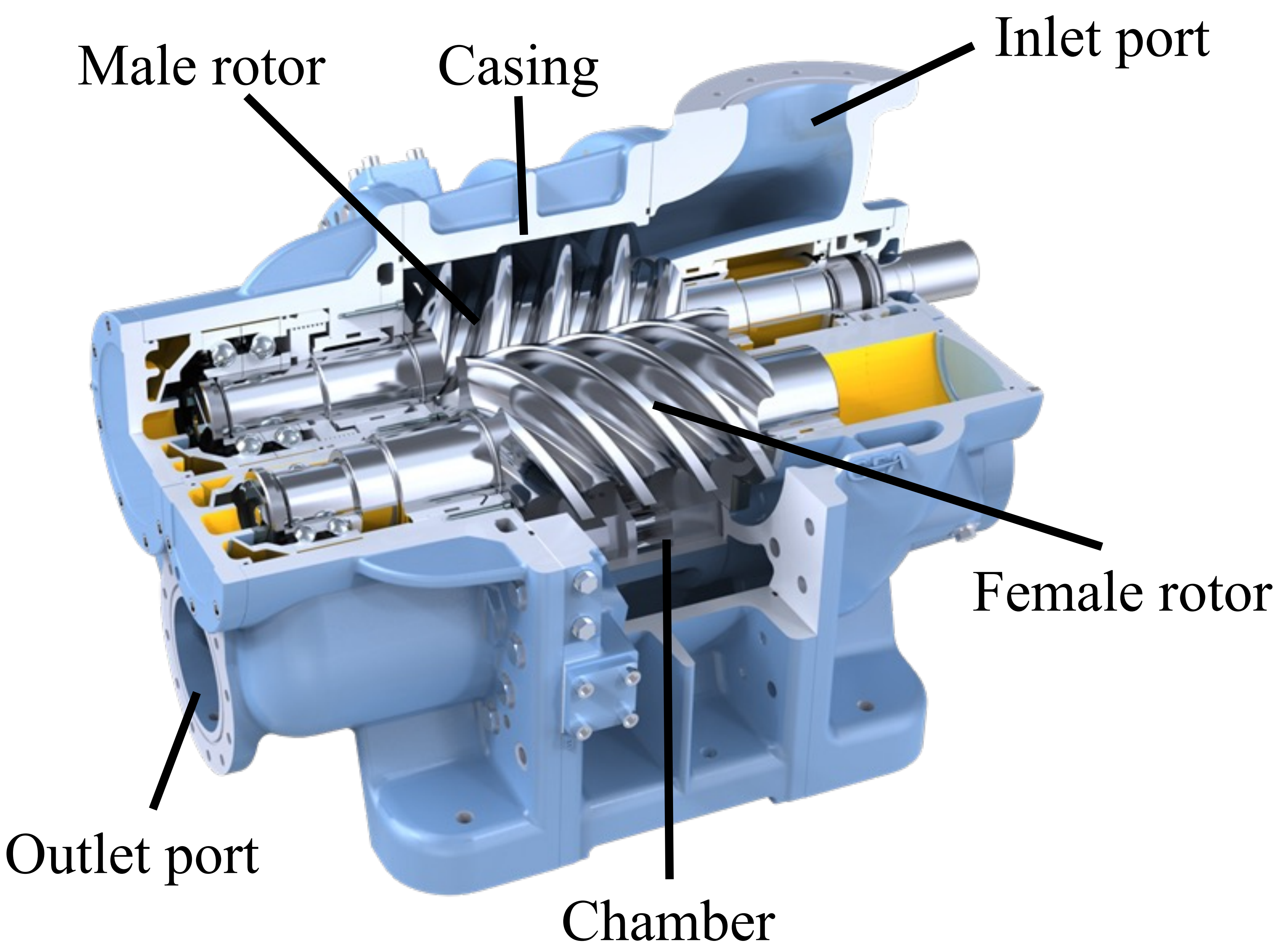


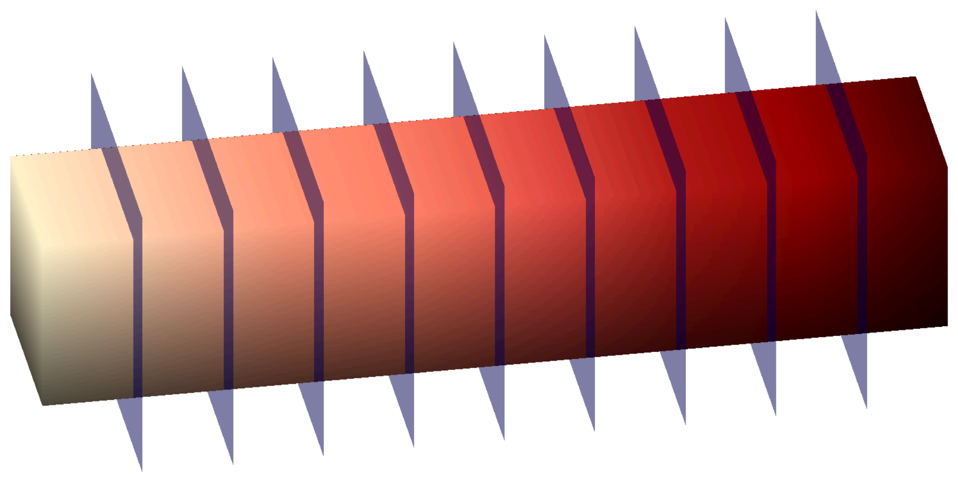
Intermezzo: spline-based mesh generation

