# 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

USACM Thematic Conference on Isogeometric Analysis 2022

7–9 November 2022, Banff, Canada

New Frontiers in IGA - II

# 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



USACM Thematic Conference on Isogeometric Analysis 2022

7–9 November 2022, Banff, Canada

New Frontiers in IGA - II

# 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 question 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, ...**

- sound mathematical foundation
- established engineering workflows

and

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

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

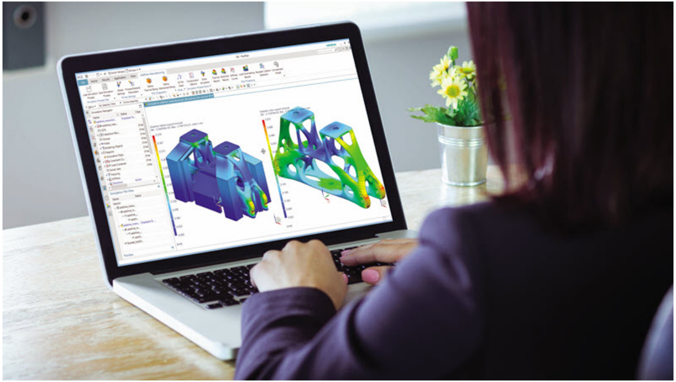
**Common misconceptions**

- *"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?
- What is the envisaged purpose of the new approach?

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

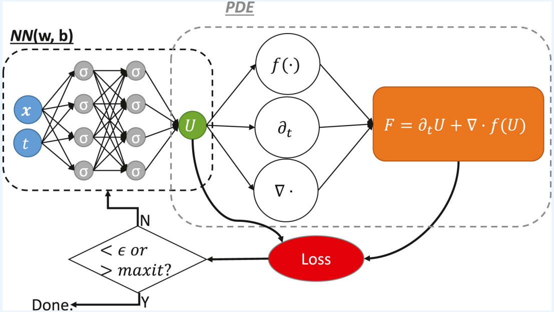


**Vision**: fast interactive qualitative analysis and accurate quantitative analysis within the same computational framework with seamless switching between both approaches

Photo: Siemens – Simulation for Design Engineers

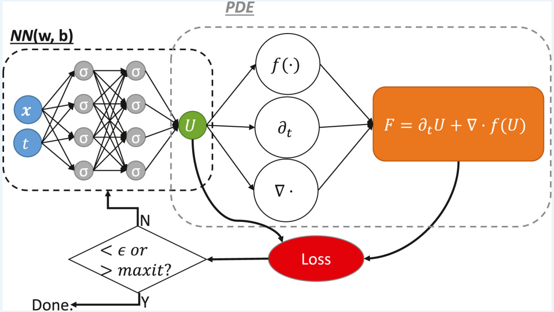
# Physics-informed machine learning

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



# Physics-informed machine learning

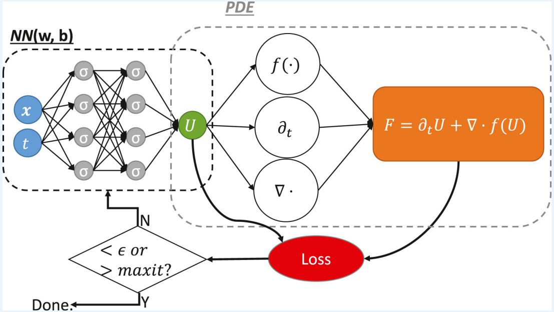
**PINN** (Raissi et al. 2018): *learns the (initial-)boundary-value problem*




- 👍 easy to implement for ‘any‘ PDE because AD magic does it for you
- 👍 combined un-/supervised learning
- 👎 poor extrapolation/generalization
- 👎 point-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*



