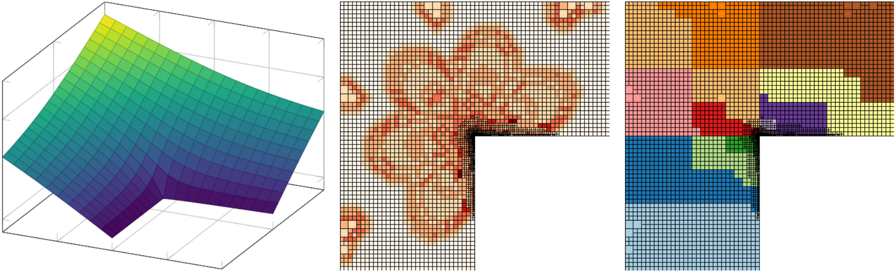


ALGORITHMS FOR MASSIVELY PARALLEL GENERIC HP-ADAPTIVE FEM

June 05, 2020 | Marc Fehling | m.fehling@fz-juelich.de |

BERGISCHE UNIVERSITÄT WUPPERTAL

JÜLICH Forschungszentrum

Table of contents

Fire safety science

- Civilian safety in metro stations
  - Smoke spread in case of fire
  - Egress routes for pedestrians

- Individual examinations necessary on complex geometries
- Experiments expensive $\longrightarrow$ Alternative: Computer simulations

Figure: Experiments in metro

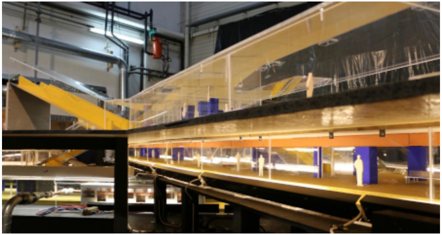


Figure: Physical model (scale 1:15)

Fire simulation

- Lots of software tools available: FDS, FireFOAM, Ansys Fluent, ...
- Flexible: Scenarios may be varied easily

- **But:** Large and complex geometries yield lots of workload
- Simulations require a lot of time

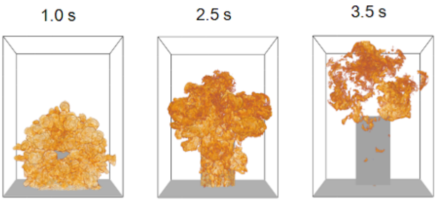


Figure: Deflagration of Heptane

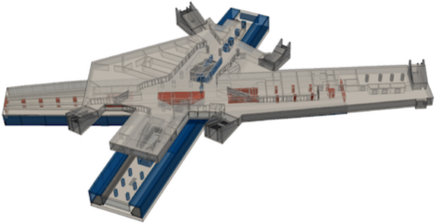


Figure: CAD model

Computational fluid dynamics

- Smoke spread modeled with incompressible Navier–Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho_0 \left[\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u}\right] &= -\nabla p + \nabla \cdot (2\, \mu\, \boldsymbol{\epsilon}) + \boldsymbol{f} \\ \rho_0 c_p \left[\partial_t T + (\boldsymbol{u} \cdot \nabla)\, T\right] &= 2\, \mu\, \boldsymbol{\epsilon} : \nabla \boldsymbol{u} + \nabla \cdot (\kappa\, \nabla T) + q - \boldsymbol{u} \cdot \boldsymbol{f} \end{aligned}$$

- Solution via numerical methods $\longrightarrow$ Computational fluid dynamics
- High resolution necessary for ...
    - Large gradients in temperature and velocity
    - Turbulence
    - Flow separation

Motivation for PhD project

- Fires stay localized in general, not only during ignition phase
- Unnecessarily fine grids bind resources that could be used near the fire
- Demand for effective use of computing power

Goal

- Balance accuracy and workload by adapting resolution
- Accelerate simulations by exploiting hardware

Numerical methods

Adaptation

Parallelization

Example: Adaptive mesh refinement

- Demonstration of adaptive mesh refinement (*h*-adaptive methods) via moving vortex test case as a shape-preserving potential stream.

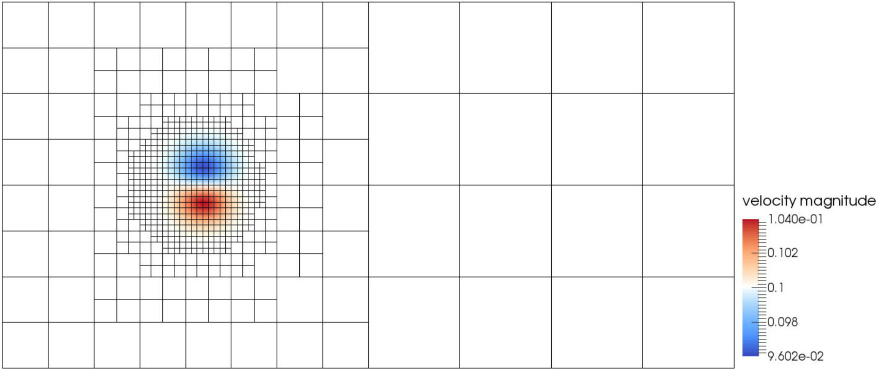


Figure: Video of velocity magnitude of moving vortex, overlaid with current mesh

Numerical methods

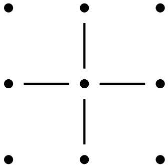
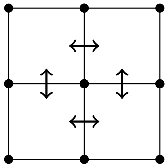
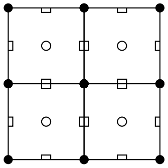
Finite differences	Finite volumes	Finite elements
		
Difference quotients as differential operators	Balance fluxes on faces between volumes Conservation laws	Limit function space to piecewise polynomials
h-adaptive methods	*h*-adaptive methods	*hp*-adaptive methods

Table of contents

Finite element method

- Shape functions form nodal basis

$$\varphi_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}$$

- Q_p elements from Lagrange interpolation with degree p

- Finite element approximation is linear combination of shape functions

$$u_{hp}(x) = \sum_i u_i \, \varphi_i(x)$$

- Coefficients u_i are degrees of freedom

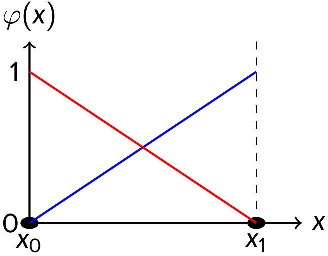


Figure: Q_1 element

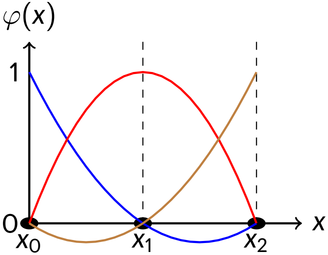


Figure: Q_2 element

Adaptive methods

- Focus computational resources on areas of interest
- Align simulation resolution with complexity of current solution

- Finite Element Method (FEM) provides two different possibilities:

***h*-adaptation:**	dynamic cell sizes	good for irregular solutions
***p*-adaptation:**	dynamic function spaces	good for smooth solutions

- Combination of both possible

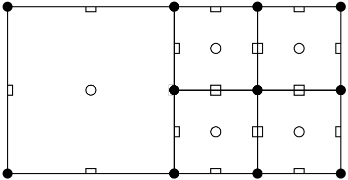


Figure: *h*-adaptivity

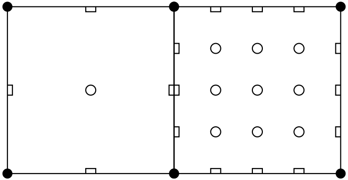


Figure: *p*-adaptivity

Adaptation criteria

- **Which** cells to adapt?
- **How** to adapt? h/p?

Adaptation criteria

- **Which** cells to adapt?
- **How** to adapt? *h*/*p*?

- Manual adaptation
- Automatic adaptation
  - Criterion to indicate adaptation
  - General approach -OR- tied to the problem

Adaptation criteria

- **Which** cells to adapt?
- **How** to adapt? *h*/*p*?

- Manual adaptation
- Automatic adaptation
  - Criterion to indicate adaptation
  - General approach -OR- tied to the problem

- Automatic *hp*-decision strategies discussed in the dissertation
  1. Error prediction based on refinement history [Melenk and Wohlmuth, 2001]
  2. Smoothness estimation by decay of Fourier coefficients [Bangerth and Kayser-Herold, 2009]
  3. Smoothness estimation by decay of Legendre coefficients [Mavriplis, 1994]

Example: Reentrant corner

- Singularity at reentrant corners for elliptic problems
- L-shaped domain:

$$\Omega = [-1,1]^2 \setminus ([0,1] \times [-1,0])$$

- Manufactured Laplace problem

$$\begin{aligned} -\nabla^2 u &= 0 \qquad \text{on} \quad \Omega \\ u &= \bar{u} \qquad \text{on} \quad \partial\Omega \end{aligned}$$

$$\begin{aligned} \bar{u} &= r^{2/3} \sin(2/3\,\varphi) \\ \|\nabla \bar{u}\| &= r^{-1/3} \end{aligned}$$

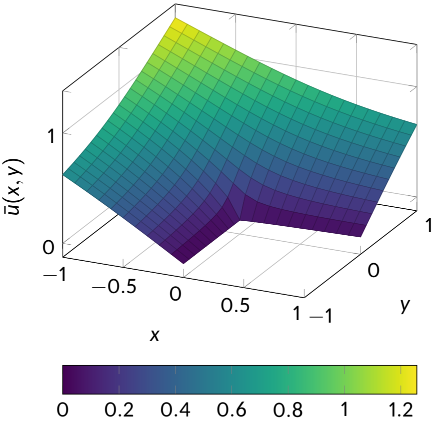


Figure: L-shaped domain

Example: Successive refinement

- Initialize coarse mesh
- Solve and refine in multiple cycles for tailored discretization

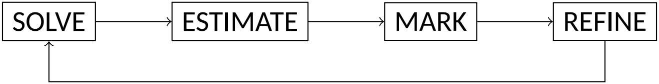


Figure: Successive refinement

1. Calculate refinement criteria (here: error estimates)
2. Flag 30%/3% of cells with highest/lowest criterion for refinement/coarsening
3. Calculate decision criteria (here: smoothness estimates)
4. Flag 90%/10% for *p*-/*h*-adaptation

Example: Successive refinement

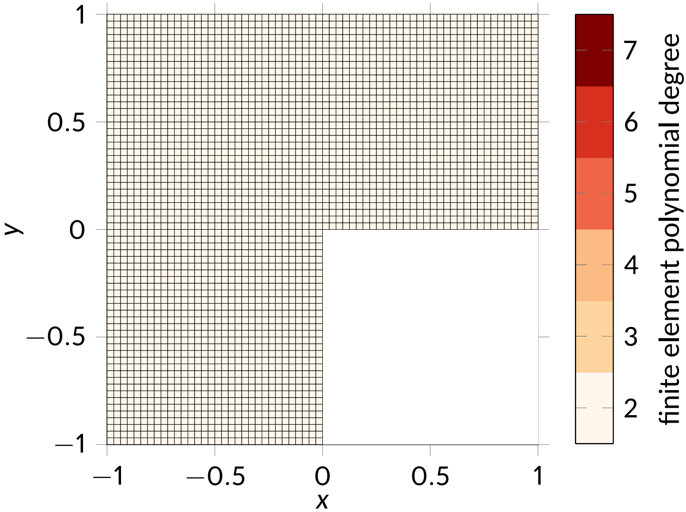


Figure: Polynomial degrees in cycle 0. Zoom 100%.