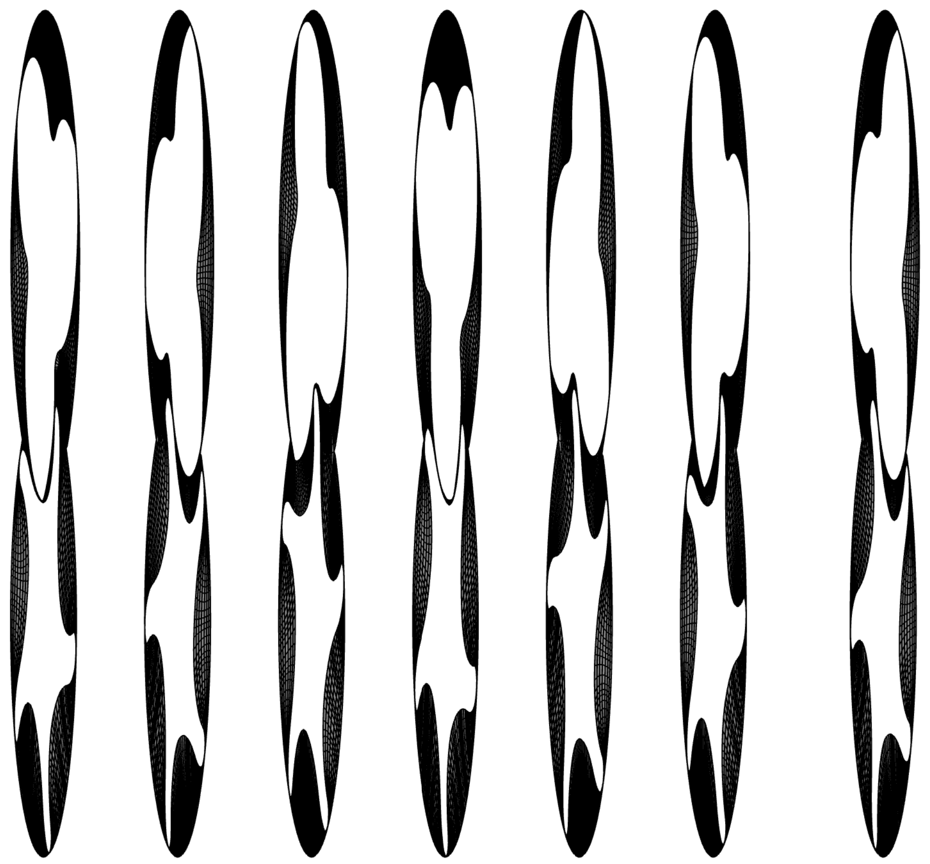


Y. Ji, 2024

Intermezzo: spline-based mesh generation

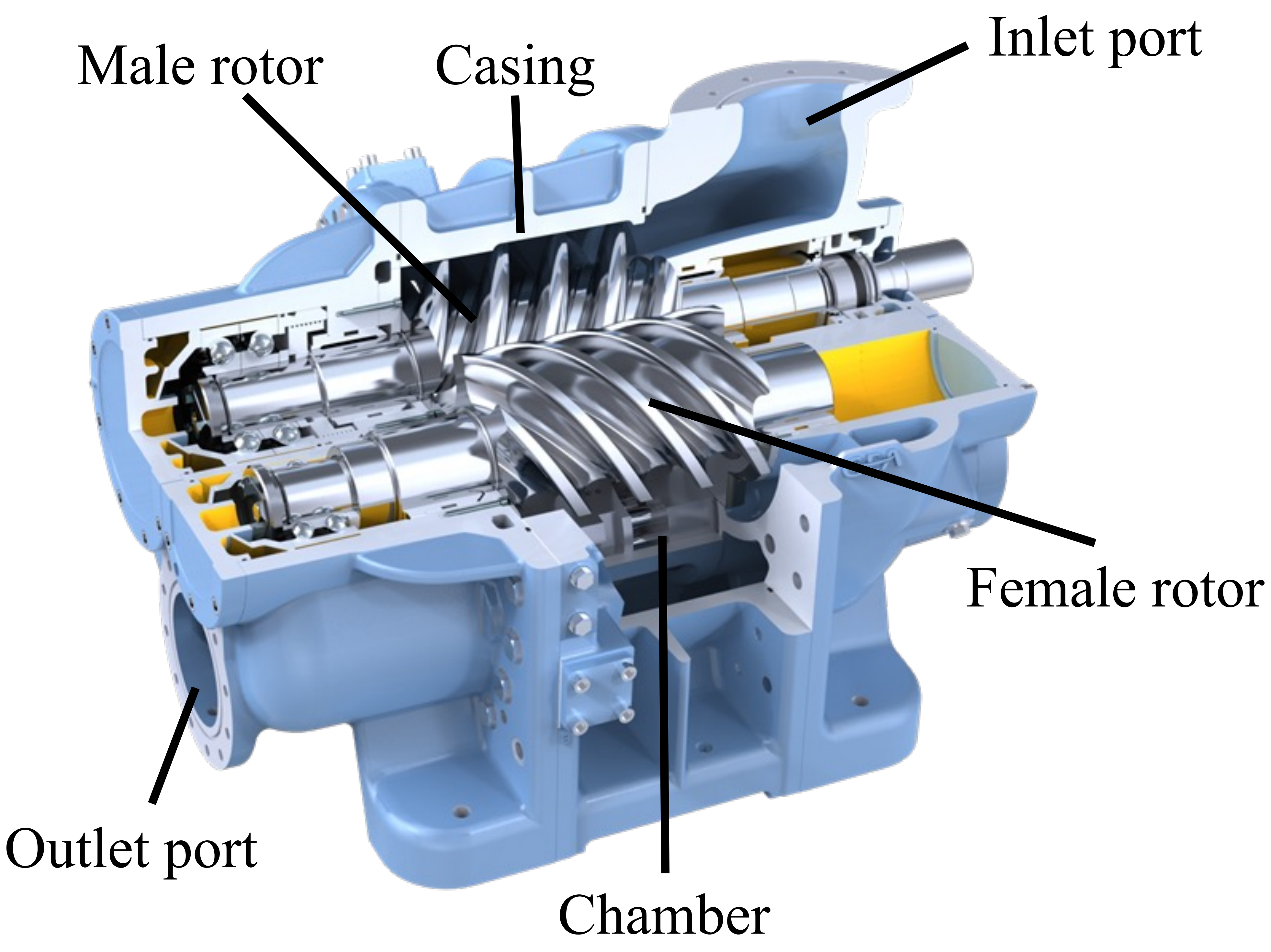


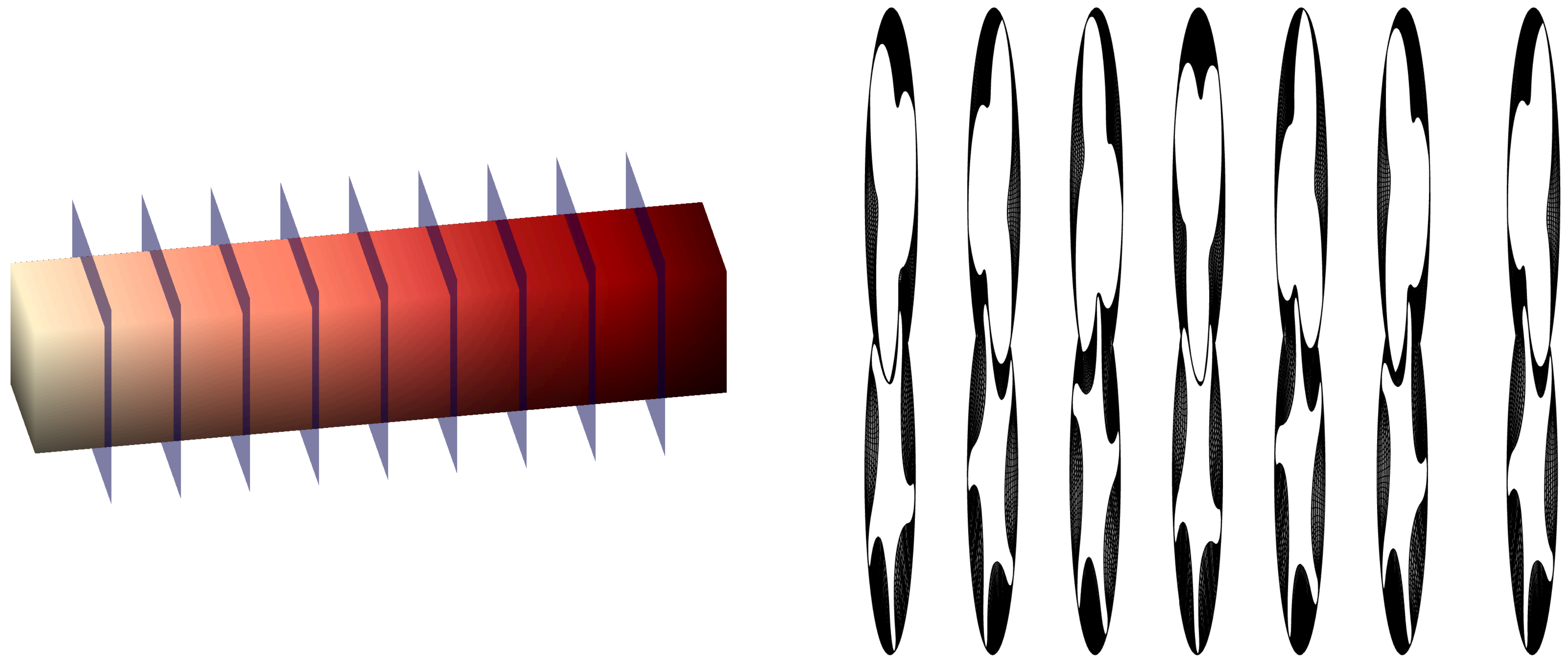


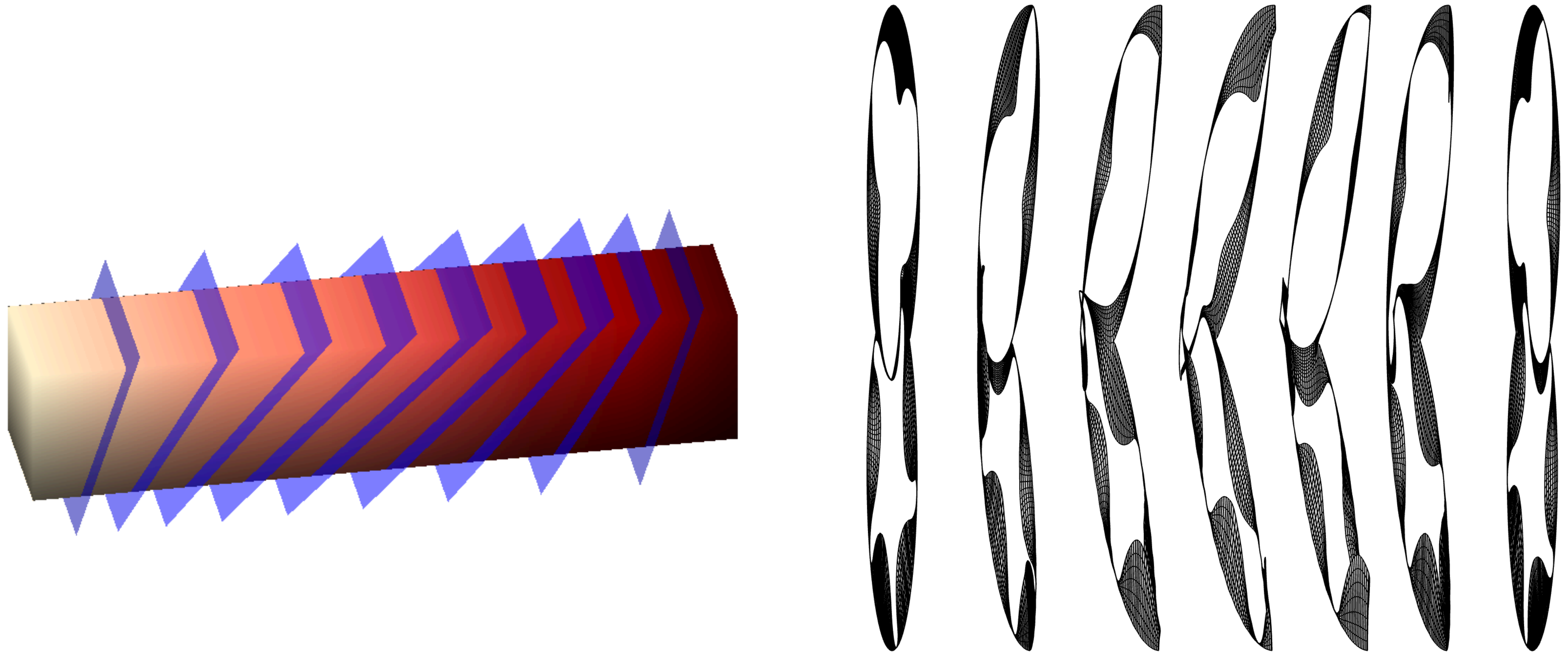


Y. Ji, 2024

Intermezzo: spline-based mesh generation

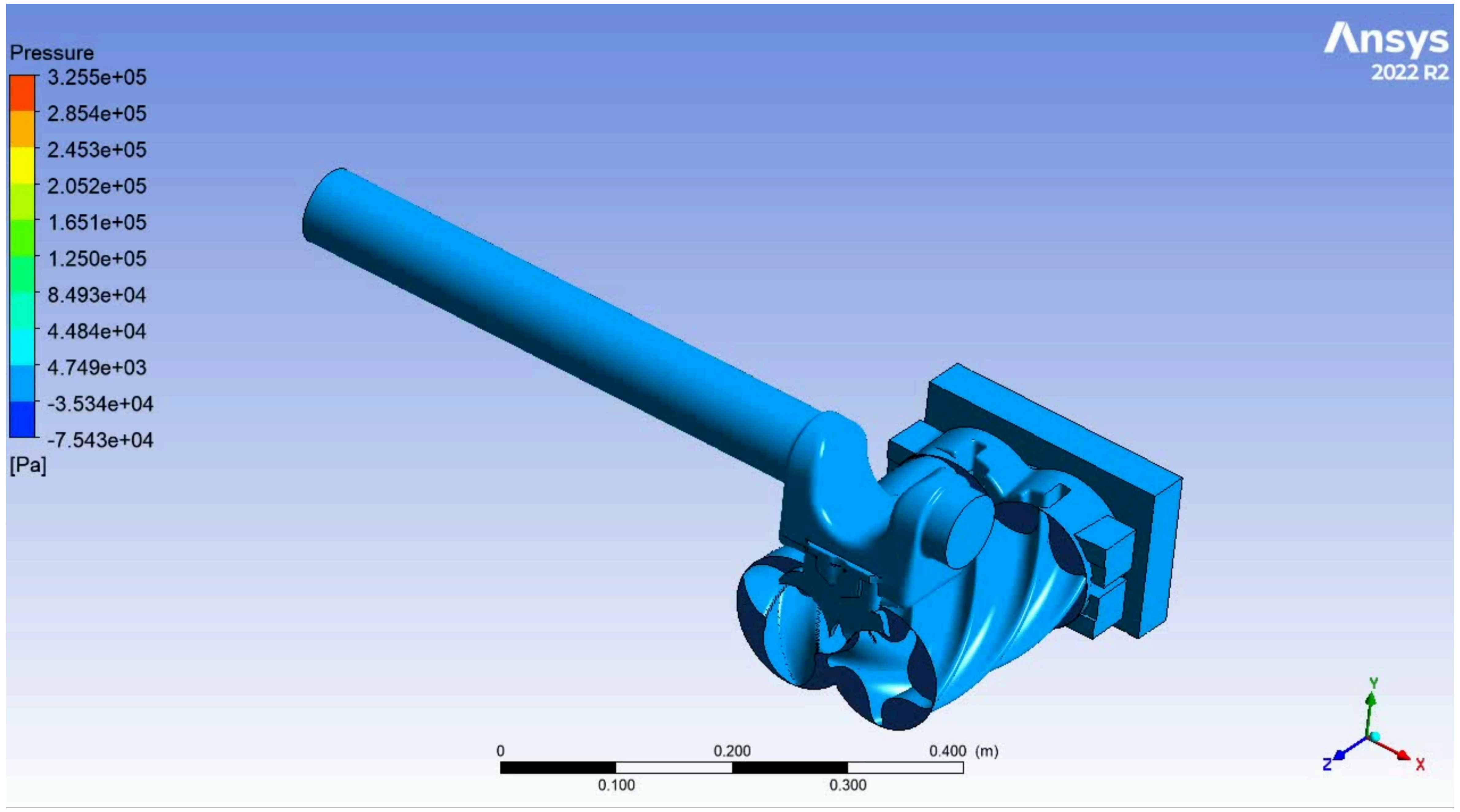


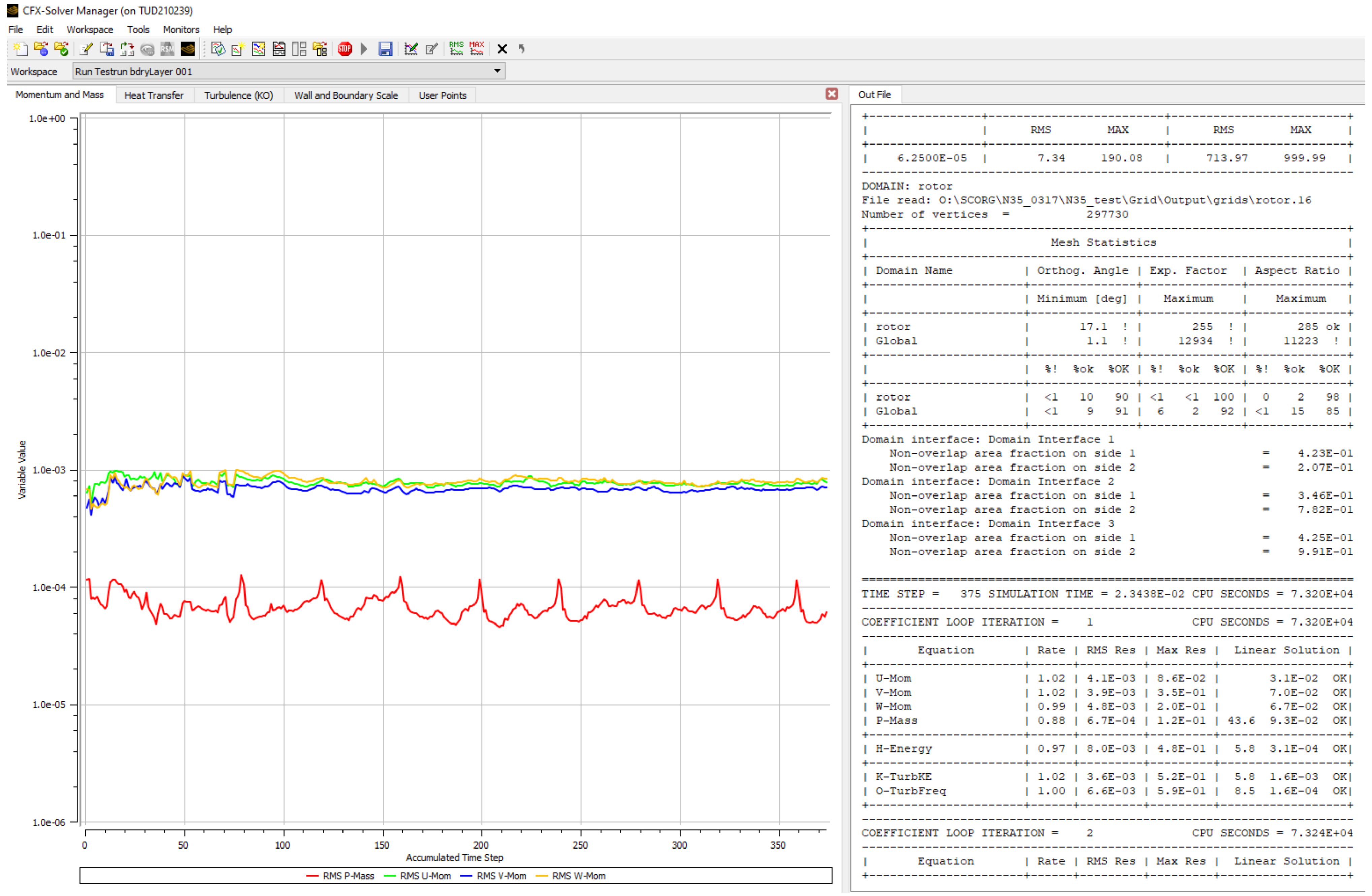




Y. Ji, 2024

Intermezzo: spline-based mesh generation + simulation

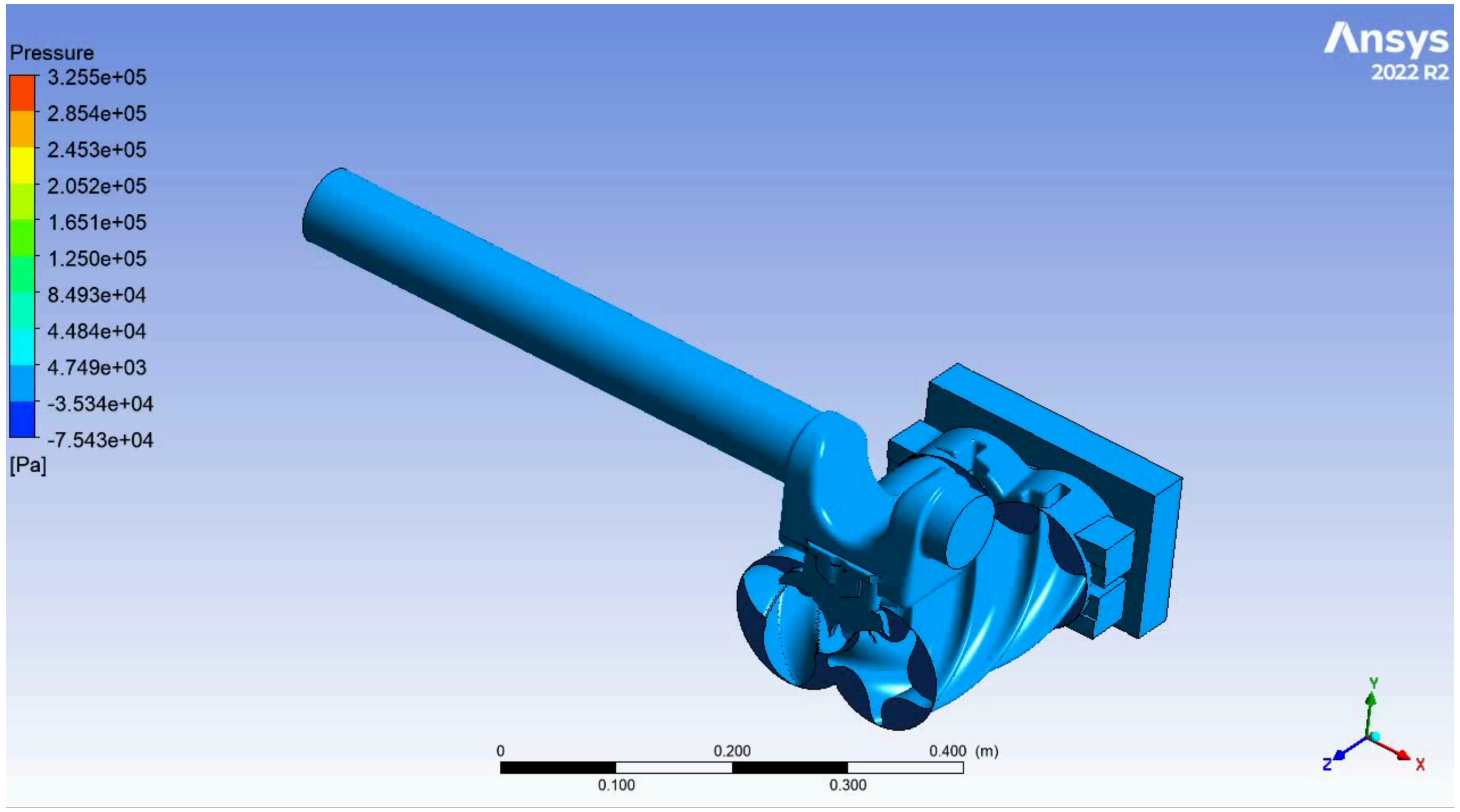


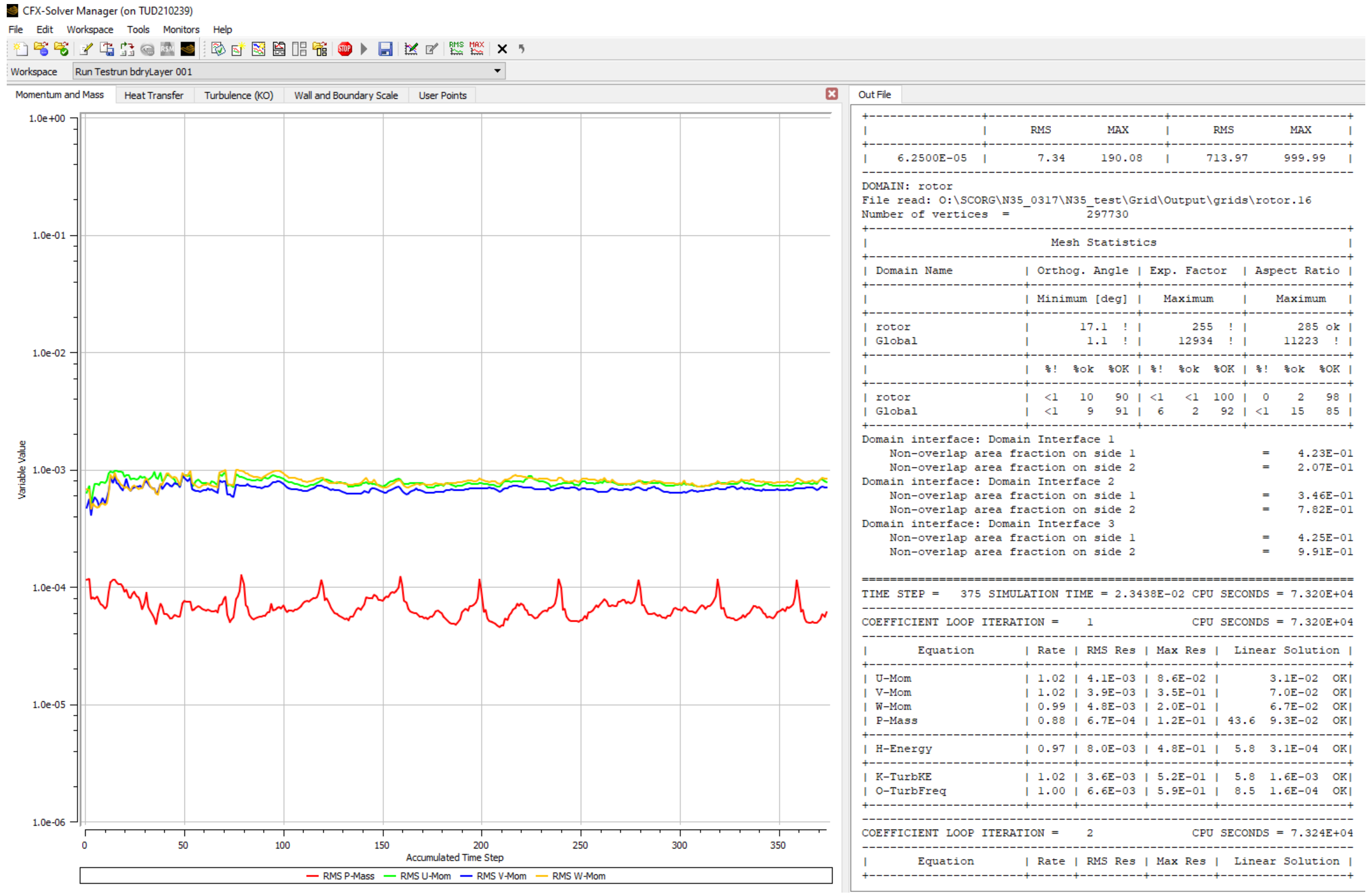


Y. Ji, 2024

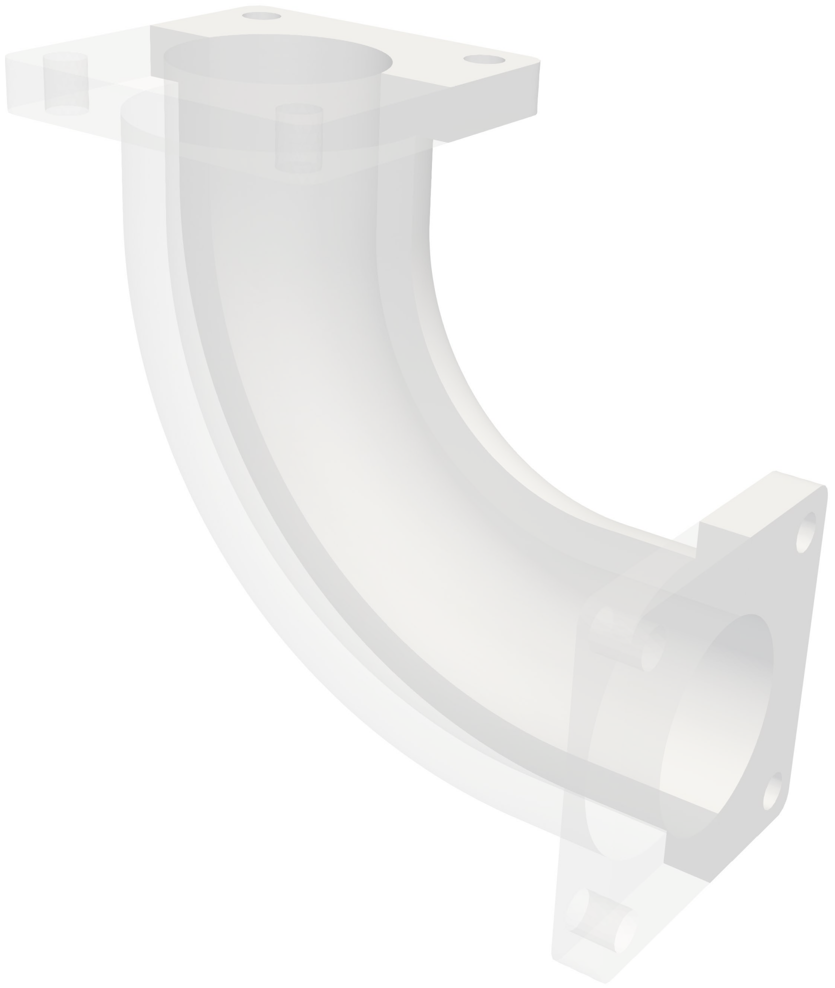
CITY
UNIVERSITY OF LONDON
EST 1894

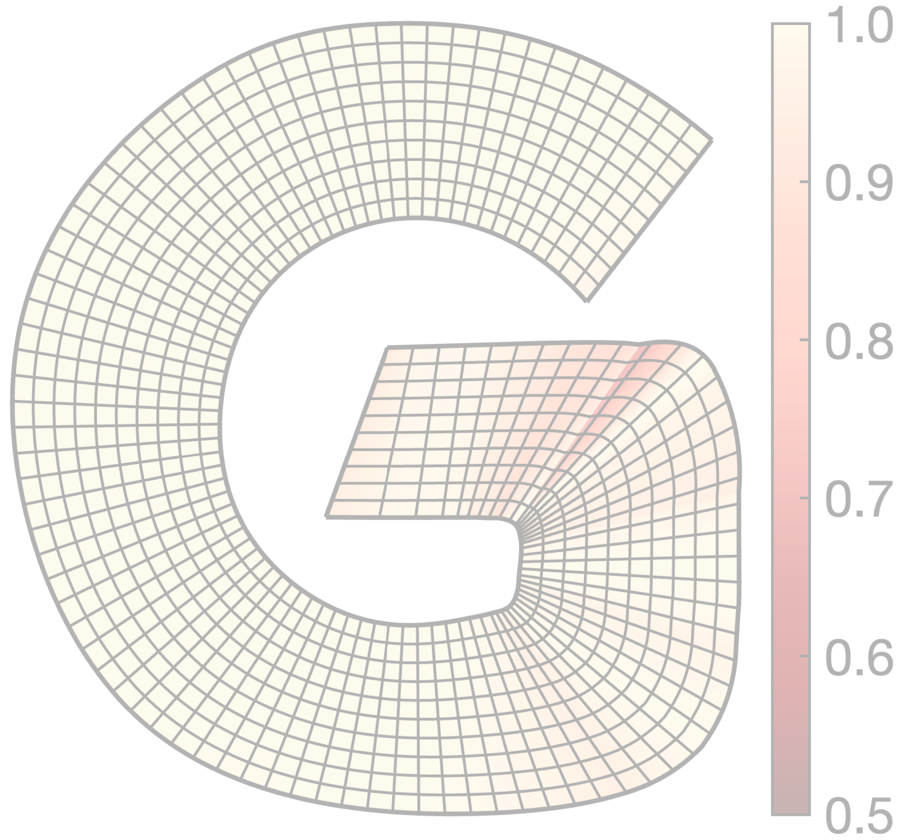
Intermezzo: spline-based mesh generation + simulation

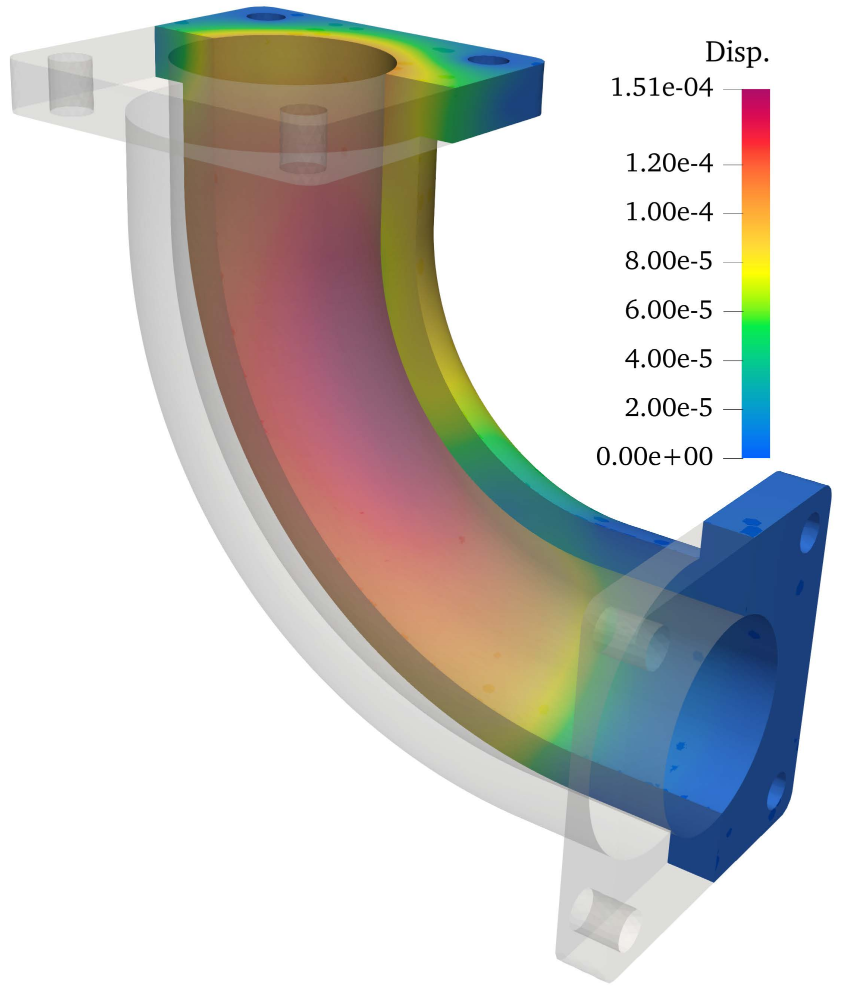


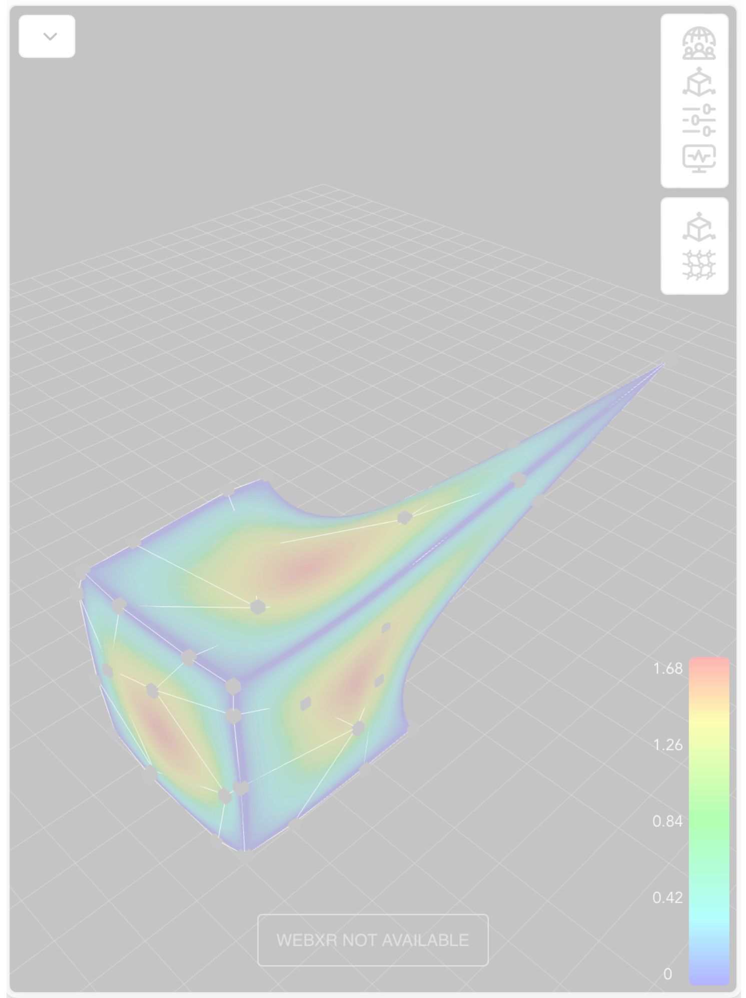


Y. Ji, 2024









1. Creation of VReps from BReps

2. Reparameterization techniques

3. Isogeometric Analysis and IgANets

4. Interactive Design-through-Analysis

Galerkin IgA

Model problem

$$\mathscr{L}u = f \quad \text{in } \Omega$$
$$\mathscr{B}u = g \quad \text{on } \Gamma$$

