CIME-EMS Summer School in Applied Mathematics

Splines and PDEs: Recent Advances
from Approximation Theory to Structured Numerical Linear Algebra

July 3 - July 7, 2017 - Cetraro

Efficient Formation of isogeometric matrices

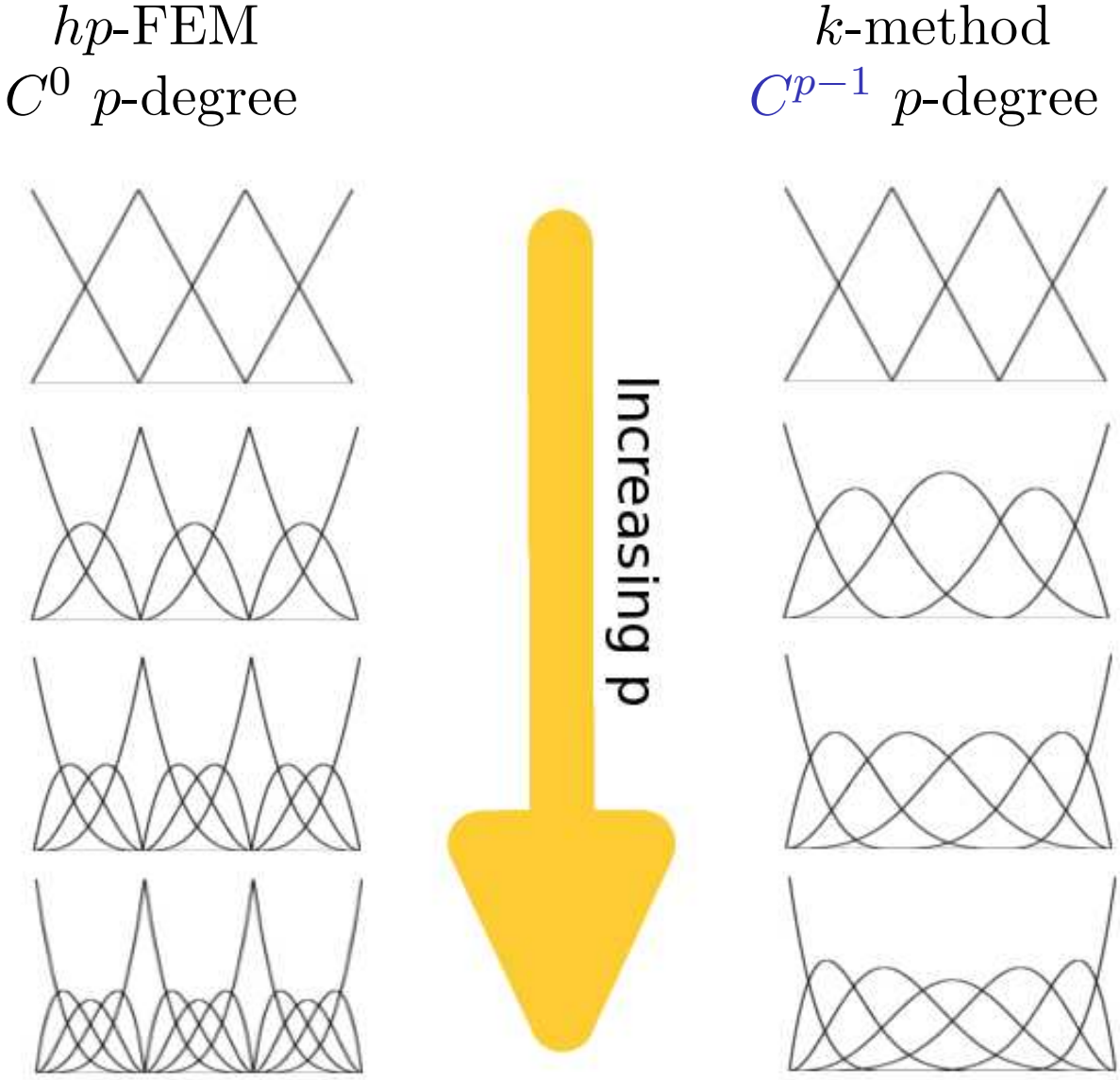
Giancarlo Sangalli

joint work with Mattia Tani and Francesco Calabrò

July 7, 2017

k-method is the high-order isogeometric method

k-method is the high-order isogeometric method



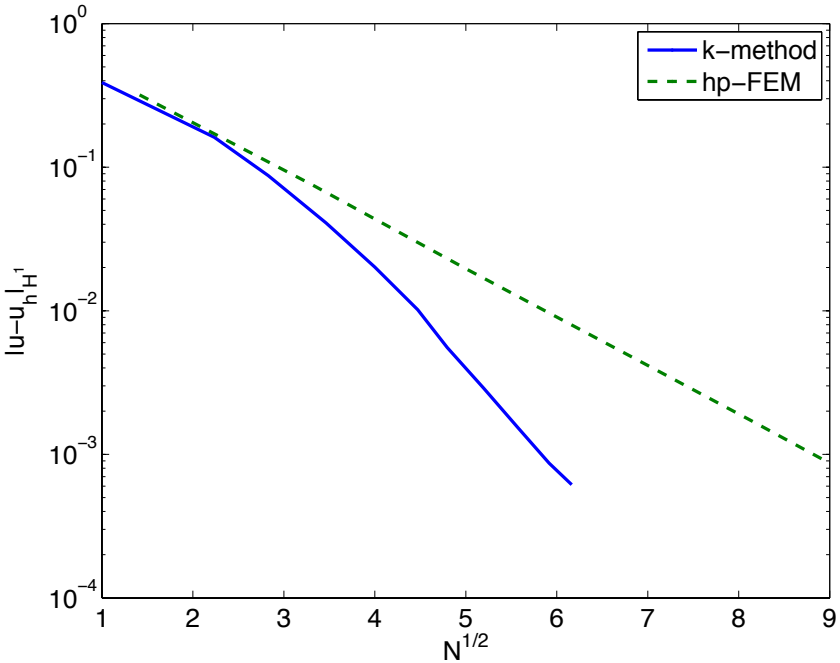
The k-method vs C^0 hp-FEM

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom



Approximation of $u(x) = x^{0.7} - x$.

The mesh is geometrically graded and the spline degree p is proportional to the number of elements for IGA. Mesh-size and degrees are optimally selected for hp-FEM.

Error is $|u - u_h|_{H^1} \leq C \exp\left(-b\sqrt{N}\right)$ in both cases, with larger b for IGA

[Buffa, Sangalli, and Schwab, 2014]

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom

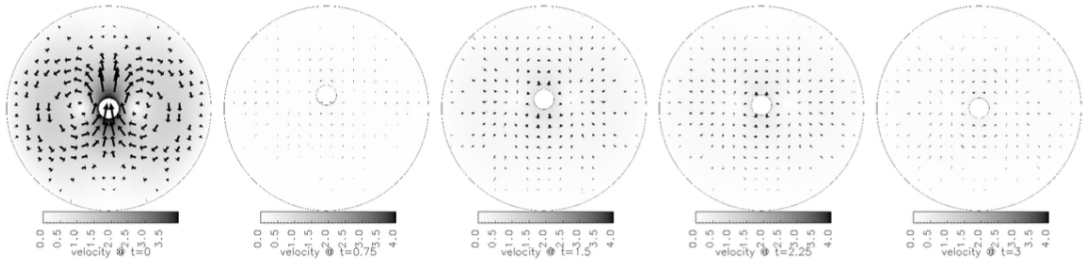
The role of continuity in RBVMS modeling of turbulence

[Akkerman, Bazilevs, Calo, Hughes, and Hulshoff, 2008]

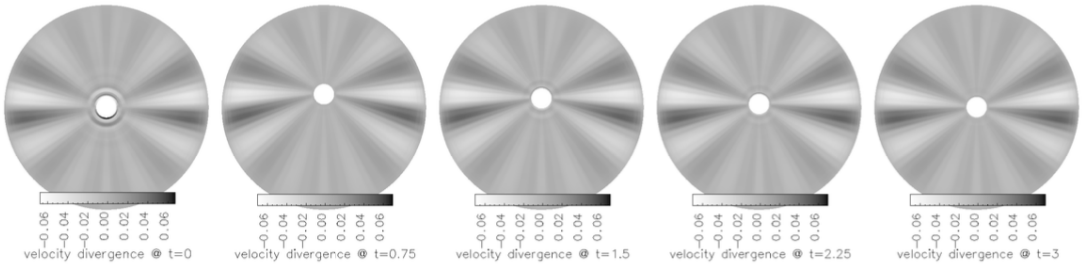
"We compared turbulent channel flow results using C^0 and C^1-continuous quadratic discretizations. Using a C^1-continuous quadratic basis yields more accurate mean flow and fluctuating quantities than C^0-continuous quadratic basis functions. We conclude that smooth NURBS basis functions have advantages over C^0-continuous finite elements in turbulent flow calculations."

The k-method vs C^0 hp-FEM

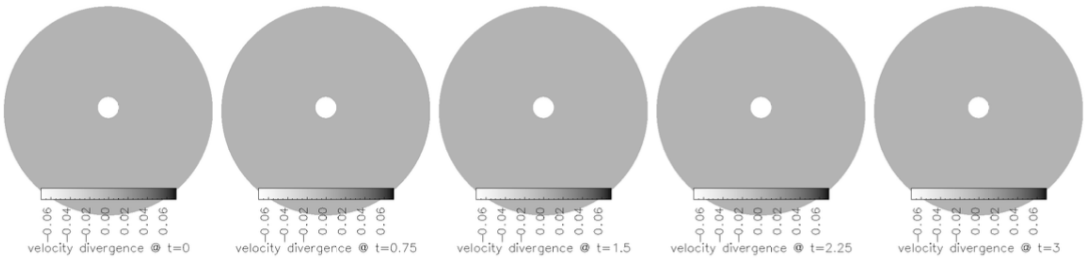
- higher accuracy per degree of freedom
- structure-preserving



(a) Velocity snapshots computed with the RT discretization



(b) Divergence of the velocity computed with the TH discretization



(c) Divergence of the velocity computed with the RT discretization

"Div-conforming" k-method

[Buffa, de Falco, and Sangalli, 2011]

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom
- structure-preserving

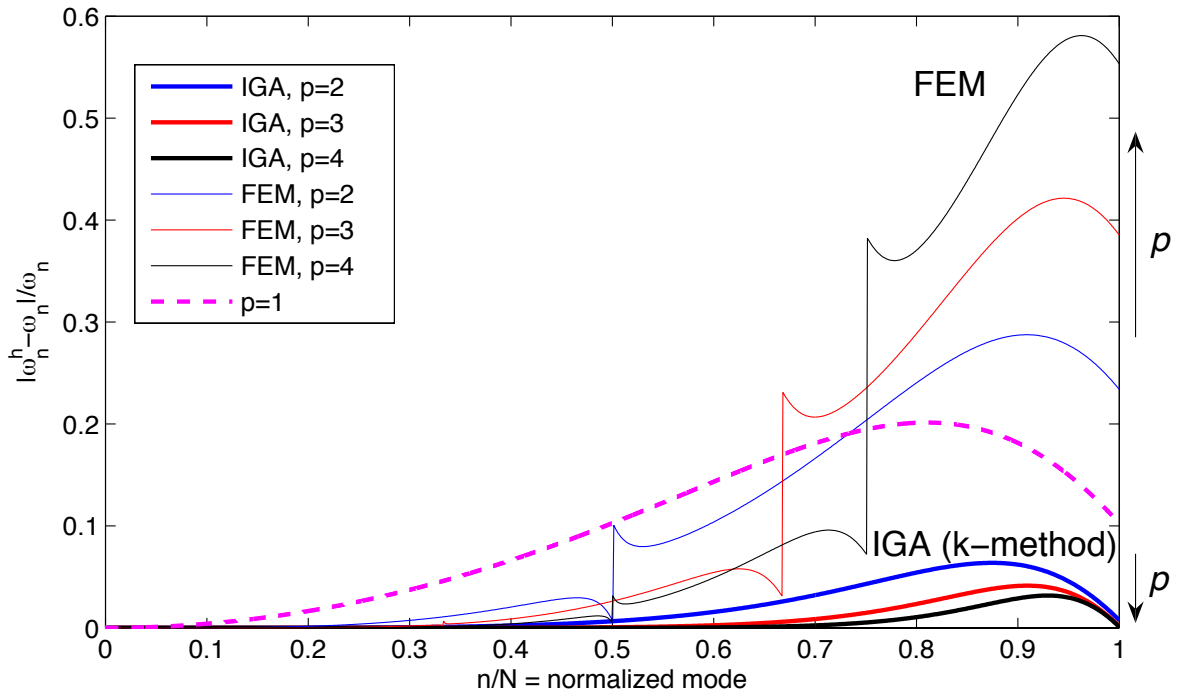
Isogeometric Div-conforming B-splines for the Unsteady NS Eq.'s