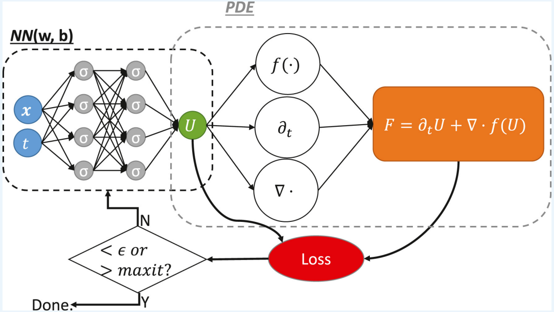

- easy to implement for ‘any‘ PDE because AD magic does it for you
- combined un-/supervised learning
- poor extrapolation/generalization
- point-based approach requires re-evaluation of NN at every point
- rudimentary convergence theory

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

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

# Physics-informed machine learning

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




- 👍 easy to implement for 'any' PDE because AD magic does it for you
- 👍 combined un-/supervised learning
- 👎 poor extrapolation/generalization
- 👎 point-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?

# Isogeometric Analysis

**Model problem**: Poisson's equation

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

with

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

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

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

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

# 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**

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

# 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 + Physics-Informed Machine Learning

**IgaNet**: replace **computation**

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

# Isogeometric Analysis + Physics-Informed Machine Learning

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

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

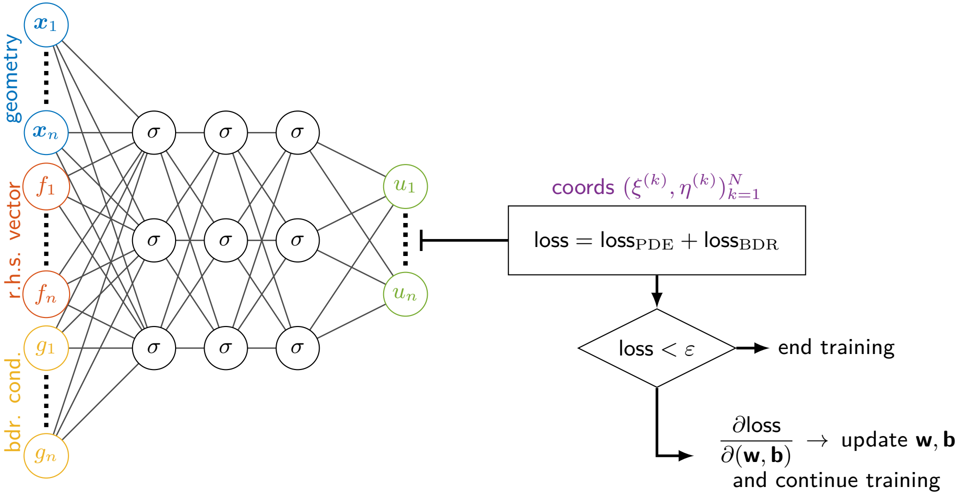
**IgaNet**: replace **computation** by **physics-informed machine learning**

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

Compute the solution from the trained neural network as follows

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

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

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

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

**EndFor**

**EndFor**

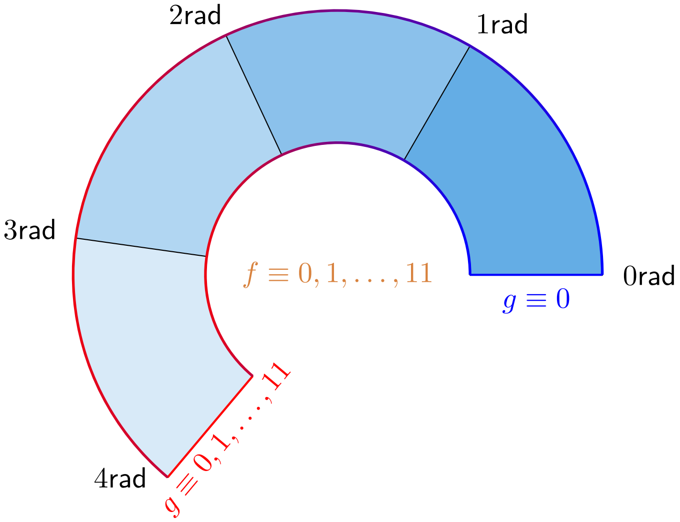
**Details**:

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

---

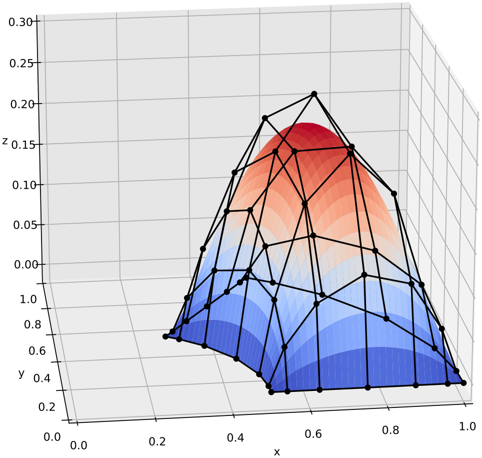
Ongoing master thesis work of Frank van Ruiten, TU Delft

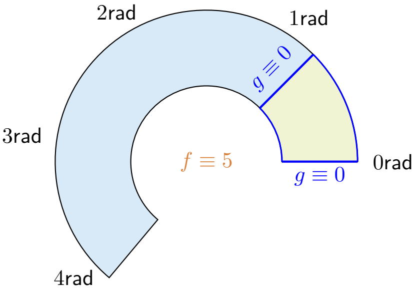
# Test case: Poisson's equation on a variable annulus