Weighted residual form

$$\int_\Omega \phi_\Omega(\mathscr{L}u - f)\,\mathrm{d}\Omega + \int_\Gamma \phi_\Gamma(\mathscr{B}u - g)\,\mathrm{d}\Gamma = 0$$

Procedure

1. Integrate by parts to equilibrate C^ℓ between test and trial functions
2. Integrate natural boundary conditions into boundary integral term
3. Discretize (evaluate integrals on parametric domain) + solve

Collocation IgA

Model problem

$$\mathscr{L}u = f \quad \text{in } \Omega$$
$$\mathscr{B}u = g \quad \text{on } \Gamma$$

Weighted residual form

$$\int_\Omega \phi_\Omega(\mathscr{L}u - f)\,\mathrm{d}\Omega + \int_\Gamma \phi_\Gamma(\mathscr{B}u - g)\,\mathrm{d}\Gamma = 0$$

Let

$$\phi_\Omega = \sum_{i=1}^{k} \delta_\Omega(\mathbf{x}-\mathbf{x}_i)\,c_i \quad (\mathbf{x}_i \in \Omega) \qquad \text{and} \qquad \phi_\Gamma = \sum_{i=k+1}^{n} \delta_\Gamma(\mathbf{x}-\mathbf{x}_i)\,c_i \quad (\mathbf{x}_i \in \Gamma)$$

then

$$\sum_{i=1}^{k} \left(\mathscr{L}u(\mathbf{x}_i) - f(\mathbf{x}_i)\right)\,c_i + \sum_{i=1+k}^{n} \left(\mathscr{B}u(\mathbf{x}_i) - g(\mathbf{x}_i)\right)\,c_i = 0$$

Collocation IgA

As the coefficients c_i are arbitrary we obtain

$$\begin{aligned} \mathscr{L}u(\mathbf{x}_i) &= f(\mathbf{x}_i) \qquad && i = 1,\ldots,k \\ \mathscr{B}u(\mathbf{x}_i) &= g(\mathbf{x}_i) \qquad && i = k+1,\ldots,n \end{aligned}$$

replacing $u \approx u_h = \sum_{j=1}^{n} u_j b_j(\mathbf{x})$ yields

$$\begin{bmatrix} \mathscr{L}b_1(\mathbf{x}_1) & \ldots & \mathscr{L}b_n(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \mathscr{L}b_1(\mathbf{x}_k) & \ldots & \mathscr{L}b_n(\mathbf{x}_k) \\ \mathscr{B}b_1(\mathbf{x}_{k+1}) & \ldots & \mathscr{B}b_n(\mathbf{x}_{k+1}) \\ \vdots & \ddots & \vdots \\ \mathscr{B}b_1(\mathbf{x}_n) & \ldots & \mathscr{B}b_n(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_k \\ u_{k+1} \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_k) \\ g(\mathbf{x}_{k+1}) \\ \vdots \\ g(\mathbf{x}_n) \end{bmatrix}$$

Collocation IgA

As the coefficients c_i are arbitrary we obtain

$$\begin{aligned} \mathscr{L}u(\mathbf{x}_i) &= f(\mathbf{x}_i) \qquad & i &= 1,\ldots,k \\ \mathscr{B}u(\mathbf{x}_i) &= g(\mathbf{x}_i) \qquad & i &= k+1,\ldots,n \end{aligned}$$

replacing $u \approx u_h = \sum_{j=1}^{n} u_j b_j(\mathbf{x})$ yields

$$\begin{bmatrix} \mathscr{L}b_1(\mathbf{x}_1) & \ldots & \mathscr{L}b_n(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \mathscr{L}b_1(\mathbf{x}_k) & \ldots & \mathscr{L}b_n(\mathbf{x}_k) \\ \mathscr{B}b_1(\mathbf{x}_{k+1}) & \ldots & \mathscr{B}b_n(\mathbf{x}_{k+1}) \\ \vdots & \ddots & \vdots \\ \mathscr{B}b_1(\mathbf{x}_n) & \ldots & \mathscr{B}b_n(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_k \\ u_{k+1} \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_k) \\ g(\mathbf{x}_{k+1}) \\ \vdots \\ g(\mathbf{x}_n) \end{bmatrix}$$

- basis functions b_i need to be at least C^ℓ such that application of $\mathscr{L}$ and $\mathscr{B}$ is well-defined

- non-singular system matrix requires

  #coll. pts = #basfunc

  and all coll. points must be pairwise distinct

Collocation IgA

As the coefficients c_i are arbitrary we obtain

$$\mathscr{L}u(\mathbf{x}_i) = f(\mathbf{x}_i) \qquad i = 1,\ldots,k$$
$$\mathscr{B}u(\mathbf{x}_i) = g(\mathbf{x}_i) \qquad i = k+1,\ldots,n$$

replacing $u \approx u_h = \sum_{j=1}^{n} u_j b_j(\mathbf{x})$ yields

$$\begin{bmatrix} \mathscr{L}b_1(\mathbf{x}_1) & \ldots & \mathscr{L}b_n(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \mathscr{L}b_1(\mathbf{x}_k) & \ldots & \mathscr{L}b_n(\mathbf{x}_k) \\ \mathscr{B}b_1(\mathbf{x}_{k+1}) & \ldots & \mathscr{B}b_n(\mathbf{x}_{k+1}) \\ \vdots & \ddots & \vdots \\ \mathscr{B}b_1(\mathbf{x}_n) & \ldots & \mathscr{B}b_n(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_k \\ u_{k+1} \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_k) \\ g(\mathbf{x}_{k+1}) \\ \vdots \\ g(\mathbf{x}_n) \end{bmatrix}$$

- basis functions b_i need to be at least C^ℓ such that application of $\mathscr{L}$ and $\mathscr{B}$ is well-defined

- non-singular system matrix requires

  #coll. pts = #basfunc

  and all coll. points must be pairwise distinct

- B-spline basis functions are defined on the parametric domain $\hat{\Omega} = (0,1)^d$, hence

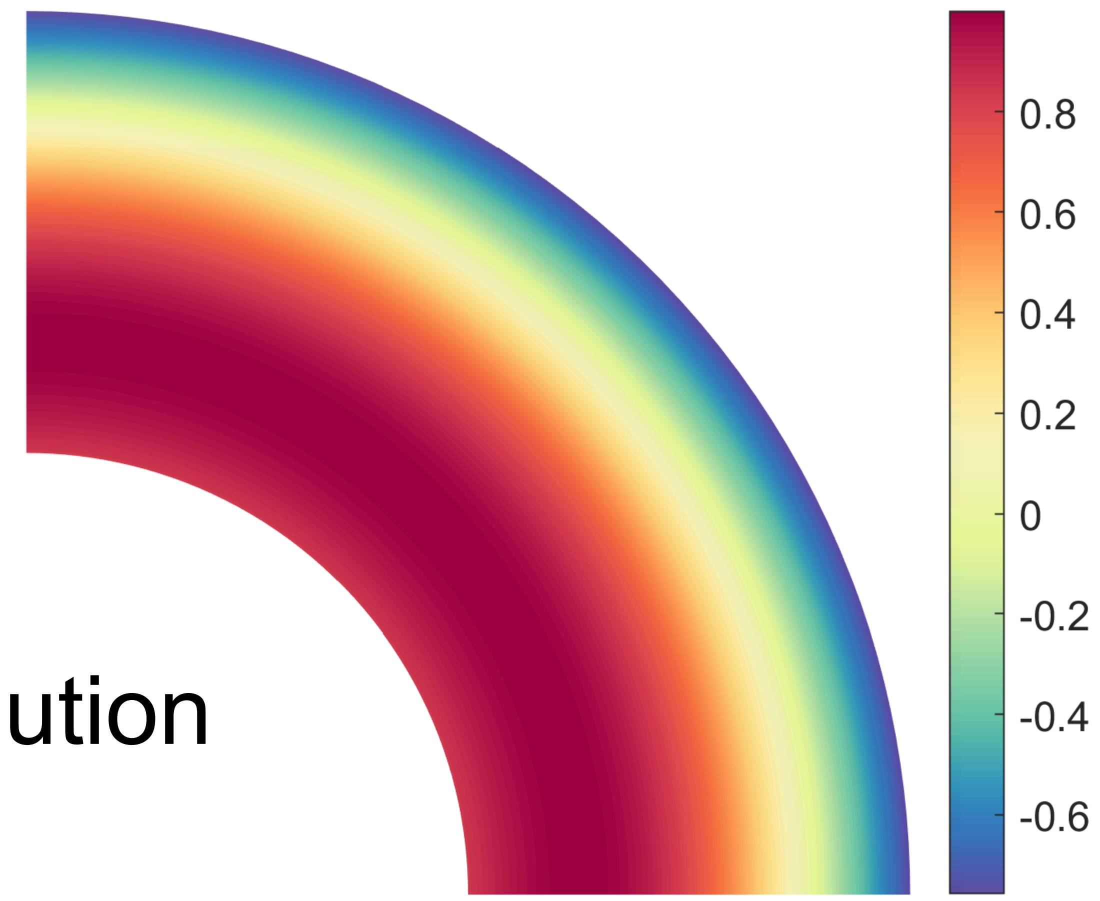
$$\mathscr{L}_{\mathbf{x}} b_i(\mathbf{x}) \to \mathscr{L}_{\boldsymbol{\xi}} b_i(\boldsymbol{\xi})$$
$$\mathscr{B}_{\mathbf{x}} b_i(\mathbf{x}) \to \mathscr{B}_{\boldsymbol{\xi}} b_i(\boldsymbol{\xi})$$

and define $\mathbf{x}_i := \mathbf{x}_h(\boldsymbol{\xi}_i)$

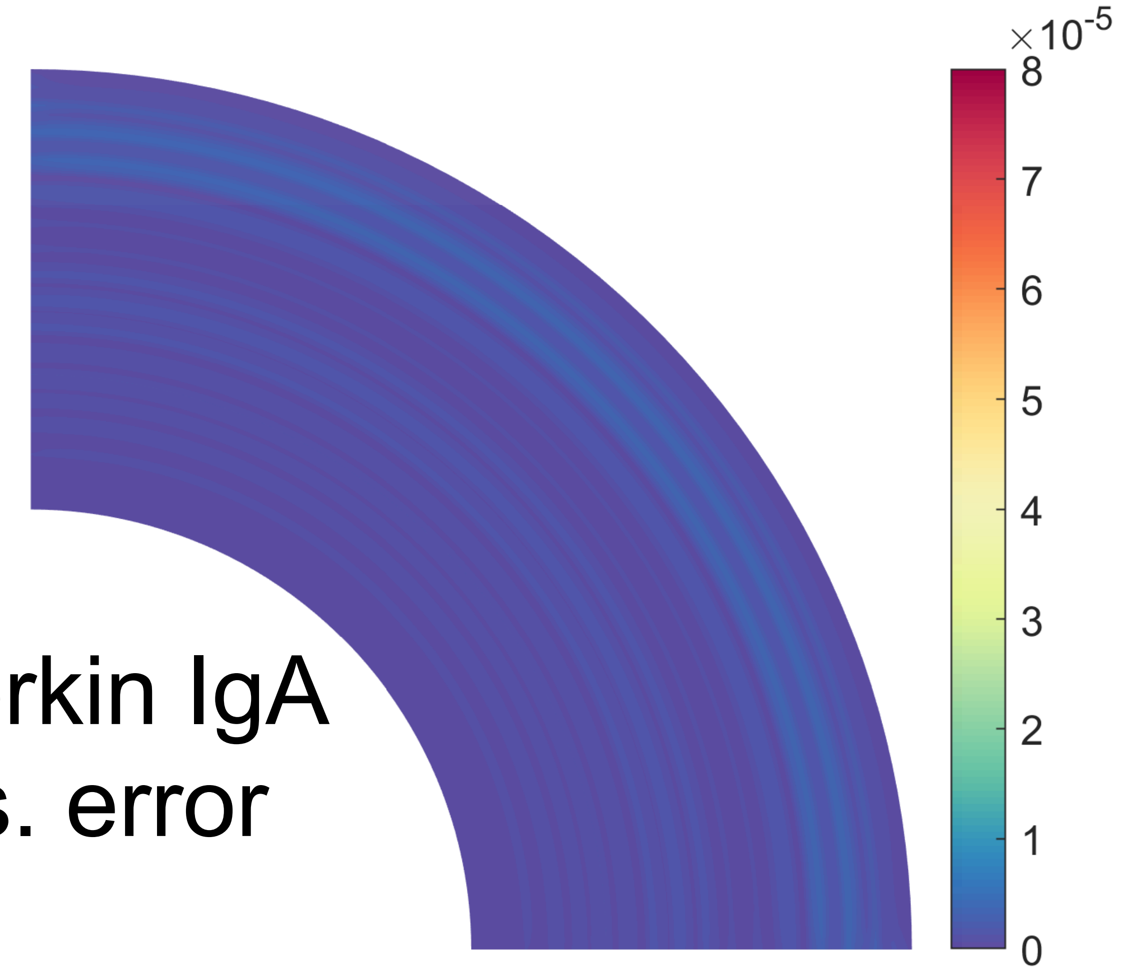
Galerkin vs. collocation IgA

Model problem

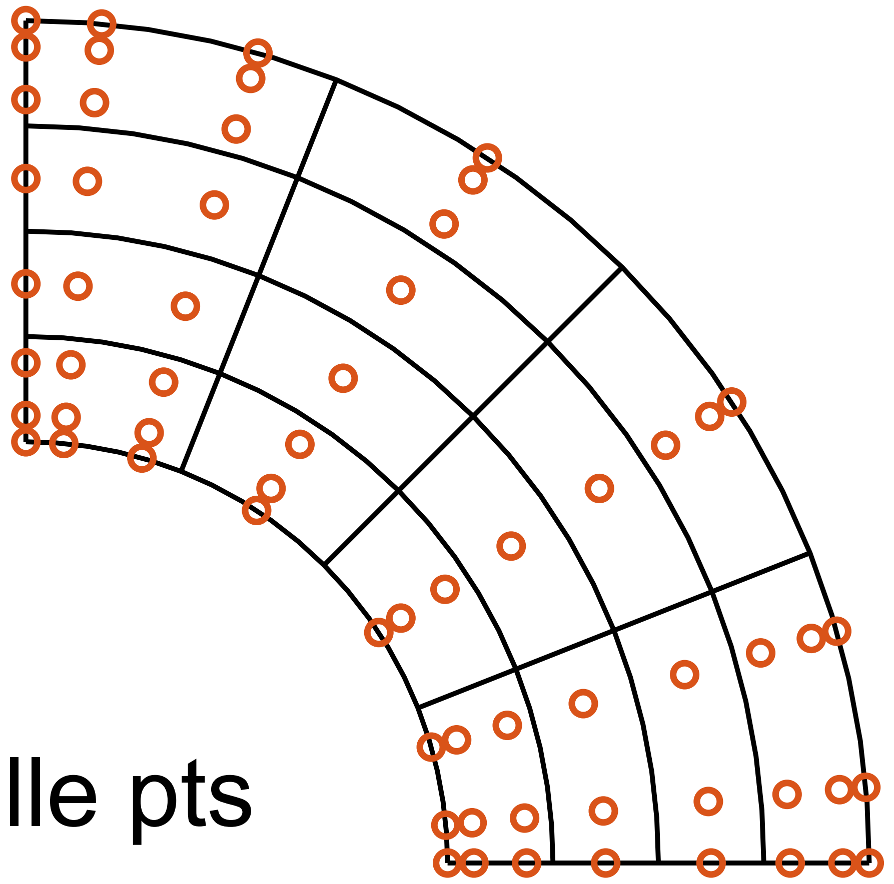
$$-\Delta u = f \quad \text{in } \Omega$$
$$u = g \quad \text{on } \Gamma$$



