IgaNets: Physics-Informed Machine Learning Embedded Into Isogeometric Analysis

Matthias Möller, Deepesh Toshniwal, Frank van Ruiten

Numerical Analysis, Delft Institute of Applied Mathematics
EEMCS, Delft University of Technology

NMC Scientific Days
19 & 20 April 2022
Numerics for PDE analysis

Many physical processes are modelled mathematically by (systems of) PDEs that require fast & accurate numerical methods to compute approximate solutions:

- particle methods: PIC (1955), SPH (1977), DPD (1992), RKPM (1995), ...
- hybrid particle-mesh methods: MPM (1990s), ...
- mesh-based methods: FEM (1940s), FDM (1950s), FVM (1971), IGA (2005), ...

Credit: www.superzelle.de – Janek Zimmer; University of Texas at Dallas (DOI: 10.1063/5.0036640); University of Minnesota – Eolos Wind Energy Research
Numerics for PDE analysis

Many physical processes are modelled mathematically by (systems of) PDEs that require fast & accurate numerical methods to compute approximate solutions:

- particle methods: PIC (1955), SPH (1977), DPD (1992), RKPM (1995), ...
- hybrid particle-mesh methods: MPM (1990s), ...
- mesh-based methods: FEM (1940s), FDM (1950s), FVM (1971), IGA (2005), ...

How fast is fast? And is it just about analysis?

Credit: www.superzelle.de – Janek Zimmer; University of Texas at Dallas (DOI: 10.1063/5.0036640); University of Minnesota – Eolos Wind Energy Research
Design through Analysis

We want it all: from really fast & moderately accurate to moderately fast & highly accurate!

Credit: Siemens – Simulation for Design Engineers
SciML for PDE analysis

- Physics-informed neural networks (PINNs) [Raissi, Perdikaris, Karniadakis, 2019]

\[ F = \partial_t U + \nabla \cdot f(U) \]
Physics-informed neural networks (PINNs) [Raissi, Perdikaris, Karniadakis, 2019]

- No pre-calculated data needed (unsupervised learning)
- Can be augmented with data (faster decay of loss function)
- Applicable to arbitrary PDEs (extra effort might be needed to impose ‘physics’)

- Convergence theory is in its infancy (different from classical numerical methods theory)
- Poor extrapolation capabilities (different geometries, problem parameters)
- Space-time treatment of time-dependent problems
SciML for PDE analysis

- Physics-informed neural networks (PINNs) [Raissi, Perdikaris, Karniadakis, 2019]
- Fourier neural operators (FNO) [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2020]
SciML for PDE analysis

- Physics-informed neural networks (PINNs) [Raissi, Perdikaris, Karniadakis, 2019]
- Fourier neural operators (FNO) [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2020]

+ Aims to learn the operator (not the PDE problem)
  - Pre-calculated training data is needed (supervised learning)
  - Assumes an efficient Fourier approximation of the solution
  - Designed for time-dependent PDEs
SciML for PDE analysis

- Physics-informed neural networks (PINNs) [Raissi, Perdikaris, Karniadakis, 2019]
- Fourier neural operators (FNO) [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2020]
- Learning nonlinear operators (DeepONets) [Lu, Jin, Pang, Zhang, Karniadakis, 2021]

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

+ Aims to learn the operator (not the PDE problem)
+ Claims to have excellent extrapolation capabilities

- Pre-calculated training data is needed (supervised learning)
- Designed for time-dependent PDEs
SciML for PDE analysis

- Physics-informed neural networks (PINNs) [Raissi, Perdikaris, Karniadakis, 2019]
- Fourier neural operators (FNO) [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2020]
- Learning nonlinear operators (DeepONets) [Lu, Jin, Pang, Zhang, Karniadakis, 2021]

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

+ Aims to learn the operator (not the PDE problem)
+ Claims to have excellent extrapolation capabilities
- Pre-calculated training data is needed (supervised learning)
- Designed for time-dependent PDEs

Combine mesh-based numerics with SciML for PDE analysis
Isogeometric Analysis

B-spline basis functions

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

\[ b^p_\ell(\xi) = \frac{\xi - \xi_\ell}{\xi_{\ell+p} - \xi_\ell} b^{p-1}_{\ell}(\xi) + \frac{\xi_{\ell+p+1} - \xi}{\xi_{\ell+p+1} - \xi_{\ell+1}} b^{p-1}_{\ell+1}(\xi) \]

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

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, ...