Example: Successive refinement

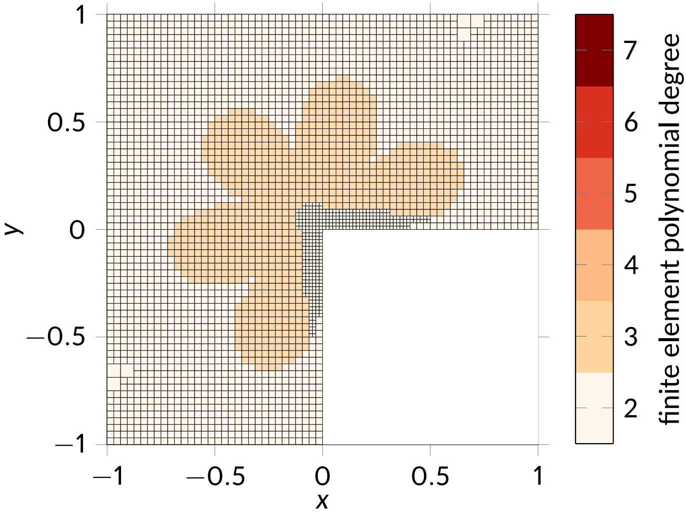


Figure: Polynomial degrees in cycle 1. Zoom 100%.

Example: Successive refinement

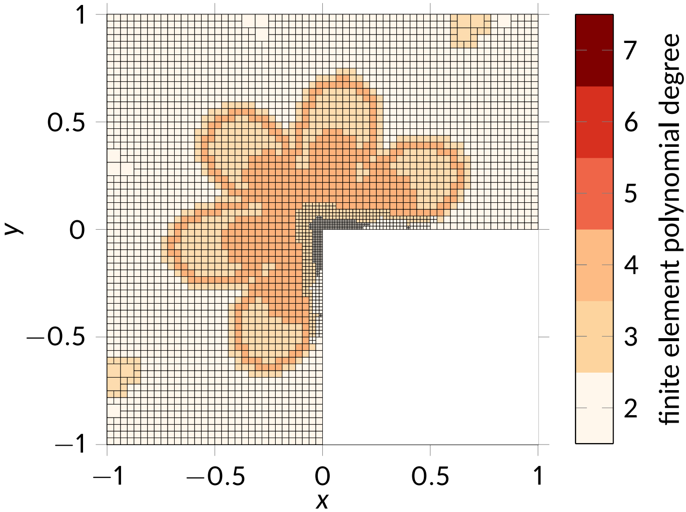


Figure: Polynomial degrees in cycle 2. Zoom 100%.

Example: Successive refinement

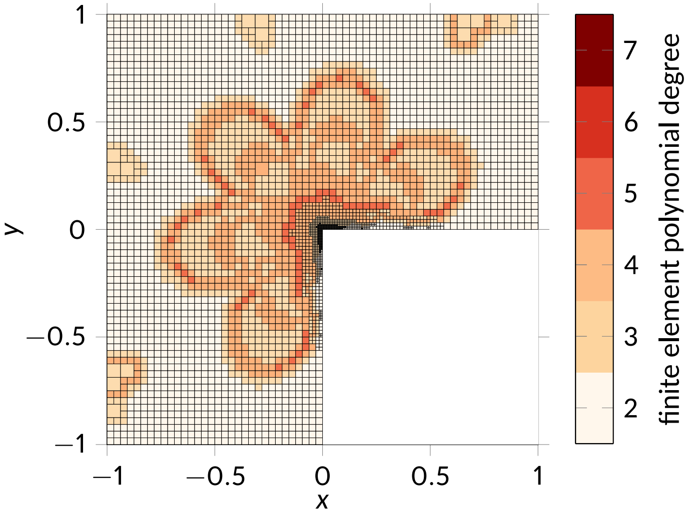


Figure: Polynomial degrees in cycle 3. Zoom 100%.

Example: Successive refinement

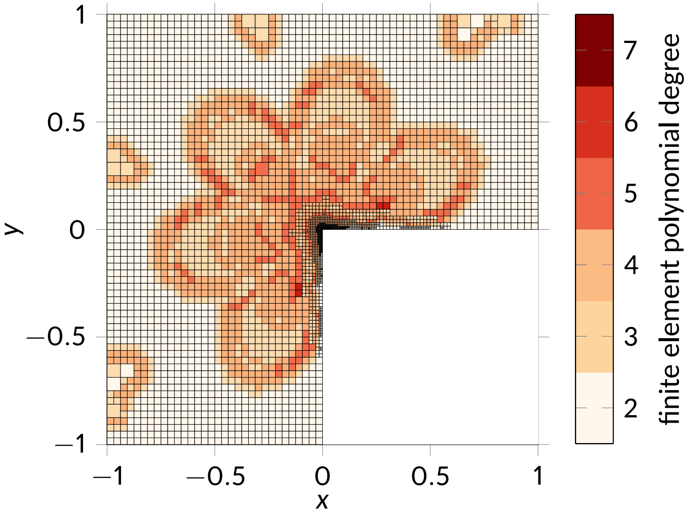


Figure: Polynomial degrees in cycle 4. Zoom 100%.

Example: Successive refinement

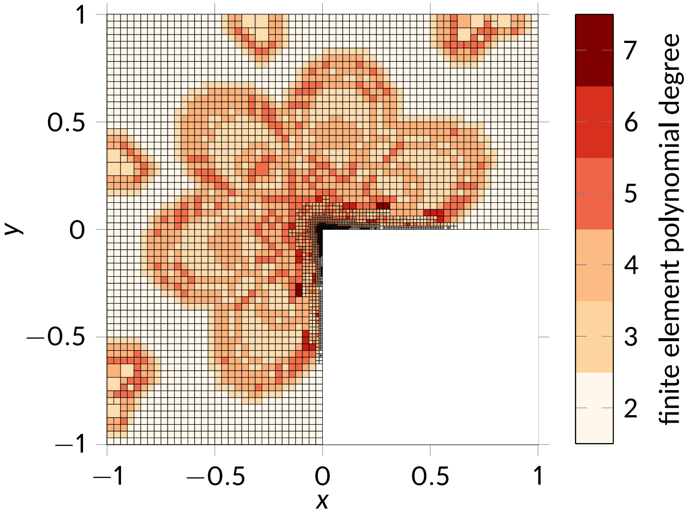


Figure: Polynomial degrees in cycle 5. Zoom 100%.

Example: Successive refinement

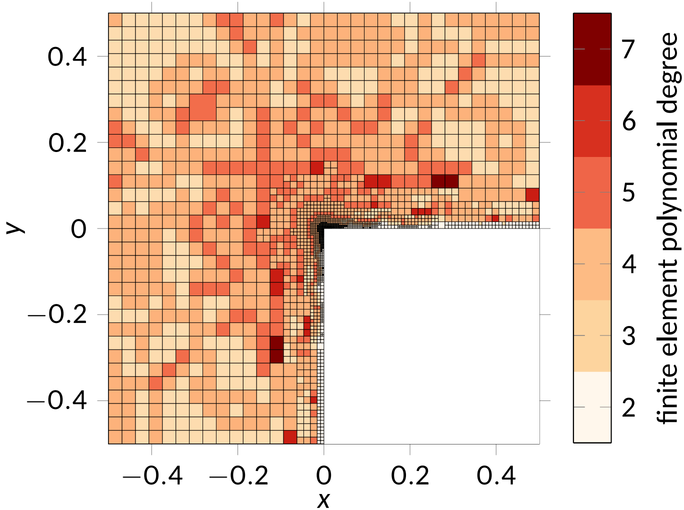


Figure: Polynomial degrees in cycle 5. Zoom 200%.

Example: Successive refinement

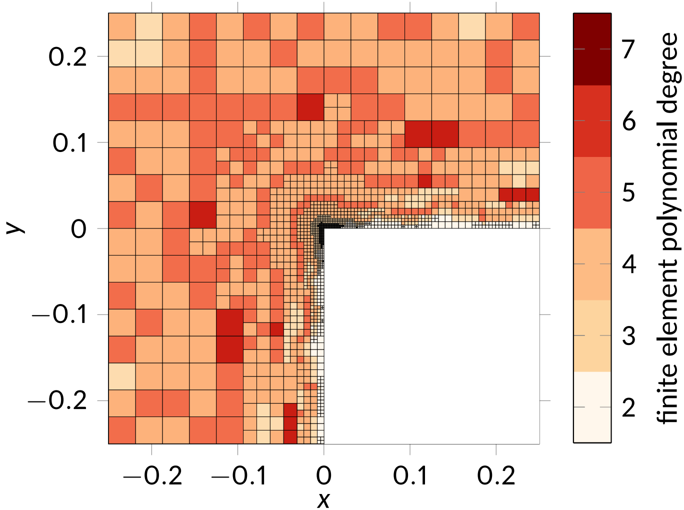


Figure: Polynomial degrees in cycle 5. Zoom 400%.

Example: Successive refinement

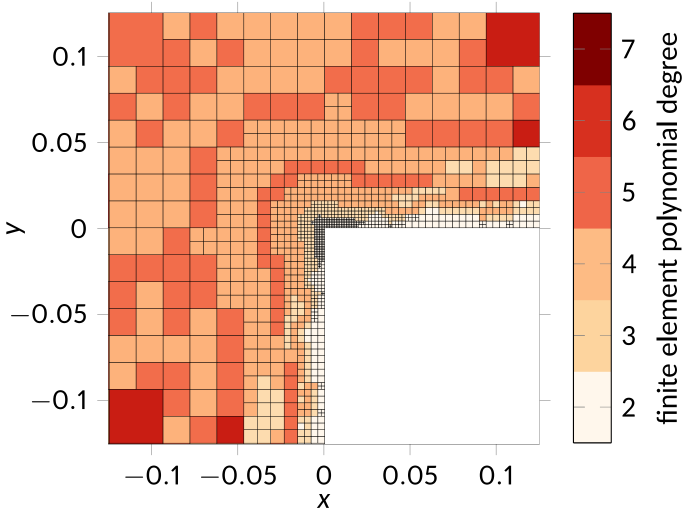
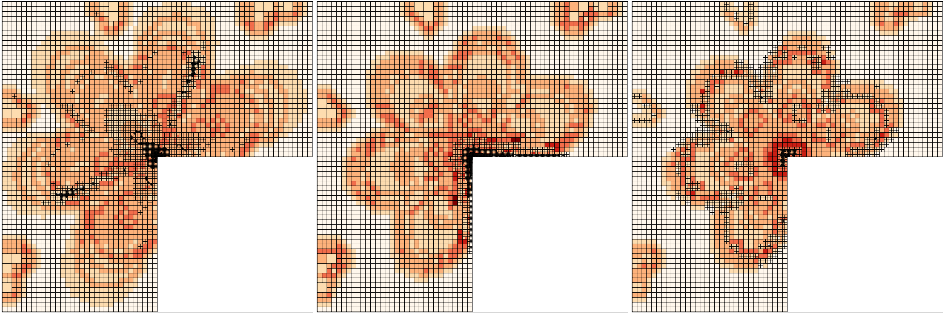


Figure: Polynomial degrees in cycle 5. Zoom 800%.

