The future of computational steering: Interactive Design-through-Analysis

Matthias Möller[1], Casper van Leeuwen[2]

[1]Department of Applied Mathematics, TU Delft

[2]Scientific Visualisation, HPCV, SURF

SURF Research Day 2023, Amersfoort

Joint work with Deepesh Toshniwal, Frank van Ruiten (TU Delft),
Paul Melis (SURF), and Jaewook Lee (TU Vienna)

Computational steering

What do you know about it?

Computational steering – Dutch roots

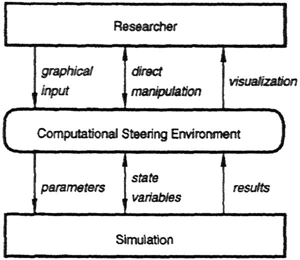
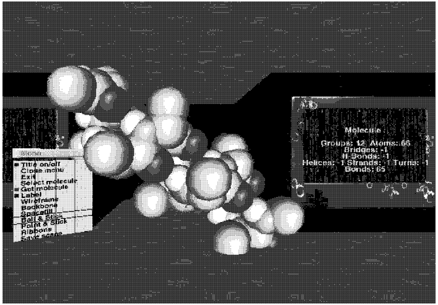
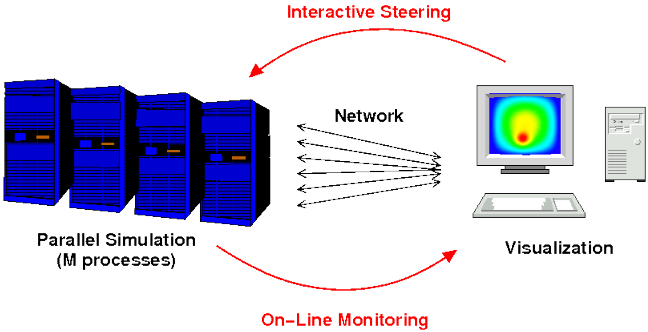


Fig. 1. Data flow between researcher, CSE, and simulation.



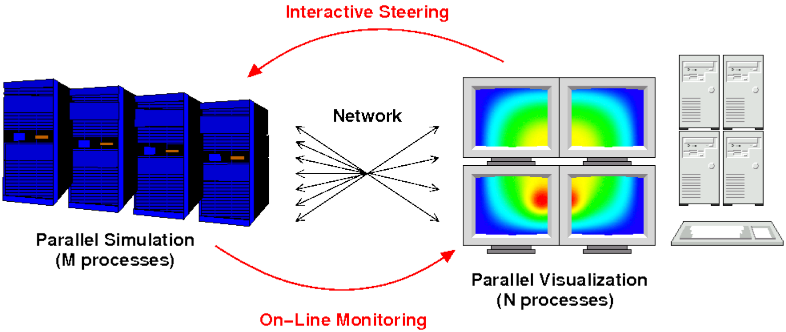
Left: R. v. Liere, J.D. Mulder, and J.J. v. Wijk, CWI 1997. Right: L.Renambot, H.E. Bal, D. Germans, and H.J.W. Spoelder, VU 2001.

Computational steering – the concept



The EPSN project by LaBRI (`https://www.labri.fr/projet/epsn/`) – 2007

Computational steering – the concept



The EPSN project by LaBRI (`https://www.labri.fr/projet/epsn/`) – 2007

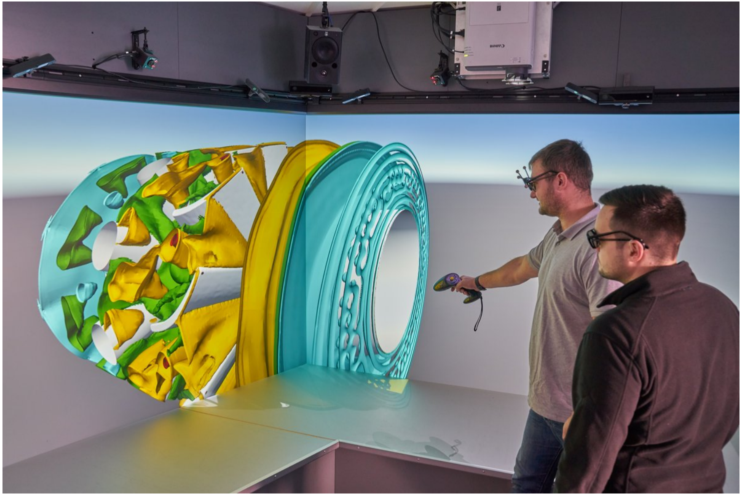
Computational steering – the early 2010s