[Evans and Hughes, 2013]

"... As incompressibility is satisfied pointwise, these semi-discretizations replicate the geometric structure of the unsteady Navier-Stokes equations and admit discrete balance laws for momentum, angular momentum, energy, vorticity, enstrophy, and helicity.... by combining the spectral-like properties of B-splines with the preservation of the geometric structure of the unsteady NS equations, our semi-discretization procedure may become a useful tool for both engineering analysis and the mathematical study of the unsteady NS equations. "

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom
- structure-preserving
- improves the approximation of the spectrum



[Cottrell, Reali, Bazilevs, and Hughes, 2006; Hughes, Reali, and Sangalli, 2008]

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom
- structure-preserving
- improves the approximation of the spectrum

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom
- structure-preserving
- improves the approximation of the spectrum
- but the computational cost per d.o.f. is higher in the k-method

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom
- structure-preserving
- improves the approximation of the spectrum
- but the computational cost per d.o.f. is higher in the k-method

However in complex isogeometric simulation, e.g. VMS turbulence

[Bazilevs, Calo, Cottrell, Hughes, Reali, and Scovazzi, 2007]

"We found quadratic NURBS to give very significant accuracy advantages over linear elements. ... Cubics, on the other hand, increased cost considerably ... These remarks need to be qualified by the fact that our implementation of higher-order elements is not yet optimized in any way."

The k-method vs C^0 hp-FEM

- higher accuracy per degree of freedom
- structure-preserving
- improves the approximation of the spectrum
- but the computational cost per d.o.f. is higher in the k-method

"The cost of continuity..." [Collier, Pardo, Dalcin, Paszynski, and Calo, 2012]

"However this advantage comes at a cost which is not seen until looking in detail at the algorithms used to solve the resulting linear systems. In this case, we have shown that direct solvers require orders of magnitude more time and memory as continuity increases."

"Isogeometric Collocation: Cost Comparison with Galerkin Methods"

[Schillinger, Evans, Reali, Scott, and Hughes, 2013]

".. the total time for the formation and assembly of the global stiffness matrix of a given size takes ... almost two minutes for FEM and almost 1.5 hours in IGA (Galerkin)"

Fast formation/assembly of the system matrix

The model problem

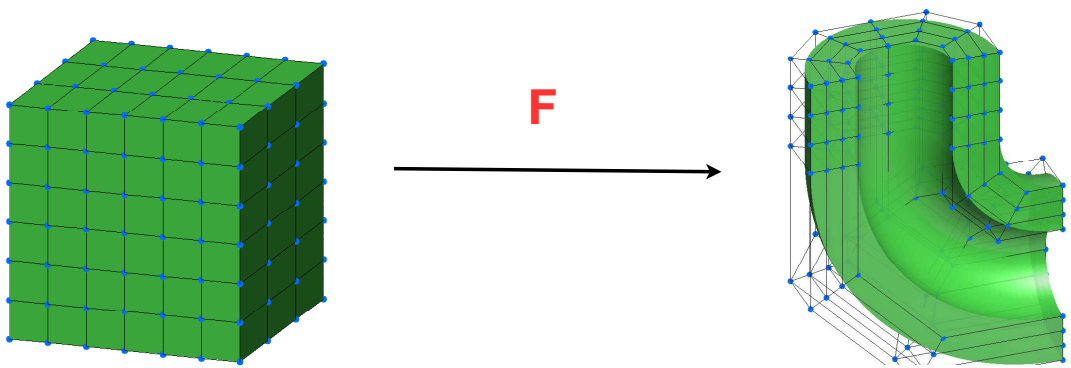
$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson

The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

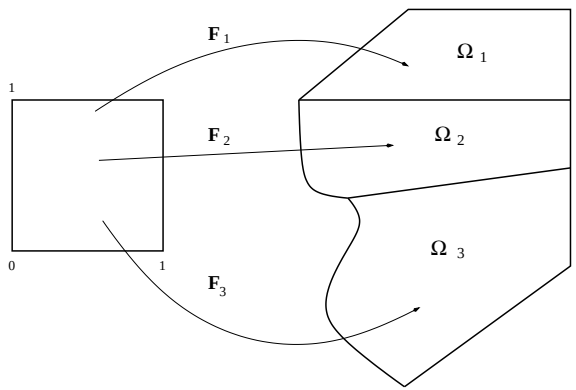
- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)



The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

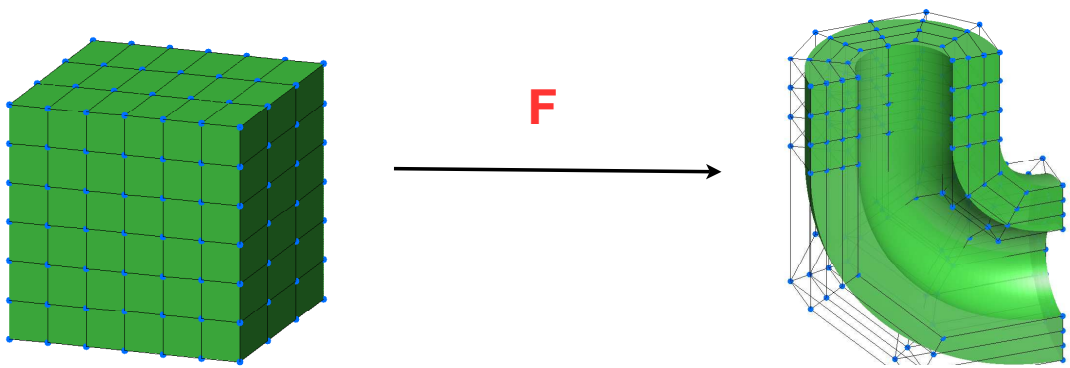
- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)



The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)
- tensor-product patches



The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)
- tensor-product patches
- focus on k-method: C^{p-1} (arbitrary continuity is the same)

The model problem

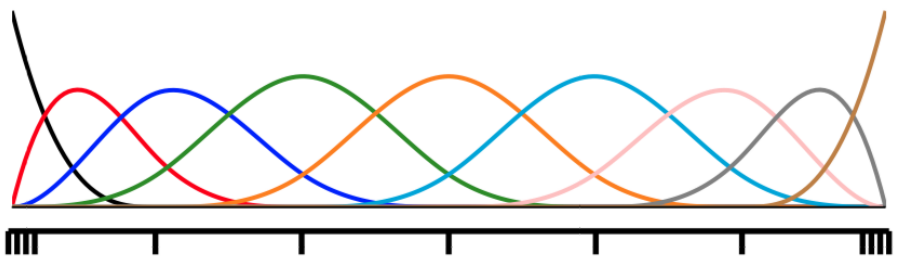
$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)
- tensor-product patches
- focus on k-method: C^{p-1} (arbitrary continuity is the same)
- isogeometric (spline) space denoted by $\mathcal{S}^p_{p-1}$ with total dimension $N_{\mathsf{DOF}} = n^d$, with $n \gg p$, and $N_{\mathsf{EL}} \approx N_{\mathsf{DOF}}$

The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)
- tensor-product patches
- focus on k-method: C^{p-1} (arbitrary continuity is the same)
- isogeometric (spline) space denoted by $\mathcal{S}^p_{p-1}$ with total dimension $N_{\mathsf{DOF}} = n^d$, with $n \gg p$, and $N_{\mathsf{EL}} \approx N_{\mathsf{DOF}}$



The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)
- tensor-product patches
- focus on k-method: C^{p-1} (arbitrary continuity is the same)
- isogeometric (spline) space denoted by $\mathcal{S}^p_{p-1}$ with total dimension $N_{\mathsf{DOF}} = n^d$, with $n \gg p$, and $N_{\mathsf{EL}} \approx N_{\mathsf{DOF}}$
- non-uniform knot vector

The model problem

$$\begin{aligned} -\nabla \cdot K \nabla \mathrm{u} &= \mathrm{f} \qquad \text{in } \Omega \subseteq \mathbb{R}^d,\ d = 2, 3 \\ \mathrm{u} &= 0 \qquad \text{in } \partial\Omega \end{aligned}$$

- d-dimensional scalar Poisson
- single patch parametrization (multi-patch through DD)
- tensor-product patches
- focus on k-method: C^{p-1} (arbitrary continuity is the same)
- isogeometric (spline) space denoted by $\mathcal{S}^p_{p-1}$ with total dimension $N_{\mathsf{DOF}} = n^d$, with $n \gg p$, and $N_{\mathsf{EL}} \approx N_{\mathsf{DOF}}$
- non-uniform knot vector
- focus on Galerkin method

Isogeometric Galerkin matrices: element-wise quadrature

$$\mathbb{A}u = f$$

- The storage cost for $\mathbb{A}$ is $O(N_{\mathsf{DOF}} p^d)$ non-zero entries.

Isogeometric Galerkin matrices: element-wise quadrature

$$\mathbb{A}u = f$$

- The storage cost for $\mathbb{A}$ is $O(N_{\mathsf{DOF}} p^d)$ non-zero entries.
- the optimal computational cost is: $O(N_{\mathsf{DOF}} p^d)$ FLOPs

Isogeometric Galerkin matrices: element-wise quadrature

$$\mathbb{A}u = f$$

- The storage cost for $\mathbb{A}$ is $O(N_{\mathsf{DOF}}p^d)$ non-zero entries.
- the optimal computational cost is: $O(N_{\mathsf{DOF}}p^d)$ FLOPs
- with element-wise assembling loop and element-wise standard Gaussian quadrature (SGQ): each elemental stiffness matrix has dimension $(p+1)^d \times (p+1)^d = O(p^{2d})$ and each entry is calculated by quadrature on $(p+1)^d = O(p^d)$ Gauss points. Total cost is $O(N_{\mathsf{EL}}p^{3d})$ that is $O(N_{\mathsf{DOF}}p^{3d})$.

Isogeometric Galerkin matrices: element-wise quadrature

$$\mathbb{A}u = f$$

- The storage cost for $\mathbb{A}$ is $O(N_{\mathsf{DOF}}p^d)$ non-zero entries.
- the optimal computational cost is: $O(N_{\mathsf{DOF}}p^d)$ FLOPs
- with element-wise assembling loop and element-wise standard Gaussian quadrature (SGQ): each elemental stiffness matrix has dimension $(p+1)^d \times (p+1)^d = O(p^{2d})$ and each entry is calculated by quadrature on $(p+1)^d = O(p^d)$ Gauss points. Total cost is $O(N_{\mathsf{EL}}p^{3d})$ that is $O(N_{\mathsf{DOF}}p^{3d})$.
- The standard way to reduce the cost is to reduce the number of quadrature points, for example by reduced Gaussian [Adam, Hughes, Bouabdallah, Zarroug, and Maitournam, 2015; Schillinger, Hossain, and Hughes, 2014; Hillman, Chen, and Bazilevs, 2015]) or generalised Gaussian quadrature (GGQ) [Hughes, Reali, and Sangalli, 2010; Bremer, Gimbutas, and Rokhlin, 2010; Cheng, Rokhlin, and Yarvin, 1999; Ma, Rokhlin, and Wandzura, 1996] .