Ongoing master thesis work of Frank van Ruiten, TU Delft

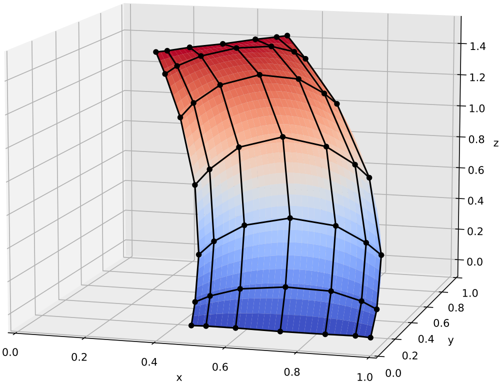
# Preliminary results

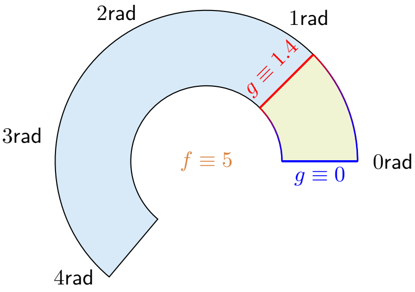



Ongoing master thesis work of Frank van Ruiten, TU Delft

# Preliminary results

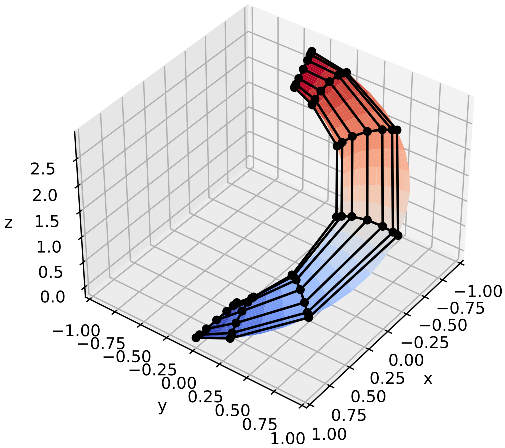


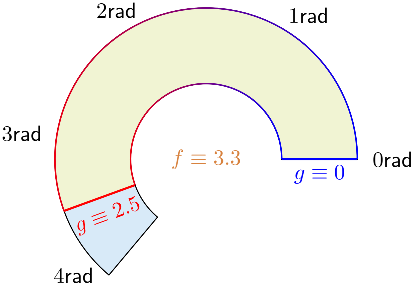




---

Ongoing master thesis work of Frank van Ruiten, TU Delft

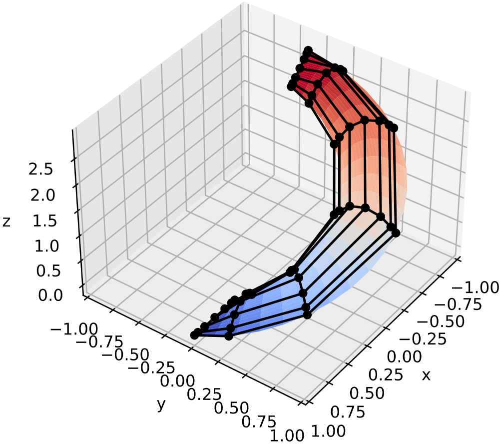
# Preliminary results

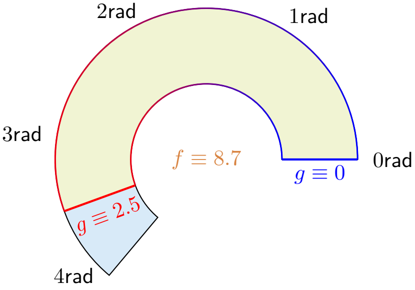



Ongoing master thesis work of Frank van Ruiten, TU Delft

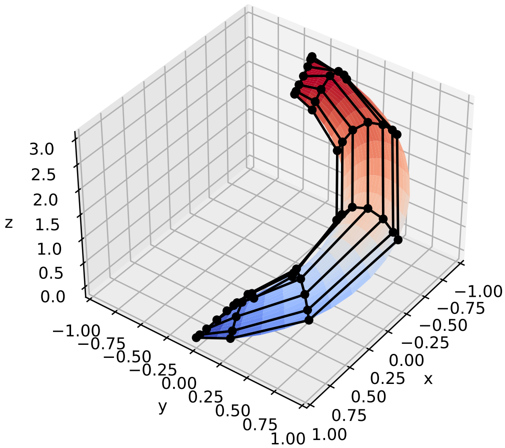
# Preliminary results

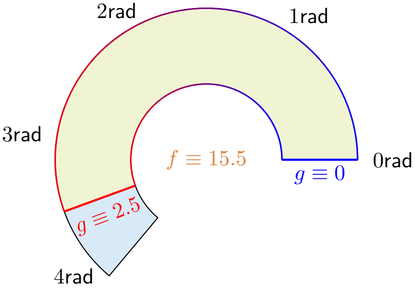



Ongoing master thesis work of Frank van Ruiten, TU Delft

# Preliminary results




Ongoing master thesis work of Frank van Ruiten, TU Delft

# Let's have a look under the hood

Computational costs of PINN vs. IgaNets, implementation aspects, ...

# Computational costs

**Working principle of PINNs**

$$\mathbf{x} \mapsto u(\mathbf{x}) := \mathsf{NN}(\mathbf{x}; f, g, G) = \sigma_L(\mathbf{W}_L\sigma(\ldots(\sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1))) + \mathbf{b}_L)$$

- use AD engine (automated chain rule) to compute derivatives, e.g., $u_x = \mathsf{NN}_x$
- use AD engine on top of AD tree (!!!) to compute gradients w.r.t. weights for training

# Computational costs

## Working principle of PINNs

$$\mathbf{x} \mapsto u(\mathbf{x}) := \mathsf{NN}(\mathbf{x}; f, g, G) = \sigma_L(\mathbf{W}_L\sigma(\ldots(\sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1))) + \mathbf{b}_L)$$

- use AD engine (automated chain rule) to compute derivatives, e.g., $u_x = \mathsf{NN}_x$
- use AD engine on top of AD tree (!!!) to compute gradients w.r.t. weights for training

## Working principle of IgaNets

$$[\mathbf{x}_i, f_i, g_i]_{i=1,\ldots,n} \mapsto [u_i]_{i=1,\ldots,n} := \mathsf{NN}(\mathbf{x}_i, f_i, g_i, i = 1, \ldots, n)$$