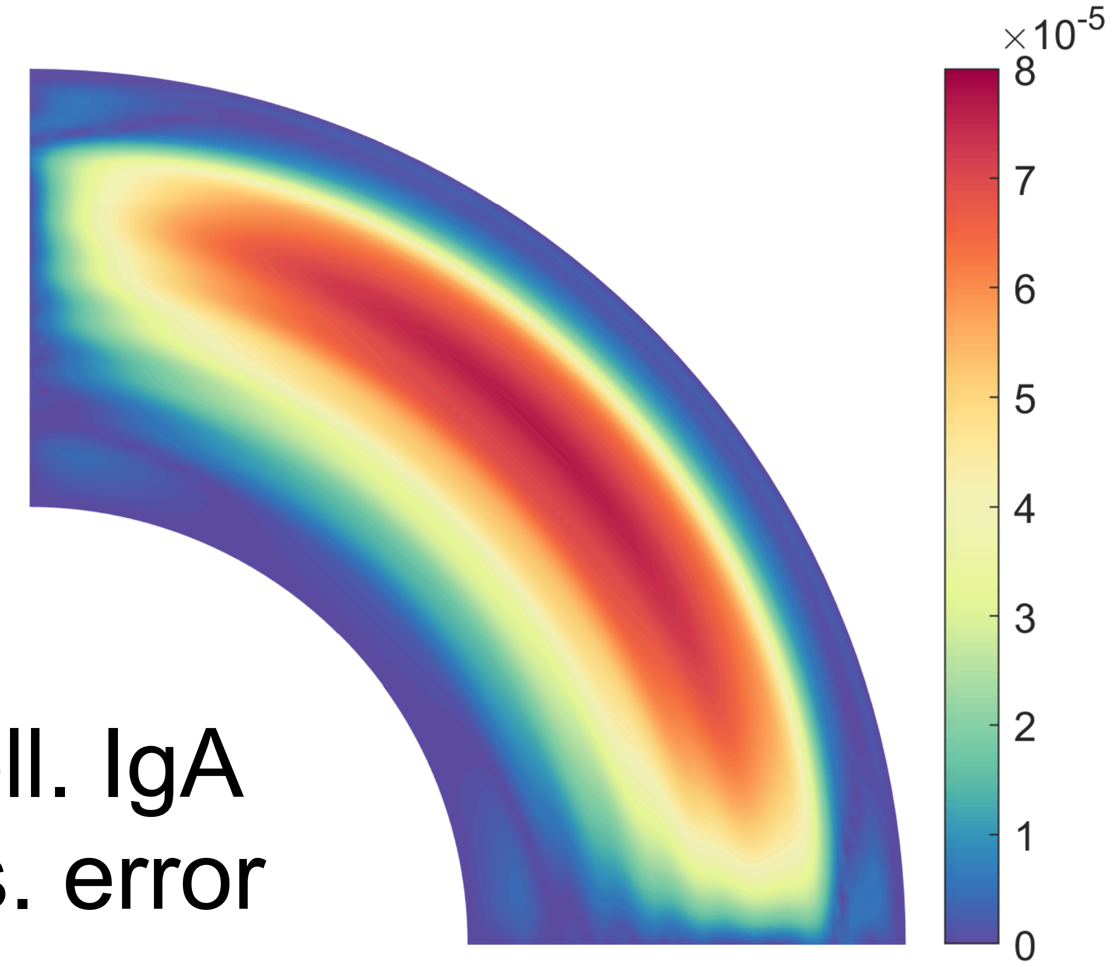
solution



Galerkin IgA
abs. error



Greville pts



Coll. IgA
abs. error

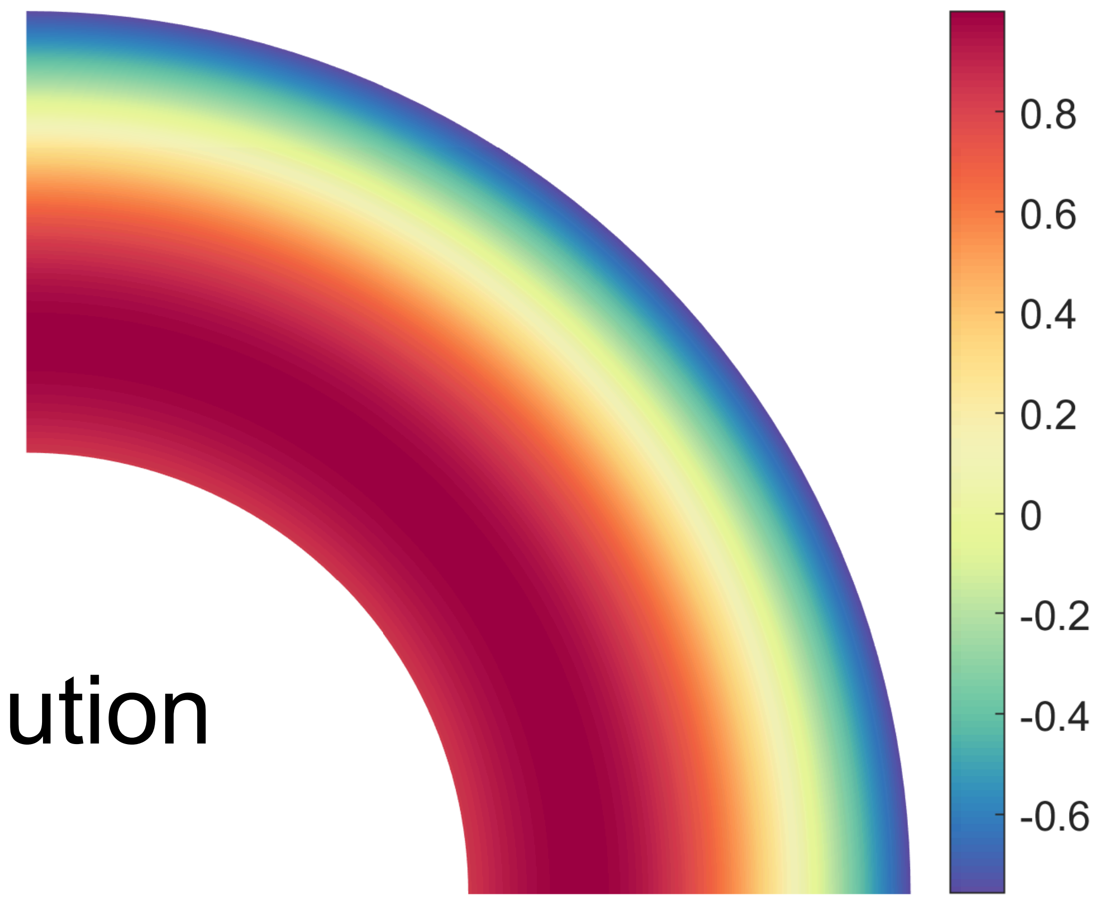
M. Wang, 2023

Galerkin vs. collocation IgA

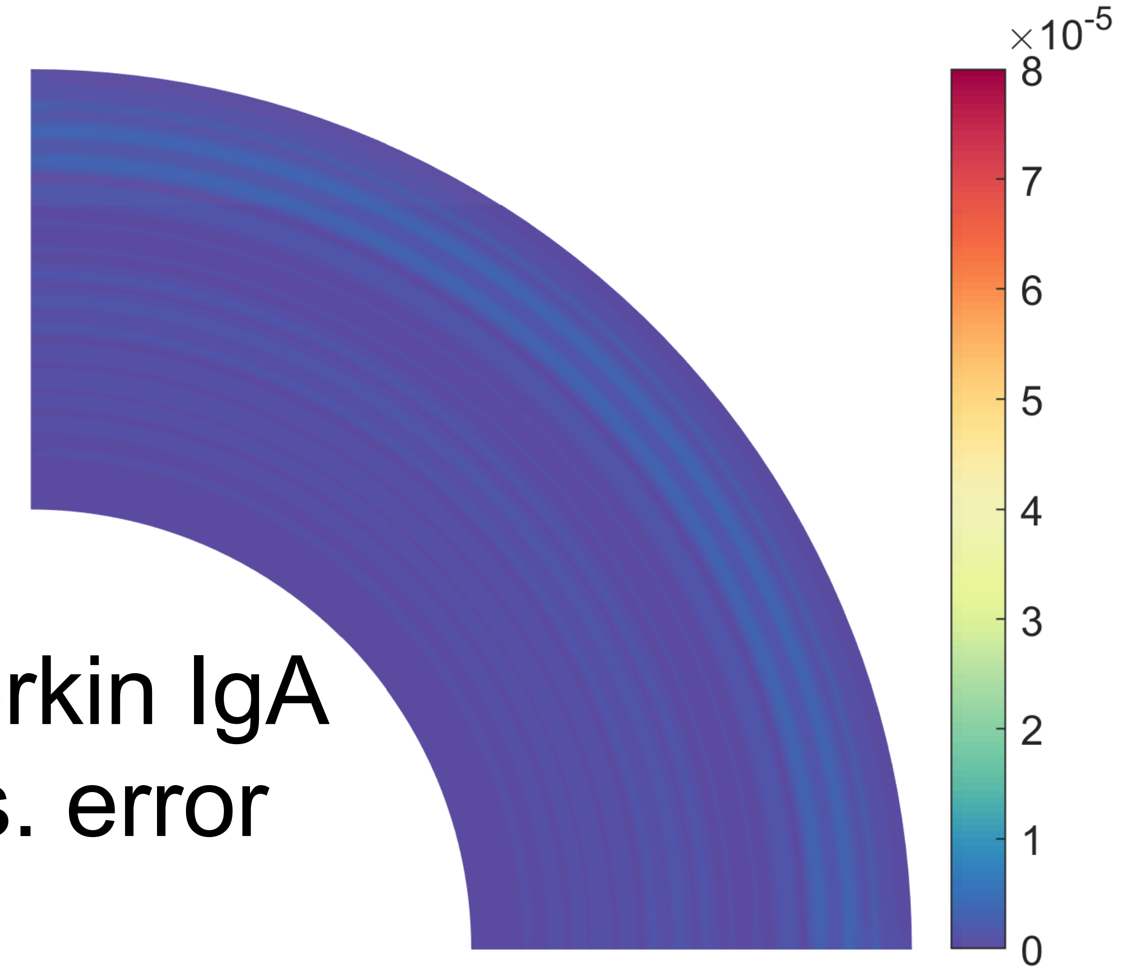
Model problem

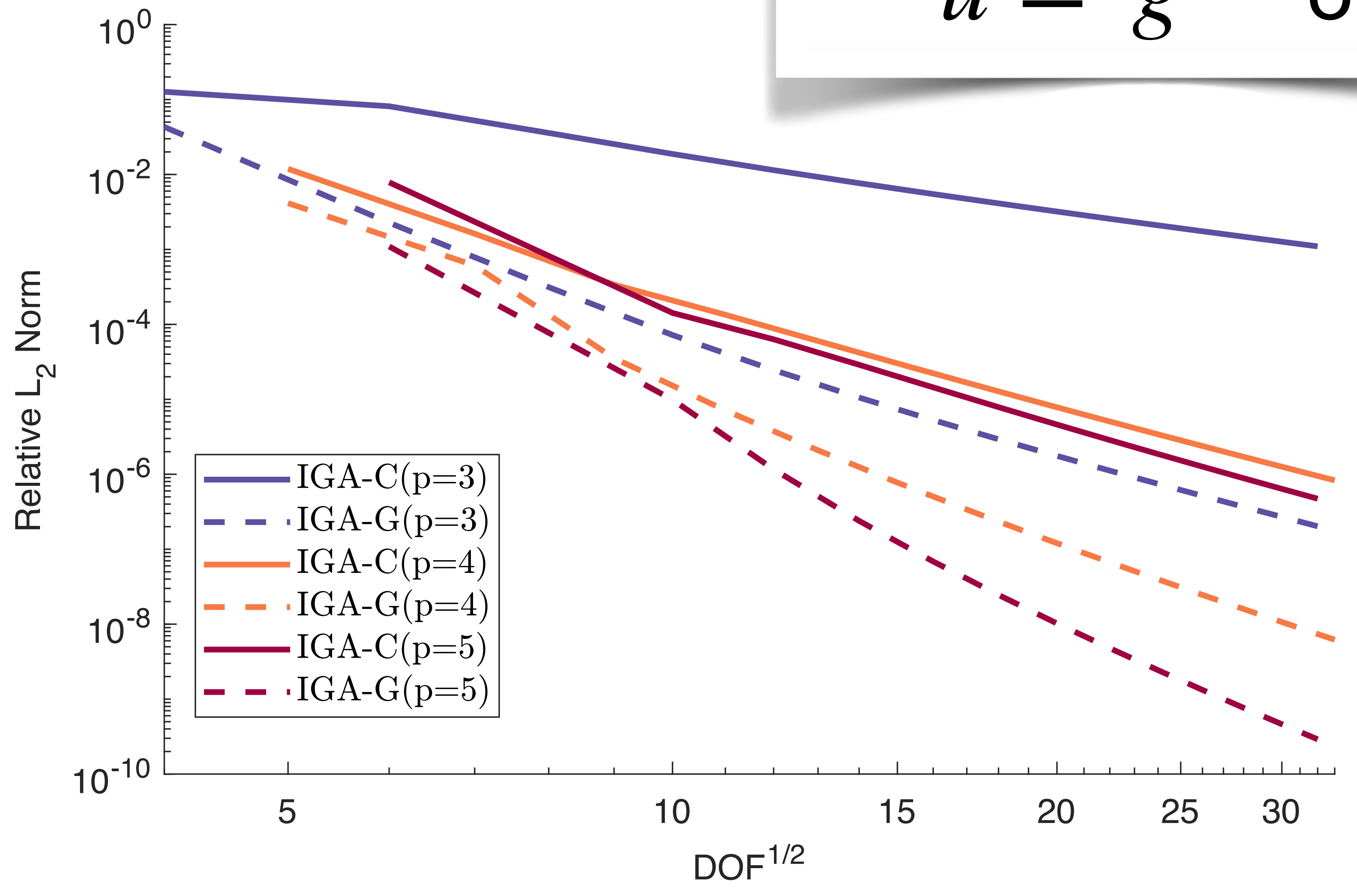
$$-\Delta u = f \quad \text{in } \Omega$$
$$u = g \quad \text{on } \Gamma$$