Opening of the CAVE at HLRS High Performance Computing Center Stuttgart in 2012 (https://www.stuttgarter-zeitung.de/inhalt.forschung-in-stuttgart-ein-wuerfel-fuer-die-virtuelle-zukunft.1272d909-f2f8-4a53-8ad4-2c1712ab8842.html)

Computational steering – today

CAVE at HLRS High Performance Computing Center Stuttgart today (`https://www.hlrs.de/solutions/systems/cave`)

Computational steering – today

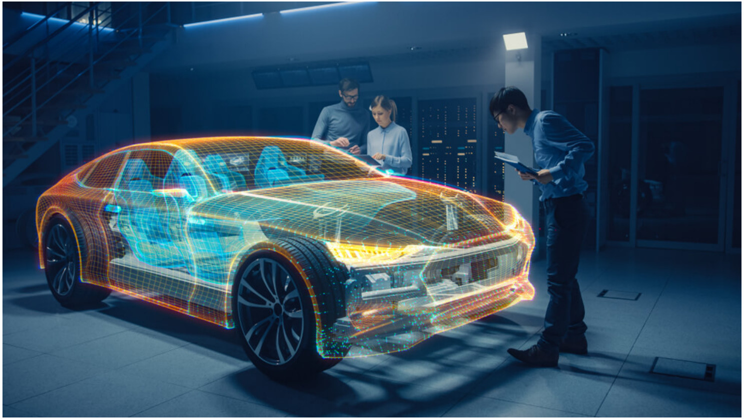


EU-funded project VITV – 2018-2021 (`https://www.b-tu.de/fg-medientechnik/forschung/virtuelles-triebwerk-v`)

Computational steering – the future

Siemens blog: *Virtual Reality in Engineering - Are You Ready?* – 7 July 2021 (https://blogs.sw.siemens.com/teamcenter/virtual-reality-in-engineering-are-you-ready/)

Computational steering – the future



Microsoft blog: *The future of mobility is now: Five themes to watch at CES 2023* – 4 January 2023 (https://blogs.microsoft.com/blog/2023/01/04/the-future-of-mobility-is-now-five-themes-to-watch-at-ces-2023/)

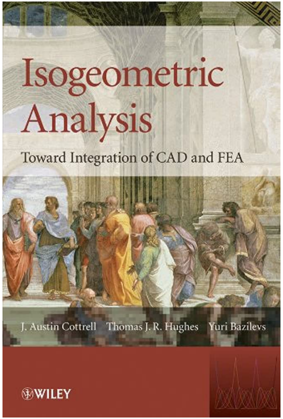
Computational steering – the future

That is, combining Computer-Aided **Design** and Computer-Aided Engineering **Analysis** to a unified **Design-through-Analysis** workflow.

Design-through-Analysis

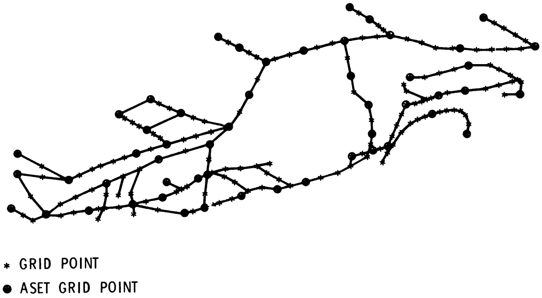
What do you know about it?