- use mathematics to compute derivatives, e.g., $\nabla_{\mathbf{x}} u = \left(\sum_{i=1}^{n} \nabla_{\boldsymbol{\xi}} B_i(\boldsymbol{\xi}) u_i\right) J_G^{-t}$
- use AD to compute gradients w.r.t. weights for training, i.e. (illustrated in 1D)

$$\frac{\partial(\mathrm{d}_\xi^r u(\xi))}{\partial w_k} = \sum_{i=1}^{n} \frac{\partial(\mathrm{d}_\xi^r b_i^p u_i)}{\partial w_k} = \cancel{\sum_{i=1}^{n} \mathrm{d}_\xi^{r+1} b_i^p \frac{\partial \xi}{\partial w_k} u_i} + \sum_{i=1}^{n} \mathrm{d}_\xi^r b_i^p \frac{\partial u_i}{\partial w_k}$$

# Towards an ML-friendly B-spline evaluation

**Major computational task** (illustrated in 1D)

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

$$\mathrm{d}_\xi^r u(\xi) = \left[\mathrm{d}_\xi^r b_{i-p}^p(\xi), \dots, \mathrm{d}_\xi^r b_i^p(\xi)\right] \cdot \underbrace{[u_{i-p}, \dots, u_i]}_{\text{network's output}}$$

Textbook derivatives

$$\mathrm{d}_\xi^r b_i^p(\xi) = (p-1)\left(\frac{-\mathrm{d}_\xi^{r-1} b_{i+1}^{p-1}(\xi)}{\xi_{i+p} - \xi_{i+1}} + \frac{\mathrm{d}_\xi^{r-1} b_i^{p-1}(\xi)}{\xi_{i+p-1} - \xi_i}\right)$$

with

$$b_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} b_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} b_{i+1}^{p-1}(\xi), \quad b_i^0(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

# Towards an ML-friendly B-spline evaluation

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

$$\left[\mathrm{d}_\xi^r b_{i-p}^p(\xi), \ldots, \mathrm{d}_\xi^r b_i^p(\xi)\right] = \frac{p!}{(p-r)!} R_1(\xi) \cdots R_{p-r}(\xi) \mathrm{d}_\xi R_{p-r+1} \cdots \mathrm{d}_\xi R_p$$

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

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

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

$$R_3(\xi) = \ldots$$

# An ML-friendly B-spline evaluation

**Algorithm 2.22** from (Lyche and Morken 2011)

1. $\mathbf{b} = 1$
2. For $k = 1, \ldots, p - r$
   1. $\mathbf{t}_1 = (\xi_{i-k+1}, \ldots, \xi_i)$
   2. $\mathbf{t}_2 = (\xi_{i+1}, \ldots, \xi_{i+k})$
   3. $\mathbf{w} = (\xi - \mathbf{t}_1) \div (\mathbf{t}_2 - \mathbf{t}_1)$
   4. $\mathbf{b} = [(1 - \mathbf{w}) \odot \mathbf{b}, 0] + [0, \mathbf{w} \odot \mathbf{b}]$
3. For $k = p - r + 1, \ldots, p$
   1. $\mathbf{t}_1 = (\xi_{i-k+1}, \ldots, \xi_i)$
   2. $\mathbf{t}_2 = (\xi_{i+1}, \ldots, \xi_{i+k})$
   3. $\mathbf{w} = 1 \div (\mathbf{t}_2 - \mathbf{t}_1)$
   4. $\mathbf{b} = [-\mathbf{w} \odot \mathbf{b}, 0] + [0, \mathbf{w} \odot \mathbf{b}]$

where $\div$ and $\odot$ denote the element-wise division and multiplication of vectors, respectively.

