IgANets: Physics-Informed Machine Learning Embedded Into Isogeometric Analysis

Matthias Möller

Department of Applied Mathematics
Delft University of Technology, The Netherlands

Aromath seminar
25 April 2023, Sophia Antipolis Cedex, France

Joint work with Deepesh Toshniwal, Frank van Ruiten (TUD),
Casper van Leeuwen, Paul Melis (SURF), Jaewook Lee (TU Vienna)
“[...] the potential value of design through analysis was demonstrated by a significant reduction in structural weight of the project vehicle.”

Design-through-Analysis 2.0

**Vision:** seamless design and analysis workflows without time-consuming (often manual) geometry cleaning and meshing → Isogeometric Analysis (Hughes et al. ’05)
**Vision**: fast interactive qualitative analysis and accurate quantitative analysis within the same computational framework with seamless switching between both approaches.
Physics-informed machine learning

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

\[ F = \partial_t U + \nabla \cdot f(U) \]

- Easier 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
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*

\[
G_{\theta}(u)(y) = \sum_{k=1}^{q} b_k(u(x_1), u(x_2), \ldots, u(x_m)) t_k(y)
\]
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*

\[
G_{\theta}(u)(y) = \sum_{k=1}^{q} b_k(u(x_1), u(x_2), \ldots, u(x_m)) t_k(y)
\]

Don’t we know good bases?
**Bases**

**AI/ML community**: Fourier series, orthogonal polynomials, problem-specific basis functions → impractical for practical computer-aided geometric design

**FEM community**: plethora of finite element basis functions defined on the computational mesh → impractical for a priori training of generic networks

**CAGD community**: trimmed NURBS → maybe, but we're not yet there

**IGA community**: multi-patch tensor-product or locally adaptive B-splines → Let's do it!
Bases

**AI/ML community**: Fourier series, orthogonal polynomials, problem-specific basis functions → impractical for practical computer-aided geometric design

**FEM community**: plethora of finite element basis functions defined on the computational mesh → impractical for a priori training of generic networks
Bases

**AI/ML community**: Fourier series, orthogonal polynomials, problem-specific basis functions $\rightarrow$ impractical for practical computer-aided geometric design

**FEM community**: plethora of finite element basis functions defined on the computational mesh $\rightarrow$ impractical for a priori training of generic networks

**CAGD community**: trimmed NURBS $\rightarrow$ maybe, but we’re not yet there
Bases

**AI/ML community**: Fourier series, orthogonal polynomials, problem-specific basis functions → impractical for practical computer-aided geometric design

**FEM community**: plethora of finite element basis functions defined on the computational mesh → impractical for a priori training of generic networks

**CAGD community**: trimmed NURBS → maybe, but we’re not yet there

**IGA community**: multi-patch tensor-product or locally adaptive B-splines → Let’s do it!
B-spline basis functions

Cox de Boor recursion formula

\[ b^0_i(\xi) = \begin{cases} 
1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\
0 & \text{otherwise}
\end{cases} \]

\[ b^p_i(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} b^{p-1}_i(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} b^{p-1}_{i+1}(\xi) \]

knot vector \( \Xi = [0, 1, 2, 3, 4] \)
B-spline basis functions

Cox de Boor recursion formula

**Knot vector**: \( \Xi = [0, 1, 2, 3, 4] \)

\[
\begin{align*}
    b^0_i(\xi) &= \begin{cases} 
        1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\
        0 & \text{otherwise}
    \end{cases} \\
    b^p_i(\xi) &= \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} b^{p-1}_i(\xi) \\
    &\quad + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} b^{p-1}_{i+1}(\xi)
\end{align*}
\]

Many good properties: compact support \([\xi_i, \xi_{i+p+1})\), 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

\[ B_i(\xi, \eta) := b^p_i(\xi) \cdot b^q_k(\eta), \quad i := (k - 1) \cdot n_i + i, \quad 1 \leq i \leq n_i, \quad 1 \leq k \leq n_k, \]
Isogeometric Analysis

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

\[ B_i(\xi, \eta) := b_i^p(\xi) \cdot b_k^q(\eta), \quad i := (k - 1) \cdot n_i + i, \quad 1 \leq i \leq n_i, \quad 1 \leq k \leq n_k, \]

**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 $\Omega_h \subset \mathbb{R}^d$

\[ x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot x_i \quad \forall (\xi, \eta) \in [0, 1]^2 =: \hat{\Omega} \]

- the shape of $\Omega_h$ is fully specified by the set of control points $x_i \in \mathbb{R}^d$
Isogeometric Analysis

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

$$x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot x_i \quad \forall (\xi, \eta) \in [0, 1]^2 =: \hat{\Omega}$$