Isogeometric Galerkin matrices: element-wise quadrature

$$\mathbb{A}u = f$$

- The storage cost for $\mathbb{A}$ is $O(N_{\mathsf{DOF}}p^d)$ non-zero entries.
- the optimal computational cost is: $O(N_{\mathsf{DOF}}p^d)$ FLOPs
- with element-wise assembling loop and element-wise standard Gaussian quadrature (SGQ): each elemental stiffness matrix has dimension $(p+1)^d \times (p+1)^d = O(p^{2d})$ and each entry is calculated by quadrature on $(p+1)^d = O(p^d)$ Gauss points. Total cost is $O(N_{\mathsf{EL}}p^{3d})$ that is $O(N_{\mathsf{DOF}}p^{3d})$.
- The standard way to reduce the cost is to reduce the number of quadrature points, for example by reduced Gaussian [Adam, Hughes, Bouabdallah, Zarroug, and Maitournam, 2015; Schillinger, Hossain, and Hughes, 2014; Hillman, Chen, and Bazilevs, 2015]) or generalised Gaussian quadrature (GGQ) [Hughes, Reali, and Sangalli, 2010; Bremer, Gimbutas, and Rokhlin, 2010; Cheng, Rokhlin, and Yarvin, 1999; Ma, Rokhlin, and Wandzura, 1996] .

Element-wise Gauss vs weighted quadrature (WQ)

Consider the mass matrix

$$\mathbb{M} = \{m_{\boldsymbol{i},\boldsymbol{j}}\} = \left\{ \int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \right\}$$

Gauss Quadrature (GQ) (or Generalized GQ)

$$m_{\boldsymbol{i},\boldsymbol{j}} \approx \mathfrak{Q}^{\mathrm{GQ}}(cB_{\boldsymbol{i}}B_{\boldsymbol{j}}) = \sum_{\boldsymbol{q}} w_{\boldsymbol{q}}^{\mathrm{GQ}} c(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{GQ}}) B_{\boldsymbol{i}}(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{GQ}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{GQ}})$$

weighted quadrature (WQ)

$$m_{\boldsymbol{i},\boldsymbol{j}} = \int_{\Omega} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(cB_{\boldsymbol{j}}) = \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}}^{\mathrm{WQ}} c(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{WQ}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{WQ}})$$

Element-wise Gauss vs weighted quadrature (WQ)

Consider the mass matrix

$$\mathbb{M} = \{m_{\boldsymbol{i},\boldsymbol{j}}\} = \left\{ \int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \right\}$$

Gauss Quadrature (GQ) (or Generalized GQ)

$$m_{\boldsymbol{i},\boldsymbol{j}} \approx \mathfrak{Q}^{\mathrm{GQ}}(cB_{\boldsymbol{i}}B_{\boldsymbol{j}}) = \sum_{\boldsymbol{q}} w_{\boldsymbol{q}}^{\mathrm{GQ}} c(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{GQ}}) B_{\boldsymbol{i}}(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{GQ}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{GQ}})$$

weighted quadrature (WQ)

$$m_{\boldsymbol{i},\boldsymbol{j}} = \int_{\Omega} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta})d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(cB_{\boldsymbol{j}}) = \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}}^{\mathrm{WQ}} c(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{WQ}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}}^{\mathrm{WQ}})$$

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}$$

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)\, B_{\boldsymbol{j}}(\cdot)) := \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}} c(\boldsymbol{x}_{\boldsymbol{q}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}})$$

- We incorporate the test function $B_{\boldsymbol{i}}(\boldsymbol{\zeta})$ into the integral measure; thus the quadrature rule $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}$, and in particular the weights $w_{\boldsymbol{q},\boldsymbol{i}}$, depend on $\boldsymbol{i}$.

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)\, B_{\boldsymbol{j}}(\cdot)) := \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}} c(\boldsymbol{x_q}) B_{\boldsymbol{j}}(\boldsymbol{x_q})$$

- We incorporate the test function $B_{\boldsymbol{i}}(\boldsymbol{\zeta})$ into the integral measure; thus the quadrature rule $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}$, and in particular the weights $w_{\boldsymbol{q},\boldsymbol{i}}$, depend on $\boldsymbol{i}$.
- The function $c(\cdot)\, B_{\boldsymbol{j}}(\cdot)$ plays the role the integrand function.

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)\, B_{\boldsymbol{j}}(\cdot)) := \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}} c(\boldsymbol{x_q}) B_{\boldsymbol{j}}(\boldsymbol{x_q})$$

- We incorporate the test function $B_{\boldsymbol{i}}(\boldsymbol{\zeta})$ into the integral measure; thus the quadrature rule $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}$, and in particular the weights $w_{\boldsymbol{q},\boldsymbol{i}}$, depend on $\boldsymbol{i}$.
- The function $c(\cdot)\, B_{\boldsymbol{j}}(\cdot)$ plays the role the integrand function.
- The quadrature points $\boldsymbol{x_q}$ are selected:
  - a-priori (they do not depend on $\boldsymbol{i}$) as suitable interpolation points,
  - looking for the best ones, and dependents on $\boldsymbol{i}$: $\rightarrow \boldsymbol{x_{q,i}}$

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)\, B_{\boldsymbol{j}}(\cdot)) := \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}} c(\boldsymbol{x}_{\boldsymbol{q}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}})$$

- We incorporate the test function $B_{\boldsymbol{i}}(\boldsymbol{\zeta})$ into the integral measure; thus the quadrature rule $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}$, and in particular the weights $w_{\boldsymbol{q},\boldsymbol{i}}$, depend on $\boldsymbol{i}$.
- The function $c(\cdot)\, B_{\boldsymbol{j}}(\cdot)$ plays the role the integrand function.
- The quadrature points $\boldsymbol{x}_{\boldsymbol{q}}$ are selected:
  - a-priori (they do not depend on $\boldsymbol{i}$) as suitable interpolation points,
  - looking for the best ones, and dependents on $\boldsymbol{i}$: $\rightarrow \boldsymbol{x}_{\boldsymbol{q},\boldsymbol{i}}$

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)\, B_{\boldsymbol{j}}(\cdot)) := \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}} c(\boldsymbol{x}_{\boldsymbol{q}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}})$$

- We incorporate the test function $B_{\boldsymbol{i}}(\boldsymbol{\zeta})$ into the integral measure; thus the quadrature rule $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}$, and in particular the weights $w_{\boldsymbol{q},\boldsymbol{i}}$, depend on $\boldsymbol{i}$.
- The function $c(\cdot)\, B_{\boldsymbol{j}}(\cdot)$ plays the role the integrand function.
- The quadrature points $\boldsymbol{x}_{\boldsymbol{q}}$ are selected:
  - a-priori (they do not depend on $\boldsymbol{i}$) as suitable interpolation points,
  - looking for the best ones, and dependents on $\boldsymbol{i}$: $\rightarrow \boldsymbol{x}_{\boldsymbol{q},\boldsymbol{i}}$

Each matrix row is computed independently: "embarrassingly parallel"

Weighted Quadrature (WQ) for the mass matrix

We want to approximate the $(\boldsymbol{i}, \boldsymbol{j})$ entry of the mass matrix:

$$\int_{[0,1]^d} c(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, (B_{\boldsymbol{i}}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}) \approx \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)\, B_{\boldsymbol{j}}(\cdot)) := \sum_{\boldsymbol{q}} w_{\boldsymbol{q},\boldsymbol{i}} c(\boldsymbol{x}_{\boldsymbol{q}}) B_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{q}})$$

- We incorporate the test function $B_{\boldsymbol{i}}(\boldsymbol{\zeta})$ into the integral measure; thus the quadrature rule $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}$, and in particular the weights $w_{\boldsymbol{q},\boldsymbol{i}}$, depend on $\boldsymbol{i}$.
- The function $c(\cdot)\, B_{\boldsymbol{j}}(\cdot)$ plays the role the integrand function.
- The quadrature points $\boldsymbol{x}_{\boldsymbol{q}}$ are selected:
  - a-priori (they do not depend on $\boldsymbol{i}$) as suitable interpolation points,
  - looking for the best ones, and dependents on $\boldsymbol{i}$: $\rightarrow \boldsymbol{x}_{\boldsymbol{q},\boldsymbol{i}}$

- How to select the quadrature points $\boldsymbol{x}_{\boldsymbol{q}}$? how many?
- How to compute the weights $w_{\boldsymbol{q},\boldsymbol{i}}$ relative to $B_{\boldsymbol{i}}$?

Weighted Quadrature

Exactness conditions:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^d} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \qquad \forall\, \boldsymbol{j}$$

Weighted Quadrature

Exactness conditions:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^d} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \qquad \forall\, \boldsymbol{j}$$

- For each given $\boldsymbol{i}$, the exactness requirements are imposed on the trial function space $\mathcal{S}_{p-1}^{p}$.

Weighted Quadrature

Exactness conditions:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^d} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \qquad \forall\, \boldsymbol{j}$$

- For each given $\boldsymbol{i}$, the exactness requirements are imposed on the trial function space $\mathcal{S}^{p}_{p-1}$.
- If c is constant, then $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)B_{\boldsymbol{j}}(\cdot)) =: \tilde{m}_{\boldsymbol{i},\boldsymbol{j}} = m_{\boldsymbol{i},\boldsymbol{j}}$.

Weighted Quadrature

Exactness conditions:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^d} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \qquad \forall\, \boldsymbol{j}$$

- For each given $\boldsymbol{i}$, the exactness requirements are imposed on the trial function space $\mathcal{S}^p_{p-1}$.
- If c is constant, then $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c(\cdot)B_{\boldsymbol{j}}(\cdot)) =: \tilde{m}_{\boldsymbol{i},\boldsymbol{j}} = m_{\boldsymbol{i},\boldsymbol{j}}$.
- For non-constant c, the new mass matrix is nonsymmetric in general ($\tilde{m}_{\boldsymbol{i},\boldsymbol{j}} \neq \tilde{m}_{\boldsymbol{j},\boldsymbol{i}}$).
- optimal accuracy follows from

$$\sup_{v_h = \sum_{\boldsymbol{i}} v_{\boldsymbol{i}} B_{\boldsymbol{i}}} \frac{\int_{[0,1]^d} c\, w\, v_h \;-\; \sum_{\boldsymbol{i}} v_{\boldsymbol{i}} \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(c\, w)}{\|v_h\|_{L^2}} \leq C h^{p+1} |c\, w|_{\mathcal{H}^{p+1}_{\mathrm{bent}}}$$

Computation of weights ($d = 1$)

We impose that

$$\text{if } x_q \notin \text{int}(\text{supp}(B_i)) \quad \Longrightarrow \quad w_{q,i} = 0$$

Computation of weights ($d = 1$)

We impose that

$$\text{if } x_q \notin \text{int}(\text{supp}(B_i)) \quad \Longrightarrow \quad w_{q,i} = 0$$

For a given $i = 1, \ldots, n$, the exactness requirements read:

$$\sum_q w_{q,i} B_j(x_q) = \int_0^1 B_i(\zeta) B_j(\zeta)\, d\zeta \qquad \forall\ j \text{ s.t. } \text{supp}(B_i) \cap \text{supp}(B_j)$$

Computation of weights ($d = 1$)

We impose that

$$\text{if } x_q \notin \text{int}(\text{supp}(B_i)) \quad \Longrightarrow \quad w_{q,i} = 0$$

For a given $i = 1, \ldots, n$, the exactness requirements read:

$$\sum_q w_{q,i} B_j(x_q) = \int_0^1 B_i(\zeta) B_j(\zeta)\, d\zeta \qquad \forall\ j \text{ s.t. } \text{supp}(B_i) \cap \text{supp}(B_j)$$

- This is a linear system with unknowns $\{w_{q,i}\}_q$ and coefficient matrix $\{B_j(x_q)\}_{j,q}$

Computation of weights ($d = 1$)

We impose that

$$\text{if } x_q \notin \text{int}(\text{supp}(B_i)) \quad \Longrightarrow \quad w_{q,i} = 0$$