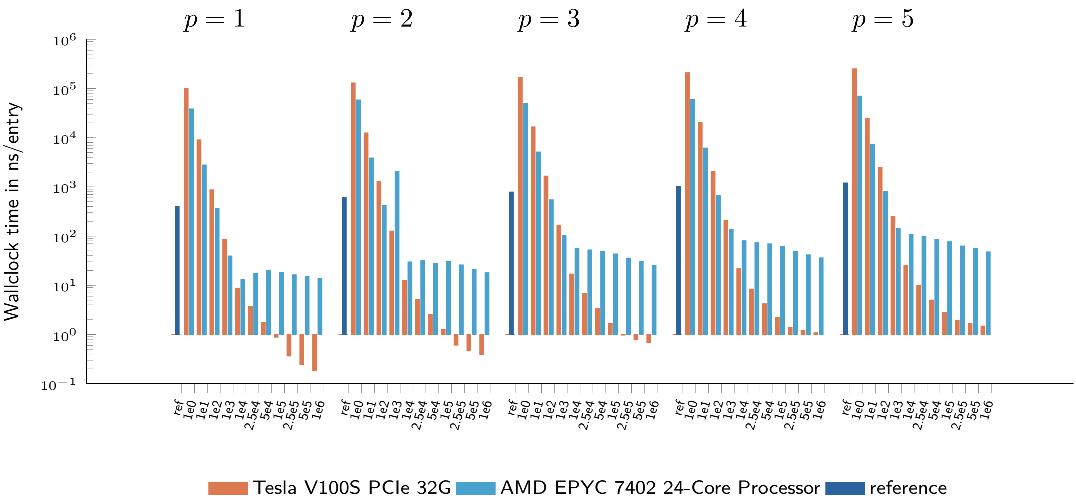
# An ML-friendly B-spline evaluation

**Algorithm 2.22** from (Lyche and Morken 2011) with slight modifications

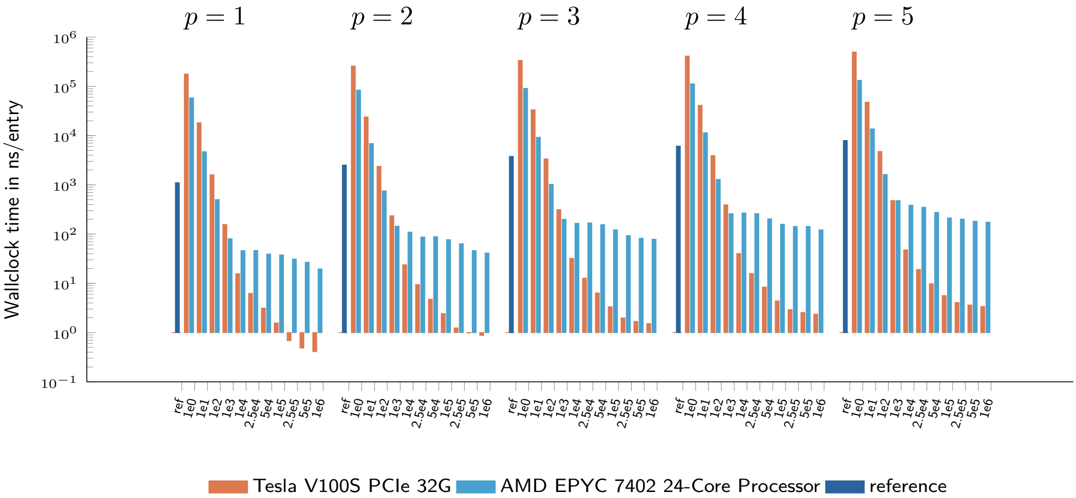
1. $\mathbf{b} = 1$
2. For $k = 1, \ldots, p - r$
   1. $\mathbf{t}_1 = (\xi_{i-k+1}, \ldots, \xi_i)$
   2. $\mathbf{t}_{21} = (\xi_{i+1}, \ldots, \xi_{i+k}) - \mathbf{t}_1$
   3. $\textbf{mask} = (\mathbf{t}_{21} < \text{tol})$
   4. $\mathbf{w} = (\xi - \mathbf{t}_1 - \textbf{mask}) \div (\mathbf{t}_{21} - \textbf{mask})$
   5. $\mathbf{b} = [(1 - \mathbf{w}) \odot \mathbf{b}, 0] + [0, \mathbf{w} \odot \mathbf{b}]$
3. For $k = p - r + 1, \ldots, p$
   1. $\mathbf{t}_1 = (\xi_{i-k+1}, \ldots, \xi_i)$
   2. $\mathbf{t}_{21} = (\xi_{i+1}, \ldots, \xi_{i+k}) - \mathbf{t}_1$
   3. $\textbf{mask} = (\mathbf{t}_{21} < \text{tol})$
   4. $\mathbf{w} = (1 - \textbf{mask}) \div (\mathbf{t}_{21} - \textbf{mask})$
   5. $\mathbf{b} = [-\mathbf{w} \odot \mathbf{b}, 0] + [0, \mathbf{w} \odot \mathbf{b}]$