- the shape of $\Omega_h$ is fully specified by the set of control points $x_i \in \mathbb{R}^d$
- interior control points must be chosen such that ‘grid lines’ do not fold as this violates the bijectivity of $x_h : \hat{\Omega} \rightarrow \Omega_h$
Isogeometric Analysis

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

$$x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot x_i \quad \forall (\xi, \eta) \in [0, 1]^2 =: \hat{\Omega}$$

- the shape of $\Omega_h$ is fully specified by the set of **control points** $x_i \in \mathbb{R}^d$
- interior control points must be chosen such that ‘grid lines’ do not fold as this violates the bijectivity of $x_h : \hat{\Omega} \to \Omega_h$
- 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
Isogeometric Analysis

**Model problem:** Poisson’s equation

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

with

(geometrical) \[x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot x_i \quad \forall (\xi, \eta) \in [0, 1]^2\]

(solution) \[u_h \circ x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot u_i \quad \forall (\xi, \eta) \in [0, 1]^2\]

(right-hand side vector) \[f_h \circ x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot f_i \quad \forall (\xi, \eta) \in [0, 1]^2\]

(boundary conditions) \[g_h \circ x_h(\xi, \eta) = \sum_{i=1}^{n} B_i(\xi, \eta) \cdot g_i \quad \forall (\xi, \eta) \in \partial [0, 1]^2\]
Isogeometric Analysis

Abstract representation
Given \( x_i \) (geometry), \( f_i \) (r.h.s. vector), and \( g_i \) (boundary conditions), compute

\[
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n
\end{bmatrix} = A^{-1} \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix}, \begin{bmatrix}
  g_1 \\
  \vdots \\
  g_n
\end{bmatrix} \cdot b \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n \\
  f_1 \\
  \vdots \\
  f_n \\
  g_1 \\
  \vdots \\
  g_n
\end{bmatrix}
\]

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

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

Abstract representation
Given $x_i$ (geometry), $f_i$ (r.h.s. vector), and $g_i$ (boundary conditions), compute

$$\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = A^{-1} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \cdot b \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

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

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

Let us interpret the sets of B-spline coefficients $\{x_i\}$, $\{f_i\}$, 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.
IgANet: replace \textit{computation}

\[
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n
\end{bmatrix} = A^{-1} \left( \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix}, \begin{bmatrix}
  g_1 \\
  \vdots \\
  g_n
\end{bmatrix} \right) \cdot b \left( \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix}, \begin{bmatrix}
  f_1 \\
  \vdots \\
  f_n
\end{bmatrix}, \begin{bmatrix}
  g_1 \\
  \vdots \\
  g_n
\end{bmatrix} \right)
\]
Isogeometric Analysis + Physics-Informed Machine Learning

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

\[
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n
\end{bmatrix}
= \text{IgANet}
\left( \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix},
\begin{bmatrix}
  f_1 \\
  \vdots \\
  f_n
\end{bmatrix},
\begin{bmatrix}
  g_1 \\
  \vdots \\
  g_n
\end{bmatrix};
(\xi^{(k)}, \eta^{(k)})^{N_{\text{samples}}} \right)_{k=1}^{N_{\text{samples}}}
\]
Isogeometric Analysis + Physics-Informed Machine Learning

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

\[
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n \\
\end{bmatrix} = \text{IgANet}
\begin{bmatrix}
  \begin{bmatrix}
    x_1 \\
    \vdots \\
    x_n \\
  \end{bmatrix},
  \begin{bmatrix}
    f_1 \\
    \vdots \\
    f_n \\
  \end{bmatrix},
  \begin{bmatrix}
    g_1 \\
    \vdots \\
    g_n \\
  \end{bmatrix}
\end{bmatrix};
\begin{bmatrix}
  (\xi^{(k)}, \eta^{(k)})^N_{k=1}
\end{bmatrix}
\]

Compute the solution from the trained neural network as follows

\[
u_h(\xi, \eta) = [B_1(\xi, \eta), \ldots, B_n(\xi, \eta)] \cdot
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n \\
\end{bmatrix},
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n \\
\end{bmatrix} = \text{IgANet}
\begin{bmatrix}
  \begin{bmatrix}
    x_1 \\
    \vdots \\
    x_n \\
  \end{bmatrix},
  \begin{bmatrix}
    f_1 \\
    \vdots \\
    f_n \\
  \end{bmatrix},
  \begin{bmatrix}
    g_1 \\
    \vdots \\
    g_n \\
  \end{bmatrix}
\end{bmatrix}
\]
IgANet architecture

\[ \text{coords } (\xi^{(k)}, \eta^{(k)})_{k=1}^N \]

\[ \text{loss} = \text{loss}_{\text{PDE}} + \text{loss}_{\text{BDR}} \]

\[ \frac{\partial \text{loss}}{\partial (w, b)} \rightarrow \text{update } w, b \]

and continue training

end training
Loss function

**Model problem:** Poisson’s equation with Dirichlet boundary conditions

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

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

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

\[
\text{For } [x_1, \ldots, x_n] \in S_{\text{geo}}, [f_1, \ldots, f_n] \in S_{\text{rhs}}, [g_1, \ldots, g_n] \in S_{\text{bcond}} \text{ do}
\]