Design-through-Analysis – the inception



J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs, Wiley 2009

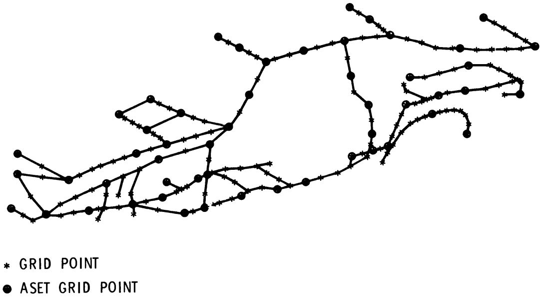
Design-through-Analysis – further back in time to the 1970s



J.A. Augustitus, M.M. Kamal, and L.J. Howell. Design through analysis of an experimental automobile structure. SAE Transactions, 86:2186-2198, 1977

Design-through-Analysis – further back in time to the 1970s

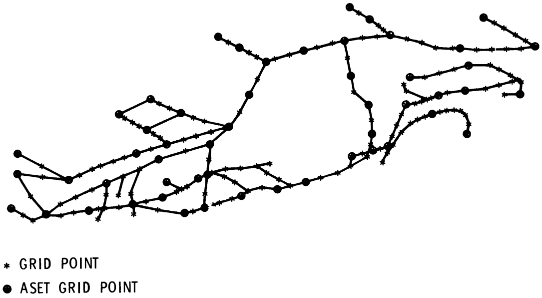
"The project described herein (which was completed early in 1975) is thought to have been the first coordinated ***design through analysis*** *of an entire automobile, followed by construction and experimental verification."*



J.A. Augustitus, M.M. Kamal, and L.J. Howell. Design through analysis of an experimental automobile structure. SAE Transactions, 86:2186-2198, 1977

Design-through-Analysis – further back in time to the 1970s

*"The project described herein (which was completed early in 1975) is thought to have been the first coordinated **design through analysis** of an entire automobile, followed by construction and experimental verification."*



*"[...] the potential value of **design through analysis** was demonstrated by a significant reduction in structural weight of the project vehicle."*

J.A. Augustitus, M.M. Kamal, and L.J. Howell. Design through analysis of an experimental automobile structure. SAE Transactions, 86:2186-2198, 1977

Computational steering: Interactive Design-through-Analysis

Vision: unified computational framework for **rapid prototyping** (design exploration phase) and **thorough analysis** (design optimization phase) of engineering designs

Ingredients

- physics-informed machine learning for rapid prototyping
- isogeometric analysis for accurate analysis

Computational steering: Interactive Design-through-Analysis

Vision: unified computational framework for **rapid prototyping** (design exploration phase) and **thorough analysis** (design optimization phase) of engineering designs

Ingredients

- physics-informed machine learning for rapid prototyping
- isogeometric analysis for accurate analysis

Let's see a live demo

Let's have a look under the hood

The big picture

Front-ends

IgANet-frontend
by SURF

gustaf
by TU Vienna

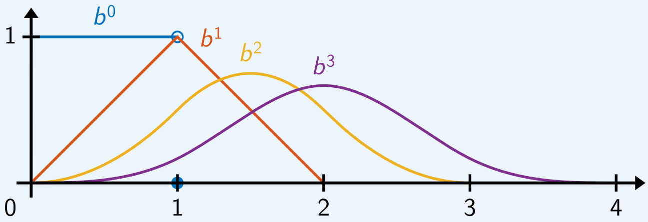
WebSockets protocol for interactive Design-through-Analysis

Back-ends

IgANet

B-spline basis functions

Cox de Boor recursion formula



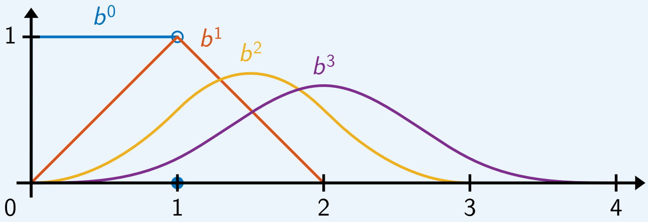
knot vector $\Xi = [0, 1, 2, 3, 4]$

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

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

B-spline basis functions

Cox de Boor recursion formula



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

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

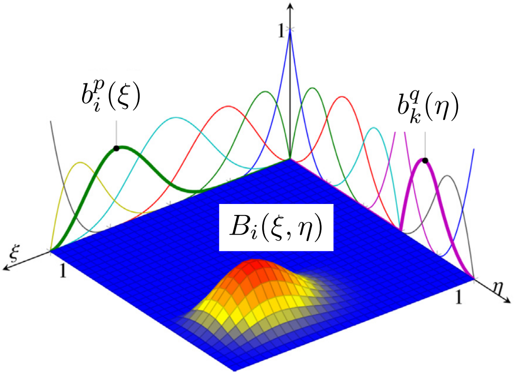
$$b_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} b_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} b_{i+1}^{p-1}(\xi)$$

