# IgaNets: Physics-Informed Machine Learning Embedded Into Isogeometric Analysis

Matthias Möller, Deepesh Toshniwal, Frank van Ruiten

Department of Applied Mathematics

Delft University of Technology, The Netherlands

9th GACM Colloquium on Computational Mechanics 2022
21–23 September 2022, Essen, Germany

MS 12: Scientific Machine Learning in Computational Mechanics

# Motivation

| FDM, FVM, FEM, BEM, IGA, ... |
|---|
| |

vs.

| PINNs, DeepONets, FourierNets, ... |
|---|
| |

# Motivation

| FDM, FVM, FEM, BEM, IGA, ... |
|---|
| |

vs.

| PINNs, DeepONets, FourierNets, ... |
|---|
| |

**Common misconceptions**

- *"Method $a$ is/is not as accurate as method $b$"*
- *"Method $a$ is x-times faster/slower than method $b$"*

# Motivation

**FDM, FVM, FEM, BEM, IGA, ...**

- 👍 sound mathematical foundation
- 👍 established engineering workflows
- 👎 no cost amortization over multiple runs, no real-time capability

vs.

**PINNs, DeepONets, FourierNets, ...**

- 👍 fast evaluation (costly training!)
- 👍 inclusion of (measurement) data
- 👎 lack of convergence theory
- 👎 lack of general acceptance

**Common misconceptions**

- *"Method $a$ is/is not as accurate as method $b$"*
- *"Method $a$ is x-times faster/slower than method $b$"*

**Better questions to ask**

- What are the specific strengths/weaknesses of the different approaches?

# Motivation

**FDM, FVM, FEM, BEM, IGA, ...**

- 👍 sound mathematical foundation
- 👍 established engineering workflows

and

**PINNs, DeepONets, FourierNets, ...**

- 👍 fast evaluation (costly training!)
- 👍 inclusion of (measurement) data

**Common misconceptions**

- *"Method $a$ is/is not as accurate as method $b$"*
- *"Method $a$ is x-times faster/slower than method $b$"*

**Better questions to ask**

- What are the specific strengths/weaknesses of the different approaches?
- How can we combine the strengths of both classes of methods?

# Motivation

| FDM, FVM, FEM, BEM, **IGA**, ... | | **PINNs**, **DeepONets**, FourierNets, ... |
|---|---|---|
| 👍 sound mathematical foundation<br>👍 established engineering workflows | and | 👍 fast evaluation (costly training!)<br>👍 inclusion of (measurement) data |

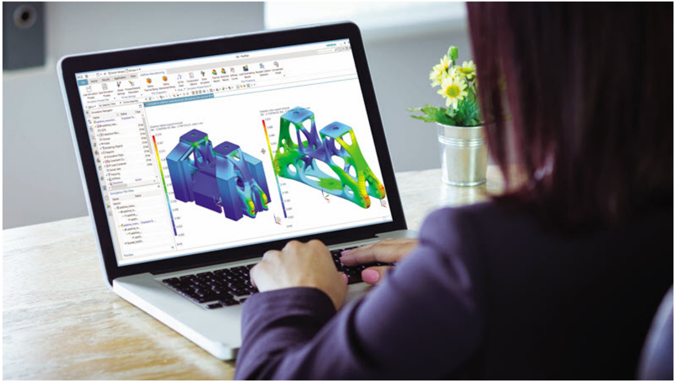
**Common misconceptions**

- *"Method $a$ is/is not as accurate as method $b$"*
- *"Method $a$ is x-times faster/slower than method $b$"*

**Better questions to ask**

- What are the specific strengths/weaknesses of the different approaches?
- How can we combine the strengths of both classes of methods?
- What is the envisaged purpose of the new approach?

# Design-through-Analysis — *IGA's ultimate goal from day one on*

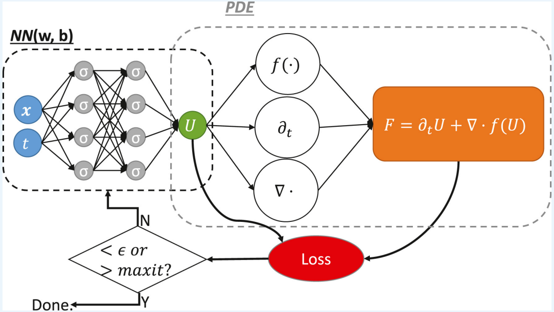


**Vision**: fast interactive qualitative analysis and accurate quantitative analysis within the same computational framework with seamless switching between both approaches