For a given $i = 1, \ldots, n$, the exactness requirements read:

$$\sum_q w_{q,i} B_j(x_q) = \int_0^1 B_i(\zeta) B_j(\zeta)\, d\zeta \qquad \forall\, j \text{ s.t. } \text{supp}(B_i) \cap \text{supp}(B_j)$$

- This is a linear system with unknowns $\{w_{q,i}\}_q$ and coefficient matrix $\{B_j(x_q)\}_{j,q}$
- The system is solvable if and only if the points x_q satisfy the Schoenberg-Whitney theorem' assumption, i.e., we can associate to each $B_j(\cdot)$ a unique x_q such that $B_j(x_q) \neq 0$. Then we need:

$$\#\{x_q \mid x_q \in \text{int}(\text{supp}(B_i))\} \geq \#\{j \mid \text{supp}(B_i) \cap \text{supp}(B_j)\}$$

Computation of weights ($d = 1$)

We impose that

$$\text{if } x_q \notin \text{int}(\text{supp}(B_i)) \quad \Longrightarrow \quad w_{q,i} = 0$$

For a given $i = 1, \ldots, n$, the exactness requirements read:

$$\sum_q w_{q,i} B_j(x_q) = \int_0^1 B_i(\zeta) B_j(\zeta)\, d\zeta \qquad \forall\, j \text{ s.t. } \text{supp}(B_i) \cap \text{supp}(B_j)$$

- This is a linear system with unknowns $\{w_{q,i}\}_q$ and coefficient matrix $\{B_j(x_q)\}_{j,q}$
- The system is solvable if and only if the points x_q satisfy the Schoenberg-Whitney theorem' assumption, i.e., we can associate to each $B_j(\cdot)$ a unique x_q such that $B_j(x_q) \neq 0$. Then we need:

$$\#\{x_q \mid x_q \in \text{int}(\text{supp}(B_i))\} \geq \#\{j \mid \text{supp}(B_i) \cap \text{supp}(B_j)\}$$

Choice of quadrature points ($d = 1$)

$$\#\{x_q \mid x_q \in \mathrm{int}(\mathrm{supp}(B_i))\} \geq \#\{j \mid \mathrm{supp}(B_i) \cap \mathrm{supp}(B_j)\}$$

The support of a function B_i far from the boundary spans $p+1$ elements and intersects the support of $2p+1$ functions B_j.

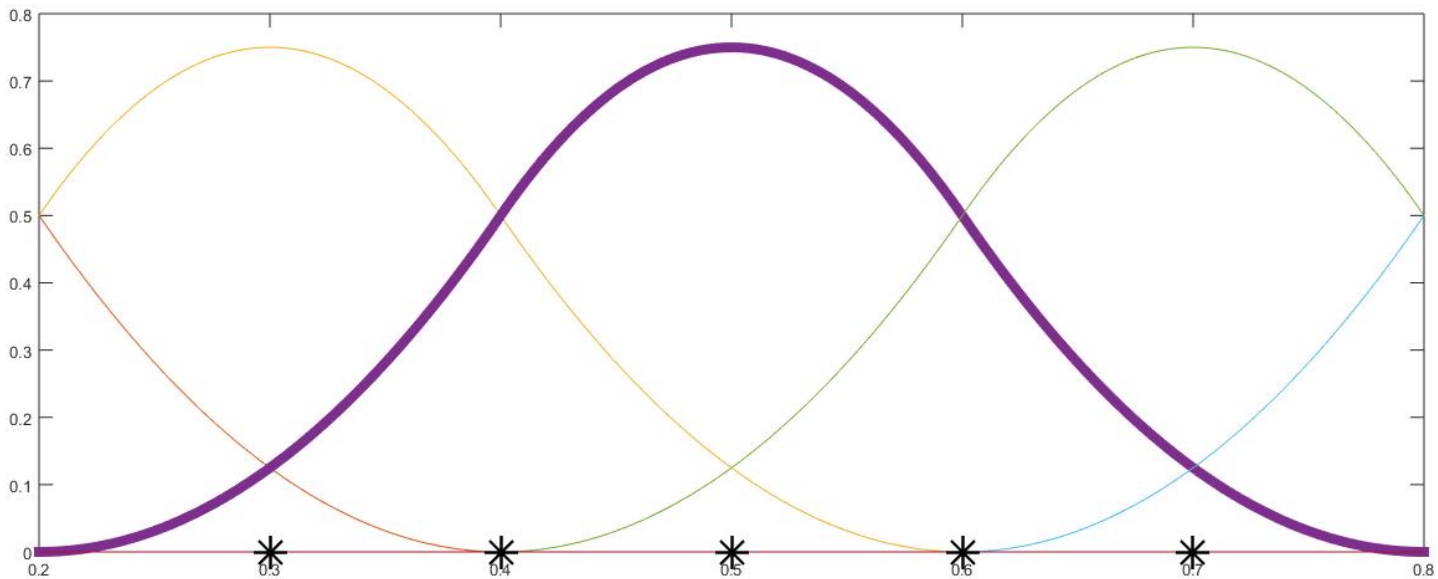
Choice of quadrature points ($d = 1$)

$$\#\{x_q \mid x_q \in \text{int}(\text{supp}(B_i))\} \geq \#\{j \mid \text{supp}(B_i) \cap \text{supp}(B_j)\}$$

The support of a function B_i far from the boundary spans $p+1$ elements and intersects the support of $2p+1$ functions B_j.

We take:

- Internal elements: 2 points (extremes and mid-points)



$p = 2$

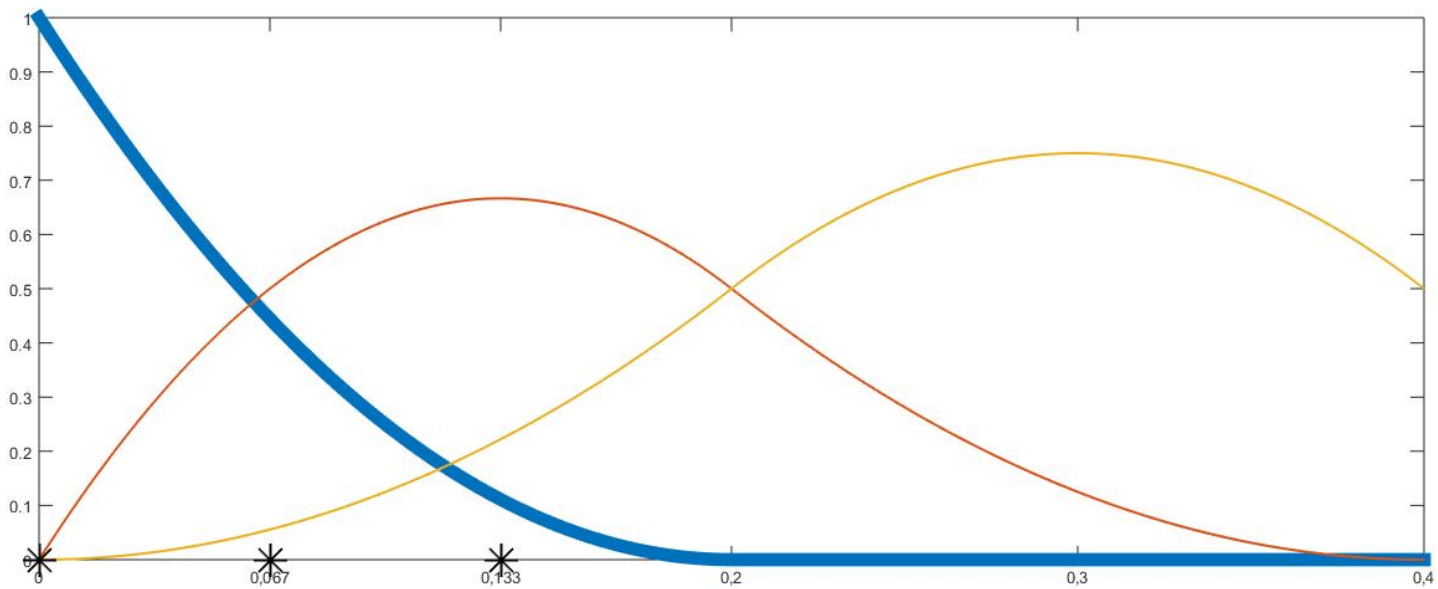
Choice of quadrature points ($d = 1$)

$$\#\{x_q \mid x_q \in \text{int}(\text{supp}(B_i))\} \geq \#\{j \mid \text{supp}(B_i) \cap \text{supp}(B_j)\}$$

The support of a function B_i far from the boundary spans $p+1$ elements and intersects the support of $2p+1$ functions B_j.

We take:

- Internal elements: 2 points (extremes and mid-points)
- Boundary elements: $p+1$ points (equally spaced).



$p = 2$

Choice of quadrature points ($d = 1$)

$$\#\{x_q \mid x_q \in \mathrm{int}(\mathrm{supp}(B_i))\} \geq \#\{j \mid \mathrm{supp}(B_i) \cap \mathrm{supp}(B_j)\}$$

The support of a function B_i far from the boundary spans $p+1$ elements and intersects the support of $2p+1$ functions B_j.

We take:

- Internal elements: 2 points (extremes and mid-points)
- Boundary elements: $p+1$ points (equally spaced).

The total number of quadrature points is $\approx 2n$, indep. of p.

Weights and points ($d > 1$)

Let $\boldsymbol{i} = (i_1, i_2)$, $\boldsymbol{j} = (j_1, j_2)$. Then

$$\begin{aligned}
\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) &= \int_{\Omega} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \\
&= \int_0^1 B_{i_1}(\zeta_1)\, B_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2)\, B_{j_2}(\zeta_2)\, d\zeta_2 \\
&= \left(\sum_{q_1} w_{q_1,i_1,1}\, B_{j_1}(x_{q_1}) \right) \left(\sum_{q_2} w_{q_2,i_2,2}\, B_{j_2}(x_{q_2}) \right)
\end{aligned}$$

Weights and points ($d > 1$)

Let $\boldsymbol{i} = (i_1, i_2)$, $\boldsymbol{j} = (j_1, j_2)$. Then

$$\begin{aligned}
\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) &= \int_{\Omega} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \\
&= \int_0^1 B_{i_1}(\zeta_1)\, B_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2)\, B_{j_2}(\zeta_2)\, d\zeta_2 \\
&= \left(\sum_{q_1} w_{q_1,i_1,1}\, B_{j_1}(x_{q_1}) \right) \left(\sum_{q_2} w_{q_2,i_2,2}\, B_{j_2}(x_{q_2}) \right)
\end{aligned}$$

We define the points $\boldsymbol{x}_{\boldsymbol{q}}$ and the weights $w_{\boldsymbol{q},\boldsymbol{i}}$ by tensor product:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(f(\cdot)) = (\mathfrak{Q}_{i_1,1} \otimes \mathfrak{Q}_{i_2,2})\,(f(\cdot)) = \sum_{q_1,q_2} w_{q_1,i_1,1}\, w_{q_2,i_2,2}\, f(x_{q_1}, x_{q_2})$$

Weights and points ($d > 1$)

Let $\boldsymbol{i} = (i_1, i_2)$, $\boldsymbol{j} = (j_1, j_2)$. Then

$$\begin{aligned}
\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) &= \int_{\Omega} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \\
&= \int_0^1 B_{i_1}(\zeta_1)\, B_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2)\, B_{j_2}(\zeta_2)\, d\zeta_2 \\
&= \left(\sum_{q_1} w_{q_1,i_1,1}\, B_{j_1}(x_{q_1}) \right) \left(\sum_{q_2} w_{q_2,i_2,2}\, B_{j_2}(x_{q_2}) \right)
\end{aligned}$$

We define the points $\boldsymbol{x}_{\boldsymbol{q}}$ and the weights $w_{\boldsymbol{q},\boldsymbol{i}}$ by tensor product:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(f(\cdot)) = (\mathfrak{Q}_{i_1,1} \otimes \mathfrak{Q}_{i_2,2})\,(f(\cdot)) = \sum_{q_1,q_2} w_{q_1,i_1,1}\, w_{q_2,i_2,2}\, f(x_{q_1}, x_{q_2})$$

- computing cost for quadrature formulae for $d > 1$ is negligible.