where $\div$ and $\odot$ denote the element-wise division and multiplication of vectors, respectively.

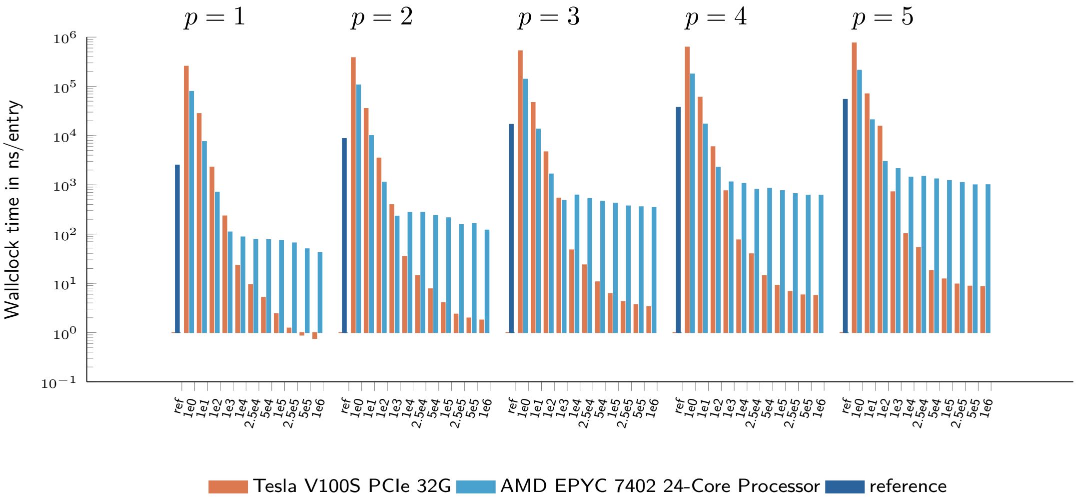
# Performance evaluation - univariate B-splines



# Performance evaluation - bivariate B-splines



# Performance evaluation - trivariate B-splines

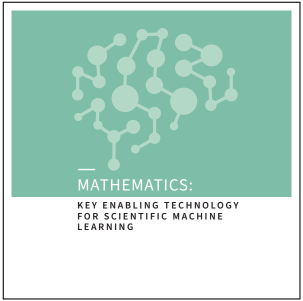


# Conclusion and outlook

**IgaNets** combine classical numerics with physics-informed machine learning and may finally enable **integrated and interactive design-through-analysis** workflows

**WIP/What's next**

- interactive modelling & visualization
- extension to multi-patch topologies
- use of IGA and IgaNets in concert
- transfer learning upon basis refinement
- theoretical foundation & error analysis




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

Journal paper and code release in preparation

# 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

USACM Thematic Conference on Isogeometric Analysis 2022

7–9 November 2022, Banff, Canada

New Frontiers in IGA - II

# 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

USACM Thematic Conference on Isogeometric Analysis 2022

7–9 November 2022, Banff, Canada

New Frontiers in IGA - II

Thank you very much!