Example: Successive refinement



(a) Fourier coefficient decay (b) Legendre coefficient decay (c) Refinement history

Figure: Mesh and polynomial degrees of finite elements after 5 consecutive *hp*-adaptations.

Example: Successive refinement

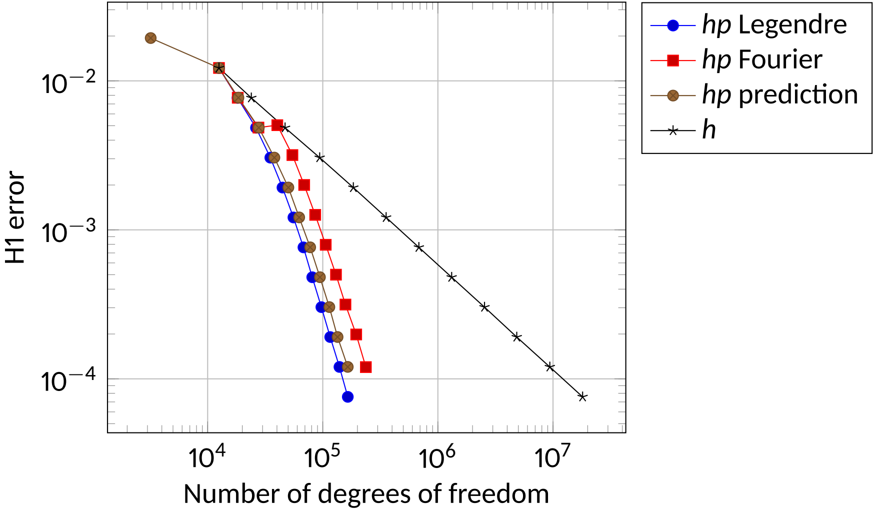
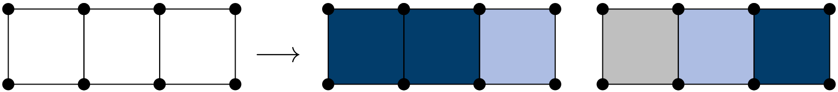


Figure: Error convergence for different strategies

Table of contents

Parallelization

- Current computer architectures provide multi-core processors
  - Supercomputers arrange those on distributed nodes
- Using all resources efficiently requires parallelization
  - Distribution of workload and memory demand
- Our approach: Distribution of domain on several processes
  - Each subdomain needs relevant part of the global solution
  - Requires a layer of so called ghost cells
  - Involves communication between processors



(a) Domain to be distributed (b) Subdomain for CPU 0 (c) Subdomain for CPU 1

Figure: Illustration of **locally owned**, ghost, and artificial cells

Parallel hp-adaptive FEM

- Combination of hp-adaptive methods with parallelisation
- The non-trivial parts are:
  1. Enumeration of degrees of freedom, independent of number of subdomains
  2. Consignment of contiguous memory chunks for data transfer
  3. Weighted repartitioning for load balancing

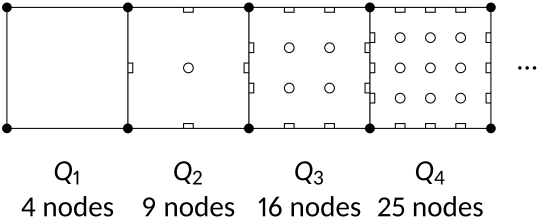


Figure: Different finite elements and their number of nodes in 2D

Example: Load balancing

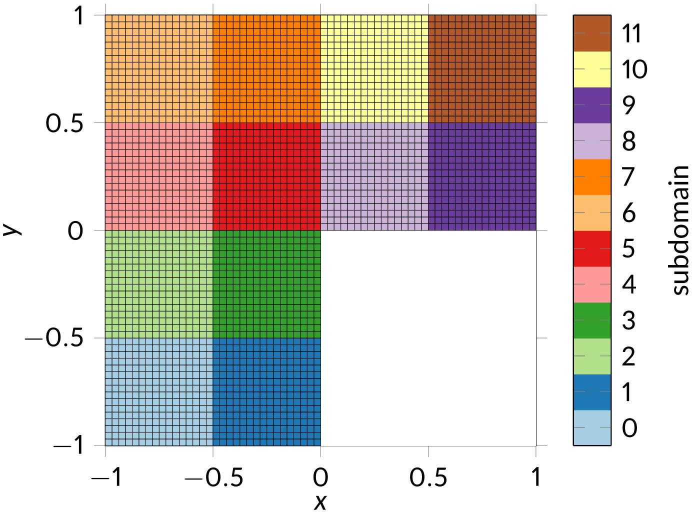


Figure: Mesh decomposition in cycle 0. Weights assigned to cells are $\propto n_{\text{dofs}}^{1.9}$.

Example: Load balancing

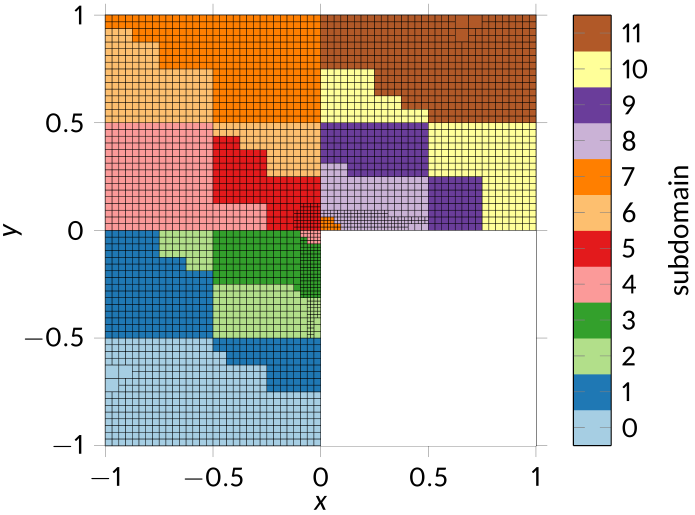


Figure: Mesh decomposition in cycle 1. Weights assigned to cells are $\propto n_{\mathrm{dofs}}^{1.9}$.

Example: Load balancing

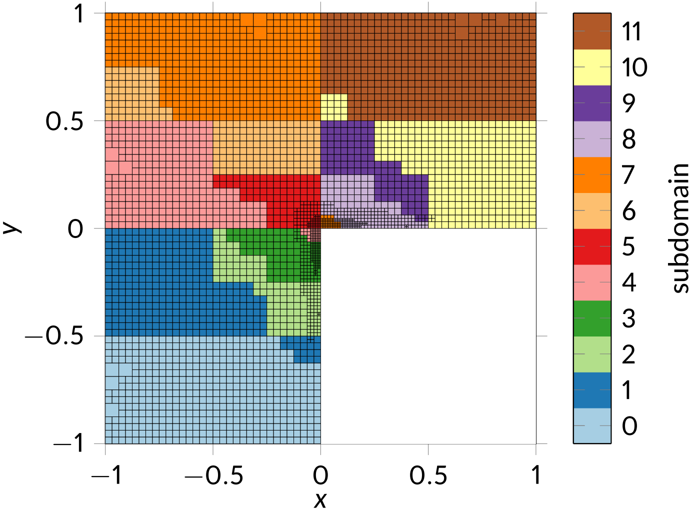


Figure: Mesh decomposition in cycle 2. Weights assigned to cells are $\propto n_{\mathrm{dofs}}^{1.9}$.

Example: Load balancing

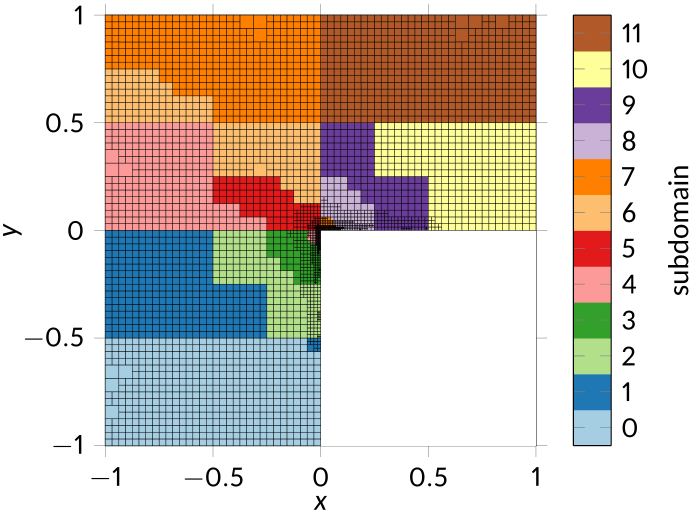


Figure: Mesh decomposition in cycle 3. Weights assigned to cells are $\propto n_{\text{dofs}}^{1.9}$.

Example: Load balancing

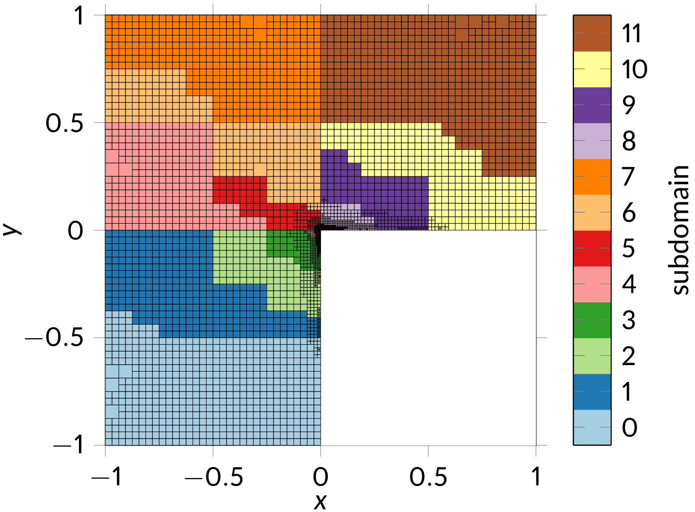


Figure: Mesh decomposition in cycle 4. Weights assigned to cells are $\propto n_{\mathrm{dofs}}^{1.9}$.