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

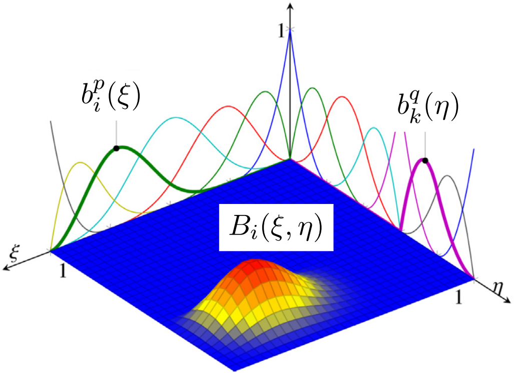
$$B_i(\xi,\eta) := b_i^p(\xi)\cdot b_k^q(\eta), \qquad i := (k-1)\cdot n_i + i, \quad 1 \le i \le n_i, \quad 1 \le k \le n_k,$$



Isogeometric Analysis

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

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

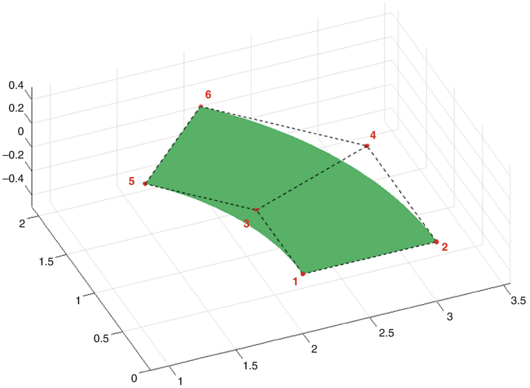


Many more good properties: partition of unity $\sum_{i=1}^{n} B_i(\xi,\eta)\equiv 1$, C^{p-1} continuity, ...

Isogeometric Analysis

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

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

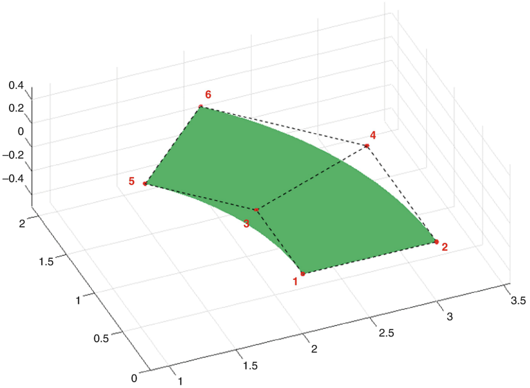


- the shape of Ω_h is fully specified by the set of **control points** $\mathbf{x}_i \in \mathbb{R}^d$

Isogeometric Analysis

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

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

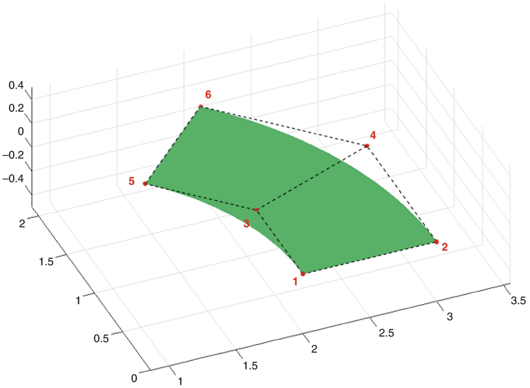


- the shape of Ω_h is fully specified by the set of **control points** $\mathbf{x}_i \in \mathbb{R}^d$
- interior control points must be chosen such that 'grid lines' do not fold as this violates the bijectivity of $\mathbf{x}_h : \hat{\Omega} \to \Omega_h$

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 Ω_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 Ω_h and can be used to generate finer computational 'grids' for the analysis

Isogeometric Analysis

Model problem: Poisson's equation

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

with

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

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

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

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

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**

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

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.