---

T.J.R. Hughes, J.A.Cottrell, Y.Bazilevs: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. CMAME 194, 2005.
Isogeometric Analysis

B-spline basis functions

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

\[ b^p_\ell(\xi) = \frac{\xi - \xi_\ell}{\xi_{\ell+p} - \xi_\ell} b^{p-1}_\ell(\xi) + \frac{\xi_{\ell+p+1} - \xi}{\xi_{\ell+p+1} - \xi_{\ell+1}} b^{p-1}_{\ell+1}(\xi) \]

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, ...

T.J.R. Hughes, J.A.Cottrell, Y.Bazilevs: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. CMAME 194, 2005.
Paradigm: represent ‘everything’ in terms of tensor products of B-spline basis functions

\[ B_i(\xi, \eta) := b^p_\ell(\xi) \cdot b^q_k(\eta), \quad i := (k - 1) \cdot n_\ell + \ell, \quad 1 \leq \ell \leq n_\ell, \quad 1 \leq k \leq n_k, \]
**Paradigm:** represent ‘everything’ in terms of tensor products of B-spline basis functions

\[ B_i(\xi, \eta) := b^p_\ell(\xi) \cdot b^q_k(\eta), \quad i := (k - 1) \cdot n_\ell + \ell, \quad 1 \leq \ell \leq n_\ell, \quad 1 \leq k \leq n_k, \]

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

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

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

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

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

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

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

Model problem: Poisson’s equation

\[ -\Delta u_h = f_h \quad \text{in} \quad \Omega_h, \quad u_h = g_h \quad \text{on} \quad \partial\Omega_h \]
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 \( x_i \) (geometry), \( f_i \) (r.h.s. vector), and \( g_i \) (boundary conditions), compute

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

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

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

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

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

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.
Isogeometric Analysis + PINNs

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

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

\[
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n
\end{bmatrix}
= \text{PINN}
\begin{bmatrix}
  \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} & f_1 \\
  \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} & f_n \\
  \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} & g_n
\end{bmatrix};
(\xi_k, \eta_k)_{k=1}^{N_{\text{samples}}}
\]

Compute the solution by evaluating the trained neural network

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

\[
\begin{align*}
\text{loss} &= \text{loss}_{\text{PDE}} + \text{loss}_{\text{BDR}} \\
\frac{\partial \text{loss}}{\partial (\mathbf{w}, \mathbf{b})} &\rightarrow \text{update } \mathbf{w}, \mathbf{b} \\
\text{loss} &< \varepsilon \\
\rightarrow &\text{end training} \\
\end{align*}
\]
Loss function

\[
\text{loss}_{\text{PDE}} = \frac{\alpha}{N_\Omega} \sum_{k=1}^{N_\Omega} |\Delta [u_h \circ x_h(\xi_k, \eta_k)] - f_h \circ x_h(\xi_k, \eta_k)|^2
\]

\[
\text{loss}_{\text{BDR}} = \frac{\beta}{N_\Gamma} \sum_{k=1}^{N_\Gamma} |u_h \circ x_h(\xi_k, \eta_k) - g_h \circ x_h(\xi_k, \eta_k)|^2
\]

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

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

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

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

EndFor

EndFor

IGA details: \(7 \times 7\) bi-cubic tensor-product B-splines for \(x_h\) 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

\[ g \equiv 0, \quad 0 \leq r \leq 1 \]

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

Ongoing master thesis work of Frank van Ruiten, TU Delft
Preliminary results

Ongoing master thesis work of Frank van Ruiten, TU Delft
Preliminary results

\[ g \equiv 0, 4 \text{ rad} \]

Ongoing master thesis work of Frank van Ruiten, TU Delft
Preliminary results

Ongoing master thesis work of Frank van Ruiten, TU Delft
Ongoing master thesis work of Frank van Ruiten, TU Delft
Preliminary results

\[ g \equiv 0 \]
\[ f \equiv 15.5 \]

Ongoing master thesis work of Frank van Ruiten, TU Delft
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.

---

We are hiring! AIO position will open soon! Thank you for your attention!