Example: Load balancing

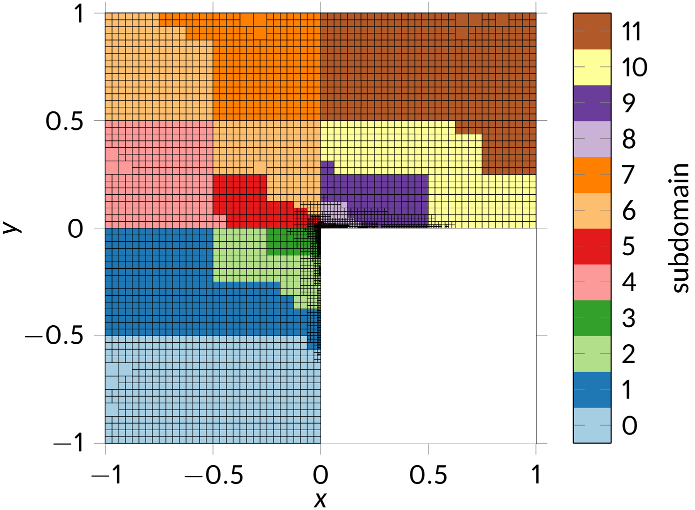


Figure: Mesh decomposition in cycle 5. Weights assigned to cells are $\propto n_{\text{dofs}}^{1.9}$.

Example: Strong scaling

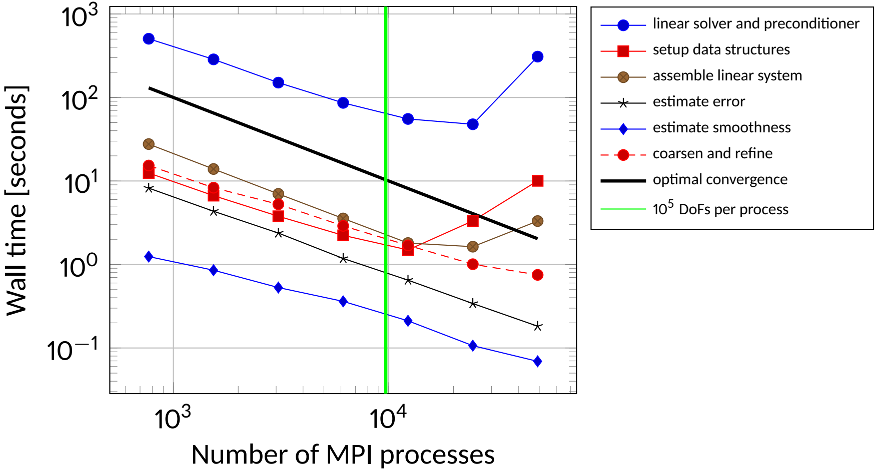


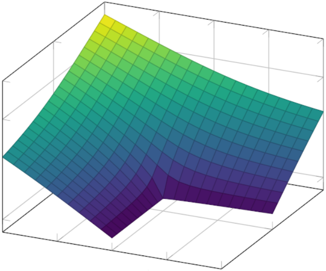
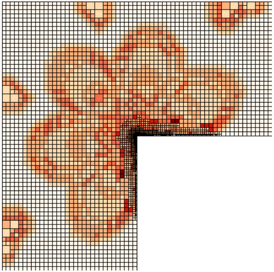
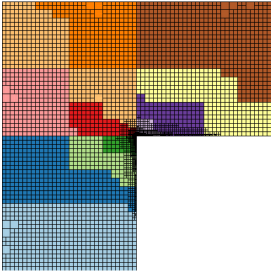
Figure: Strong scaling for fixed problem size of $\sim$ 970 million degrees of freedom

Table of contents

Summary & Outlook

- New algorithm for massively parallel hp-adaptive methods, generally applicable for any FEM software
- Reference implementation in `deal.II` involves:
  - Enumeration of degrees of freedom, independent of number of subdomains
  - Consignment of contiguous memory chunks for data transfer
  - Weighted repartitioning for load balancing
  - Selection of adaptation strategies for *hp*-FEM

- Future steps:
  - p-Multigrid methods
  - MatrixFree methods
  - Provide tutorial in `deal.II` as a manual for a broader audience
  - More applications

MASSIVELY PARALLEL HP-ADAPTIVE FEM

Bibliography

June 05, 2020 | Marc Fehling | m.fehling@fz-juelich.de |

BERGISCHE UNIVERSITÄT WUPPERTAL

JÜLICH Forschungszentrum

Bibliography I

Ainsworth, M. and Senior, B. (1998).
An adaptive refinement strategy for hp-finite element computations.
Applied Numerical Mathematics, 26(1):165–178.

Arndt, D., Bangerth, W., Clevenger, T. C., Davydov, D., Fehling, M., Garcia-Sanchez, D., Harper, G., Heister, T., Heltai, L., Kronbichler, M., Kynch, R. M., Maier, M., Pelteret, J.-P., Turcksin, B., and Wells, D. (2019).
The `deal.II` Library, Version 9.1.
Journal of Numerical Mathematics, 27(4):203–213.

Babuška, I. and Guo, B. (1996).
Approximation properties of the h-p version of the finite element method.
Computer Methods in Applied Mechanics and Engineering, 133(3):319–346.

Bibliography II

Babuška, I. and Suri, M. (1990).
The p- and h-p versions of the finite element method, an overview.
Computer Methods in Applied Mechanics and Engineering, 80(1):5–26.

Bangerth, W., Burstedde, C., Heister, T., and Kronbichler, M. (2012).
Algorithms and Data Structures for Massively Parallel Generic Adaptive Finite Element Codes.
ACM Transactions on Mathematical Software, 38(2):14:1–14:28.

Bangerth, W. and Kayser-Herold, O. (2009).
Data Structures and Requirements for Hp Finite Element Software.
ACM Transactions on Mathematical Software, 36(1):4:1–4:31.

Burstedde, C. (2018).
Parallel tree algorithms for AMR and non-standard data access.
arXiv e-prints, page arXiv:1803.08432.

Bibliography III

Eibner, T. and Melenk, J. M. (2007).
An adaptive strategy for *hp*-FEM based on testing for analyticity.
Computational Mechanics, 39(5):575–595.

Houston, P. and Süli, E. (2005).
A note on the design of hp-adaptive finite element methods for elliptic partial differential equations.
Computer Methods in Applied Mechanics and Engineering, 194(2):229–243.

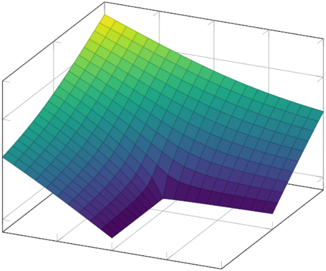
Kelly, D. W., De S. R. Gago, J. P., Zienkiewicz, O. C., and Babuška, I. (1983).
A posteriori error analysis and adaptive processes in the finite element method: Part I–Error Analysis.
International Journal for Numerical Methods in Engineering, 19(11):1593–1619.

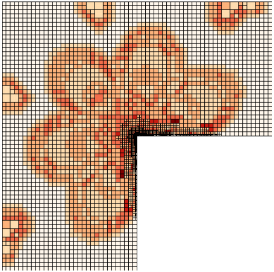
Bibliography IV

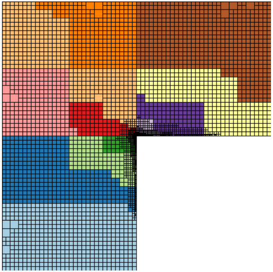
Mavriplis, C. (1994).
Adaptive mesh strategies for the spectral element method.
Computer Methods in Applied Mechanics and Engineering, 116(1):77–86.

Melenk, J. M. and Wohlmuth, B. I. (2001).
On residual-based a posteriori error estimation in hp-FEM.
Advances in Computational Mathematics, 15(1):311–331.

Mitchell, W. F. and McClain, M. A. (2014).
A Comparison of hp-Adaptive Strategies for Elliptic Partial Differential Equations.
ACM Transactions on Mathematical Software, 41(1):2:1–2:39.







MASSIVELY PARALLEL HP-ADAPTIVE FEM

Expected questions

June 05, 2020 | Marc Fehling | m.fehling@fz-juelich.de |

BERGISCHE
UNIVERSITÄT
WUPPERTAL

JÜLICH
Forschungszentrum

INS assumptions

- Incompressibility assumption: Mach number < 0.3
- Buoyancy force with Boussinesq approximation:
  $f \approx \rho_0 g - \rho_0 g \beta (T - T_0)$

Karniadakis scheme

- Explicit convection step
  $(u^* - u^n)/k = -\nabla \cdot (u^n u^n) + f^{n+1}$
- Pressure Poisson equation and velocity reconstruction
  $\nabla^2 p^{n+1} = -1/k\, \nabla \cdot u^*$
  $u^{**} = u^* - k\, \nabla p^{n+1}$
- Implicit diffusion step
  $(u^{n+1} - u^{**})/k = \nabla \cdot (2\nu\epsilon)$

Dimensionless quantities

- Reynolds number: inertial/viscous forces (turbulence behavior)
- Rayleigh number: time scales for thermal transport diffusion/convection
- Grashof number: buoyant/viscous forces
- Prandtl number: momentum/thermal diffusivity

Types of PDEs

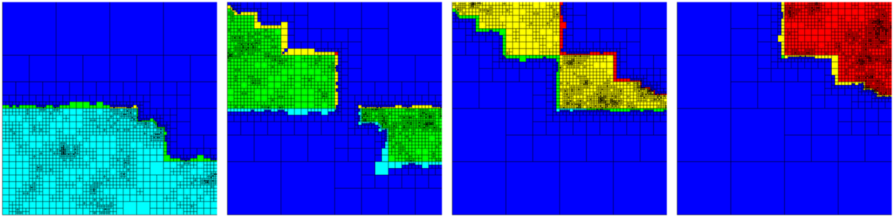
General form of second order PDE with two variables

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + +Fu + G = 0$$

- Elliptic equations: $B^2 - AC < 0$
  Laplace equation: $-u_{xx} - u_{yy} = 0$
- Hyperbolic equations: $B^2 - AC > 0$
  Wave equation: $u_{tt} - c^2 u_{xx} = 0$
- Parabolic equations: $B^2 - AC = 0$
  Heat equation: $u_t - \alpha u_{xx} = 0$

Artificial cells

- Every process knows about global coarse mesh
- Correct level representation only on locally owned cells
- 2:1 mesh balance
- Artificial cells as coarse as possible