Photo: Siemens – Simulation for Design Engineers

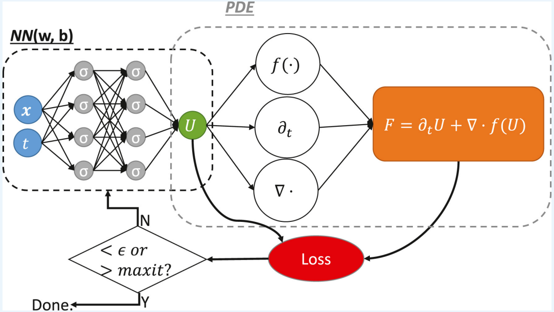
# Physics-informed machine learning

**PINN** (Raissi et al. 2018): *learns the (initial-)boundary-value problem*



# Physics-informed machine learning

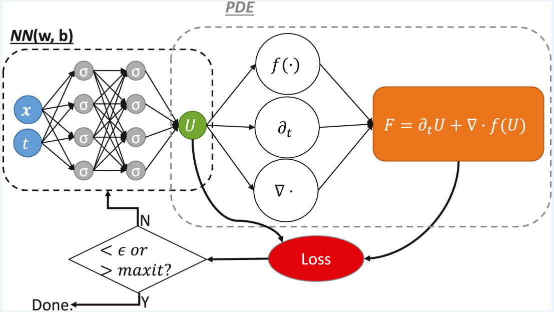
**PINN** (Raissi et al. 2018): *learns the (initial-)boundary-value problem*




- easy to implement for ‘any‘ PDE
- combined un-/supervised learning
- poor extrapolation/generalization
- collocation-based approach requires re-evaluation of NN at every point
- rudimentary convergence theory

# Physics-informed machine learning

**PINN** (Raissi et al. 2018): *learns the (initial-)boundary-value problem*



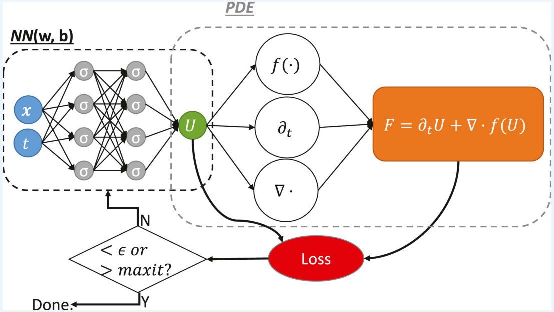

- easy to implement for ‘any‘ PDE
- combined un-/supervised learning
- poor extrapolation/generalization
- collocation-based approach requires re-evaluation of NN at every point
- rudimentary convergence theory

**DeepONet** (Lu et al. 2019): *learns the differential operator*

$$G_\theta(u)(y) = \sum_{k=1}^{q} \underbrace{b_k(u(x_1), u(x_2), \dots, u(x_m))}_{\text{branch}} \underbrace{t_k(y)}_{\text{trunk}}$$

# Physics-informed machine learning

## **PINN** (Raissi et al. 2018): *learns the (initial-)boundary-value problem*




- 👍 easy to implement for ‘any‘ PDE
- 👍 combined un-/supervised learning
- 👎 poor extrapolation/generalization
- 👎 collocation-based approach requires re-evaluation of NN at every point
- 👎 rudimentary convergence theory

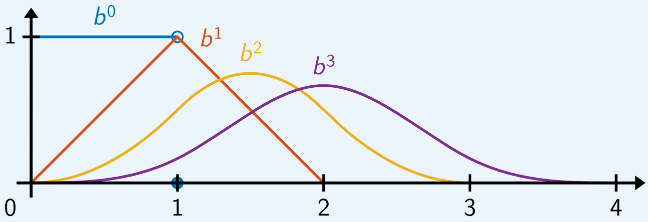
## **DeepONet** (Lu et al. 2019): *learns the differential operator*

$$G_\theta(u)(y) = \sum_{k=1}^{q} \underbrace{b_k(u(x_1), u(x_2), \ldots, u(x_m))}_{\text{branch}} \underbrace{t_k(y)}_{\text{trunk}}$$

Don’t we know a good basis?

# B-spline basis functions

## Cox de Boor recursion formula



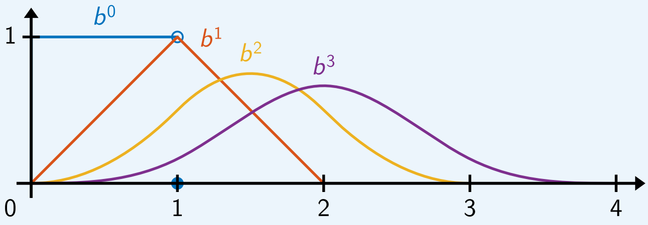

knot vector $\Xi = [0, 1, 2, 3, 4]$

$$b_\ell^0(\xi) = \begin{cases} 1 & \text{if } \xi_\ell \leq \xi < \xi_{\ell+1} \\ 0 & \text{otherwise} \end{cases}$$

$$b_\ell^p(\xi) = \frac{\xi - \xi_\ell}{\xi_{\ell+p} - \xi_\ell} b_\ell^{p-1}(\xi) + \frac{\xi_{\ell+p+1} - \xi}{\xi_{\ell+p+1} - \xi_{\ell+1}} b_{\ell+1}^{p-1}(\xi)$$

# B-spline basis functions

## Cox de Boor recursion formula




knot vector $\Xi = [0, 1, 2, 3, 4]$

$$b_\ell^0(\xi) = \begin{cases} 1 & \text{if } \xi_\ell \leq \xi < \xi_{\ell+1} \\ 0 & \text{otherwise} \end{cases}$$