solution

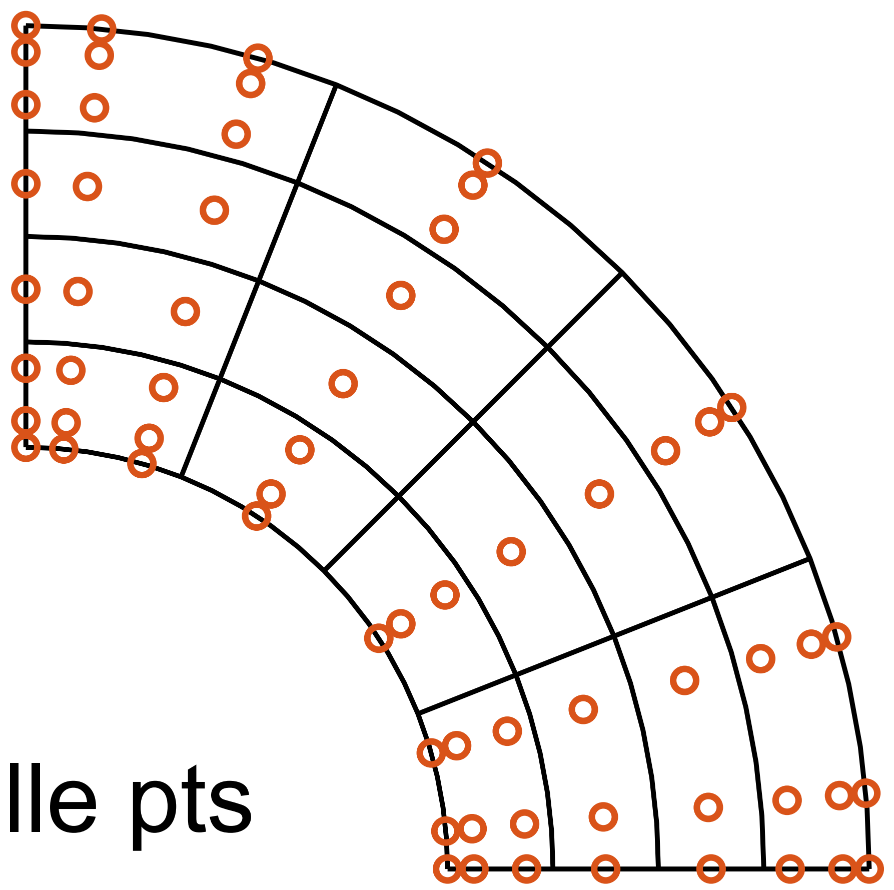


Galerkin IgA
abs. error

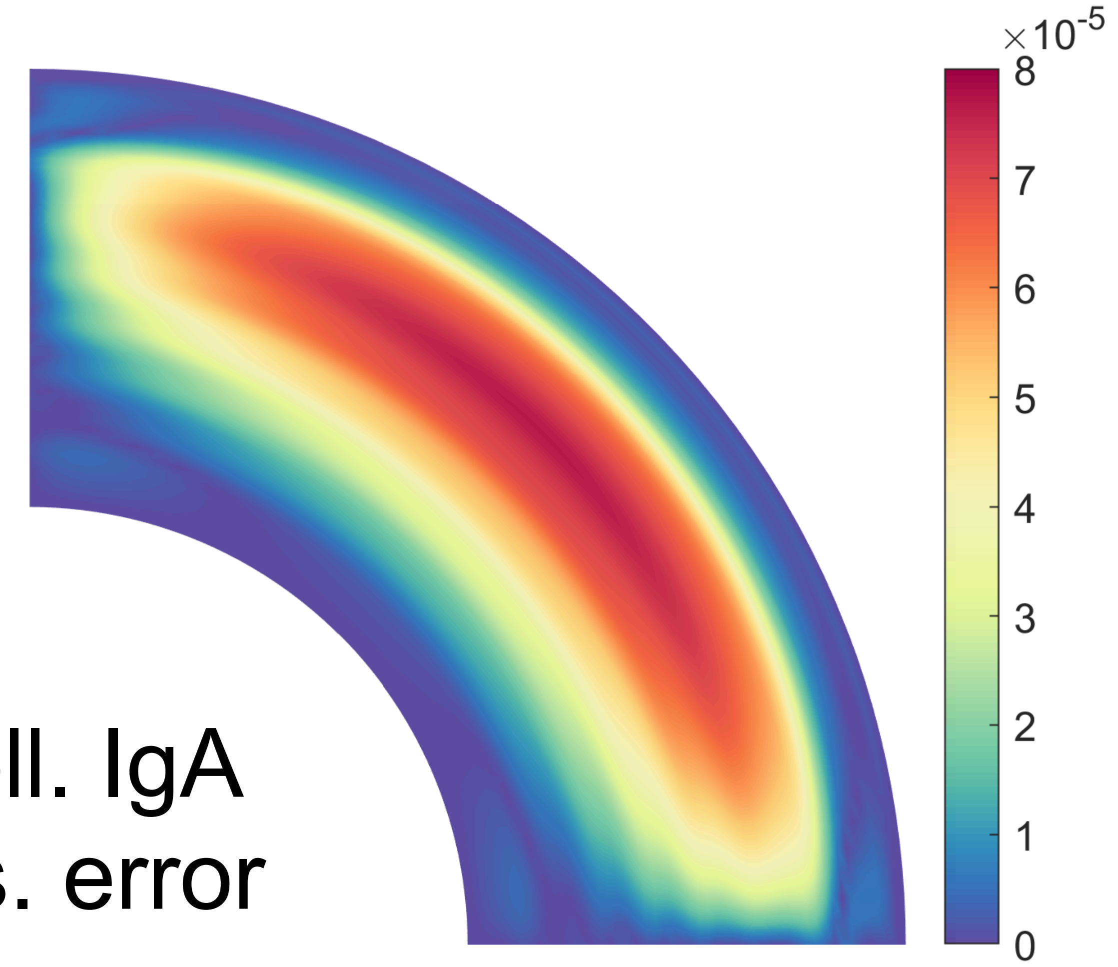


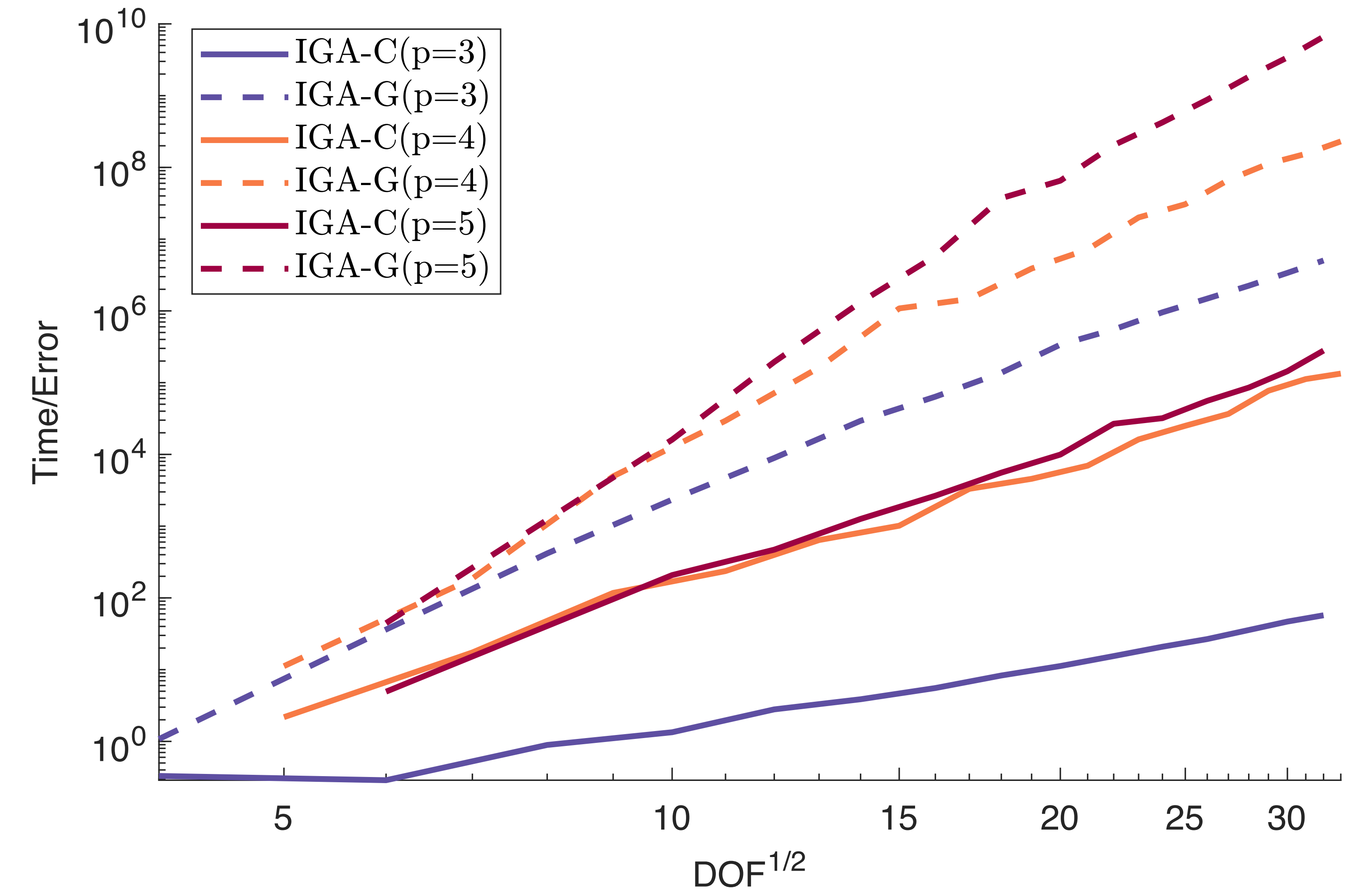


Greville pts



Coll. IgA
abs. error



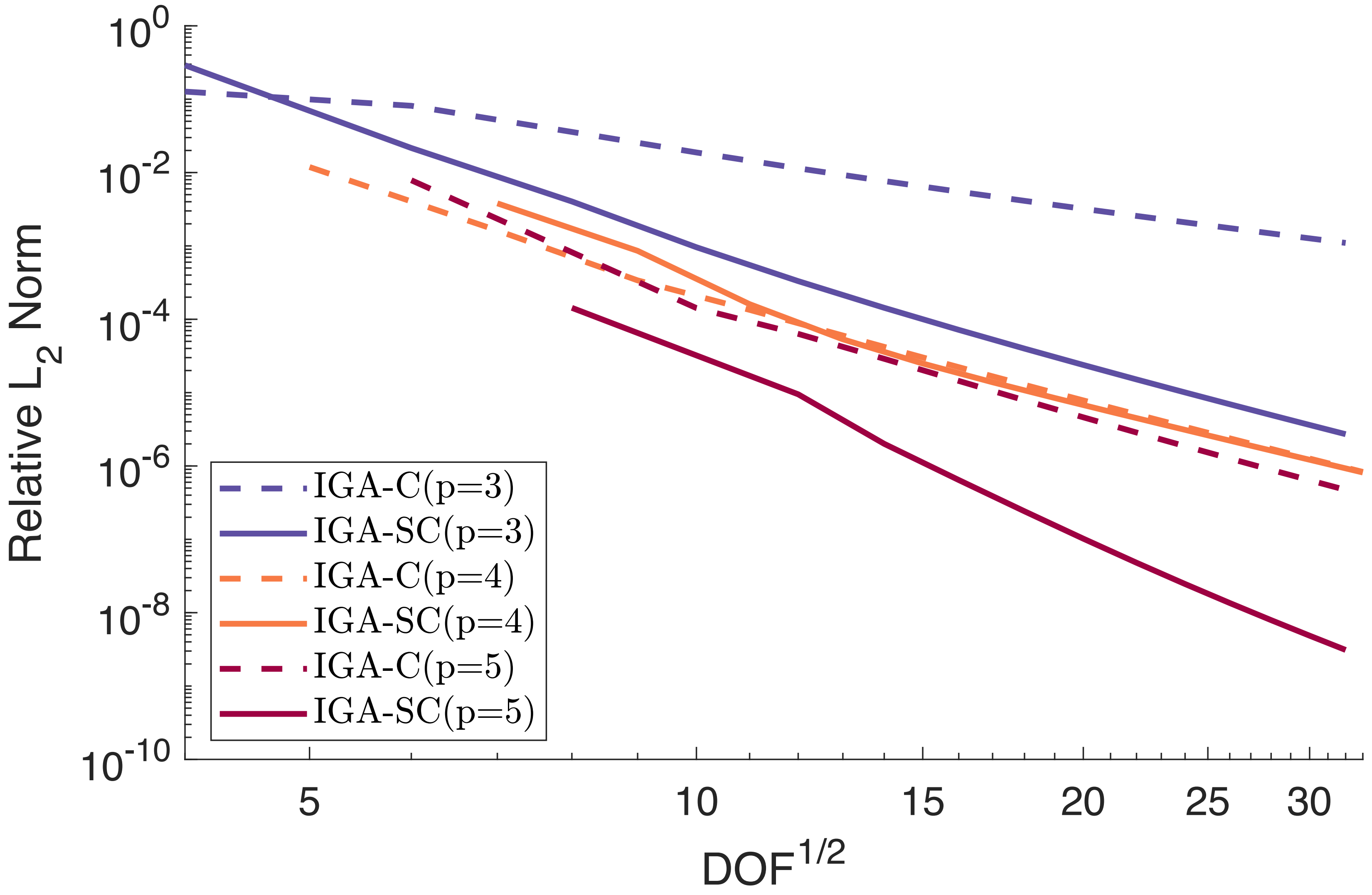


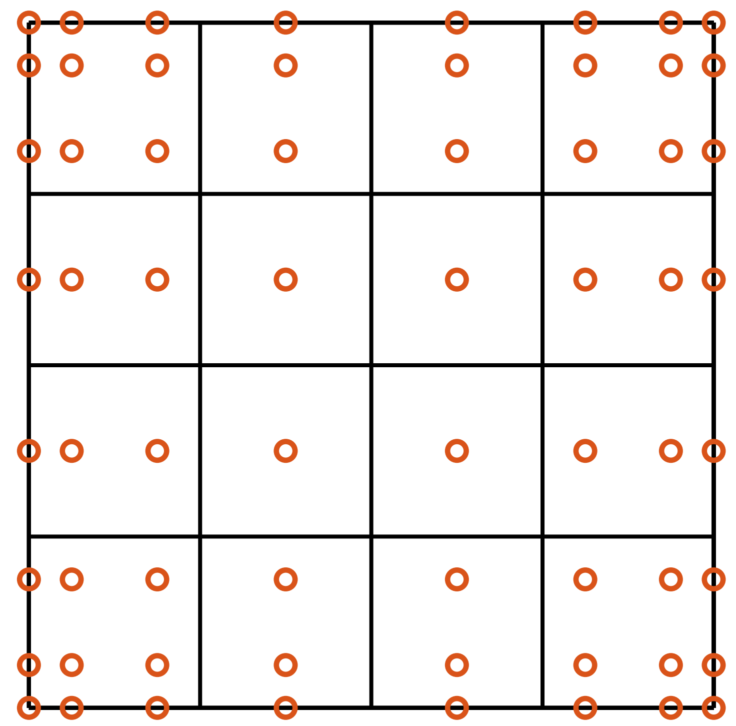
M. Wang, 2023

Influence of collocation points

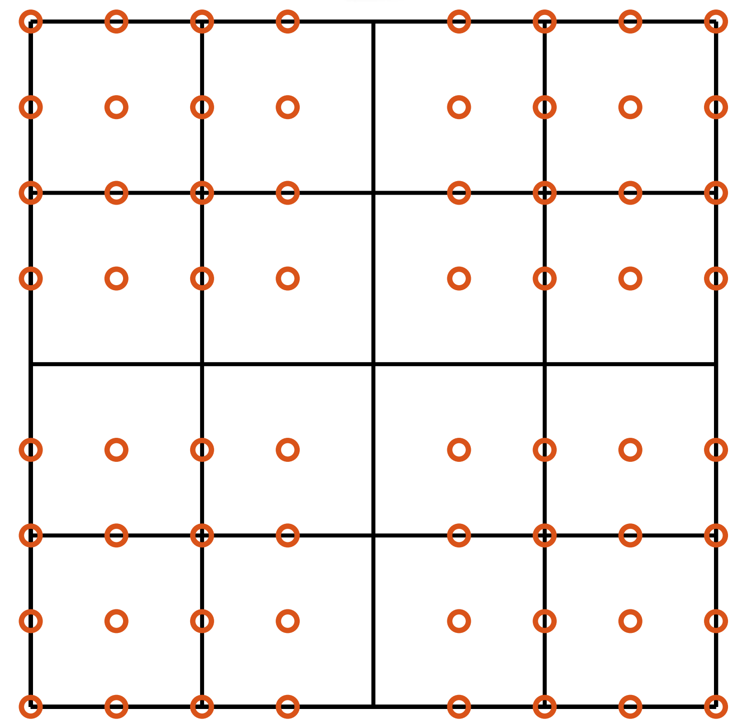
Model problem

$$
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega \\
u &= g \quad \text{on } \Gamma
\end{aligned}
$$