Space-filling curve

- Z-order space filling curve
- Partitioning via leaves of tree structure

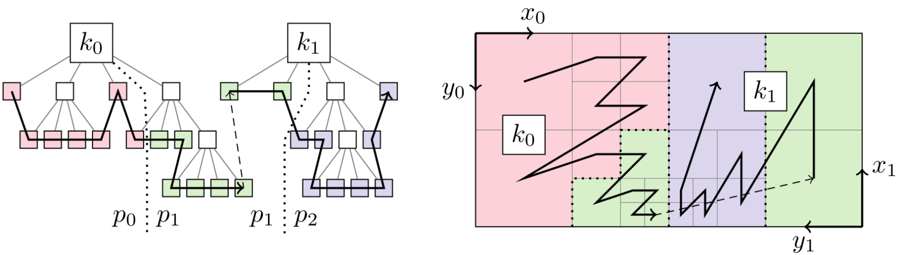
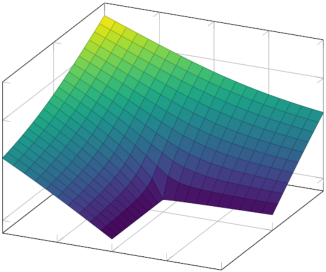
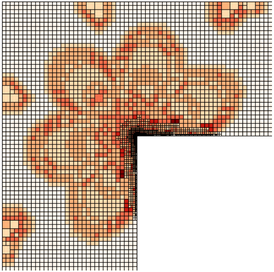
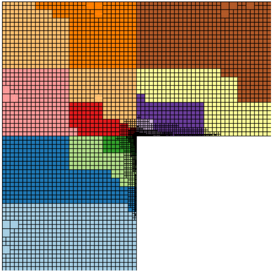


Figure: Correspondence of hierarchic tree structure and geometry, incorporating partitioning and the space-filling curve







MASSIVELY PARALLEL HP-ADAPTIVE FEM

Details on algorithms

June 05, 2020 | Marc Fehling | m.fehling@fz-juelich.de |

BERGISCHE UNIVERSITÄT WUPPERTAL

JÜLICH Forschungszentrum

Finite element method

- Solve variational equation from 'weak' formulation of the differential equation with bilinear form $a(u, v)$:

$$\exists u \in V : \forall v \in V : a(u, v) = f(v)$$

- Choose subspace V_h with basis w_i, out of which the approximate solution $u_\mathrm{h} = \sum u_i w_i \in V_\mathrm{h}$ will be constructed:

$$a(u_\mathrm{h}, w_\mathrm{j}) = \sum_i a(w_i, w_\mathrm{j})\, u_i = f(w_\mathrm{j}) \quad \rightarrow \quad AU = F$$

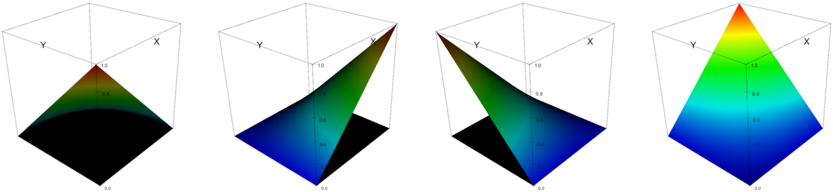


Figure: Q_1 elements in 2D

Laplace equation

Multiply with test function, integrate and apply Gauss theorem

$$-\nabla^2 u = f$$

$$-\int_\Omega \nabla^2 u\varphi = \int_\Omega \nabla u \nabla \varphi - \int_{\partial\Omega} (\boldsymbol{n} \cdot \nabla \boldsymbol{u})\, \varphi = \int_\Omega f$$

$\varphi = 0$ on boundary

$$(\nabla u, \nabla \varphi) = (f, \varphi)$$

Discretized

$$\sum_j (\nabla \varphi_i, \nabla \varphi_j)\, u_j = (f, \varphi_i)$$

$$AU = F$$

Gaussian quadrature

$$A_{ij}^K = \int_K \nabla\varphi_i \cdot \nabla\varphi_j \approx \sum_q \nabla\varphi_i(\mathbf{x}_q^K) \cdot \nabla\varphi_j(\mathbf{x}_q^K) w_q^K,$$

$$F_i^K = \int_K \varphi_i f \approx \sum_q \varphi_i(\mathbf{x}_q^K) f(\mathbf{x}_q^K) w_q^K,$$

Choose finite elements upon adaptation

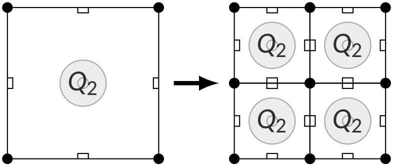


Figure: *hp* refinement

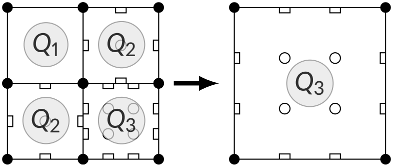


Figure: *hp* coarsening

Enumeration of degrees of freedom

- Numbering of DoFs necessary to build linear equation
- Parallelization and p-adaptive methods require different algorithms
  - See parallel [Bangerth et al., 2012] and hp [Bangerth and Kayser-Herold, 2009] papers for details

Combination of both algorithms **not trivial**

- Why not number DoFs locally and handle relations with constraints?
  - Number of DoFs would vary with number of processors
  - Larger matrices & vectors, thus higher memory demand

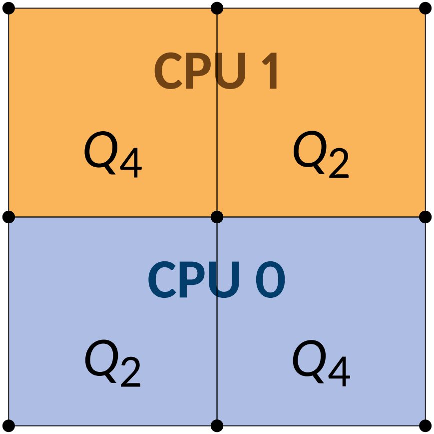
- Need for algorithm independent of number of processors
  - Results independent of number of processors
    - Proof that solvers work
  - Simplifies debugging

Enumeration algorithm

- Developed 6-phase algorithm
  - Requires two ghost exchanges

1. Local enumeration of DoFs
2. Invalidate DoFs on ghost interfaces to processors with lower rank
3. Unification of DoFs on local domain **and** ghost interfaces
   - Ownership of DoFs clarified
4. Global re-enumeration of DoFs
   - Local DoF indices set
5. Exchange of locally owned DoFs
6. Merge DoFs on ghost interfaces
   - Global DoF indices set

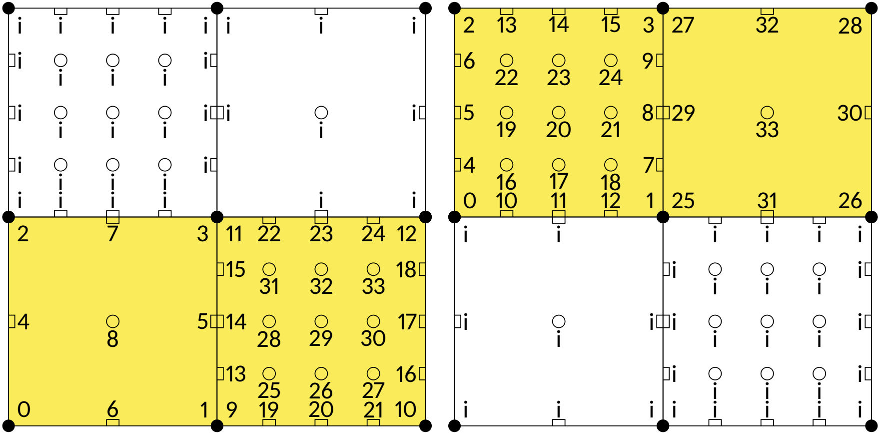
Example of application of enumeration algorithm



(Phase 1) Local enumeration

Figure: Example of application of our enumeration algorithm for degrees of freedom

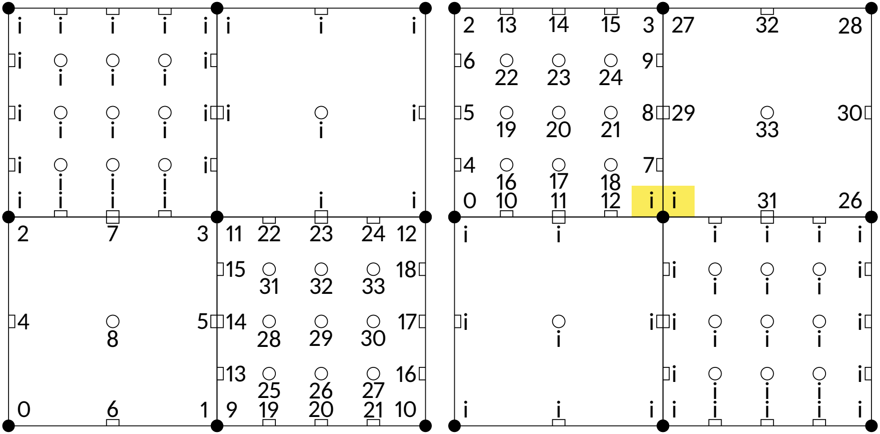
Example of application of enumeration algorithm



(Phase 1) Local enumeration

Figure: Example of application of our enumeration algorithm for degrees of freedom

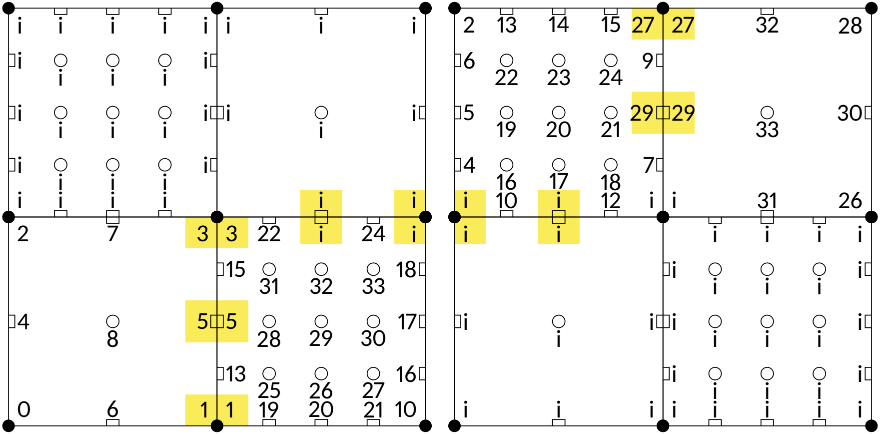
Example of application of enumeration algorithm



(Phase 2) Invalidation

Figure: Example of application of our enumeration algorithm for degrees of freedom

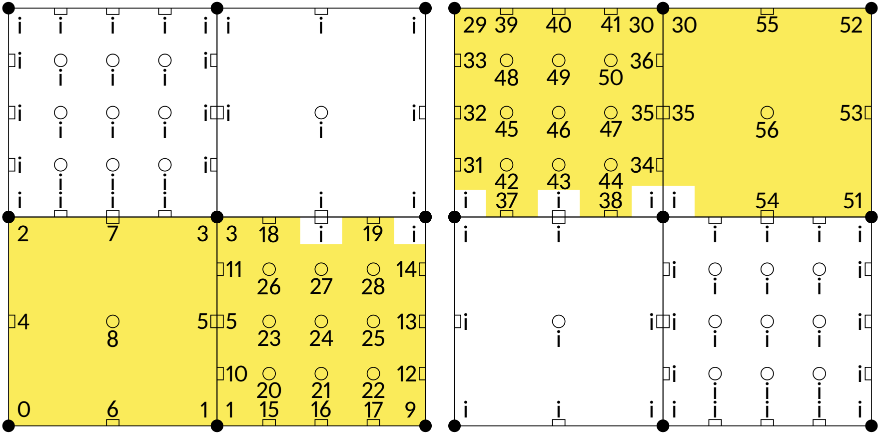
Example of application of enumeration algorithm