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

\[
\text{Train IgANet } \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ f_1 \\ \vdots \\ f_n \\ g_1 \\ \vdots \\ g_n \end{pmatrix} ; (\xi_k, \eta_k)_{k=1}^{N_{\text{samples}}} \rightarrow \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}
\]

\text{EndFor}

\text{EndFor}

Details:

- \(7 \times 7\) bi-cubic tensor-product B-splines for \(x_h\) 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
Test case: Poisson’s equation on a variable annulus

\[ g \equiv 0 \]

\[ f \equiv 0, 1, \ldots, 11 \]

Master thesis work by Frank van Ruiten, TU Delft
Preliminary results
Preliminary results

$g \equiv 0$.

$g \equiv 1.4$.

$g \equiv 1$.

$f \equiv 5$.

$g \equiv 0$.

$0\text{rad}$.

$1\text{rad}$.

$2\text{rad}$.

$3\text{rad}$.

$4\text{rad}$.
Preliminary results

Master thesis work by Frank van Ruiten, TU Delft
Preliminary results

\[ g \equiv 0 \]

\[ f \equiv 15.5 \]

Master thesis work by 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

\[ x \mapsto u(x) := \text{NN}(x; f, g, G) = \sigma_L(W_L\sigma(\ldots(\sigma_1(W_1x + b_1))) + b_L) \]

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

Working principle of PINNs

\[ x \mapsto u(x) := \text{NN}(x; f, g, G) = \sigma_L(W_L \sigma(\ldots (\sigma_1(W_1x + b_1)))) + b_L \]

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

Working principle of IgANets

\[ [x_i, f_i, g_i]_{i=1,...,n} \mapsto [u_i]_{i=1,...,n} := \text{NN}(x_i, f_i, g_i, i = 1, \ldots, n) \]

- use mathematics to compute derivatives, e.g., \( \nabla_x u = (\sum_{i=1}^n \nabla_\xi B_i(\xi)u_i) J_{G}^{-t} \)
- use AD to compute gradients w.r.t. weights for training, i.e. (illustrated in 1D)

\[
\frac{\partial (d^r_{\xi} u(\xi))}{\partial w_k} = \sum_{i=1}^n \frac{\partial (d^r_{\xi} b^p_i u_i)}{\partial w_k} = \sum_{i=1}^n d^r_{\xi} b^p_i \frac{\partial \xi}{\partial w_k} u_i + \sum_{i=1}^n d^r_{\xi} b^p_i \frac{\partial u_i}{\partial w_k}
\]
Towards an ML-friendly B-spline evaluation

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

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

$$d^r_\xi u(\xi) = \left[d^r_\xi b^p_{i-p}(\xi), \ldots, d^r_\xi b^p_i(\xi)\right] \cdot \left[u_{i-p}, \ldots, u_i\right]$$

Textbook derivatives

$$d^r_\xi b^p_i(\xi) = (p - 1)\left(-d^{r-1}_\xi b^{p-1}_{i+1}(\xi) \frac{-d^{r-1}_\xi b^{p-1}_{i+1}(\xi)}{\xi_{i+p} - \xi_{i+1}} + d^{r-1}_\xi b^{p-1}_i(\xi) \frac{d^{r-1}_\xi b^{p-1}_i(\xi)}{\xi_{i+p-1} - \xi_i}\right)$$

with

$$b^p_i(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i}b^{p-1}_i(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}}b^{p-1}_{i+1}(\xi), \quad b^0_i(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$
Towards an ML-friendly B-spline evaluation

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

\[
\begin{bmatrix}
d_{\xi}^{r} b_{i-p}^{p}(\xi), \ldots, d_{\xi}^{r} b_{i}^{p}(\xi)
\end{bmatrix} = \frac{p!}{(p-r)!} R_{1}(\xi) \cdots R_{p-r}(\xi) d_{\xi} R_{p-r+1} \cdots d_{\xi} R_{p}
\]

with \( k \times k + 1 \) matrices \( R_{k}(\xi) \), e.g.