Weights and points ($d > 1$)

Let $\boldsymbol{i} = (i_1, i_2)$, $\boldsymbol{j} = (j_1, j_2)$. Then

$$\begin{aligned}
\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(B_{\boldsymbol{j}}(\cdot)) &= \int_{\Omega} B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} \\
&= \int_0^1 B_{i_1}(\zeta_1)\, B_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2)\, B_{j_2}(\zeta_2)\, d\zeta_2 \\
&= \left(\sum_{q_1} w_{q_1,i_1,1}\, B_{j_1}(x_{q_1}) \right) \left(\sum_{q_2} w_{q_2,i_2,2}\, B_{j_2}(x_{q_2}) \right)
\end{aligned}$$

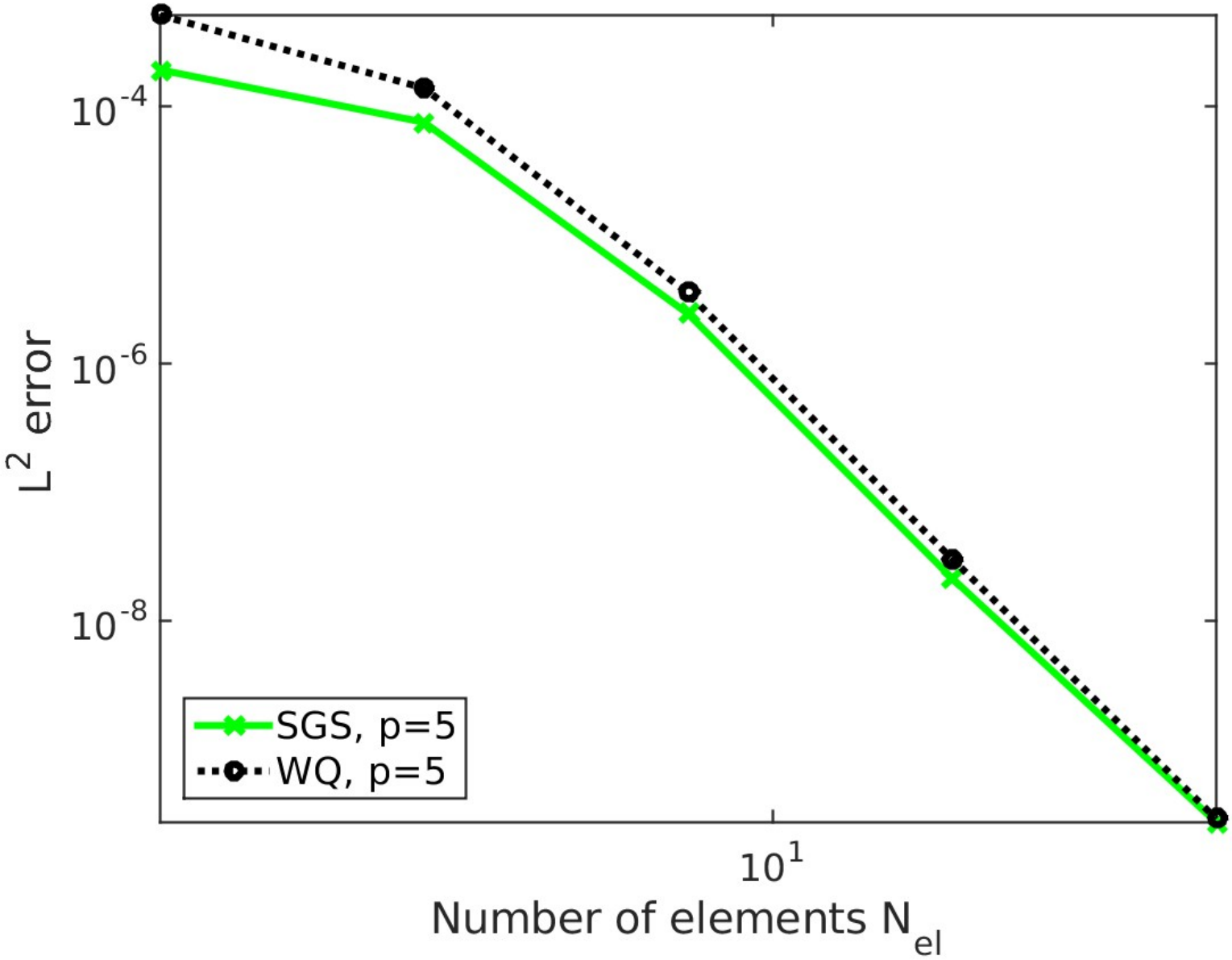
We define the points $\boldsymbol{x}_{\boldsymbol{q}}$ and the weights $w_{\boldsymbol{q},\boldsymbol{i}}$ by tensor product:

$$\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(f(\cdot)) = (\mathfrak{Q}_{i_1,1} \otimes \mathfrak{Q}_{i_2,2})\,(f(\cdot)) = \sum_{q_1,q_2} w_{q_1,i_1,1}\, w_{q_2,i_2,2}\, f(x_{q_1}, x_{q_2})$$

- computing cost for quadrature formulae for $d > 1$ is negligible.
- The total number of quadrature points is $\approx 2^d N_{\mathsf{DOF}}$.

Convergence plot

Thick quarter of ring domain.



Assembly algorithm for the mass matrix

Using sum-factorization approach and matrix-tensor products.

Assembly algorithm for the mass matrix

Using sum-factorization approach and matrix-tensor products.

for $\boldsymbol{i} = 1, \ldots, N_{\mathsf{DOF}}$

$\quad \mathcal{C}_{\boldsymbol{i}}^{(0)} := \{c(\boldsymbol{x_q})\}_{\boldsymbol{q}} \qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **for** $l = 1, \ldots, d$

$\quad\quad \mathrm{B}^{(i_l)} := \{\hat{B}_j(x_q)\}_{j_l, q_l} \qquad \in \mathbb{R}^{(2p+1)\times(2p+1)}$

$\quad\quad \mathcal{C}_{\boldsymbol{i}}^{(l)} := \mathcal{C}_{\boldsymbol{i}}^{(l-1)} \times_l \left(\mathrm{B}^{(i_l)} \times \mathsf{diag}\{w_{q,i}\}_q\right) \qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **end**

$\quad m_{\boldsymbol{i},\ldots} = \mathcal{C}_{\boldsymbol{i}}^{(d)} \qquad \in \mathbb{R}^{(2p+1)^d}$

end

Assembly algorithm for the mass matrix

Using sum-factorization approach and matrix-tensor products.

for $\boldsymbol{i} = 1, \ldots, N_{\mathsf{DOF}}$

$\quad \mathcal{C}_{\boldsymbol{i}}^{(0)} := \{c(\boldsymbol{x_q})\}_{\boldsymbol{q}}$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **for** $l = 1, \ldots, d$

$\quad\quad \mathrm{B}^{(i_l)} := \{\hat{B}_j(x_q)\}_{j_l, q_l}$ $\qquad \in \mathbb{R}^{(2p+1)\times(2p+1)}$

$\quad\quad \mathcal{C}_{\boldsymbol{i}}^{(l)} := \mathcal{C}_{\boldsymbol{i}}^{(l-1)} \times_l \left(\mathrm{B}^{(i_l)} \times \mathsf{diag}\{w_{q,i}\}_q\right)$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **end**

$\quad m_{\boldsymbol{i},\ldots} = \mathcal{C}_{\boldsymbol{i}}^{(d)}$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

end

- The inner loop is repeated N_{DOF} and perform $O(p)$ (BLAS level 3) operations to compute $O(p^d)$ results: total $O(N_{\mathsf{DOF}} p^{d+1})$ FLOPS

Assembly algorithm for the mass matrix

Using sum-factorization approach and matrix-tensor products.

for $\boldsymbol{i} = 1, \ldots, N_{\mathsf{DOF}}$

$\quad \mathcal{C}_{\boldsymbol{i}}^{(0)} := \{c(\boldsymbol{x_q})\}_{\boldsymbol{q}}$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **for** $l = 1, \ldots, d$

$\quad\quad \mathrm{B}^{(i_l)} := \{\hat{B}_j(x_q)\}_{j_l, q_l}$ $\qquad \in \mathbb{R}^{(2p+1)\times(2p+1)}$

$\quad\quad \mathcal{C}_{\boldsymbol{i}}^{(l)} := \mathcal{C}_{\boldsymbol{i}}^{(l-1)} \times_l \left(\mathrm{B}^{(i_l)} \times \mathsf{diag}\{w_{q,i}\}_q\right)$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **end**

$\quad m_{\boldsymbol{i},\ldots} = \mathcal{C}_{\boldsymbol{i}}^{(d)}$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

end

- The inner loop is repeated N_{DOF} and perform $O(p)$ (BLAS level 3) operations to compute $O(p^d)$ results: total $O(N_{\mathsf{DOF}} p^{d+1})$ FLOPS
- The sparse matrix memory operations (allocation and write) are $O(N_{\mathsf{DOF}} p^d)$ and dominate in practice.

Assembly algorithm for the mass matrix

Using sum-factorization approach and matrix-tensor products.

for $\boldsymbol{i} = 1, \ldots, N_{\mathsf{DOF}}$

$\quad \mathcal{C}_{\boldsymbol{i}}^{(0)} := \{c(\boldsymbol{x_q})\}_{\boldsymbol{q}}$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

$\quad$ **for** $l = 1, \ldots, d$

$\quad\quad \mathrm{B}^{(i_l)} := \{\hat{B}_j(x_q)\}_{j_l, q_l}$ $\qquad \in \mathbb{R}^{(2p+1)\times(2p+1)}$

$\quad\quad \mathcal{C}_{\boldsymbol{i}}^{(l)} := \mathcal{C}_{\boldsymbol{i}}^{(l-1)} \times_l \left(\mathrm{B}^{(i_l)} \times \mathsf{diag}\{w_{q,i}\}_q\right)$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