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**

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

Isogeometric Analysis + Physics-Informed Machine Learning

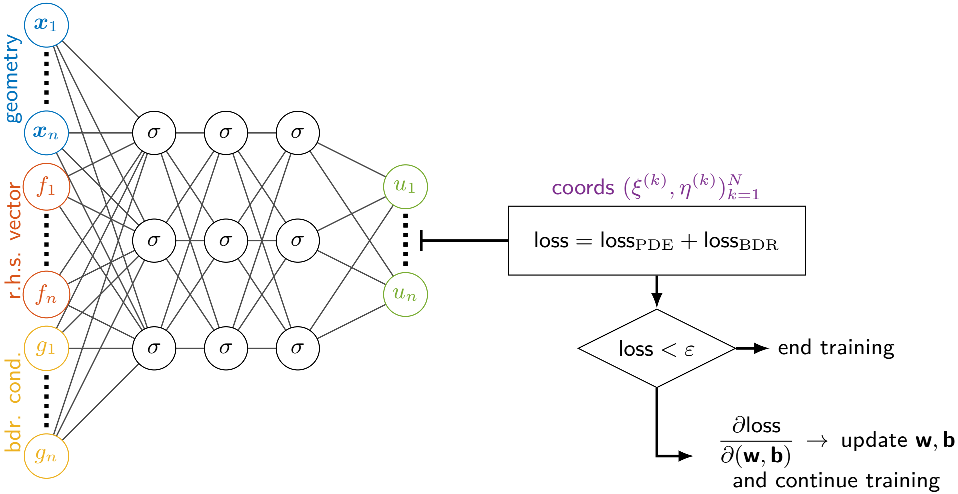
IgANet: replace **computation** by **physics-informed machine learning**

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

Compute the solution from the trained neural network as follows

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

IgANet architecture



Loss function

Model problem: Poisson's equation with Dirichlet boundary conditions

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

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

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 $(\xi_k, \eta_k) \in [0,1]^2$ (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