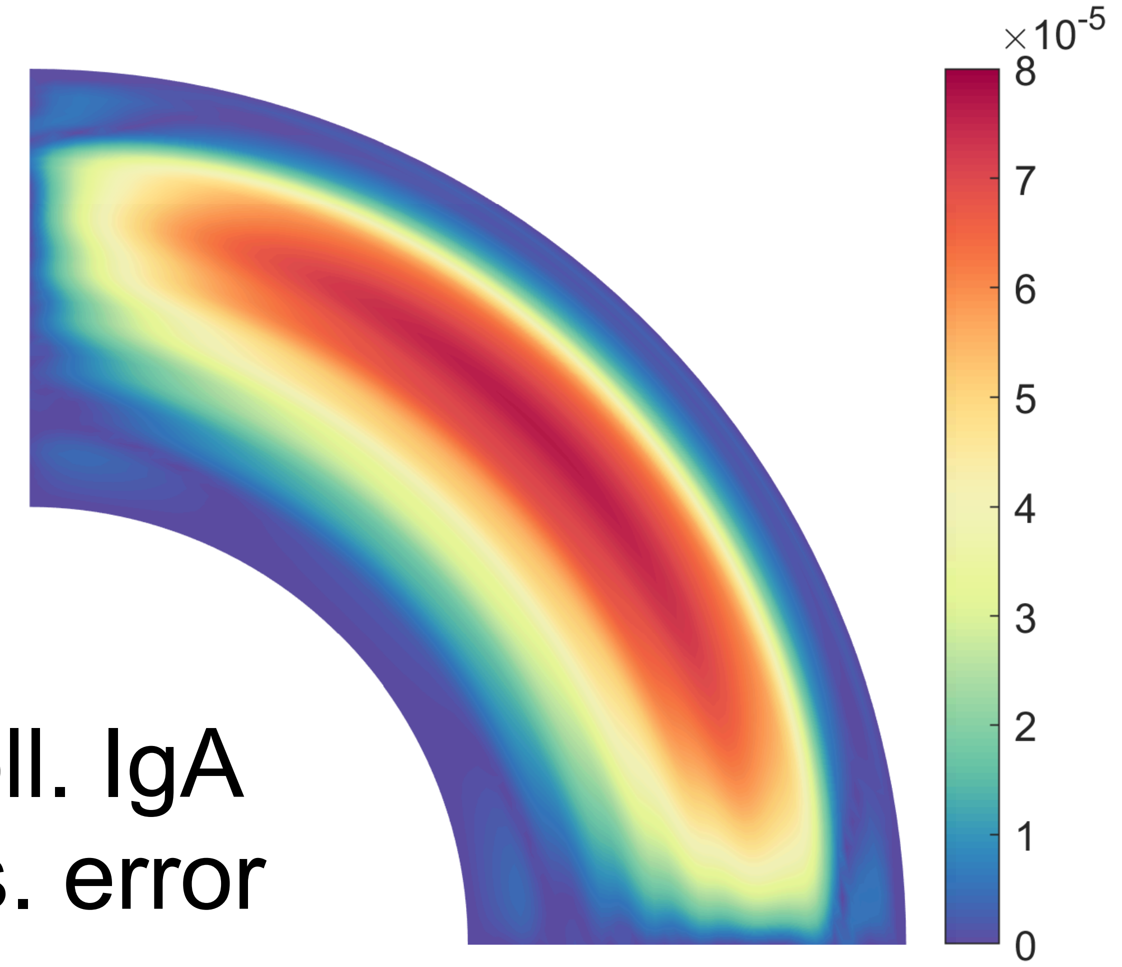




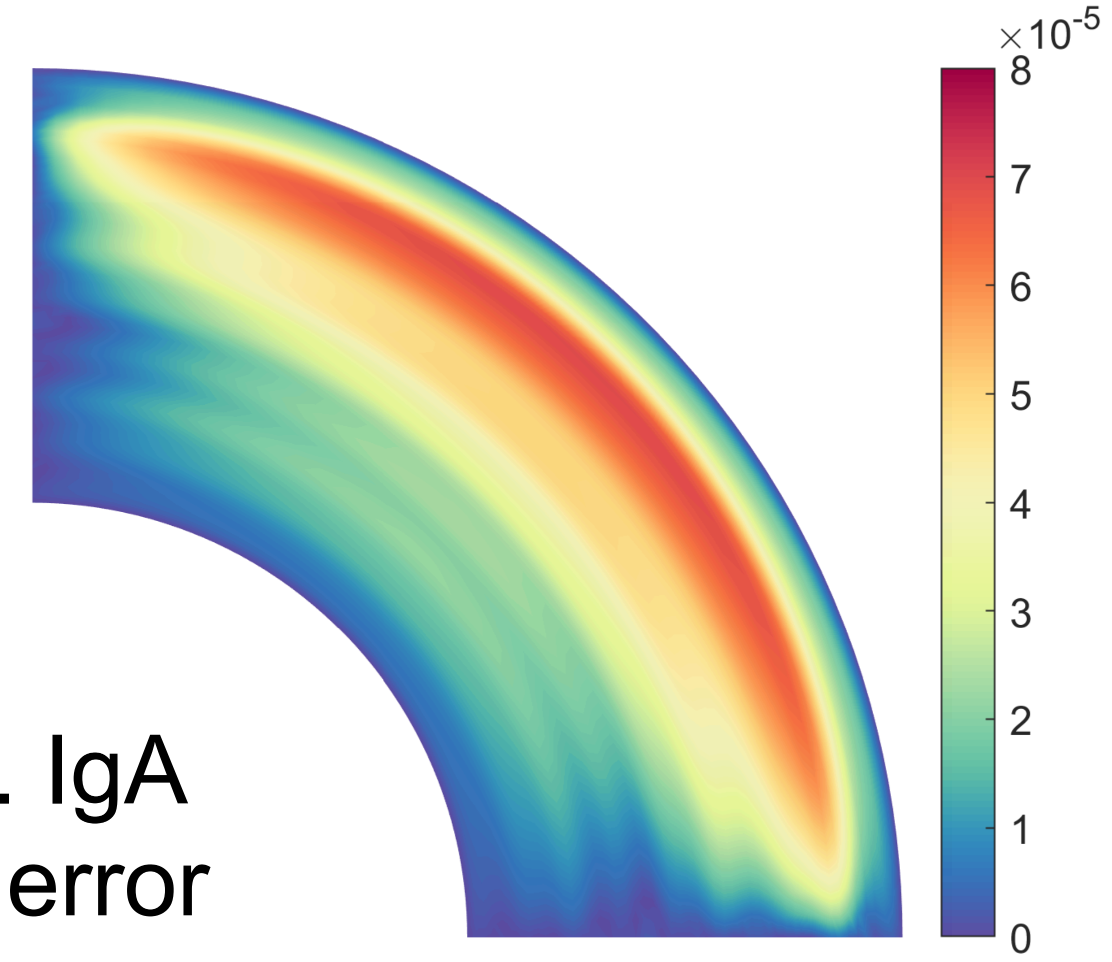
Greville pts



Clustered SC pts



Coll. IgA abs. error



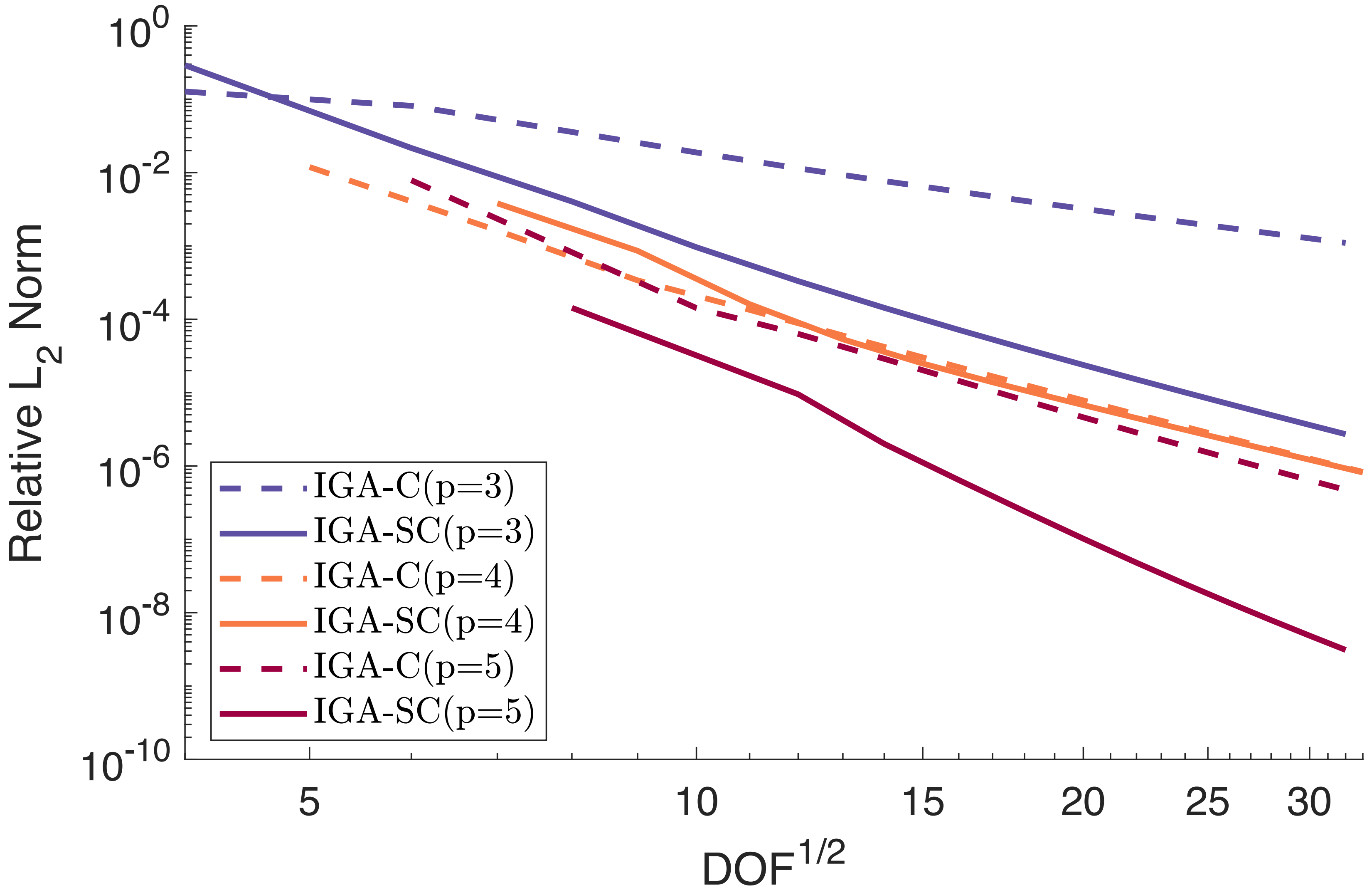
Coll. IgA abs. error

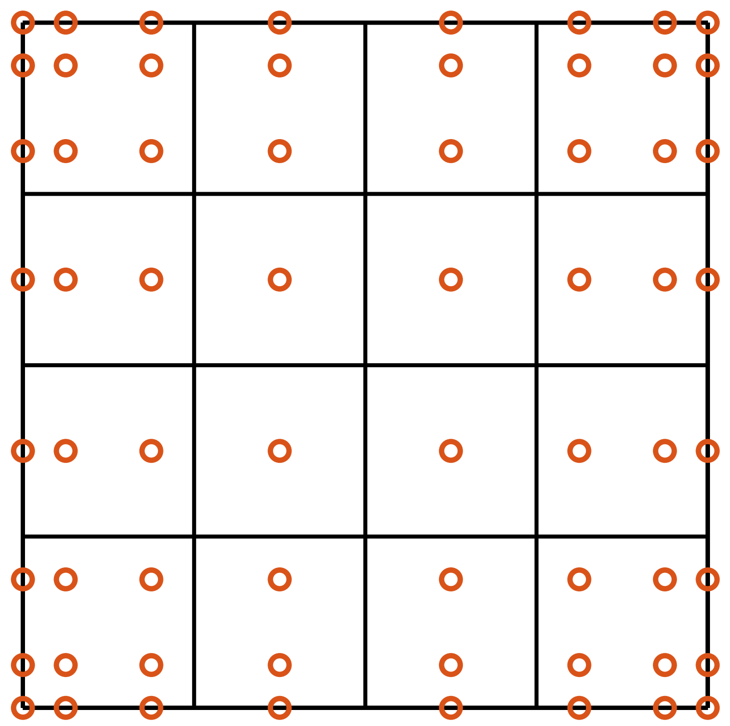
M. Wang, 2023

Influence of collocation points

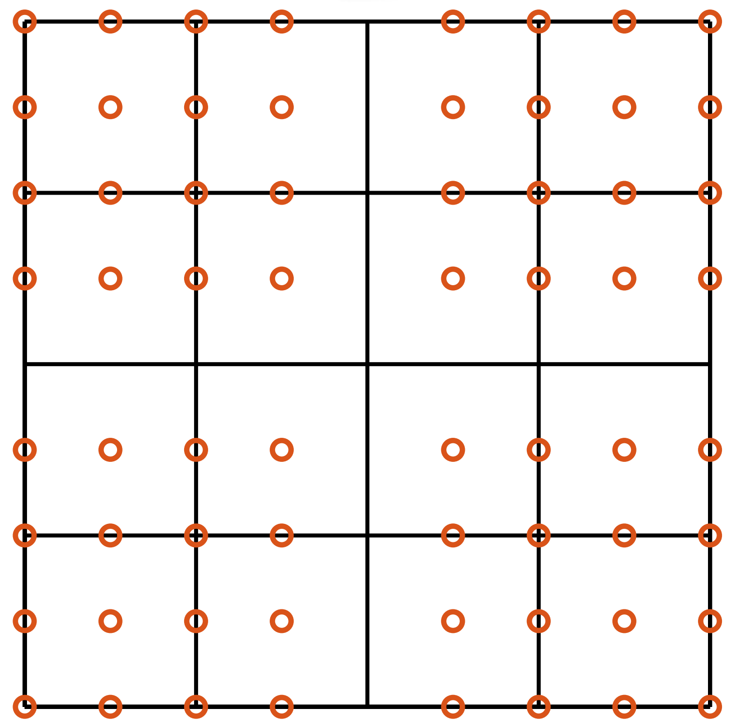
Model problem

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

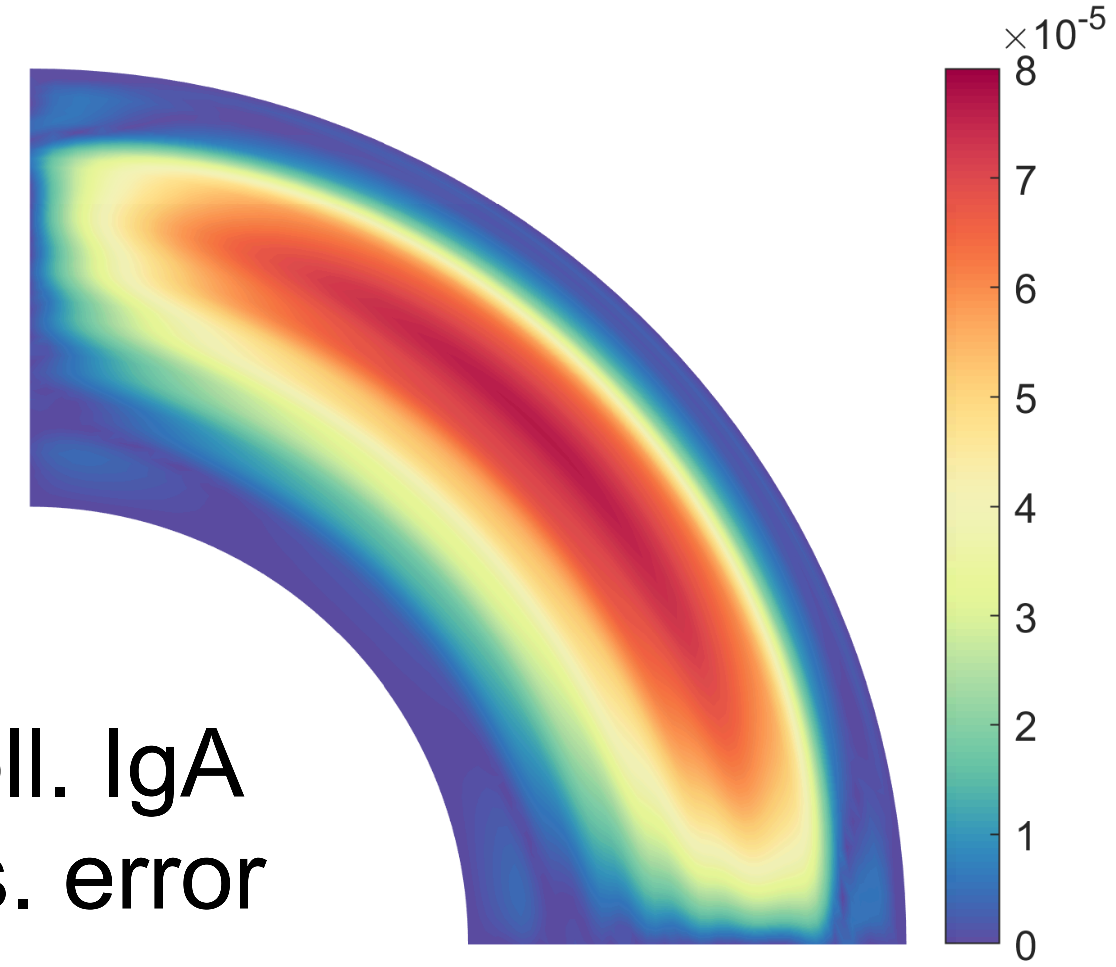




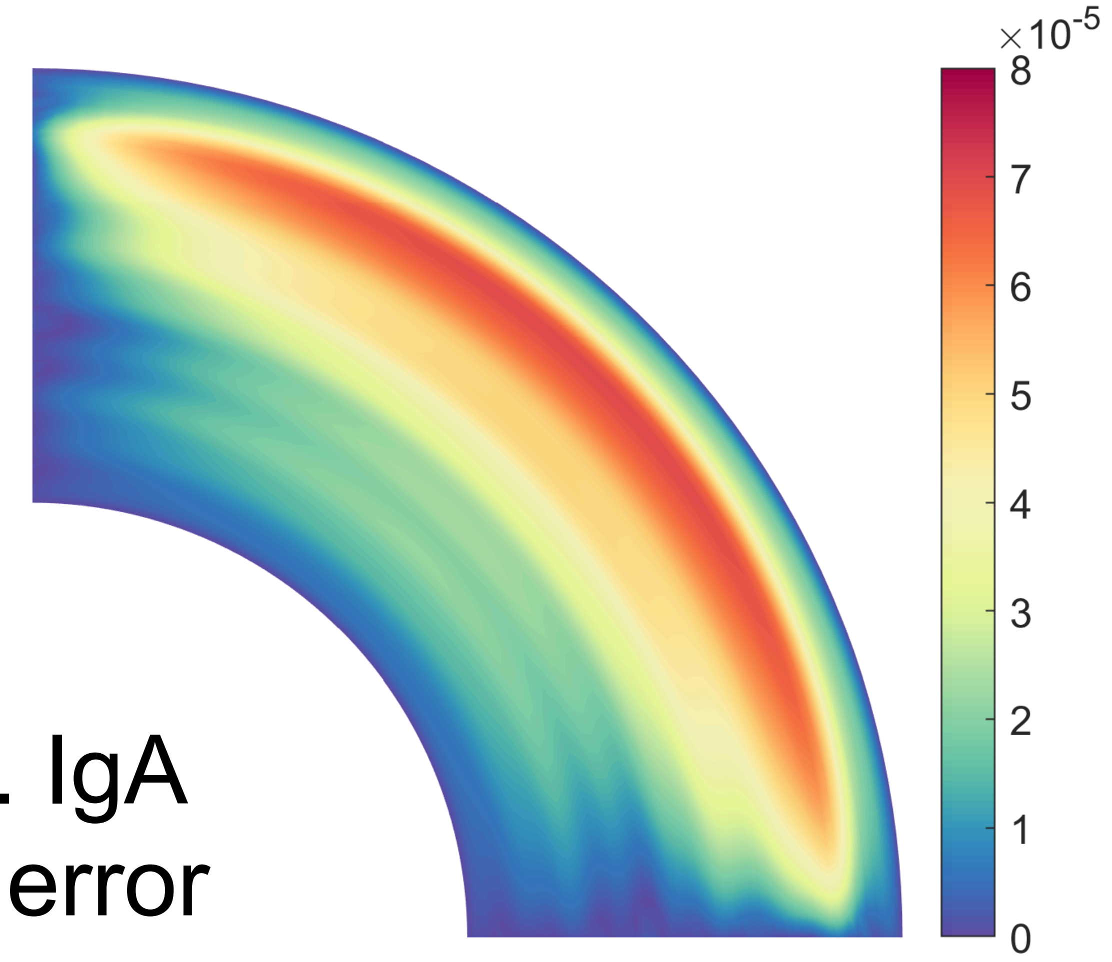
Greville pts



Clustered SC pts



Coll. IgA abs. error



Coll. IgA abs. error

As finding the **optimal position of collocation points** is difficult, can't we instead increase their number to reduce the error?

M. Wang, 2023

Least-squares collocation IgA

When #collocation points ($m_1 + m_2$) > #degrees of freedom (n) then the system matrix is over-determined and the system cannot be solved regularly. However, we can solve it in **least-squares sense**

$$\min_{u_h} \frac{1}{m_1} \sum_{i=1}^{m_1} \|\mathscr{L}_{\mathbf{x}} u_h(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \|\mathscr{B}_{\mathbf{x}} u_h(\mathbf{x}_i) - g(\mathbf{x}_i)\|^2$$

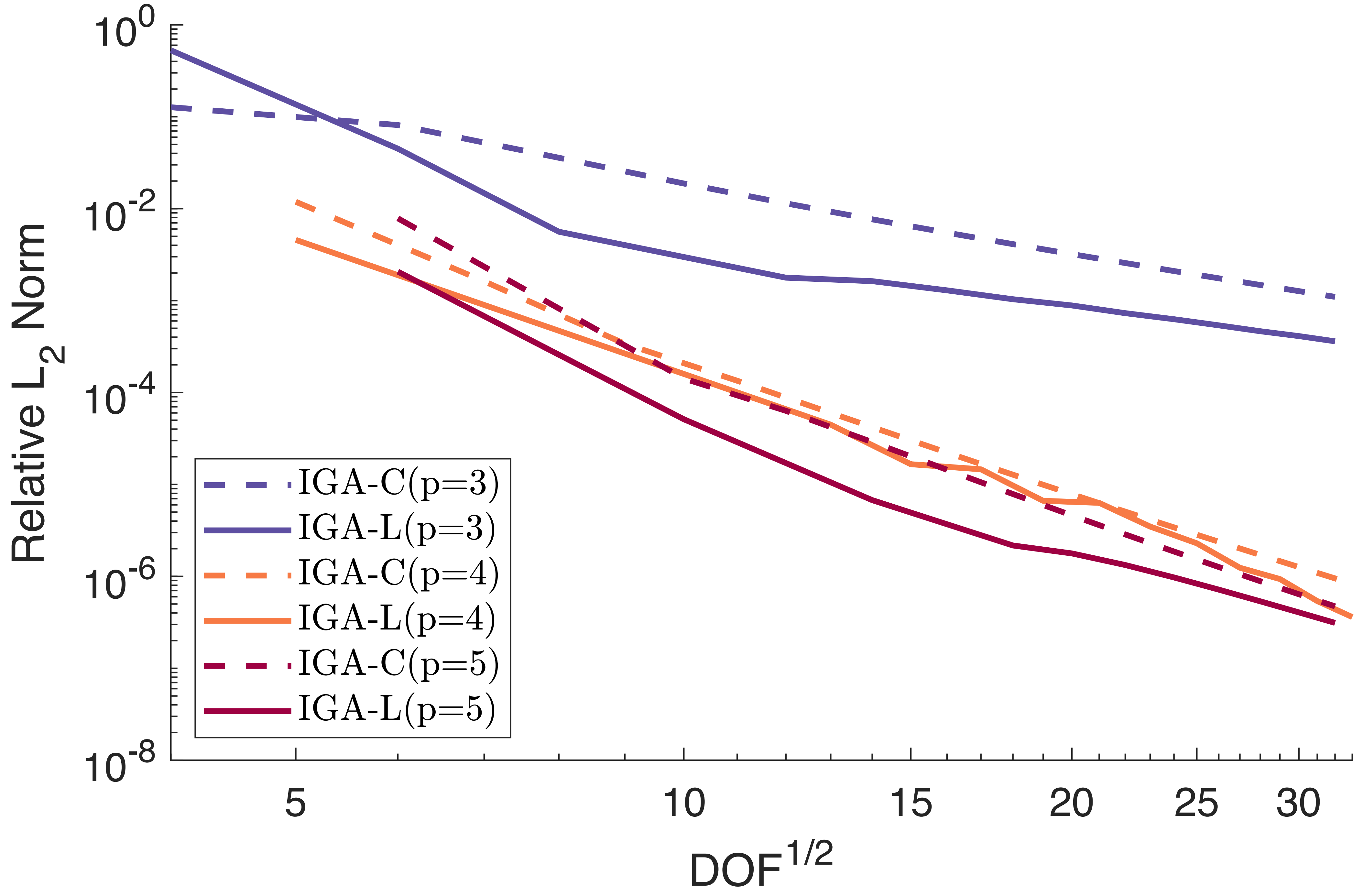
Least-squares collocation IgA

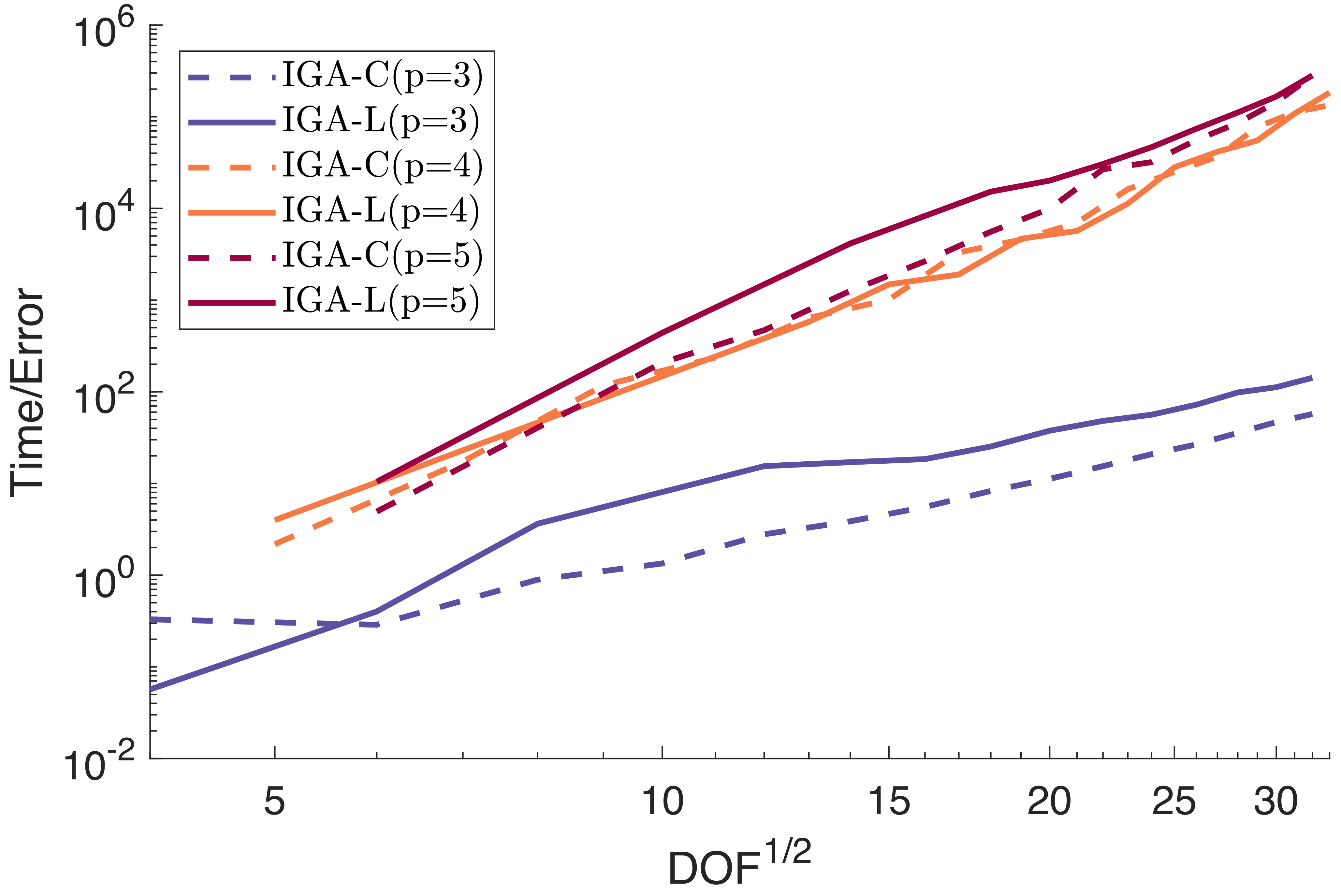
When #collocation points $(m_1 + m_2)$ > #degrees of freedom (n) then the system matrix is over-determined and the system cannot be solved regularly. However, we can solve it in **least-squares sense**

$$\min_{u_h} \frac{1}{m_1} \sum_{i=1}^{m_1} \|\mathscr{L}_{\mathbf{x}} u_h(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \|\mathscr{B}_{\mathbf{x}} u_h(\mathbf{x}_i) - g(\mathbf{x}_i)\|^2$$

Theoretical justification [Lin et al. 2020]: under certain conditions, the least-squares collocation IgA method is consistent and convergent. In essence, there must be at *least one collocation point per element* (e.g., Greville point) but we can use more to increase the sampling resolution.

Least-squares collocation IgA





From least-squares collocation IgA to IgANets

$$\min_{u_h} \frac{1}{m_1} \sum_{i=1}^{m_1} \|\mathscr{L}_{\mathbf{x}} u_h(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \|\mathscr{B}_{\mathbf{x}} u_h(\mathbf{x}_i) - g(\mathbf{x}_i)\|^2$$

From least-squares collocation IgA to IgANets

$$\min_{u_h} \frac{1}{m_1} \sum_{i=1}^{m_1} \| \mathscr{L}_{\mathbf{x}} u_h(\mathbf{x}_i) - f(\mathbf{x}_i) \|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \| \mathscr{B}_{\mathbf{x}} u_h(\mathbf{x}_i) - g(\mathbf{x}_i) \|^2$$

$$\min_{\{u_j\}_j} \frac{1}{m_1} \sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) f_j \|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) g_j \|^2$$

From least-squares collocation IgA to IgANets

$$\min_{u_h} \frac{1}{m_1} \sum_{i=1}^{m_1} \| \mathscr{L}_{\mathbf{x}} u_h(\mathbf{x}_i) - f(\mathbf{x}_i) \|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \| \mathscr{B}_{\mathbf{x}} u_h(\mathbf{x}_i) - g(\mathbf{x}_i) \|^2$$

$$\min_{\{u_j\}_j} \frac{1}{m_1} \sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) f_j \|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) g_j \|^2$$

$$\mathbf{x}_h(\boldsymbol{\xi}) = \sum_{j=1}^{n} b_j(\boldsymbol{\xi})\, \mathbf{x}_j$$

From least-squares collocation IgA to IgANets

$$\min_{u_h} \frac{1}{m_1} \sum_{i=1}^{m_1} \|\mathscr{L}_{\mathbf{x}} u_h(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \|\mathscr{B}_{\mathbf{x}} u_h(\mathbf{x}_i) - g(\mathbf{x}_i)\|^2$$

$$\min_{\boxed{\{u_j\}_j}} \frac{1}{m_1} \sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) \boxed{f_j} \|^2 + \frac{1}{m_2} \sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) \boxed{g_j} \|^2$$

solution

rhs

bc cond.

$$\mathbf{x}_h(\boldsymbol{\xi}) = \sum_{j=1}^{n} b_j(\boldsymbol{\xi}) \boxed{\mathbf{x}_j}$$

geometry

From least-squares collocation IgA to deep operator learning

$$\min_{\{u_j\}_j} \boxed{\frac{1}{m_1}\sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\xi} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) f_j \|^2} + \boxed{\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\xi} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) g_j \|^2}$$

PDE loss function

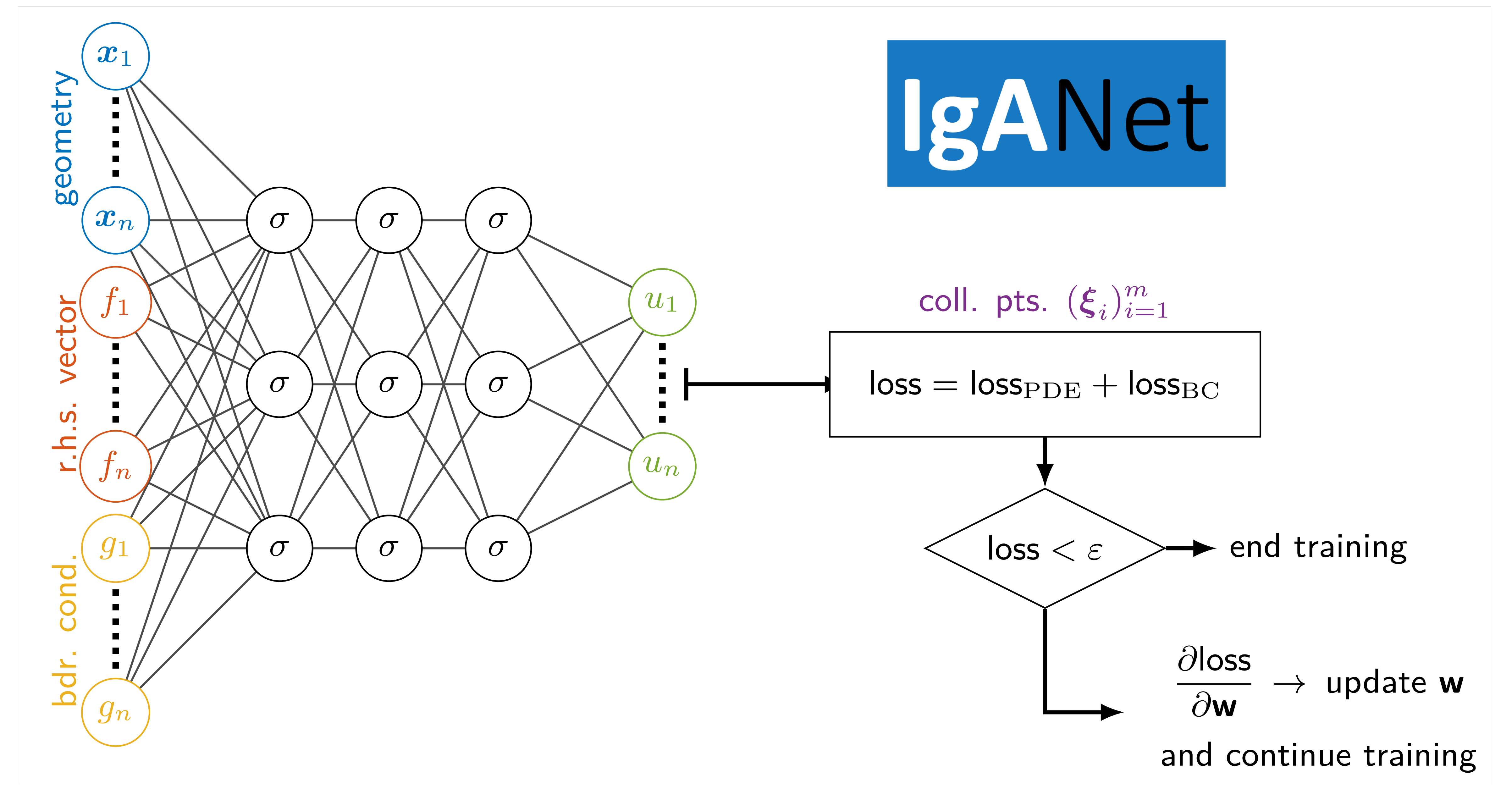
boundary loss function

From least-squares collocation IgA to deep operator learning

$$\min_{\{u_j\}_j} \boxed{\frac{1}{m_1}\sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i)u_j - b_j(\boldsymbol{\xi}_i)f_j \|^2} + \boxed{\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i)u_j - b_j(\boldsymbol{\xi}_i)g_j \|^2}$$

PDE loss function

boundary loss function

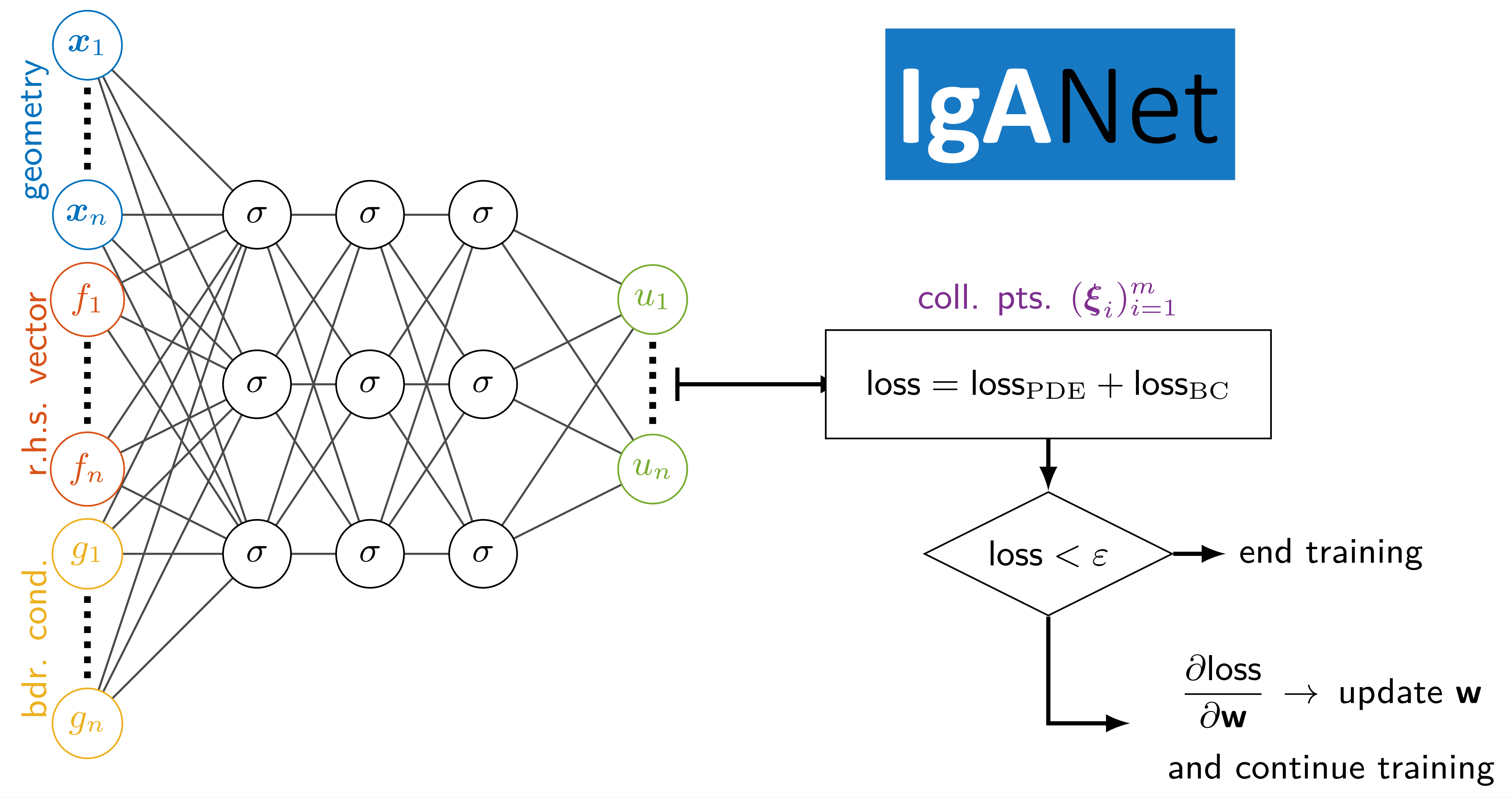


From least-squares collocation IgA to deep operator learning

$$\min_{\{u_j\}_j} \boxed{\frac{1}{m_1}\sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\xi} b_j(\xi_i) u_j - b_j(\xi_i) f_j \|^2} + \boxed{\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\xi} b_j(\xi_i) u_j - b_j(\xi_i) g_j \|^2}$$