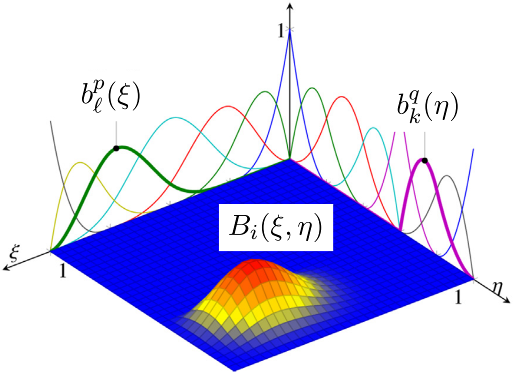
$$b_\ell^p(\xi) = \frac{\xi - \xi_\ell}{\xi_{\ell+p} - \xi_\ell} b_\ell^{p-1}(\xi) + \frac{\xi_{\ell+p+1} - \xi}{\xi_{\ell+p+1} - \xi_{\ell+1}} b_{\ell+1}^{p-1}(\xi)$$

**Many good properties**: compact support $[\xi_\ell, \xi_{\ell+p+1})$, positive function values over support interval, derivatives of B-splines are combinations of lower-order B-splines, ...

# Isogeometric Analysis

**Paradigm**: represent 'everything' in terms of tensor products of B-spline basis functions

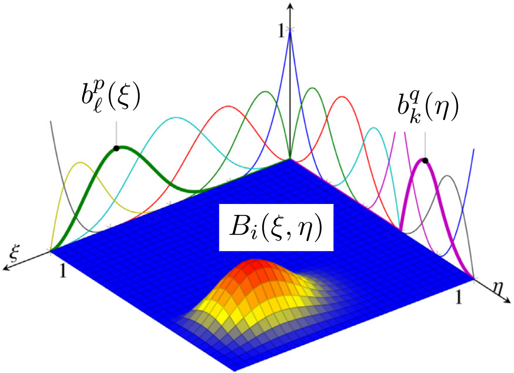
$$B_i(\xi,\eta) := b_\ell^p(\xi)\cdot b_k^q(\eta), \qquad i := (k-1)\cdot n_\ell + \ell, \quad 1 \le \ell \le n_\ell, \quad 1 \le k \le n_k,$$



# Isogeometric Analysis

**Paradigm**: represent 'everything' in terms of tensor products of B-spline basis functions

$$B_i(\xi,\eta) := b_\ell^p(\xi)\cdot b_k^q(\eta), \qquad i := (k-1)\cdot n_\ell + \ell, \quad 1 \le \ell \le n_\ell, \quad 1 \le k \le n_k,$$

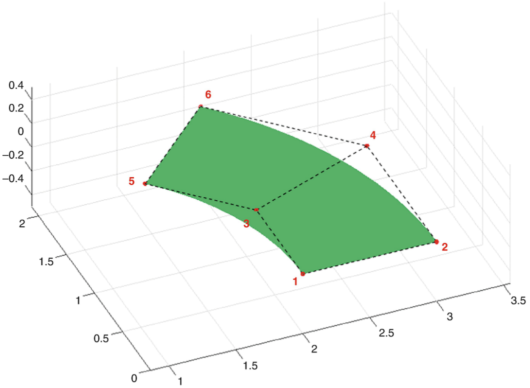



**Many more good properties**: partition of unity $\sum_{i=1}^{n} B_i(\xi,\eta) \equiv 1$, $C^{p-1}$ continuity, ...

# Isogeometric Analysis

**Geometry**: bijective mapping from the unit square to the physical domain $\Omega_h \subset \mathbb{R}^d$

$$\mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot \mathbf{x}_i \qquad \forall (\xi,\eta) \in [0,1]^2 =: \hat{\Omega}$$

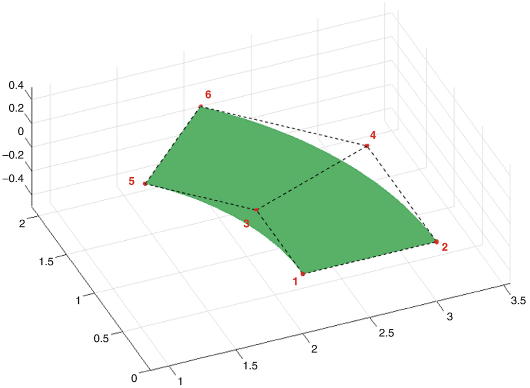



- the shape of $\Omega_h$ is fully specified by the set of **control points** $\mathbf{x}_i \in \mathbb{R}^d$

# Isogeometric Analysis

**Geometry**: bijective mapping from the unit square to the physical domain $\Omega_h \subset \mathbb{R}^d$

$$\mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot \mathbf{x}_i \qquad \forall(\xi,\eta) \in [0,1]^2 =: \hat{\Omega}$$

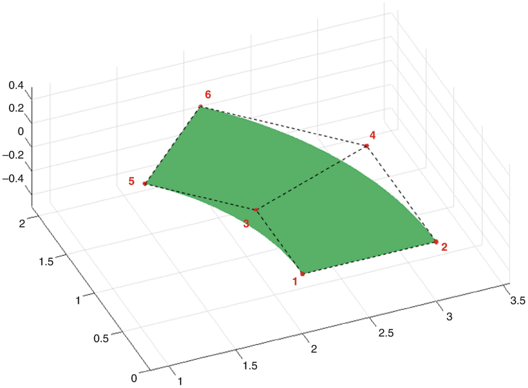



- the shape of $\Omega_h$ is fully specified by the set of **control points** $\mathbf{x}_i \in \mathbb{R}^d$
- interior control points must be chosen such that 'grid lines' do not fold as this violates the bijectivity of $\mathbf{x}_h : \hat{\Omega} \to \Omega_h$