Computational costs

Working principle of PINNs

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

- 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

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

- 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

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

- 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

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

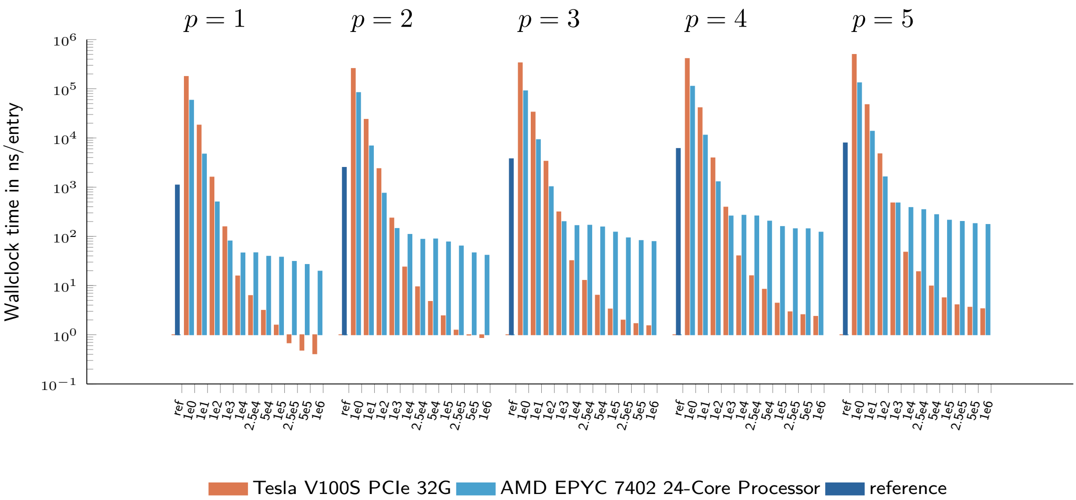
An ML-friendly B-spline evaluation

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

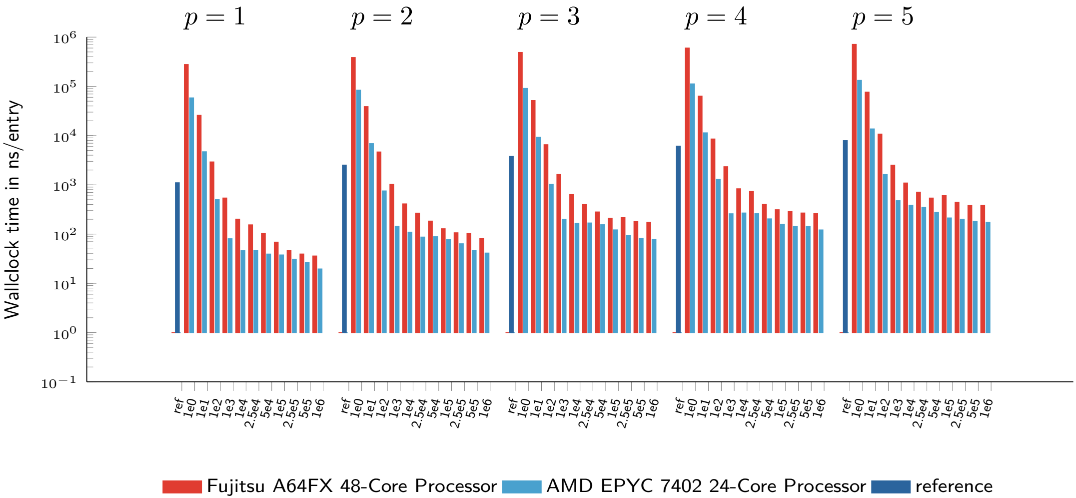
1. $\mathbf{b} = 1$
2. For $k = 1, \ldots, p - r$
   1. $\mathbf{t}_1 = (\xi_{i-k+1}, \ldots, \xi_i)$
   2. $\mathbf{t}_{21} = (\xi_{i+1}, \ldots, \xi_{i+k}) - \mathbf{t}_1$
   3. $\textbf{mask} = (\mathbf{t}_{21} < \text{tol})$
   4. $\mathbf{w} = (\xi - \mathbf{t}_1 - \textbf{mask}) \div (\mathbf{t}_{21}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

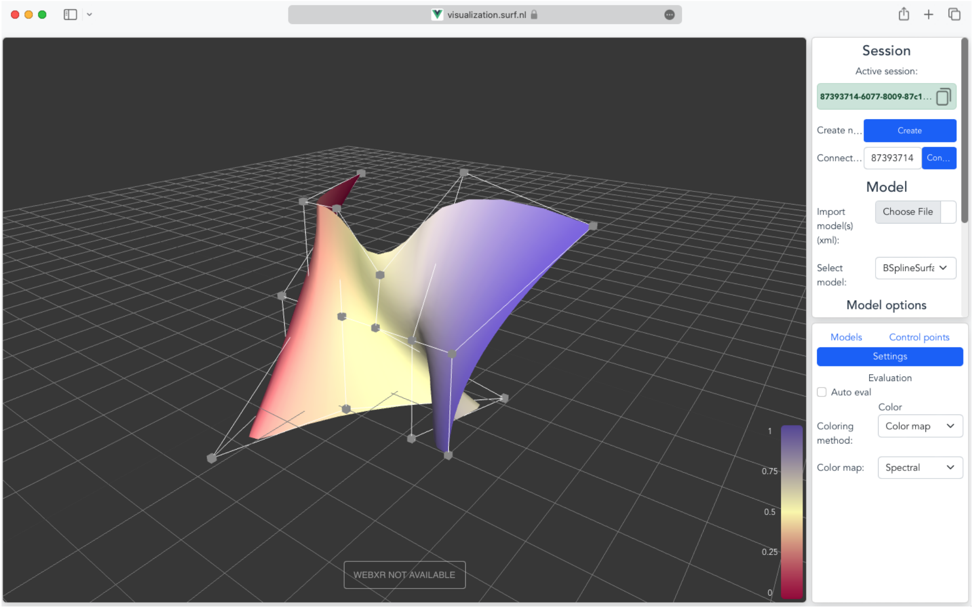
Performance evaluation - bivariate B-splines



Performance evaluation - bivariate B-splines



Let's move on to the front-end



The future of computational steering: Interactive Design-through-Analysis

Matthias Möller[1], Casper van Leeuwen[2]

[1]Department of Applied Mathematics, TU Delft

[2]Scientific Visualisation, HPCV, SURF

SURF Research Day 2023, Amersfoort

Joint work with Deepesh Toshniwal, Frank van Ruiten (TU Delft), Paul Melis (SURF), and Jaewook Lee (TU Vienna)

Thank you very much!