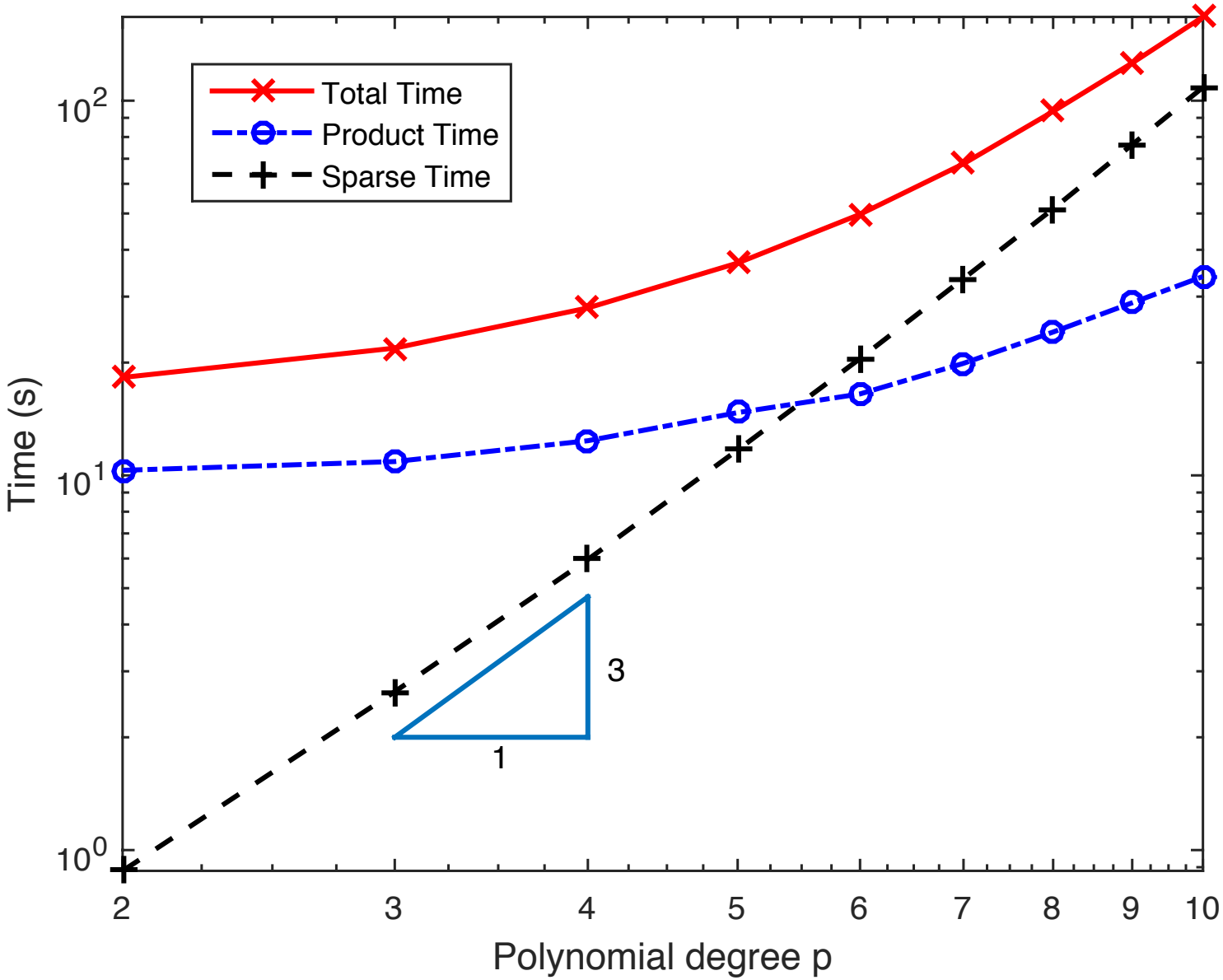
$\quad$ **end**

$\quad m_{\boldsymbol{i},\ldots} = \mathcal{C}_{\boldsymbol{i}}^{(d)}$ $\qquad \in \mathbb{R}^{(2p+1)^d}$

end

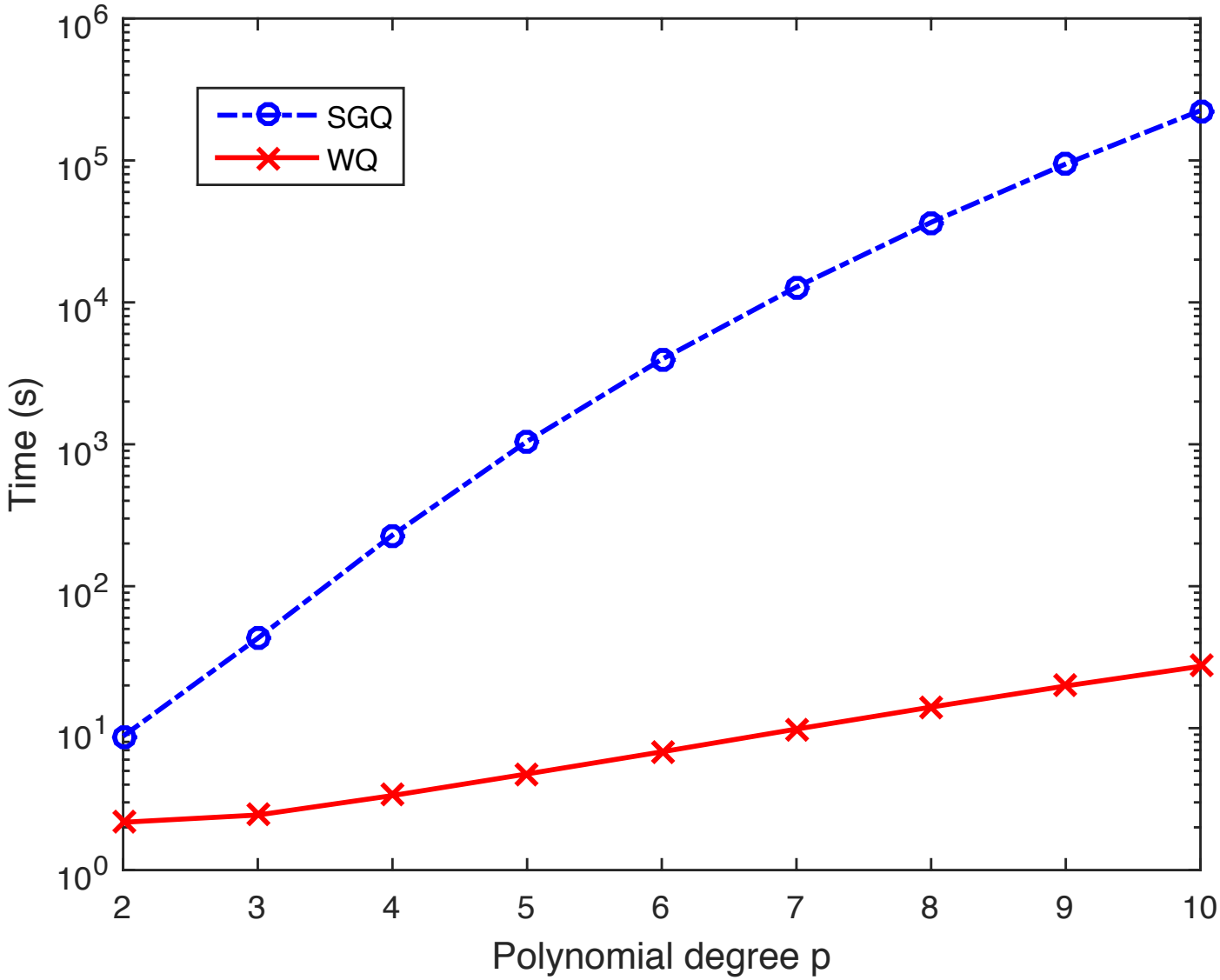
- The inner loop is repeated N_{DOF} and perform $O(p)$ (BLAS level 3) operations to compute $O(p^d)$ results: total $O(N_{\mathsf{DOF}} p^{d+1})$ FLOPS
- The sparse matrix memory operations (allocation and write) are $O(N_{\mathsf{DOF}} p^d)$ and dominate in practice.

Assembly time ($\Omega = [0,1]^3$, $N_{\mathsf{EL}} = 40^3$)



Mass matrix assembly time: WQ vs. Geopdes 3.0

$$\Omega = [0,1]^3,\ N_{\mathsf{EL}} = 20^3.$$



SGQ: $O(N_{\mathsf{DOF}}\, p^{3d})$ FLOPS WQ: $O(N_{\mathsf{DOF}}\, p^{d+1})$ FLOPS

Weighted Quadrature for the stiffness matrix

$$
\begin{aligned}
k_{\boldsymbol{i},\boldsymbol{j}} &= \int_{[0,1]^d} \nabla B_{\boldsymbol{i}}(\boldsymbol{\zeta})^T \mathbf{c}(\boldsymbol{\zeta}) \nabla B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\
&= \sum_{\alpha,\beta=1}^{d} \int_{[0,1]^d} c_{\alpha,\beta}(\boldsymbol{\zeta}) \, \partial_\alpha B_{\boldsymbol{i}}(\boldsymbol{\zeta}) \, \partial_\beta B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\
&\approx \sum_{\alpha,\beta=1}^{d} \mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}} \left(c_{\alpha,\beta}(\cdot) \partial_\beta B_{\boldsymbol{j}}(\cdot) \right);
\end{aligned}
$$

Weighted Quadrature for the stiffness matrix

$$
\begin{aligned}
k_{\boldsymbol{i},\boldsymbol{j}} &= \int_{[0,1]^d} \nabla B_{\boldsymbol{i}}(\boldsymbol{\zeta})^T \mathbf{c}(\boldsymbol{\zeta}) \, \nabla B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\
&= \sum_{\alpha,\beta=1}^{d} \int_{[0,1]^d} c_{\alpha,\beta}(\boldsymbol{\zeta}) \, \partial_\alpha B_{\boldsymbol{i}}(\boldsymbol{\zeta}) \, \partial_\beta B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\
&\approx \sum_{\alpha,\beta=1}^{d} \mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}} \left(c_{\alpha,\beta}(\cdot) \partial_\beta B_{\boldsymbol{j}}(\cdot) \right);
\end{aligned}
$$

$$
\int_{[0,1]^d} c_{\alpha,\beta}(\boldsymbol{\zeta}) \, \partial_\beta B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, \left(\partial_\alpha B_{\boldsymbol{i}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \right) \approx \mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}} \left(c_{\alpha\beta}(\cdot) \partial_\beta B_{\boldsymbol{j}}(\cdot) \right)
$$

Weighted Quadrature for the stiffness matrix

$$
\begin{aligned}
k_{\boldsymbol{i},\boldsymbol{j}} &= \int_{[0,1]^d} \nabla B_{\boldsymbol{i}}(\boldsymbol{\zeta})^T \mathbf{c}(\boldsymbol{\zeta}) \nabla B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\
&= \sum_{\alpha,\beta=1}^{d} \int_{[0,1]^d} c_{\alpha,\beta}(\boldsymbol{\zeta}) \, \partial_\alpha B_{\boldsymbol{i}}(\boldsymbol{\zeta}) \, \partial_\beta B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\
&\approx \sum_{\alpha,\beta=1}^{d} \mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}} \left(c_{\alpha,\beta}(\cdot) \partial_\beta B_{\boldsymbol{j}}(\cdot) \right);
\end{aligned}
$$

$$
\int_{[0,1]^d} c_{\alpha,\beta}(\boldsymbol{\zeta}) \, \partial_\beta B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, \left(\partial_\alpha B_{\boldsymbol{i}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \right) \approx \mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}} \left(c_{\alpha\beta}(\cdot) \partial_\beta B_{\boldsymbol{j}}(\cdot) \right)
$$

Exactness conditions

$$
\mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}} (\partial_\beta B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^d} \partial_\alpha B_{\boldsymbol{i}}(\boldsymbol{\zeta}) \, \partial_\beta B_{\boldsymbol{j}}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \qquad \forall \, \boldsymbol{j}
$$

Weighted Quadrature for the stiffness matrix

- For $d = 2$, two of the exactness conditions read:

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,1,\boldsymbol{i}}(\partial_1 B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^2} \partial_1 B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, \partial_1 B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta}$$

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,2,\boldsymbol{i}}(\partial_2 B_{\boldsymbol{j}}(\cdot)) = \int_{[0,1]^2} \partial_1 B_{\boldsymbol{i}}(\boldsymbol{\zeta})\, \partial_2 B_{\boldsymbol{j}}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta}$$

Weighted Quadrature for the stiffness matrix

- For $d = 2$, two of the exactness conditions read:

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,1,\boldsymbol{i}}(\partial_1 B_{\boldsymbol{j}}(\cdot)) = \int_0^1 B'_{i_1}(\zeta_1) B'_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2) B_{j_2}(\zeta_2)\, d\zeta_2$$

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,2,\boldsymbol{i}}(\partial_2 B_{\boldsymbol{j}}(\cdot)) = \int_0^1 B'_{i_1}(\zeta_1) B_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2) B'_{j_2}(\zeta_2)\, d\zeta_2$$

Weighted Quadrature for the stiffness matrix

- For $d = 2$, two of the exactness conditions read:

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,1,\boldsymbol{i}}(\partial_1 B_{\boldsymbol{j}}(\cdot)) = \int_0^1 B'_{i_1}(\zeta_1) B'_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2) B_{j_2}(\zeta_2)\, d\zeta_2$$

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,2,\boldsymbol{i}}(\partial_2 B_{\boldsymbol{j}}(\cdot)) = \int_0^1 B'_{i_1}(\zeta_1) B_{j_1}(\zeta_1)\, d\zeta_1 \int_0^1 B_{i_2}(\zeta_2) B'_{j_2}(\zeta_2)\, d\zeta_2$$

- We take 4 univariate WQ rules (per direction).