# Isogeometric Analysis

**Geometry**: bijective mapping from the unit square to the physical domain $\Omega_h \subset \mathbb{R}^d$

$$\mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot \mathbf{x}_i \qquad \forall(\xi,\eta) \in [0,1]^2 =: \hat{\Omega}$$




- the shape of $\Omega_h$ is fully specified by the set of **control points** $\mathbf{x}_i \in \mathbb{R}^d$
- interior control points must be chosen such that 'grid lines' do not fold as this violates the bijectivity of $\mathbf{x}_h : \hat{\Omega} \to \Omega_h$
- refinement in $h$ (knot insertion) and $p$ (order elevation) preserves the shape of $\Omega_h$ and can be used to generate finer computational 'grids' for the analysis

**Data, boundary conditions, and solution**: forward mappings from the unit square

(r.h.s vector) $$f_h \circ \mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot f_i \qquad \forall(\xi,\eta) \in [0,1]^2$$

(boundary conditions) $$g_h \circ \mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot g_i \qquad \forall(\xi,\eta) \in \partial[0,1]^2$$

(solution) $$u_h \circ \mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot u_i \qquad \forall(\xi,\eta) \in [0,1]^2$$

# Isogeometric Analysis

**Data, boundary conditions, and solution**: forward mappings from the unit square

$$\text{(r.h.s vector)} \qquad f_h \circ \mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot f_i \qquad \forall(\xi,\eta) \in [0,1]^2$$

$$\text{(boundary conditions)} \qquad g_h \circ \mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot g_i \qquad \forall(\xi,\eta) \in \partial[0,1]^2$$

$$\text{(solution)} \qquad u_h \circ \mathbf{x}_h(\xi,\eta) = \sum_{i=1}^{n} B_i(\xi,\eta) \cdot u_i \qquad \forall(\xi,\eta) \in [0,1]^2$$

**Model problem**: Poisson's equation

$$-\Delta u_h = f_h \quad \text{in} \quad \Omega_h, \qquad u_h = g_h \quad \text{on} \quad \partial\Omega_h$$

# Isogeometric Analysis

**Different solution approaches**

- Galerkin-type IGA (Hughes *et al.* 2005 and many more)
- Isogeometric collocation methods (Reali, Hughes, 2015)
- Variational collocation method (Gomez, De Lorenzis, 2016)

# Isogeometric Analysis

**Different solution approaches**

- Galerkin-type IGA (Hughes *et al.* 2005 and many more)
- Isogeometric collocation methods (Reali, Hughes, 2015)
- Variational collocation method (Gomez, De Lorenzis, 2016)

**Abstract representation**

Given $\mathbf{x}_i$ (geometry), $f_i$ (r.h.s. vector), and $g_i$ (boundary conditions), **compute**

$$\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = A^{-1}\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}\right) \cdot b\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}\right)$$

Any point of the solution can afterwards be obtained by a simple **function evaluation**

$$(\xi,\eta) \in [0,1]^2 \quad \mapsto \quad u_h \circ \mathbf{x}_h(\xi,\eta) = [B_1(\xi,\eta),\dots,B_n(\xi,\eta)] \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

# Isogeometric Analysis

**Abstract representation**

Given $\mathbf{x}_i$ (geometry), $f_i$ (r.h.s. vector), and $g_i$ (boundary conditions), **compute**

$$\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = A^{-1}\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}\right) \cdot b\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}\right)$$

Any point of the solution can afterwards be obtained by a simple **function evaluation**

$$(\xi,\eta) \in [0,1]^2 \quad \mapsto \quad u_h \circ \mathbf{x}_h(\xi,\eta) = [B_1(\xi,\eta),\dots,B_n(\xi,\eta)] \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

Let us interpret the sets of B-spline coefficients $\{\mathbf{x}_i\}$, $\{f_i\}$, and $\{g_i\}$ as an efficient encoding of our PDE problem that is fed into our IGA machinery as **input**.

The **output** of our IGA machinery are the B-spline coefficients $\{u_i\}$ of the solution.