PDE loss function

boundary loss function



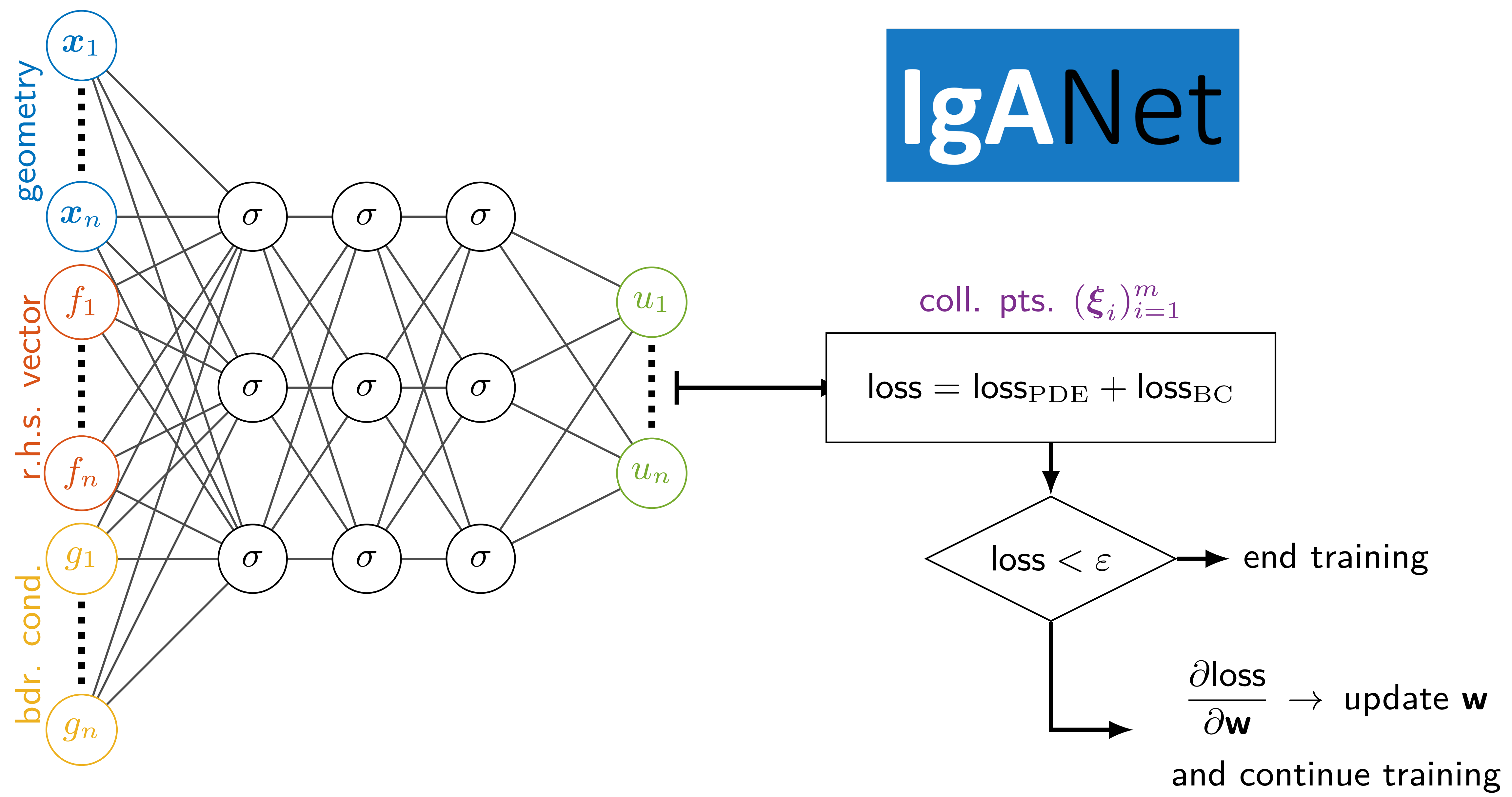
How is this related to PINNs, DeepONets, etc.?

From least-squares collocation IgA to deep operator learning

$$\min_{\{u_j\}_j} \boxed{\frac{1}{m_1}\sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) f_j \|^2} + \boxed{\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i) u_j - b_j(\boldsymbol{\xi}_i) g_j \|^2}$$

PDE loss function

boundary loss function



How is this related to PINNs, DeepONets, etc.?

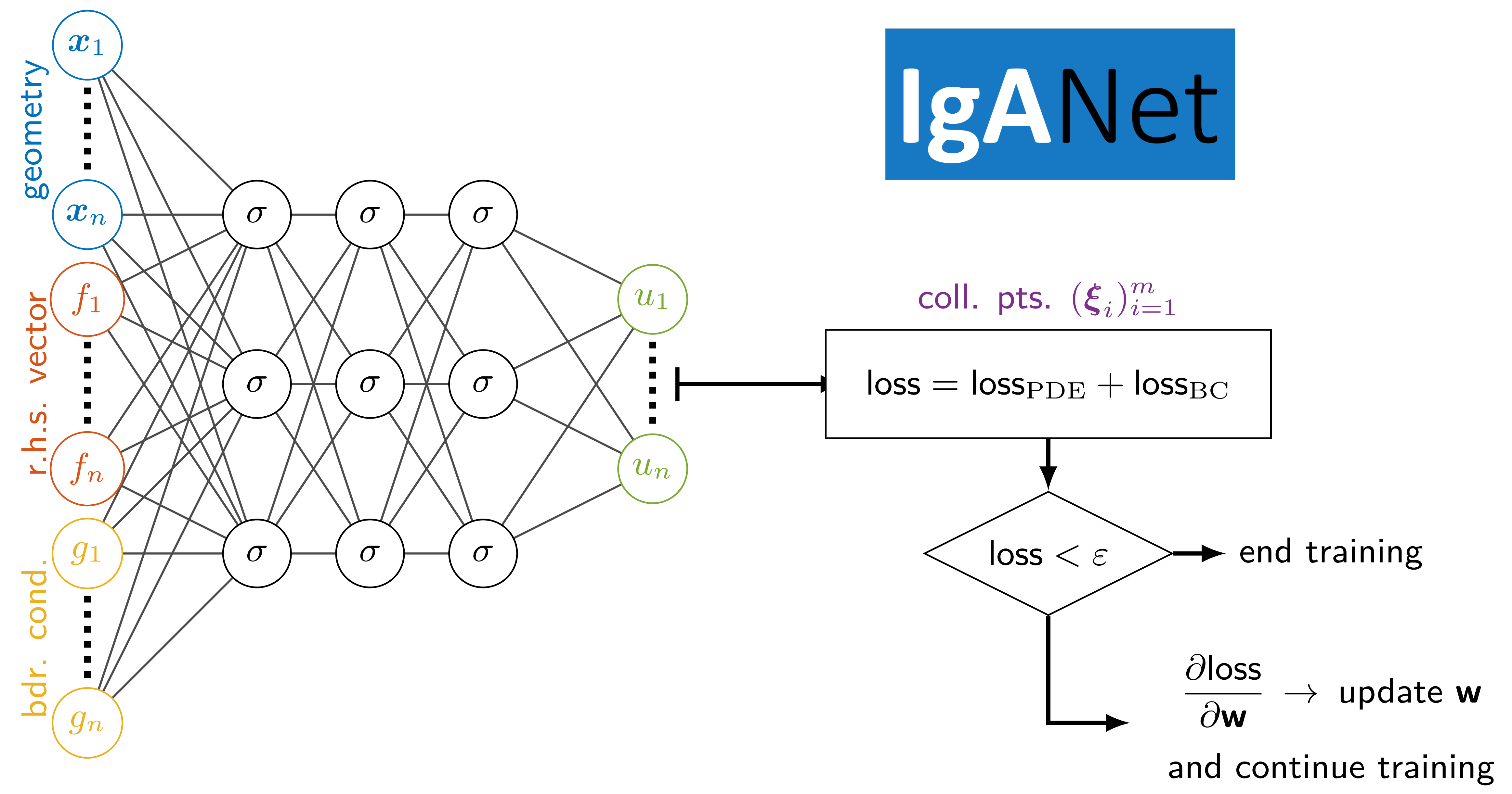
Can be seen as deep operator network with *given* B-spline/NURBS basis!

From least-squares collocation IgA to deep operator learning

$$\min_{\{u_j\}_j} \boxed{\frac{1}{m_1}\sum_{i=1}^{m_1}\| \sum_{j=1}^{n} \mathscr{L}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i)u_j - b_j(\boldsymbol{\xi}_i)f_j\|^2} + \boxed{\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2}\| \sum_{j=1}^{n} \mathscr{B}_{\boldsymbol{\xi}} b_j(\boldsymbol{\xi}_i)u_j - b_j(\boldsymbol{\xi}_i)g_j\|^2}$$

PDE loss function

boundary loss function



How is this related to PINNs, DeepONets, etc.?

Can be seen as deep operator network with *given* B-spline/NURBS basis!

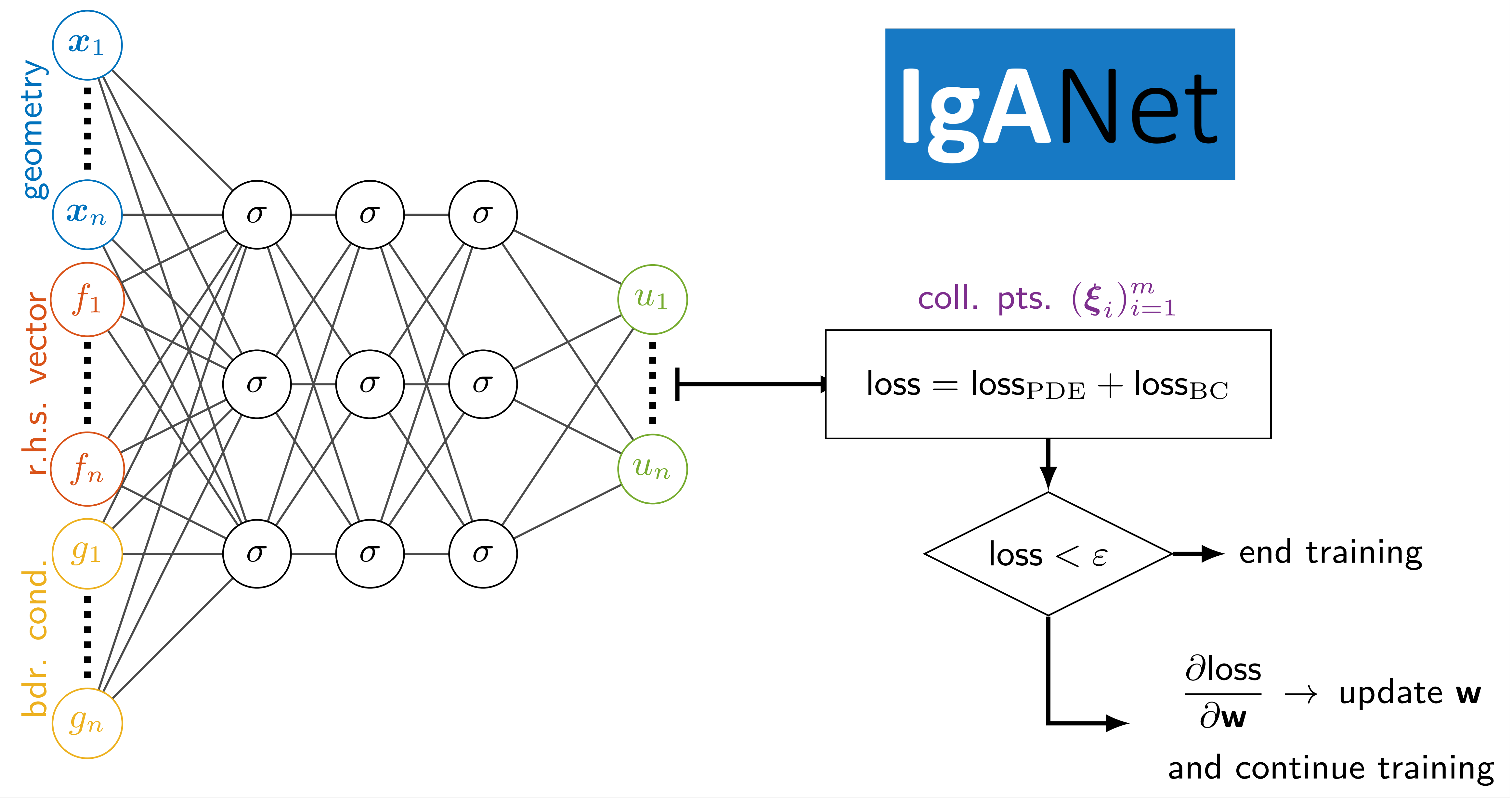
Do we need to separate training from inference?

From least-squares collocation IgA to deep operator learning

$$\min_{\{u_j\}_j} \boxed{\frac{1}{m_1}\sum_{i=1}^{m_1} \| \sum_{j=1}^{n} \mathscr{L}_{\xi} b_j(\xi_i) u_j - b_j(\xi_i) f_j \|^2} + \boxed{\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2} \| \sum_{j=1}^{n} \mathscr{B}_{\xi} b_j(\xi_i) u_j - b_j(\xi_i) g_j \|^2}$$

PDE loss function

boundary loss function



How is this related to PINNs, DeepONets, etc.?

Can be seen as deep operator network with *given* B-spline/NURBS basis!

Do we need to separate training from inference?

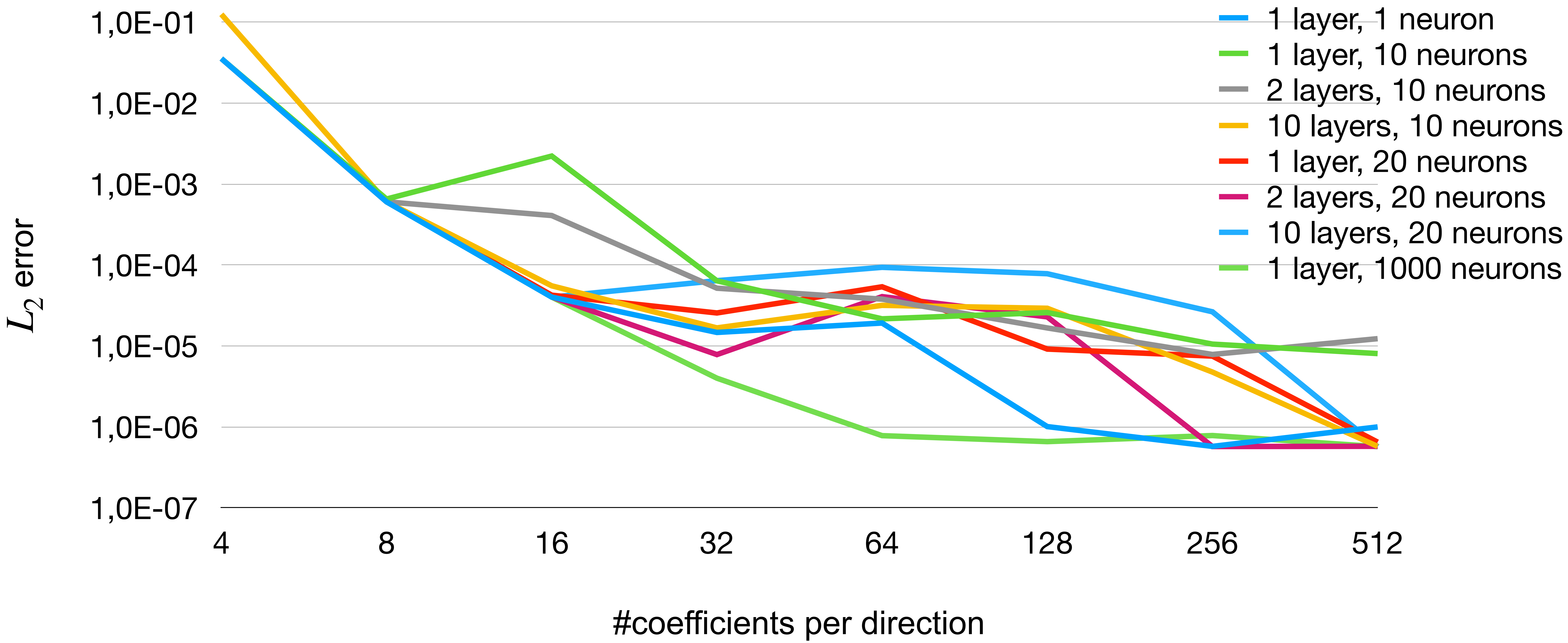
No, the network learns on the fly. But we can also pre-train it.