$$\mathfrak{Q}_i^{(0,0)}\left(B_j(\cdot)\right) = \int_0^1 B_i(\zeta) B_j(\zeta) d\zeta \qquad \mathfrak{Q}_i^{(1,0)}\left(B_j(\cdot)\right) = \int_0^1 B'_i(\zeta) B_j(\zeta) d\zeta$$

$$\mathfrak{Q}_i^{(0,1)}\left(B'_j(\cdot)\right) = \int_0^1 B_i(\zeta) B'_j(\zeta) d\zeta \qquad \mathfrak{Q}_i^{(1,1)}\left(B'_j(\cdot)\right) = \int_0^1 B'_i(\zeta) B'_j(\zeta) d\zeta$$

Weighted Quadrature for the stiffness matrix

- For $d = 2$, two of the exactness conditions read:

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,1,\boldsymbol{i}}(\partial_1 B_{\boldsymbol{j}}(\cdot)) = \int_0^1 B'_{i_1}(\zeta_1)B'_{j_1}(\zeta_1)\,d\zeta_1 \int_0^1 B_{i_2}(\zeta_2)B_{j_2}(\zeta_2)\,d\zeta_2$$

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,2,\boldsymbol{i}}(\partial_2 B_{\boldsymbol{j}}(\cdot)) = \int_0^1 B'_{i_1}(\zeta_1)B_{j_1}(\zeta_1)\,d\zeta_1 \int_0^1 B_{i_2}(\zeta_2)B'_{j_2}(\zeta_2)\,d\zeta_2$$

- We take 4 univariate WQ rules (per direction).

$$\mathfrak{Q}_i^{(0,0)}\left(B_j(\cdot)\right) = \int_0^1 B_i(\zeta)B_j(\zeta)d\zeta \qquad \mathfrak{Q}_i^{(1,0)}\left(B_j(\cdot)\right) = \int_0^1 B'_i(\zeta)B_j(\zeta)d\zeta$$

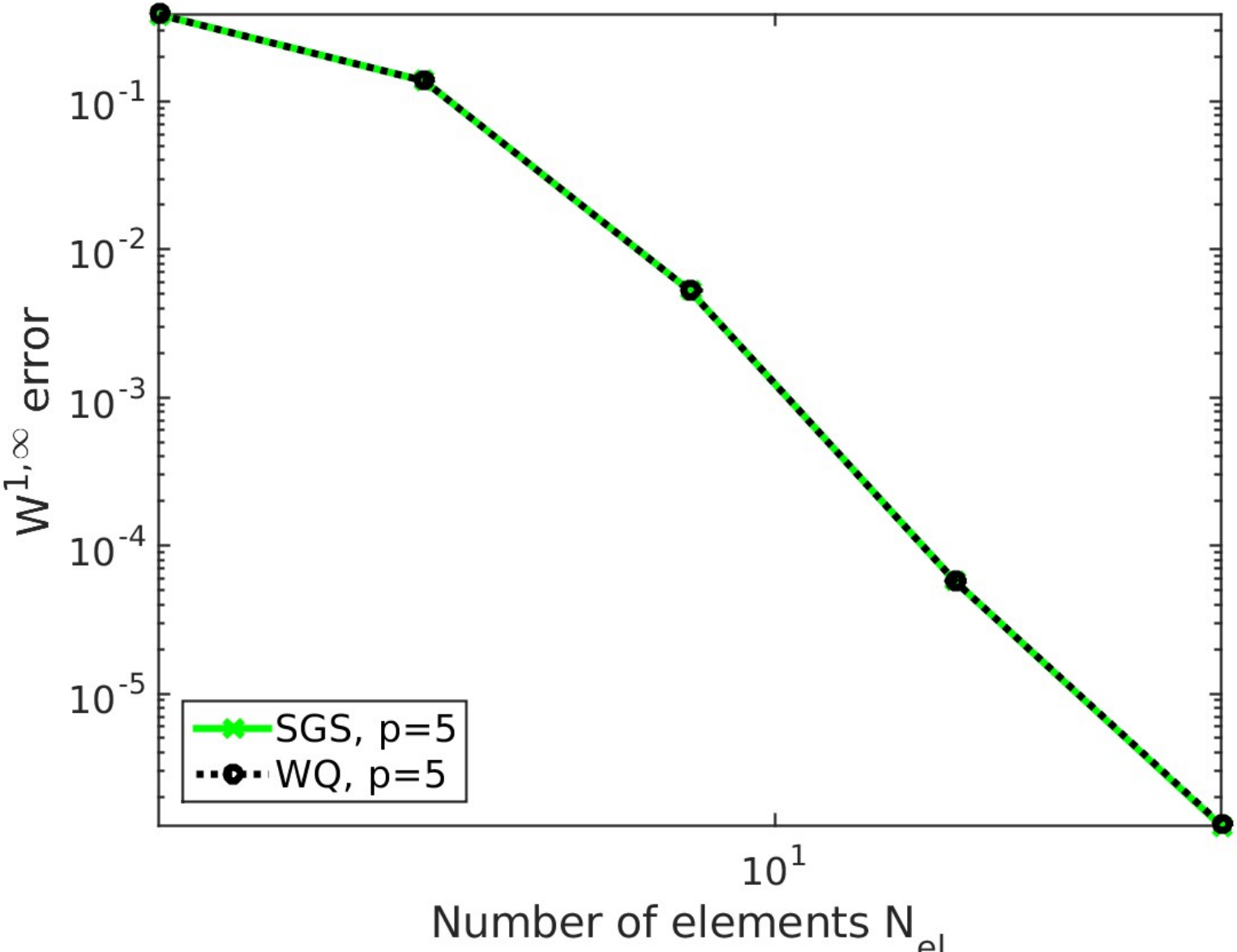
$$\mathfrak{Q}_i^{(0,1)}\left(B'_j(\cdot)\right) = \int_0^1 B_i(\zeta)B'_j(\zeta)d\zeta \qquad \mathfrak{Q}_i^{(1,1)}\left(B'_j(\cdot)\right) = \int_0^1 B'_i(\zeta)B'_j(\zeta)d\zeta$$

- These 4 univariate rules are the building blocks we use to define the rules $\mathfrak{Q}^{\mathrm{WQ}}_{\alpha,\beta,\boldsymbol{i}}$ (also for $d > 2$). For example:

$$\mathfrak{Q}^{\mathrm{WQ}}_{1,1,\boldsymbol{i}} = \mathfrak{Q}^{(1,1)}_{i_1} \otimes \mathfrak{Q}^{(0,0)}_{i_2} \qquad \mathfrak{Q}^{\mathrm{WQ}}_{1,2,\boldsymbol{i}} = \mathfrak{Q}^{(1,0)}_{i_1} \otimes \mathfrak{Q}^{(0,1)}_{i_2}$$

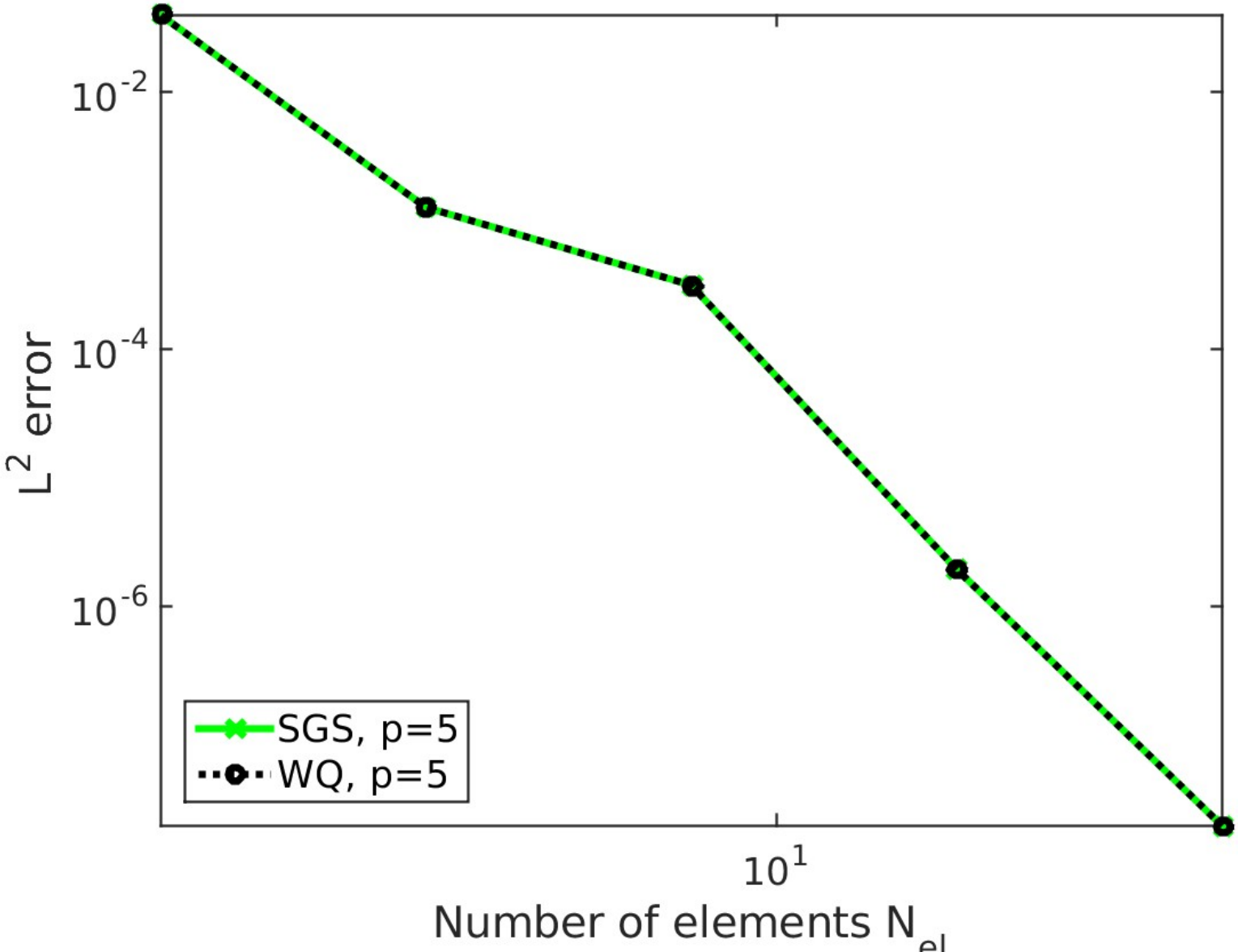
Convergence plot 1D: $W^{1,\infty}$ norm

Thick quarter of ring domain.



Convergence plot 1D: L^2 norm

Thick quarter of ring domain.



A matrix-free implementation

We only want to compute matrix-vector products with the mass matrix $\widetilde{\mathcal{M}}$:

$$u \longrightarrow \widetilde{\mathcal{M}} \cdot u$$

A matrix-free implementation

We only want to compute matrix-vector products with the mass matrix $\widetilde{\mathcal{M}}$:

$$u \longrightarrow \widetilde{\mathcal{M}} \cdot u$$

Let $u = (u_1, \ldots, u_{N_{DOF}}) \in \mathbb{R}^{N_{\mathsf{DOF}}}$. Since $\mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}(\cdot)$ is a linear functional, we have

$$\left(\widetilde{\mathcal{M}}u\right)_{\boldsymbol{i}} = \mathfrak{Q}_{\boldsymbol{i}}^{\mathrm{WQ}}\left(c(\cdot)u_h(\cdot)\right) = \sum_{\boldsymbol{q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}}),$$

where

$$u_h(\xi) = \sum_{\boldsymbol{j}} u_{\boldsymbol{j}} \hat{B}_{\boldsymbol{j}}(\xi)$$

A matrix-free implementation

$$\left(\widetilde{\mathcal{M}}u\right)_{\boldsymbol{i}} = \sum_{\boldsymbol{q}\in\mathcal{Q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}}), \qquad |\mathcal{Q}| = O(N_{\mathsf{DOF}})$$

The Setup

- Weights and basis spline values for univariate spaces.
- Coefficient values $c(x_{\boldsymbol{q}})$.