# Isogeometric Analysis + PINNs

**IgaNet**: replace **computation** by **physics-informed machine learning**

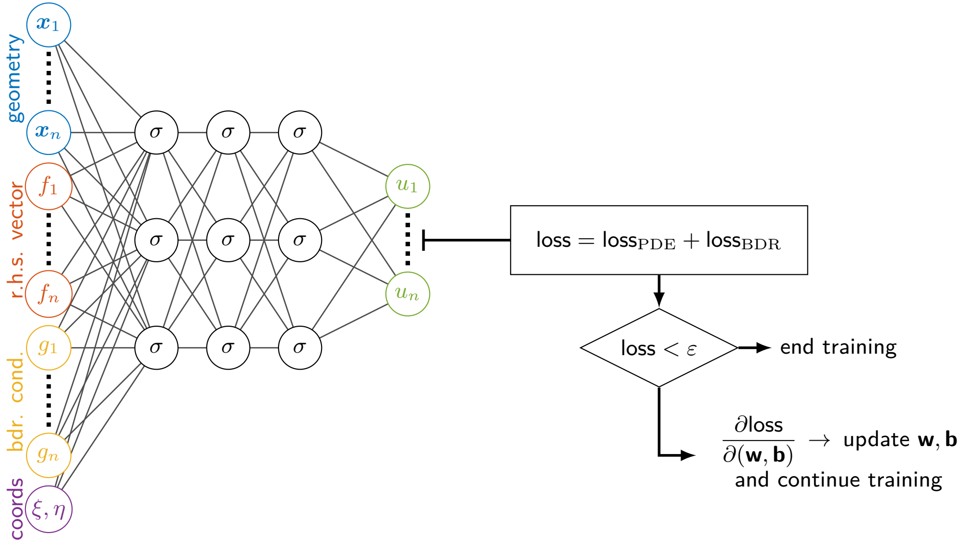
$$\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = A^{-1}\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}\right) \cdot b\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}\right)$$

$$\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \text{PINN}\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}; (\xi^{(k)}, \eta^{(k)})_{k=1}^{N_{\text{samples}}}\right)$$

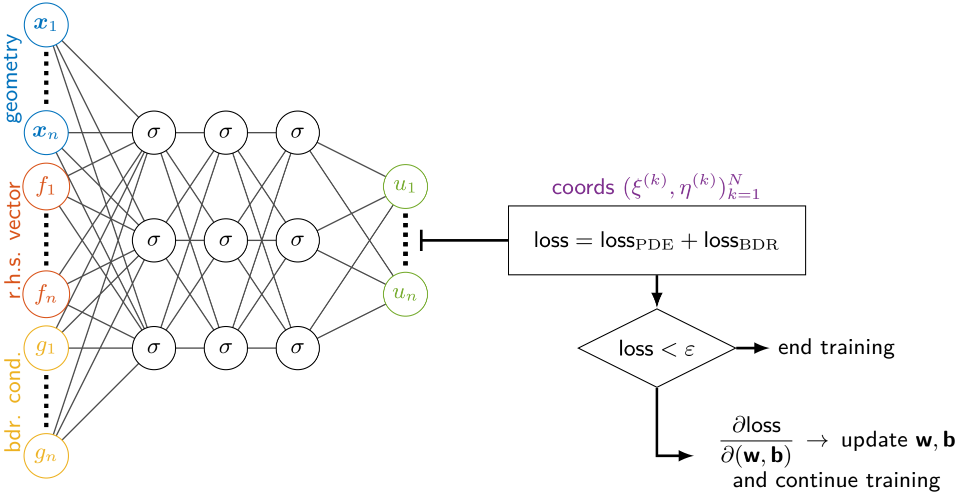
Compute the solution by evaluating the trained neural network

$$u_h(\xi, \eta) \approx [B_1(\xi, \eta), \dots, B_n(\xi, \eta)] \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \text{PINN}\left(\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}; (\xi, \eta)\right)$$

# IgaNet architecture



# IgaNet architecture



# Loss function

$$\text{loss}_{\mathrm{PDE}} = \frac{\alpha}{N_\Omega} \sum_{k=1}^{N_\Omega} \left| \Delta \left[ u_h \circ \mathbf{x}_h \left( \xi^{(k)}, \eta^{(k)} \right) \right] - f_h \circ \mathbf{x}_h \left( \xi^{(k)}, \eta^{(k)} \right) \right|^2$$

$$\text{loss}_{\mathrm{BDR}} = \frac{\beta}{N_\Gamma} \sum_{k=1}^{N_\Gamma} \left| u_h \circ \mathbf{x}_h \left( \xi^{(k)}, \eta^{(k)} \right) - g_h \circ \mathbf{x}_h \left( \xi^{(k)}, \eta^{(k)} \right) \right|^2$$

Express derivatives with respect to physical space variables using the Jacobian $J$, the Hessian $H$ and the matrix of squared first derivatives $Q$ (Schillinger *et al.* 2013):

$$\begin{bmatrix} \frac{\partial^2 B}{\partial x^2} \\ \frac{\partial^2 B}{\partial x \partial y} \\ \frac{\partial^2 B}{\partial y^2} \end{bmatrix} = Q^{-\top} \left( \begin{bmatrix} \frac{\partial^2 B}{\partial \xi^2} \\ \frac{\partial^2 B}{\partial \xi \partial \eta} \\ \frac{\partial^2 B}{\partial \eta^2} \end{bmatrix} - H^\top J^{-\top} \begin{bmatrix} \frac{\partial B}{\partial \xi} \\ \frac{\partial B}{\partial \eta} \end{bmatrix} \right)$$

# Two-level training strategy

**For** $[\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathcal{S}_{\text{geo}}$, $[f_1, \ldots, f_n] \in \mathcal{S}_{\text{rhs}}$, $[g_1, \ldots, g_n] \in \mathcal{S}_{\text{bcond}}$ **do**