IgANet

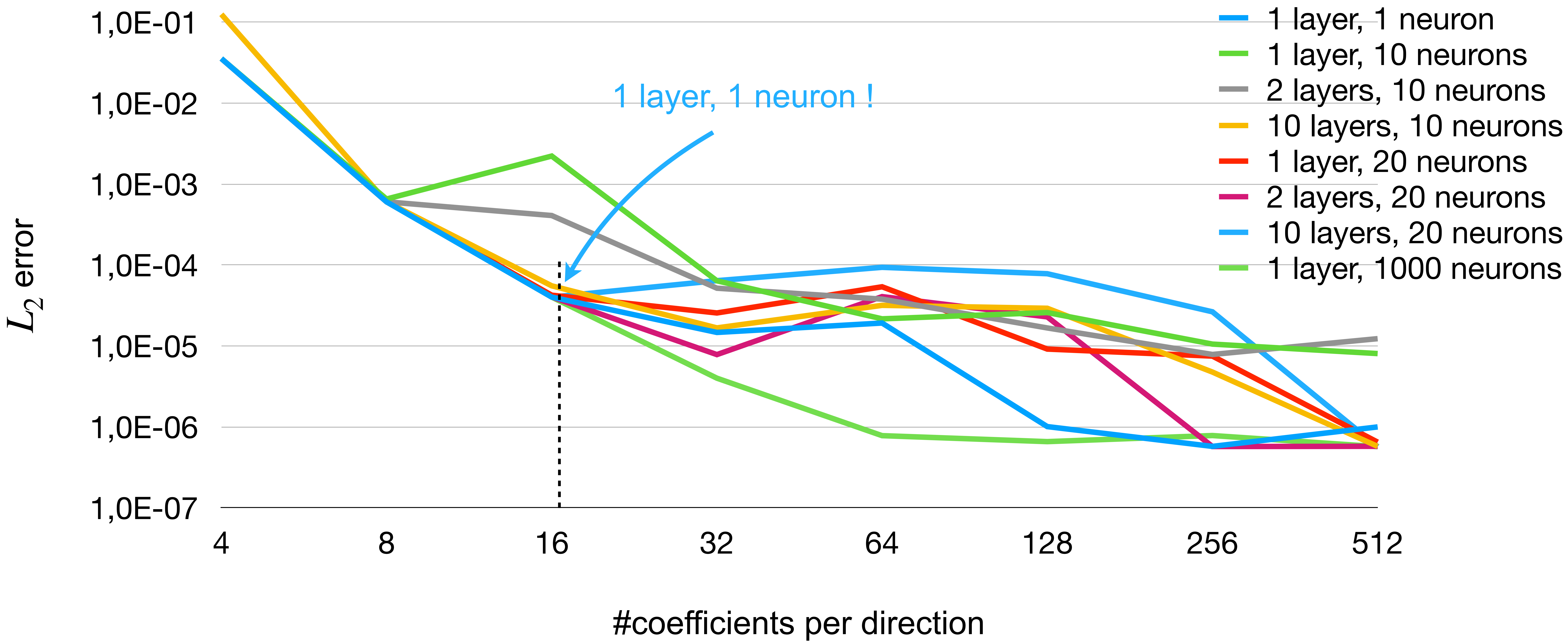
GitHub.com/IgANets

- C++17 (soon open-source) library implemented atop LibTorch 2.x (C++ API of PyTorch)
- Dimension-independent B-splines and NURBS and a customisable IgANets deep learning framework
- Distributed computing on CPUs and GPUs (NVIDIA & AMD, Google TPUs WIP)
- Coupled with G+Smo (Geometry plus Simulation Modules) as IgA reference library and as toolkit with surface and volume reparameterization techniques

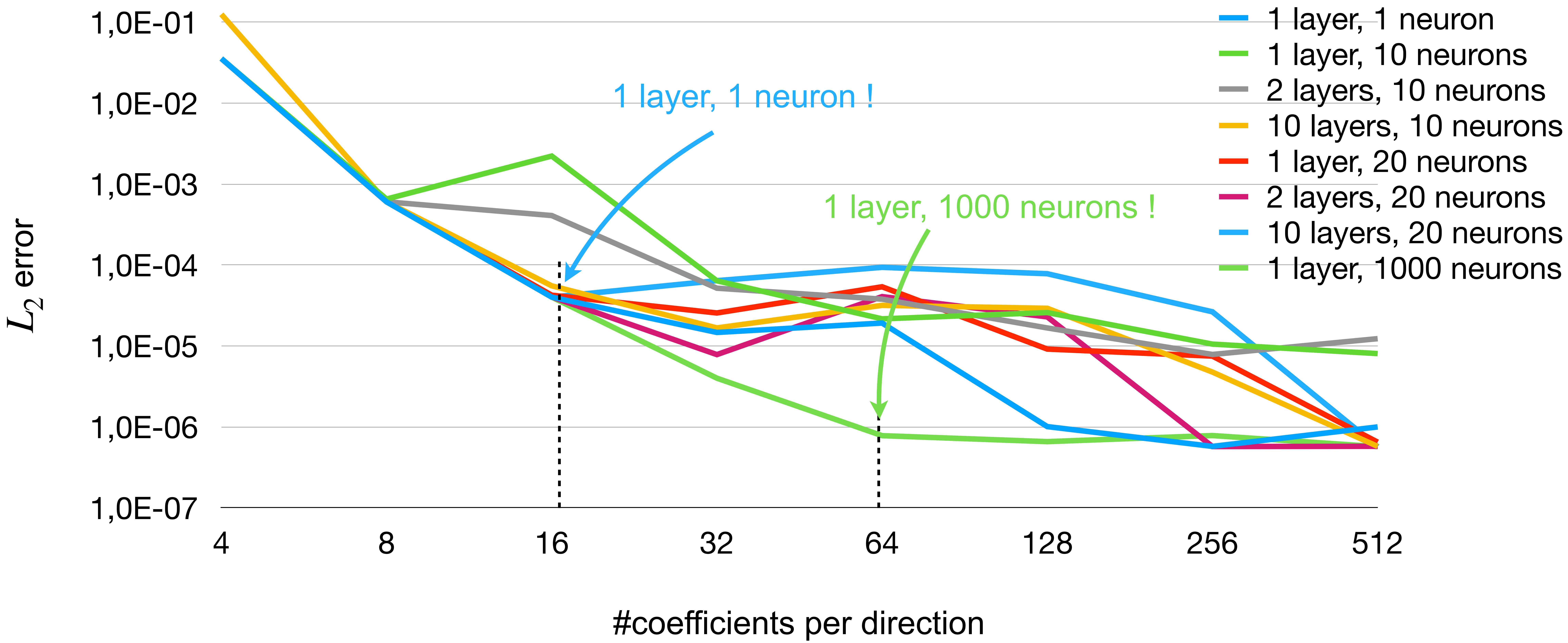
Hyper-parameter tuning (loss function tolerance 10^{-12})



Hyper-parameter tuning (loss function tolerance 10^{-12})



Hyper-parameter tuning (loss function tolerance 10^{-12})



Cost analysis

$$\mathrm{loss}_{\mathrm{PDE}} = \frac{1}{m_1} \sum_{i=1}^{m_1} \left| \Delta \left[u_h \circ \mathbf{x}_h \left(\boldsymbol{\xi}_i \right) \right] - f_h \circ \mathbf{x}_h \left(\boldsymbol{\xi}_i \right) \right|^2$$

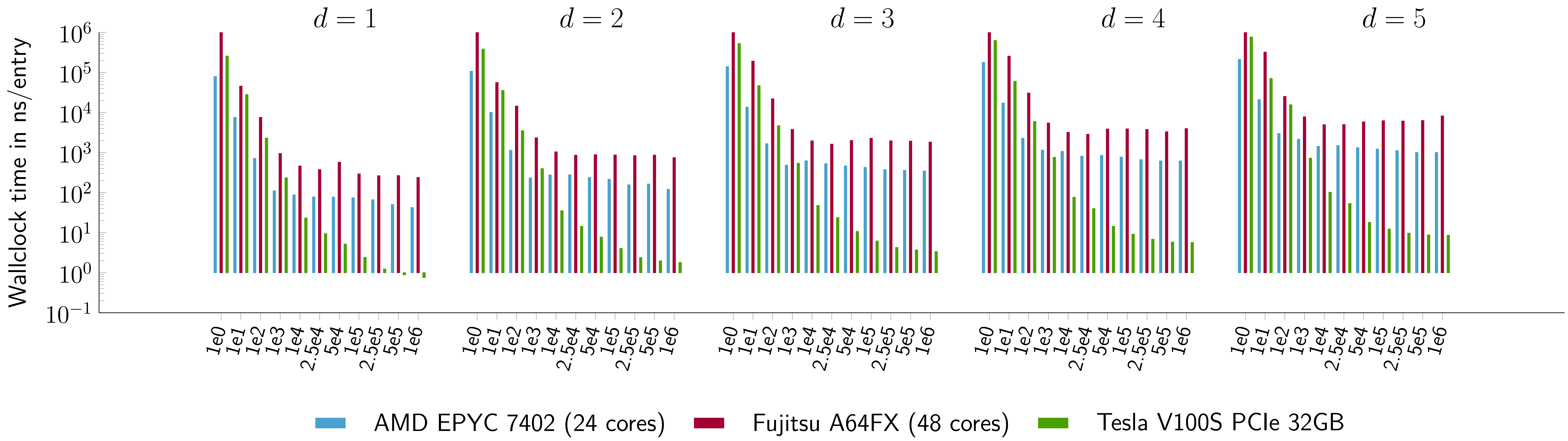
Differential operators in the loss function are computed efficiently in the ‘traditional’ IgA manner by differentiating the B-spline basis functions (no need to differentiate the network as in PINNs!)

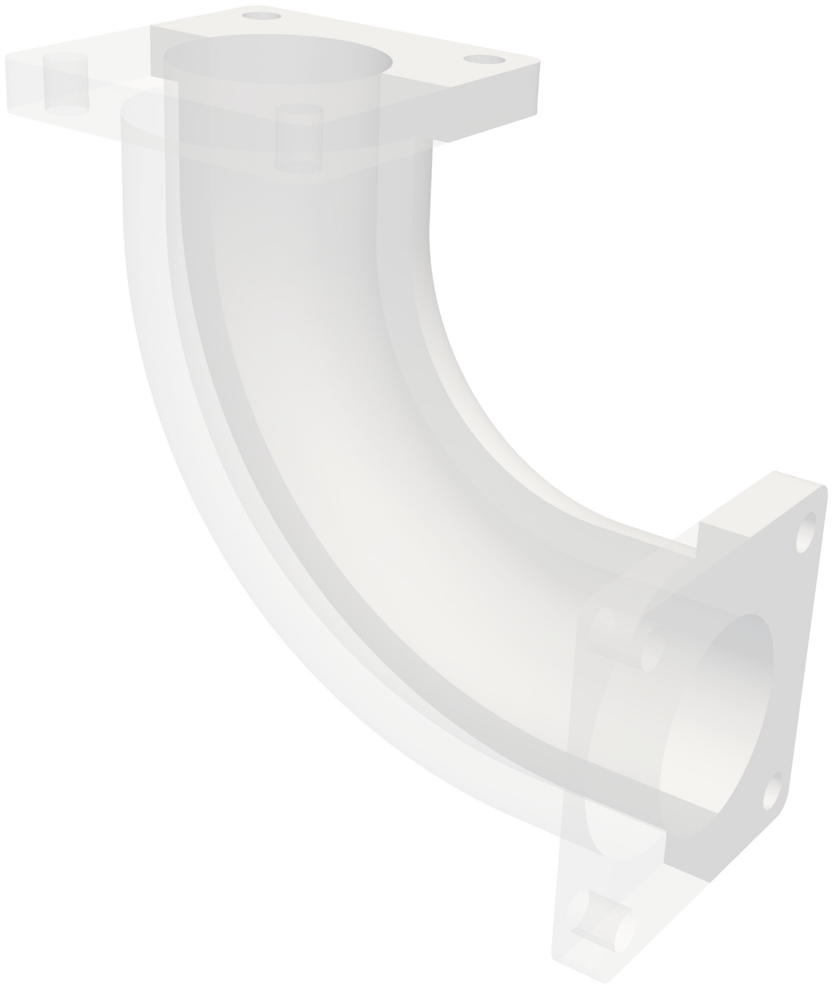
```
bspline.ilaplace(Geo, Cpts);
```

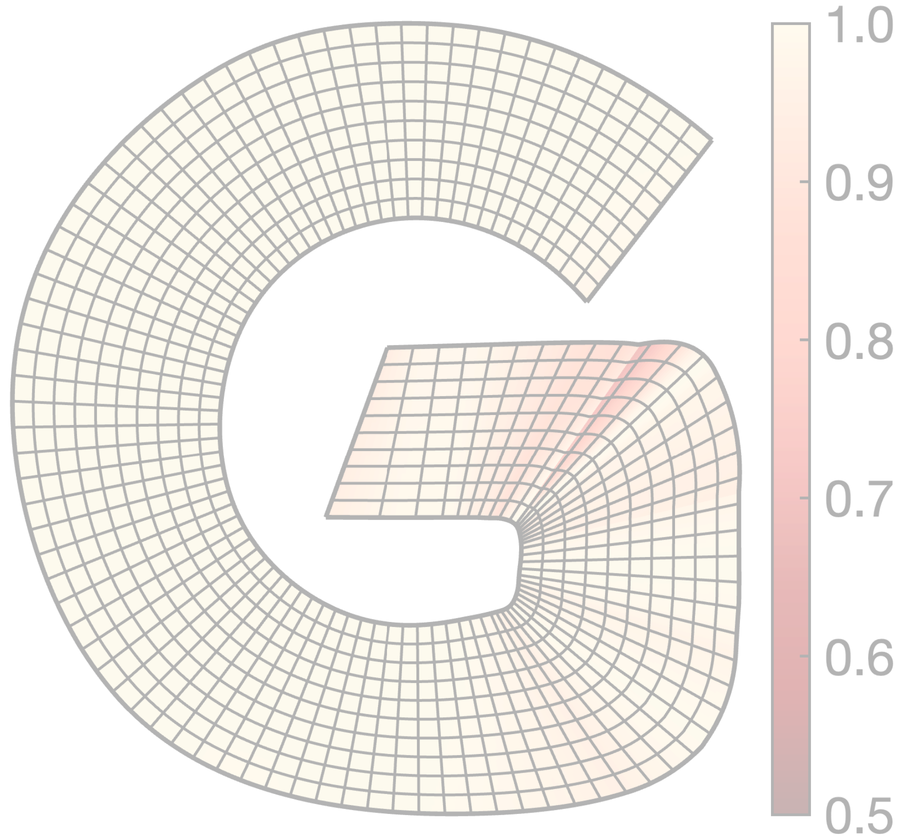
Derivatives of the loss function w.r.t. to the weights and biases are computed via back propagation

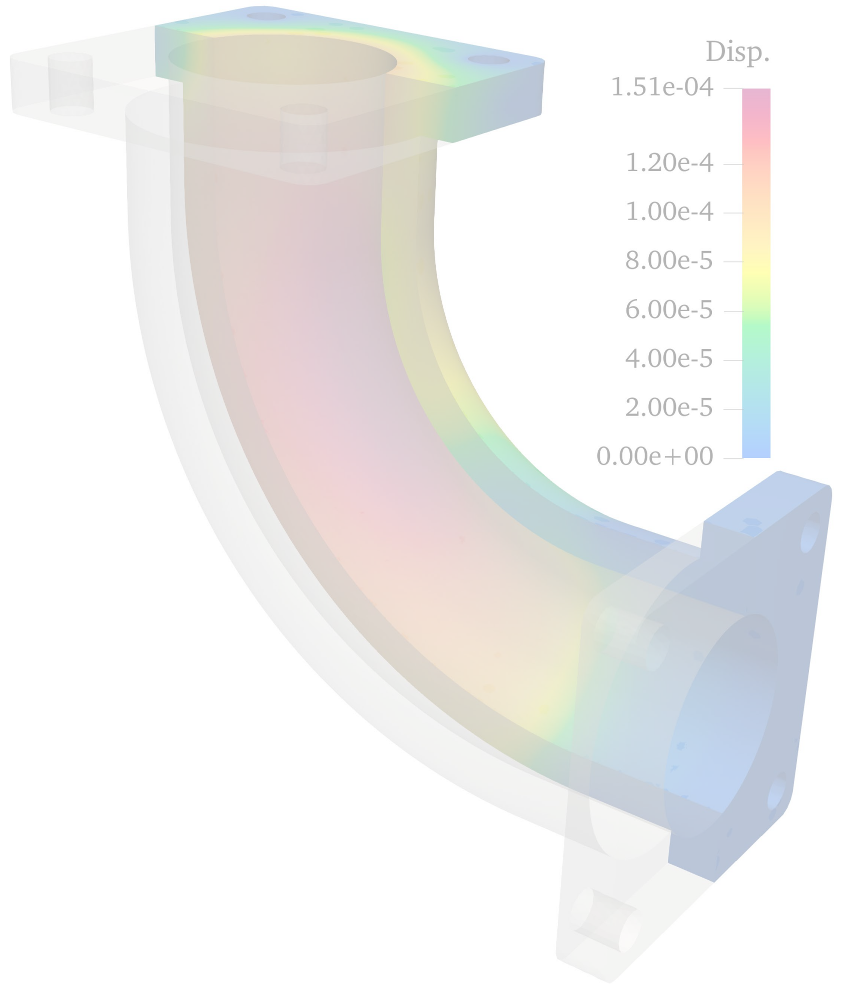
```
nn.zero_grad(); loss = L(nn.forward(Geo, Cpts),…); loss.backward();
```

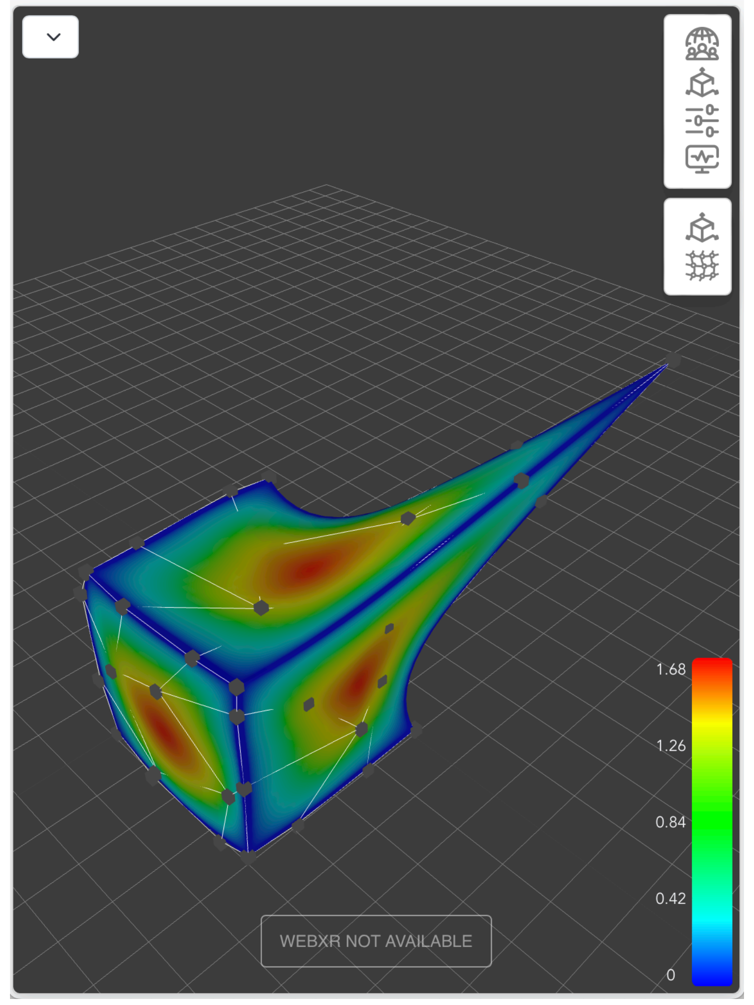
Cost analysis











1. Creation of VReps from BReps

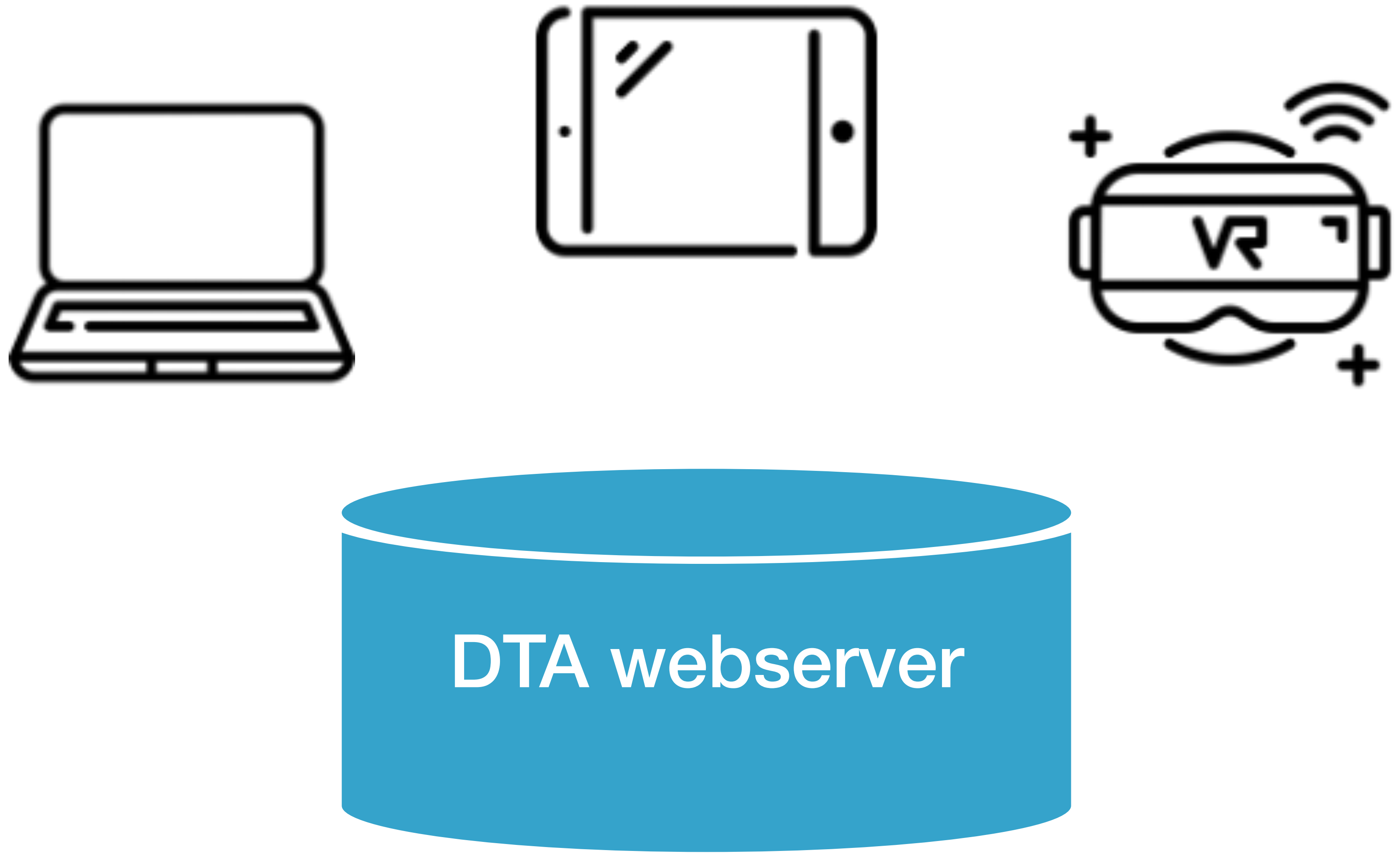
2. Reparameterization techniques

3. Isogeometric Analysis and IgANets

4. Interactive Design-through-Analysis

Interactive design-through-analysis

Vision: Enable CAD and CAE experts to collaborate in a distributed DTA framework



IgANet

https://visualization.surf.nl/iganet

Summary

Seamless **in-paradigm integration** of deep learning techniques into the IgA framework

Concept of an **interactive Design-through-Analysis** pipeline for CAD and CAE experts

Software toolbox for **automatic (re-)parameterization** to motivate CAD expert use VReps

Acknowledgments
Ye Ji, Hugo Verhelst, Mengyun Wang, Deepesh Toshniwal, Frank van Ruiten (TU Delft), Jochen Hinz, Merle Backmeyer (ETH Zurich), Casper van Leeuwen, Paul Melis, Yue Zhao (SURF), Jaewook Lee (TU Vienna), Veronika Trávníková (RWTH Aachen)

H2020: MOTOR (GA No. 678727), SURF: EINF-3689, EINF-7907, ETP-0001