(Phase 3) Unification

Figure: Example of application of our enumeration algorithm for degrees of freedom

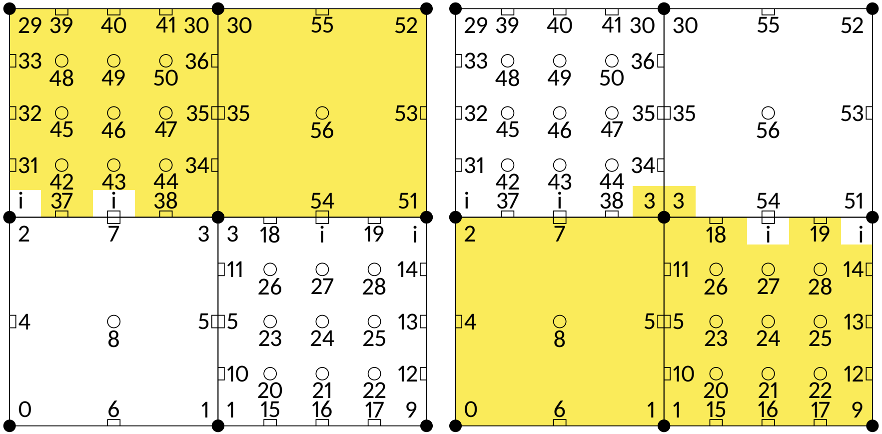
Example of application of enumeration algorithm



(Phase 4) Re-enumeration

Figure: Example of application of our enumeration algorithm for degrees of freedom

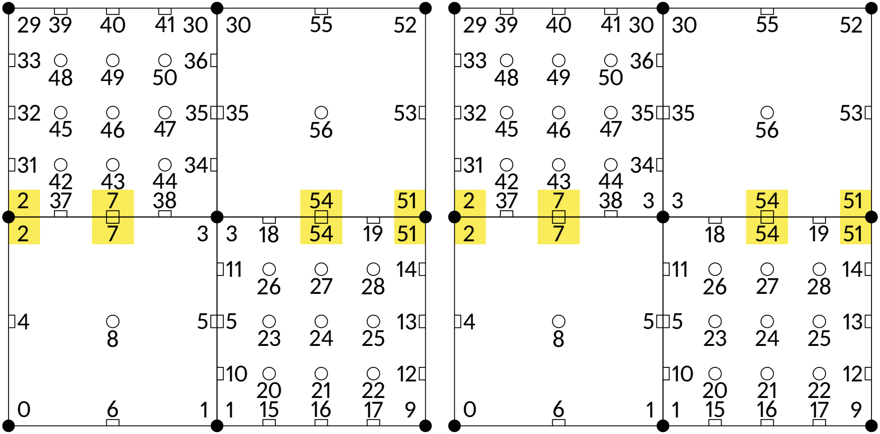
Example of application of enumeration algorithm



(Phase 5) Ghost exchange

Figure: Example of application of our enumeration algorithm for degrees of freedom

Example of application of enumeration algorithm



(Phase 6) Merge

Figure: Example of application of our enumeration algorithm for degrees of freedom

Data transfer across subdomains

- On distributed triangulations, each subdomain needs access to relevant fraction of global quantities
- Changes on cell ownership requires transfer of these quantities
  - Active finite element indices
  - Solution
  - ...
- With p-adaptive methods, per cell data sizes may differ

Communication between involved processors required

- Algorithm should be generic, i.e. independent of scenario
  1. Creation of **memory buffers**
  2. Transfer data of **fixed** and **variable** size

Structure of memory buffers

- Register functions that prepare data to be transferred via callback
- **On each locally owned cell**:
  1. Pack data for each registered callback individually
  2. Store memory sizes of each callback's data pack
  3. Store memory size of cell's complete data pack
- **After each cell** is processed:
  4. Move each cell's packed data into contiguous buffer

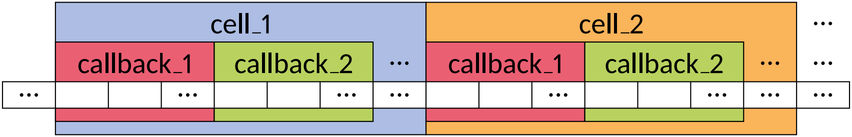


Figure: Division of contiguous memory chunk

Data consignment

- Treat fixed and variable size data separately
  - Each transfer algorithm optimized for their specific task
  - Potentially slower variable size transfer will only be used when necessary

1. **Fixed size data**:
   - Refrain from using compression
   - Additionally pack `CellStatus` information
   - Gathering callback sizes on first packed cell will suffice
     - Communicate callback sizes to all processors
2. **Variable size data**:
   - Compression allowed (using `ZLIB`)
   - Size of each callback's data differs from cell to cell
     - Register additional callback for fixed size data transfer

Data transfer

- We have contiguous memory chunks for data transfer during repartitioning, refinement/coarsening, serialization
  - Program may be resumed with a different number of processors

- Data consignment **independent** of transfer algorithms used for repartitioning, refinement/coarsening, serialization
  - Use non-blocking `MPI` communication for all operations
  - `deal.II` utilizes interface to `p4est` [Burstedde, 2018]

FEM error behavior

- Error behavior for *hp*-FEM is well understood [Babuška and Suri, 1990]

$$\|\nabla (u - u_{\mathrm{hp}})\|_{H^1(\Omega)} \leq C \frac{h^p}{p^{m-1}} \|u\|_{H^m(\Omega)}$$

- Exponential convergence rate possible with *hp*-adaptation [Babuška and Guo, 1996]

$$\|\nabla (u - u_{\mathrm{hp}})\|_{H^1(\Omega)} \leq C \exp\left(-b\, N_{\mathrm{dofs}}^{\alpha}\right)$$

with $\alpha = 1/3$ in 2D and $\alpha = 1/5$ in 3D

Kelly error estimation

- Error estimator by [Kelly et al., 1983]
- Meant for generalized Poisson equation $-\nabla \cdot (a\nabla u) = f$
- Proved to be useful in other applications as well

$$\|e_{\mathrm{hp}}\|^2_{H_1(\Omega)} \leq C \sum_{K\in\Omega} \eta_K^2$$

$$\eta_K^2 = \sum_{F\in\partial K} c_F \int\limits_F \left[a \frac{\partial u_{\mathrm{hp}}}{\partial n}\right]^2 do$$

$$\left[a \frac{\partial u_{\mathrm{hp}}}{\partial n}\right] = a \left.\frac{\partial u_{\mathrm{hp}}}{\partial n_K}\right|_K + a \left.\frac{\partial u_{\mathrm{hp}}}{\partial n_J}\right|_J$$

Refinement history

- Verify prediction's accuracy
- Decide on either h- or p-adaptation on marked cells

  Keep $\boldsymbol{h}$ small: $\eta_K > \eta_{K,\text{pred}}$

  Keep $\boldsymbol{p}$ large: $\eta_K \leq \eta_{K,\text{pred}}$

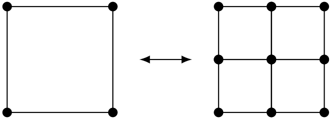


Figure: h-adaptation

Figure: Error prediction algorithm based on [Melenk and Wohlmuth, 2001]

Adaptation type	Prediction formula
no adaptation	$\eta_{K,\text{pred}} = \eta_K \, \gamma_\text{n}$
p-adaptation	$\eta_{K,\text{pred}} = \eta_K \, \gamma_\text{p}^{(p_{K,\text{future}} - p_K)}$
hp-refinement	$\left(\eta_{K_c,\text{pred}}\right)^2 = n_{K_c}^{-1} \left(\eta_{K_p} \, \gamma_\text{h} \, 0.5^{p_{K_c,\text{future}}} \, \gamma_\text{p}^{(p_{K_c,\text{future}} - p_{K_p})} \right)^2$
hp-coarsening	$\left(\eta_{K_p,\text{pred}}\right)^2 = \sum_{K_c} \left(\eta_{K_c} \left(\gamma_\text{h} \, 0.5^{p_{K_p,\text{future}}} \right)^{-1} \gamma_\text{p}^{(p_{K_p,\text{future}} - p_{K_c})} \right)^2$

Smoothness estimation

- Different representations of finite element approximation

$$u_{hp}(x) = \sum_i u_i \, \varphi_i(x) = \sum_k c_k \, P_k(x)$$

- Choose orthogonal basis $\{P_k\}$ of increasing frequency
- Uniquely attribute frequency of oscillations to each basis function

Orthogonal basis

- Legendre: Orthogonal basis of polynomials

$$P_k(x) = \frac{1}{2^k k!} \frac{d^k}{dx^k} \left((x^2 - 1)^k\right)$$

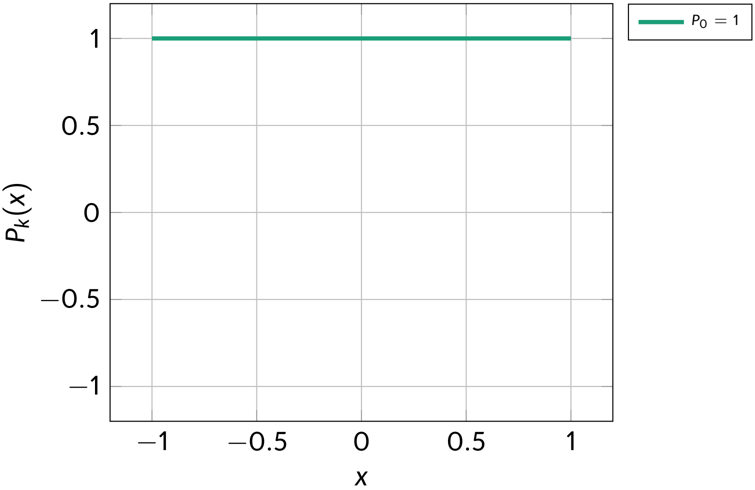
$$\langle P_k, P_l \rangle = \int P_k(x)\, P_l(x)\, dx = \frac{2}{2k+1} \delta_{kl}$$