A matrix-free implementation

$$\left(\widetilde{\mathcal{M}}u\right)_{\boldsymbol{i}} = \sum_{\boldsymbol{q}\in\mathcal{Q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}}), \qquad |\mathcal{Q}| = O(N_{\mathsf{DOF}})$$

The Setup

- Weights and basis spline values for univariate spaces.
  Negligible memory and cost.
- Coefficient values $c(x_{\boldsymbol{q}})$. $O(N_{\mathsf{DOF}})$ memory.

A matrix-free implementation

$$\left(\widetilde{\mathcal{M}}u\right)_{\boldsymbol{i}} = \sum_{\boldsymbol{q}\in\mathcal{Q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}}), \qquad |\mathcal{Q}| = O(N_{\mathsf{DOF}})$$

The Setup

- Weights and basis spline values for univariate spaces.
  Negligible memory and cost.
- Coefficient values $c(x_{\boldsymbol{q}})$. $O(N_{\mathsf{DOF}})$ memory.

The Product

1. Given $u \in \mathbb{R}^{N_{\mathsf{DOF}}}$, compute $u_h(x_{\boldsymbol{q}})$ for every $\boldsymbol{q}$.
2. Compute $c(x_{\boldsymbol{q}}) \cdot u_h(x_{\boldsymbol{q}})$ for every $\boldsymbol{q}$.
3. Compute $\sum_{\boldsymbol{q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}})$ for every $\boldsymbol{i}$.

A matrix-free implementation

$$\left(\widetilde{\mathcal{M}}u\right)_{\boldsymbol{i}} = \sum_{\boldsymbol{q}\in\mathcal{Q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}}), \qquad |\mathcal{Q}| = O(N_{\mathsf{DOF}})$$

The Setup

- Weights and basis spline values for univariate spaces.
  Negligible memory and cost.
- Coefficient values $c(x_{\boldsymbol{q}})$. $O(N_{\mathsf{DOF}})$ memory.

The Product

1. Given $u \in \mathbb{R}^{N_{\mathsf{DOF}}}$, compute $u_h(x_{\boldsymbol{q}})$ for every $\boldsymbol{q}$.
   $O(N_{\mathsf{DOF}}\, p)$ FLOPS exploiting the tensor structure.
2. Compute $c(x_{\boldsymbol{q}}) \cdot u_h(x_{\boldsymbol{q}})$ for every $\boldsymbol{q}$.
   $O(N_{\mathsf{DOF}})$ FLOPS.
3. Compute $\sum_{\boldsymbol{q}} w_{\boldsymbol{iq}}\, c(x_{\boldsymbol{q}})\, u_h(x_{\boldsymbol{q}})$ for every $\boldsymbol{i}$.
   $O(N_{\mathsf{DOF}}\, p)$ FLOPS exploiting the tensor structure.

A matrix-free implementation

	Standard WQ	Matrix-free WQ
Allocated memory	$O(N_{\mathsf{DOF}}\, p^d)$	$O(N_{\mathsf{DOF}})$
Setup cost	$O(N_{\mathsf{DOF}}\, p^{d+1}) + \mathrm{EV}$	EV
Product cost	$O(N_{\mathsf{DOF}}\, p^d)$	$O(N_{\mathsf{DOF}}\, p)$

EV = Cost of coefficient evaluation on $O(N_{\mathsf{DOF}})$ points

A matrix-free implementation

	Standard WQ	Matrix-free WQ
Allocated memory	$O(N_{\mathsf{DOF}}\, p^d)$	$O(N_{\mathsf{DOF}})$
Setup cost	$O(N_{\mathsf{DOF}}\, p^{d+1}) + \mathrm{EV}$	EV
Product cost	$O(N_{\mathsf{DOF}}\, p^d)$	$O(N_{\mathsf{DOF}}\, p)$

EV = Cost of coefficient evaluation on $O(N_{\mathsf{DOF}})$ points

- Analogous for the matrix-free stiffness matrix $\mathcal{A}$.

A matrix-free implementation

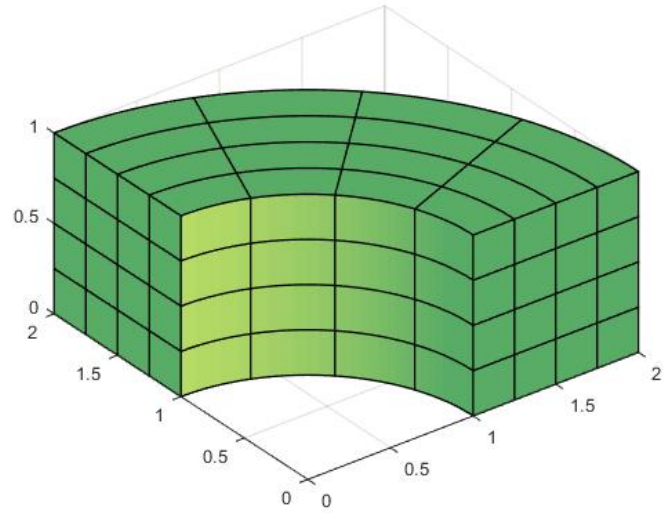
	Standard WQ	Matrix-free WQ
Allocated memory	$O(N_{\mathsf{DOF}}\, p^d)$	$O(N_{\mathsf{DOF}})$
Setup cost	$O(N_{\mathsf{DOF}}\, p^{d+1}) + \text{EV}$	EV
Product cost	$O(N_{\mathsf{DOF}}\, p^d)$	$O(N_{\mathsf{DOF}}\, p)$

EV = Cost of coefficient evaluation on $O(N_{\mathsf{DOF}})$ points

- Analogous for the matrix-free stiffness matrix $\mathcal{A}$.
- Can be coupled with a $p-$robust preconditioner to solve iteratively the system $\mathcal{A}x = b$ ☞ yesterday Mattia Tani talk.

Standard WQ (S-WQ) vs. Matrix-free WQ (MF-WQ)

$N_{\mathsf{EL}} = 64^3$



	Setup Time		Product Time	
	MF-WQ	S-WQ	MF-WQ	S-WQ
$p = 2$	1.28	72.79	0.04	0.05
$p = 3$	1.30	89.06	0.05	0.14
$p = 4$	1.36	112.52	0.05	0.34
$p = 5$	1.58	174.19	0.06	0.58

Conclusions

It is possible to form and solve high-order isogeometric Galerkin k-method is an efficient way, but this is beyond standard FE routines.

- with row-loop and WQ: calculating the matrix entries is faster than saving it in a sparse matrix (MATLAB `sparse`)
- with Fast Diagonalization direct solver as preconditioner: CPU time for the preconditioner setup+application is less than CPU time for the residual calculation in CG

Current work in Pavia is on:

- matrix-free approach
- dealing with "bad" geometries ($\sup_\Omega \kappa(J_F) \gg 1$).
- develop all this for NS

References

Adam, C., T. Hughes, S. Bouabdallah, M. Zarroug, and H. Maitournam (2015). Selective and reduced numerical integrations for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 284*, 732–761.

Akkerman, I., Y. Bazilevs, V. Calo, T. J. Hughes, and S. Hulshoff (2008). The role of continuity in residual-based variational multiscale modeling of turbulence. *Computational Mechanics 41*(3), 371–378.

Bartels, R. H. and G. W. Stewart (1972). Solution of the matrix equation AX+ XB= C. *Communications of the ACM 15*(9), 820–826.

Bazilevs, Y., V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, and G. Scovazzi (2007). Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Comput. Methods Appl. Mech. Engrg. 197*(1-4), 173–201.

Beirão da Veiga, L., D. Cho, L. F. Pavarino, and S. Scacchi (2013). Bddc preconditioners for isogeometric analysis. *Mathematical Models and Methods in Applied Sciences 23*(06), 1099–1142.

Bercovier, M. and I. Soloveichik (2015). Overlapping non matching meshes domain decomposition method in isogeometric analysis. *arXiv preprint arXiv:1502.03756*.

References

Bremer, J., Z. Gimbutas, and V. Rokhlin (2010). A nonlinear optimization procedure for generalized gaussian quadratures. *SIAM Journal on Scientific Computing 32*(4), 1761–1788.

Buffa, A., C. de Falco, and G. Sangalli (2011). Isogeometric Analysis: stable elements for the 2D Stokes equation. *Internat. J. Numer. Methods Fluids 65*(11-12), 1407–1422.

Buffa, A., H. Harbrecht, A. Kunoth, and G. Sangalli (2013). Bpx-preconditioning for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 265*, 63–70.

Buffa, A., G. Sangalli, and C. Schwab (2014). Exponential convergence of the hp version of isogeometric analysis in 1d. In M. Azaiez, H. El Fekih, and J. S. Hesthaven (Eds.), *Spectral and High Order Methods for Partial Differential Equations - ICOSAHOM 2012*, Volume 95 of *Lecture Notes in Computational Science and Engineering*, pp. 191–203. Springer.

Cheng, H., V. Rokhlin, and N. Yarvin (1999). Nonlinear optimization, quadrature, and interpolation. *SIAM Journal on Optimization 9*(4), 901–923.

Collier, N., D. Pardo, L. Dalcin, M. Paszynski, and V. M. Calo (2012). The cost of continuity: a study of the performance of isogeometric finite elements using direct solvers. *Comput. Methods Appl. Mech. Engrg. 213/216*, 353–361.

References

Cottrell, J. A., A. Reali, Y. Bazilevs, and T. J. R. Hughes (2006). Isogeometric analysis of structural vibrations. *Comput. Methods Appl. Mech. Engrg. 195*(41-43), 5257–5296.

da Veiga, L. B., D. Cho, L. F. Pavarino, and S. Scacchi (2012). Overlapping schwarz methods for isogeometric analysis. *SIAM Journal on Numerical Analysis 50*(3), 1394–1416.

Donatelli, M., C. Garoni, C. Manni, S. Serra-Capizzano, and H. Speleers (2015). Robust and optimal multi-iterative techniques for iga galerkin linear systems. *Computer Methods in Applied Mechanics and Engineering 284*, 230–264.

Elman, H. C., D. J. Silvester, and A. J. Wathen (2014). *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press (UK).

Evans, J. A. and T. J. R. Hughes (2013). Isogeometric divergence-conforming B-splines for the Unsteady Navier-Stokes Equations. *J. Comput. Phys. 241*, 141 – 167.

Gahalaut, K. P. S., J. K. Kraus, and S. K. Tomar (2013). Multigrid methods for isogeometric discretization. *Computer methods in applied mechanics and engineering 253*, 413–425.

Hillman, M., J. Chen, and Y. Bazilevs (2015). Variationally consistent domain integration for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 284*, 521–540.

References

Hofreither, C., S. Takacs, and W. Zulehner (2015). A robust multigrid method for isogeometric analysis using boundary correction. Technical Report 33, NFN.

Hughes, T., A. Reali, and G. Sangalli (2010). Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 199*(5), 301–313.

Hughes, T. J. R., A. Reali, and G. Sangalli (2008). Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: comparison of p-method finite elements with k-method NURBS. *Comput. Methods Appl. Mech. Engrg. 197*(49-50), 4104–4124.

Kleiss, S. K., C. Pechstein, B. Jüttler, and S. Tomar (2012). IETI-Isogeometric Tearing and Interconnecting. *Comput. Methods Appl. Mech. Engrg. 247–248*, 201 – 215.

Ma, J., V. Rokhlin, and S. Wandzura (1996). Generalized gaussian quadrature rules for systems of arbitrary functions. *SIAM Journal on Numerical Analysis 33*(3), 971–996.

Mantzaflaris, A., B. Jüttler, B. N. Khoromskij, and U. Langer (2017). Low rank tensor methods in galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 316*, 1062–1085.

Schillinger, D., J. Evans, A. Reali, M. Scott, and T. Hughes (2013). Isogeometric collocation: cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Comput. Methods Appl. Mech. Engrg. 267*, 170 – 232.

References

Schillinger, D., S. Hossain, and T. Hughes (2014). Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 277*, 1–45.

Simoncini, V. (2013). Computational methods for linear matrix equations. *to appear on SIAM Review*.