**For** a batch of randomly sampled $(\xi_k, \eta_k) \in [0,1]^2$ **do**

$$\text{Train PINN}\left( \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}; (\xi_k, \eta_k)_{k=1}^{N_{\text{samples}}} \right) \mapsto \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

**EndFor**

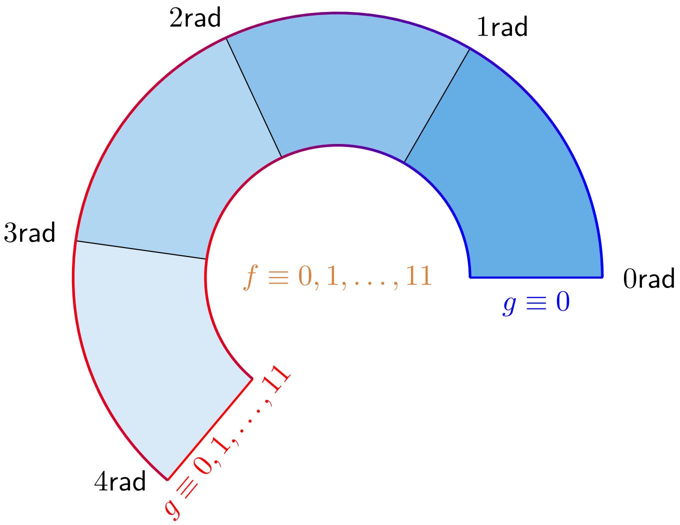
**EndFor**

**IGA details**: $7 \times 7$ bi-cubic tensor-product B-splines for $\mathbf{x}_h$ and $u_h$, $C^2$-continuous

**PINN details**: TensorFlow 2.6, 7-layer neural network with 50 neurons per layer and ReLU activation function (except for output layer), Adam optimizer, 30.000 epochs, training is stopped after 3.000 epochs w/o improvement of the loss value

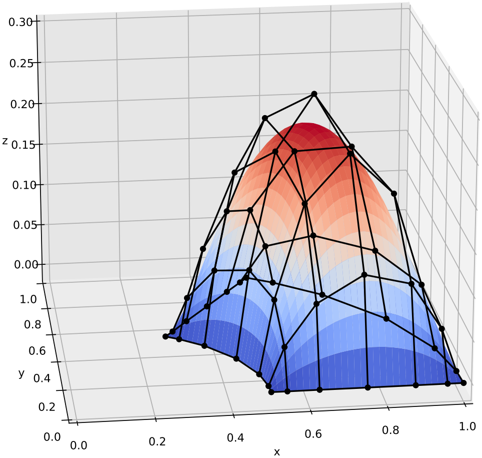
Ongoing master thesis work of Frank van Ruiten, TU Delft

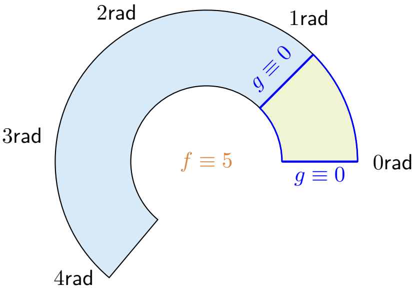
# Test case: Poisson's equation on a variable annulus