\[
R_{1}(\xi) = \begin{bmatrix}
\frac{\xi_{i+1} - \xi}{\xi_{i+1} - \xi_{i}} & \frac{\xi - \xi_{i}}{\xi_{i+1} - \xi_{i}} \\
\frac{\xi_{i+1} - \xi_{i}}{\xi_{i+1} - \xi_{i}} & \xi_{i+1} - \xi_{i}
\end{bmatrix}
\]

\[
R_{2}(\xi) = \begin{bmatrix}
\frac{\xi_{i+1} - \xi}{\xi_{i+1} - \xi_{i-1}} & \frac{\xi - \xi_{i-1}}{\xi_{i+1} - \xi_{i-1}} & 0 \\
\frac{\xi_{i+1} - \xi_{i}}{\xi_{i+1} - \xi_{i-1}} & \frac{\xi_{i+2} - \xi}{\xi_{i+1} - \xi_{i}} & \frac{\xi - \xi_{i}}{\xi_{i+2} - \xi_{i}}
\end{bmatrix}
\]

\[
R_{3}(\xi) = \ldots
\]
An ML-friendly B-spline evaluation

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

1. \( b = 1 \)
2. For \( k = 1, \ldots, p - r \)
   1. \( t_1 = (\xi_{i-k+1}, \ldots, \xi_i) \)
   2. \( t_2 = (\xi_{i+1}, \ldots, \xi_{i+k}) \)
   3. \( w = (\xi - t_1) \div (t_2 - t_1) \)
   4. \( b = [(1 - w) \circ b, 0] + [0, w \circ b] \)
3. For \( k = p - r + 1, \ldots, p \)
   1. \( t_1 = (\xi_{i-k+1}, \ldots, \xi_i) \)
   2. \( t_2 = (\xi_{i+1}, \ldots, \xi_{i+k}) \)
   3. \( w = 1 \div (t_2 - t_1) \)
   4. \( b = [-w \circ b, 0] + [0, w \circ b] \)

where \( \div \) and \( \circ \) 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. \( b = 1 \)
2. For \( k = 1, \ldots, p - r \)
   1. \( t_1 = (\xi_{i-k+1}, \ldots, \xi_i) \)
   2. \( t_{21} = (\xi_{i+1}, \ldots, \xi_{i+k}) - t_1 \)
   3. \( \text{mask} = (t_{21} < \text{tol}) \)
   4. \( w = (\xi - t_1 - \text{mask}) \div (t_{21} - \text{mask}) \)
   5. \( b = [(1 - w) \odot b, 0] + [0, w \odot b] \)
3. For \( k = p - r + 1, \ldots, p \)
   1. \( t_1 = (\xi_{i-k+1}, \ldots, \xi_i) \)
   2. \( t_{21} = (\xi_{i+1}, \ldots, \xi_{i+k}) - t_1 \)
   3. \( \text{mask} = (t_{21} < \text{tol}) \)
   4. \( w = (1 - \text{mask}) \div (t_{21} - \text{mask}) \)
   5. \( b = [-w \odot b, 0] + [0, w \odot b] \)

where \( \div \) and \( \odot \) denote the element-wise division and multiplication of vectors, respectively.
Performance evaluation - bivariate B-splines
Performance evaluation - trivariate B-splines

![Graph showing Wallclock time in ns/entry for different p values with reference and two processor configurations: Tesla V100S PCIe 32G and AMD EPYC 7402 24-Core Processor.]

- $p = 1$
- $p = 2$
- $p = 3$
- $p = 4$
- $p = 5$
Performance evaluation - bivariate B-splines

Wallclock time in ns/entry

Ookami Cluster @ Stony Brook: gcc12.2 ’-Ofast -mcpu=a64fx’
Performance evaluation - trivariate B-splines

Ookami Cluster @ Stony Brook: gcc12.2 ‘-0fast -mcpu=a64fx’
## Interactive Design-through-Analysis

### Front-ends

<table>
<thead>
<tr>
<th>Platform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gustaf</code></td>
<td>by TU Vienna</td>
</tr>
<tr>
<td>Three.js modeler</td>
<td>by SURF</td>
</tr>
</tbody>
</table>

WebSocket protocol for interactive spline modeling and visualization

### Back-ends

<table>
<thead>
<tr>
<th>Platform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Σ</code></td>
<td>by SURF</td>
</tr>
<tr>
<td>IgANet</td>
<td>by SURF</td>
</tr>
</tbody>
</table>

???
Conclusion and outlook

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

WIP

- interactive DTA workflow (/w SURF)
- use of IgA and IgANets in concert
- transfer learning upon basis refinement


What’s next

1. Journal paper and code release (including Python API) in preparation
2. CISM-ECCOMAS Summer School Scientific Machine Learning in Design Optimization
IgANets: Physics-Informed Machine Learning Embedded Into Isogeometric Analysis

Matthias Möller

Department of Applied Mathematics
Delft University of Technology, The Netherlands

Aromath seminar
25 April 2023, Sophia Antipolis Cedex, France

Joint work with Deepesh Toshniwal, Frank van Ruiten (TUD),
Casper van Leeuwen, Paul Melis (SURF), Jaewook Lee (TU Vienna)

Thank you very much!