- Fourier: Orthogonal basis of sinusoids

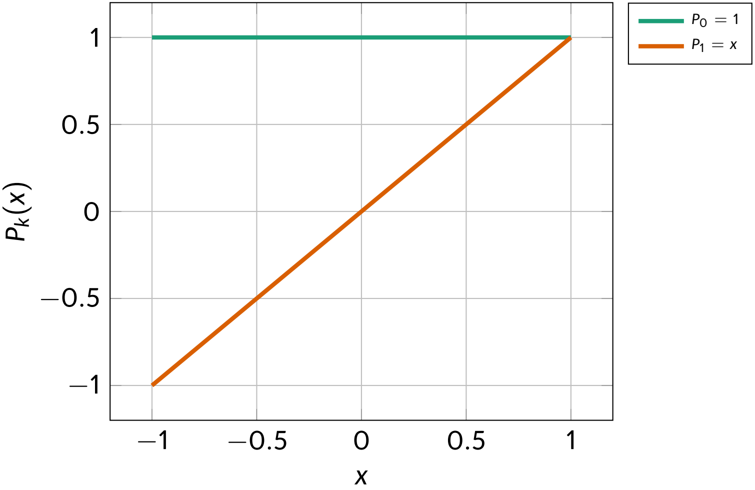
$$P_k(x) = \exp\left(-i\, 2\pi k\, x\right)$$