Ongoing master thesis work of Frank van Ruiten, TU Delft

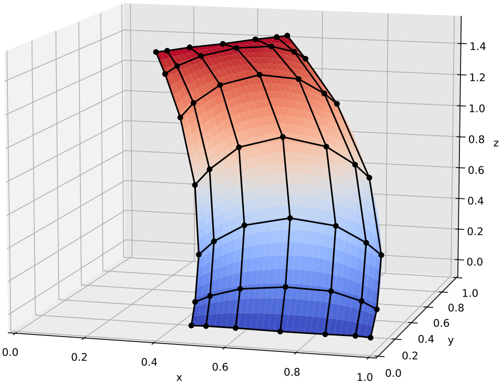
# Preliminary results

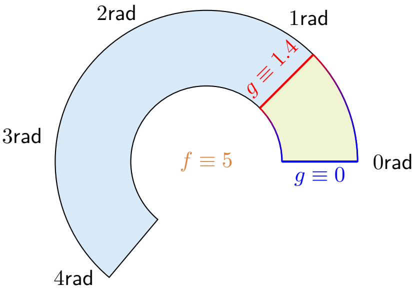



Ongoing master thesis work of Frank van Ruiten, TU Delft

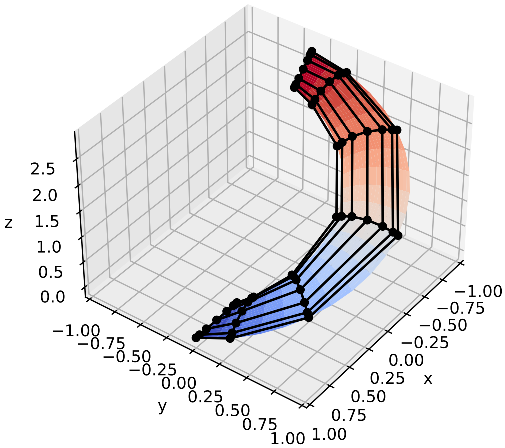
# Preliminary results

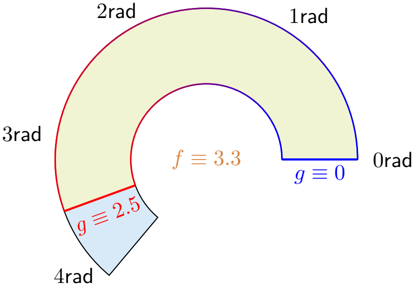



Ongoing master thesis work of Frank van Ruiten, TU Delft

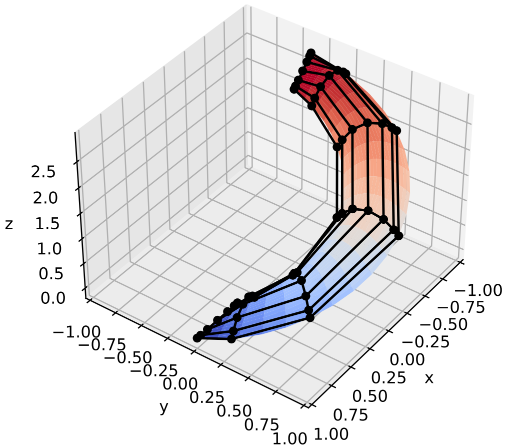
# Preliminary results

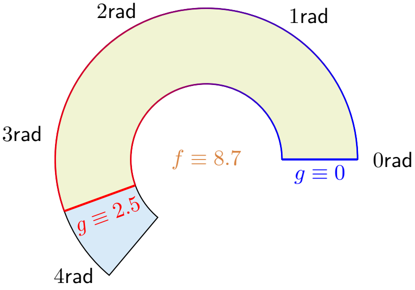



Ongoing master thesis work of Frank van Ruiten, TU Delft

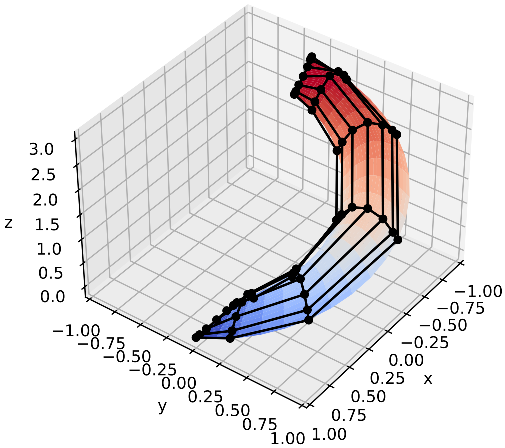
# Preliminary results

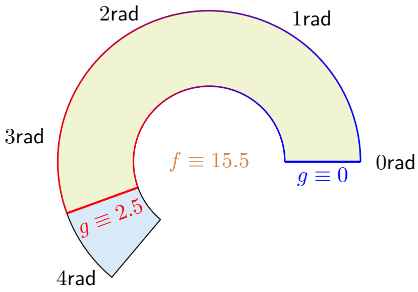



Ongoing master thesis work of Frank van Ruiten, TU Delft

# Preliminary results




Ongoing master thesis work of Frank van Ruiten, TU Delft

# Towards an ML-friendly B-spline formulation

**Common computational task**

Given sampling point $\xi \in [\xi_\ell, \xi_{\ell+1})$ compute for $r \geq 0$

$$\frac{\mathrm{d}^r}{\mathrm{d}\xi}\chi(\xi) = \left[\frac{\mathrm{d}^r}{\mathrm{d}\xi}b^p_{\ell-p}(\xi), \dots, \frac{\mathrm{d}^r}{\mathrm{d}\xi}b^p_{\ell}(\xi)\right] \cdot \underbrace{[\chi_{\ell-p}, \dots, \chi_\ell]}_{\text{network's output}}$$

# Towards an ML-friendly B-spline formulation

**Common computational task**

Given sampling point $\xi \in [\xi_\ell, \xi_{\ell+1})$ compute for $r \geq 0$

$$\frac{\mathrm{d}^r}{\mathrm{d}\xi}\chi(\xi) = \left[\frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_{\ell-p}(\xi), \ldots, \frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_\ell(\xi)\right] \cdot \underbrace{[\chi_{\ell-p}, \ldots, \chi_\ell]}_{\text{network's output}}$$

- The above needs to be performed for all sampling points $\xi^{(k)}$ in the batch

$$\mathtt{sum}(\mathrm{d}^r \mathcal{B}^p \odot \mathcal{X}, 2)$$

# Towards an ML-friendly B-spline formulation

**Common computational task**

Given sampling point $\xi \in [\xi_\ell, \xi_{\ell+1})$ compute for $r \geq 0$

$$\frac{\mathrm{d}^r}{\mathrm{d}\xi}\chi(\xi) = \left[\frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_{\ell-p}(\xi), \ldots, \frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_{\ell}(\xi)\right] \cdot \underbrace{[\chi_{\ell-p}, \ldots, \chi_\ell]}_{\text{network's output}}$$

- The above needs to be performed for all sampling points $\xi^{(k)}$ in the batch

$$\texttt{sum}(\mathrm{d}^r \mathcal{B}^p \odot \mathcal{X}, 2)$$

- The above needs to be differentiated by the AD engine during backpropagation

$$\frac{\partial \left(\mathrm{d}^r b^p_\ell \chi_\ell\right)}{\partial w} = \mathrm{d}^{r+1} b^p_\ell \frac{\partial \xi}{\partial w} \chi + \mathrm{d}^r b^p_\ell \frac{\partial \chi}{\partial \xi}$$

# Towards an ML-friendly B-spline formulation

## Common computational task

Given sampling point $\xi \in [\xi_\ell, \xi_{\ell+1})$ compute for $r \geq 0$

$$\frac{\mathrm{d}^r}{\mathrm{d}\xi}\chi(\xi) = \left[\frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_{\ell-p}(\xi), \dots, \frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_\ell(\xi)\right] \cdot \underbrace{[\chi_{\ell-p}, \dots, \chi_\ell]}_{\text{network's output}}$$

Textbook derivatives

$$\frac{\mathrm{d}^r}{\mathrm{d}\xi} b^p_\ell(\xi) = (p-1)\left(\frac{1}{\xi_{\ell+p} - \xi_{\ell+1}} \frac{-\mathrm{d}^{r-1}}{\mathrm{d}\xi} b^{p-1}_{\ell+1}(\xi) + \frac{1}{\xi_{\ell+p-1} - \xi_\ell} \frac{\mathrm{d}^{r-1}}{\mathrm{d}\xi} b^{p-1}_\ell(\xi)\right)$$

with

$$b^p_\ell(\xi) = \frac{\xi - \xi_\ell}{\xi_{\ell+p} - \xi_\ell} b^{p-1}_\ell(\xi) + \frac{\xi_{\ell+p+1} - \xi}{\xi_{\ell+p+1} - \xi_{\ell+1}} b^{p-1}_{\ell+1}(\xi), \quad b^0_\ell(\xi) = \begin{cases} 1 & \text{if } \xi_\ell \leq \xi < \xi_{\ell+1} \\ 0 & \text{otherwise} \end{cases}$$

# Towards an ML-friendly B-spline formulation

Matrix representation of B-splines (Lyche and Morken 2011)

$$\left[\frac{\mathrm{d}^r}{\mathrm{d}\xi}b^p_{\ell-p}(\xi),\dots,\frac{\mathrm{d}^r}{\mathrm{d}\xi}b^p_{\ell}(\xi)\right] = \frac{p!}{(p-r)!}R_1(\xi)\cdots R_{p-r}(\xi)\mathrm{d}R_{p-r+1}\cdots\mathrm{d}R_p$$

with $k\times k+1$ matrices $R_k(\xi)$, e.g.

$$R_1(\xi) = \begin{bmatrix}\frac{\xi_{\ell+1}-\xi}{\xi_{\ell+1}-\xi_\ell} & \frac{x-\xi_\ell}{\xi_{\ell+1}-\xi_\ell}\end{bmatrix}$$

$$R_2(\xi) = \begin{bmatrix}\frac{\xi_{\ell+1}-\xi}{\xi_{\ell+1}-\xi_{\ell-1}} & \frac{x-\xi_{\ell-1}}{\xi_{\ell+1}-\xi_{\ell-1}} & 0\\ 0 & \frac{\xi_{\ell+2}-\xi}{\xi_{\ell+2}-\xi_\ell} & \frac{x-\xi_\ell}{\xi_{\ell+2}-\xi_\ell}\end{bmatrix}$$

$$R_3(\xi) = \dots$$

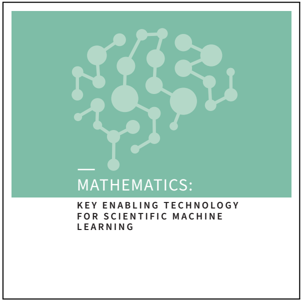
There exists an efficient algorithm based on elementwise operations on vectors.

# Conclusion and outlook

**IgaNets** combine classical numerics with scientific machine learning and may finally enable integrated and interactive computer-aided **design-through-analysis** workflows

## Todo

- performance and hyper-parameter tuning
- extension to multi-patch topologies
- use of IGA and IgaNets in concert
- transfer learning upon basis refinement



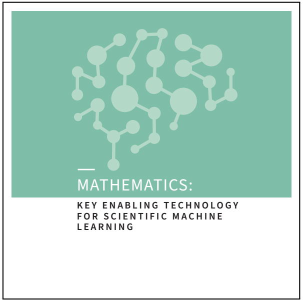

**Short paper**: Möller, Toshniwal, van Ruiten: *Physics-informed machine learning embedded into isogeometric analysis*, 2021.

# Conclusion and outlook

**IgaNets** combine classical numerics with scientific machine learning and may finally enable integrated and interactive computer-aided **design-through-analysis** workflows

## Todo

- performance and hyper-parameter tuning
- extension to multi-patch topologies
- use of IGA and IgaNets in concert
- transfer learning upon basis refinement




**Short paper**: Möller, Toshniwal, van Ruiten: *Physics-informed machine learning embedded into isogeometric analysis*, 2021.

Thank you for your attention!