$$\langle P_k, P_l \rangle = \int P_k(x)\, P_l^*(x)\, dx = \delta_{kl}$$

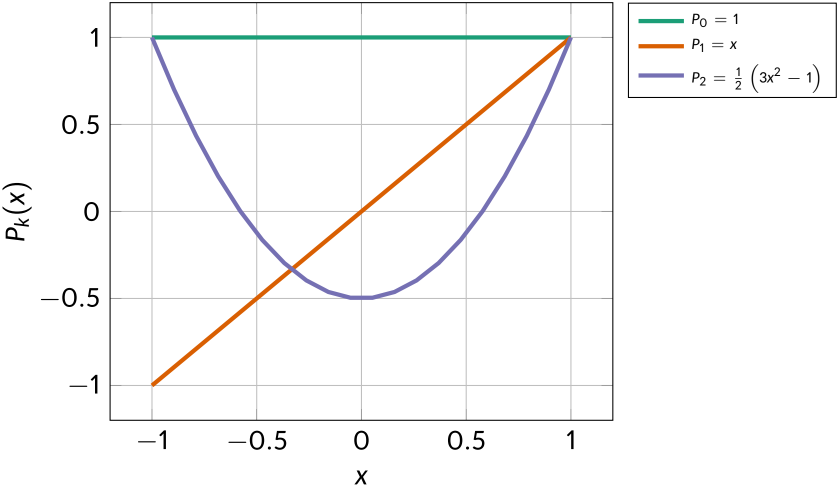
Legendre polynomials



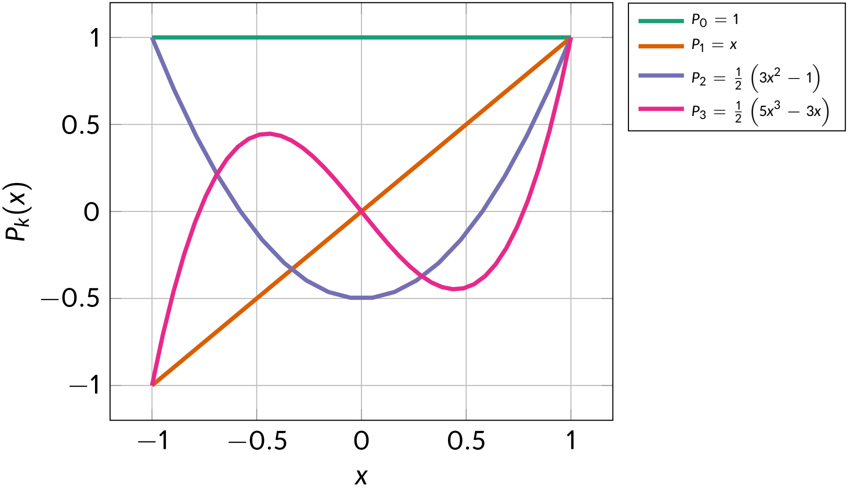
Legendre polynomials



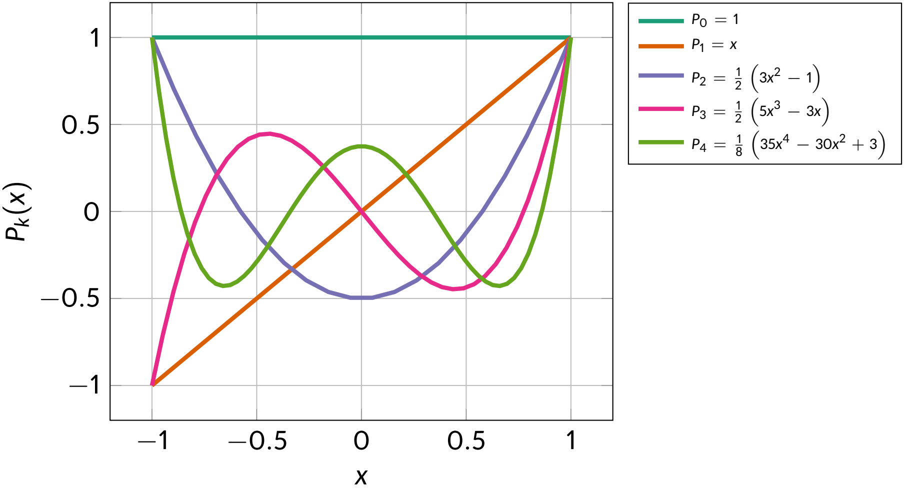
Legendre polynomials



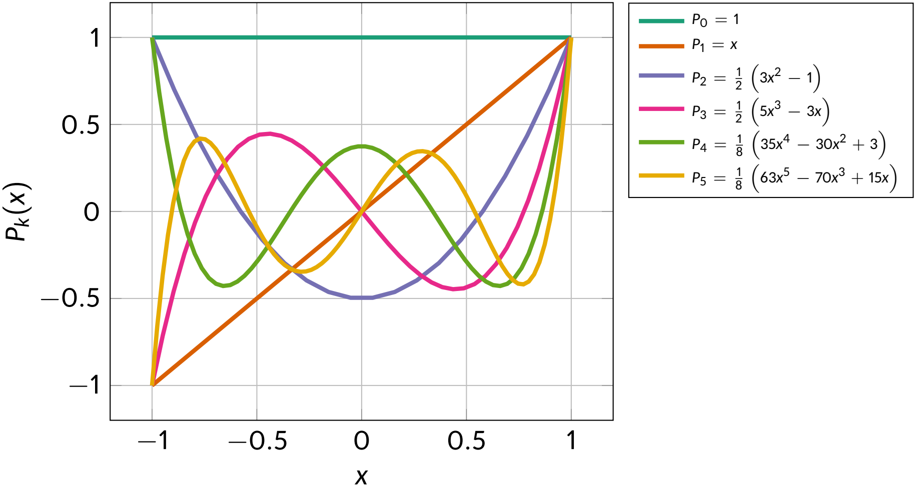
Legendre polynomials



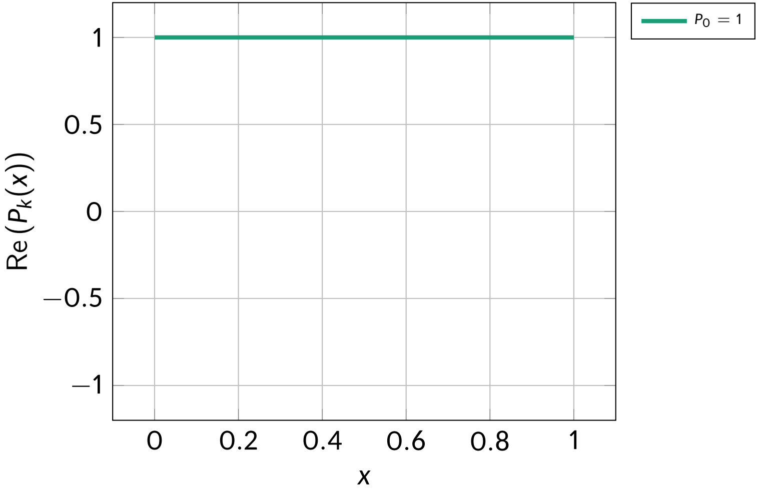
Legendre polynomials



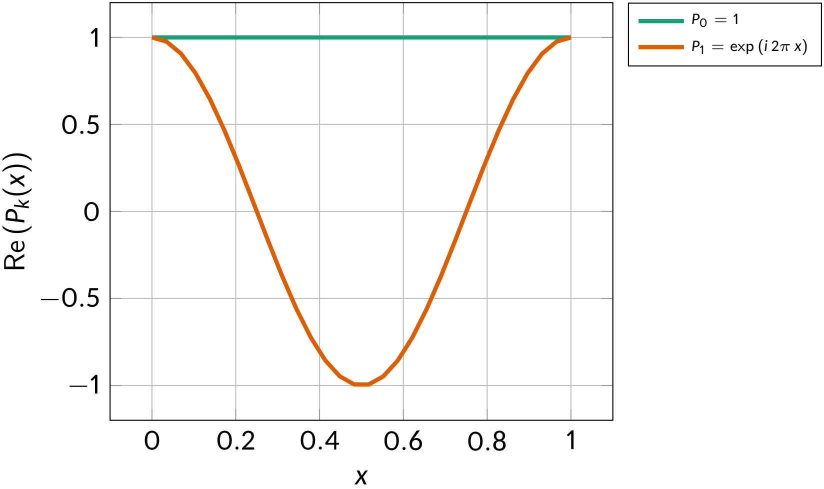
Legendre polynomials



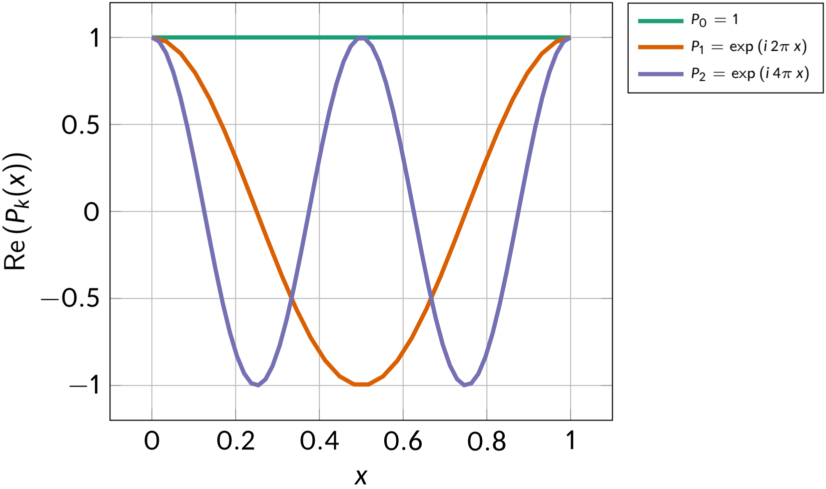
Fourier sinusoids



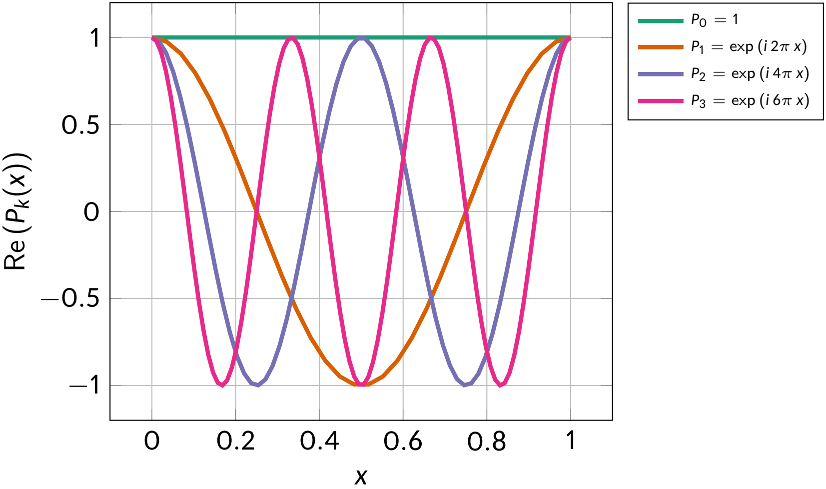
Fourier sinusoids



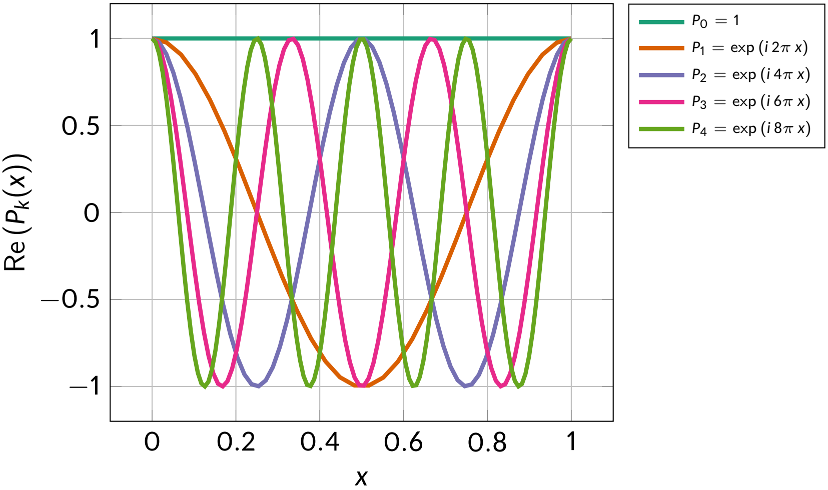
Fourier sinusoids



Fourier sinusoids



Fourier sinusoids



Decay of Legendre coefficients

- Determine coefficients $\{c_k\}$ on every cell K:

$$c_k = \int_K u_{hp}(x)\, P_k(x)\, dx$$

- FE approximation is analytic if coefficients decay exponentially [Eibner and Melenk, 2007]

$$|c_k| \leq C \exp\left(-\sigma |k|\right)$$

- Determine decay rate σ via least-squares fit

$$\ln(|c_k|) \sim \tilde{C} - \sigma |k|$$

Decay rates σ considered as smoothness estimates
Keep p large if σ is large

Decay of Fourier coefficients

- FE approximation is part of Hilbert space $\mathcal{H}^s$ if integral exists

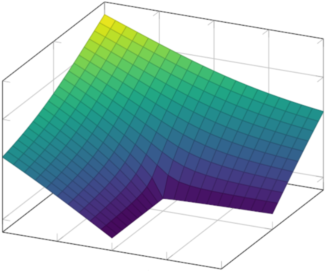
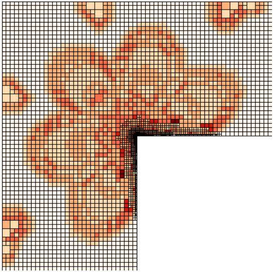
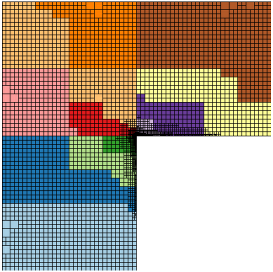
$$\int_K \left|\nabla^s u_{\mathrm{hp}}(x)\right|^2 dx = (2\pi)^{2s} \sum_k |c_k|^2 k^{2s} < \infty$$

$$\Rightarrow \qquad |c_k| = \mathcal{O}\left(k^{-\sigma-\epsilon}\right) \quad \text{with} \quad \sigma = s + \mathrm{dim}/2$$

- Determine coefficients $\{c_k\}$ and their decay via least-squares fit on every cell K:

$$c_k = \int_K u_{hp}(x)\, P_k^*(x)\, dx \qquad\qquad \ln(|c_k|) \sim C - \sigma \ln\left(|k|\right)$$

Decay rates σ considered as smoothness estimates
Keep p large if σ is large

MASSIVELY PARALLEL HP-ADAPTIVE FEM

More results

June 05, 2020 | Marc Fehling | m.fehling@fz-juelich.de |

BERGISCHE
UNIVERSITÄT
WUPPERTAL

JÜLICH
Forschungszentrum

Error convergence

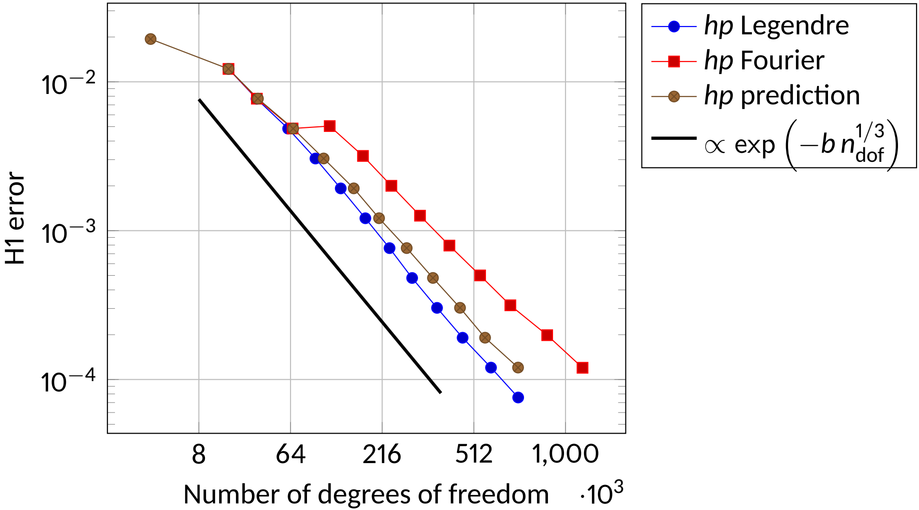


Figure: Customly scaled error convergence plot

Error vs runtime

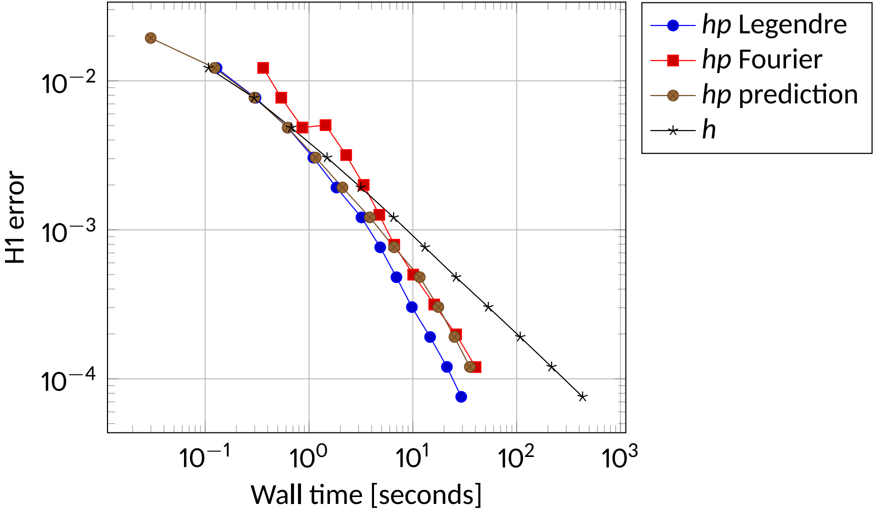


Figure: Error against walltime

Weighting exponent

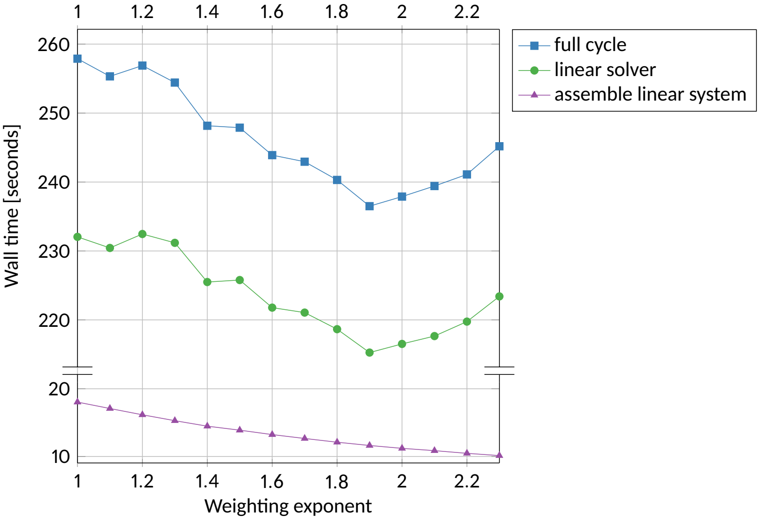


Figure: Wall times for load balancing with varying weighting exponents

Scaling on successive refinement

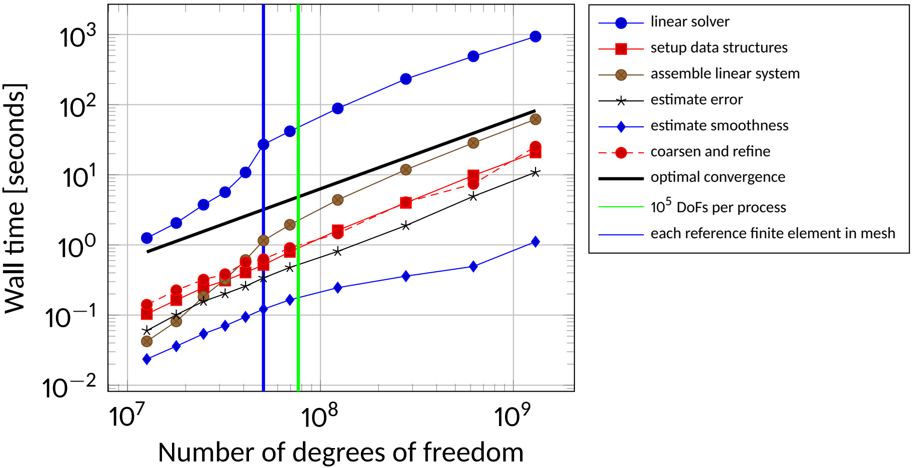


Figure: Scaling on successively refined grids