Towards r-h-adaptivity in FEM

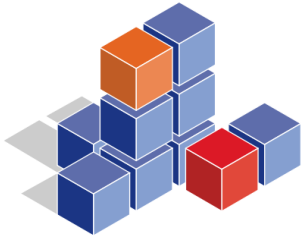
Matthias Grajewski

Matthias.Grajewski@math.uni-dortmund.de

Graduate School for Production Engineering and Logistics;

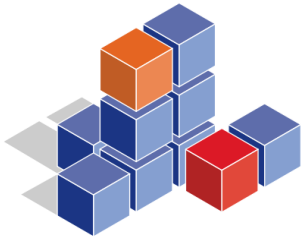
Institute for Applied Mathematics,

University of Dortmund



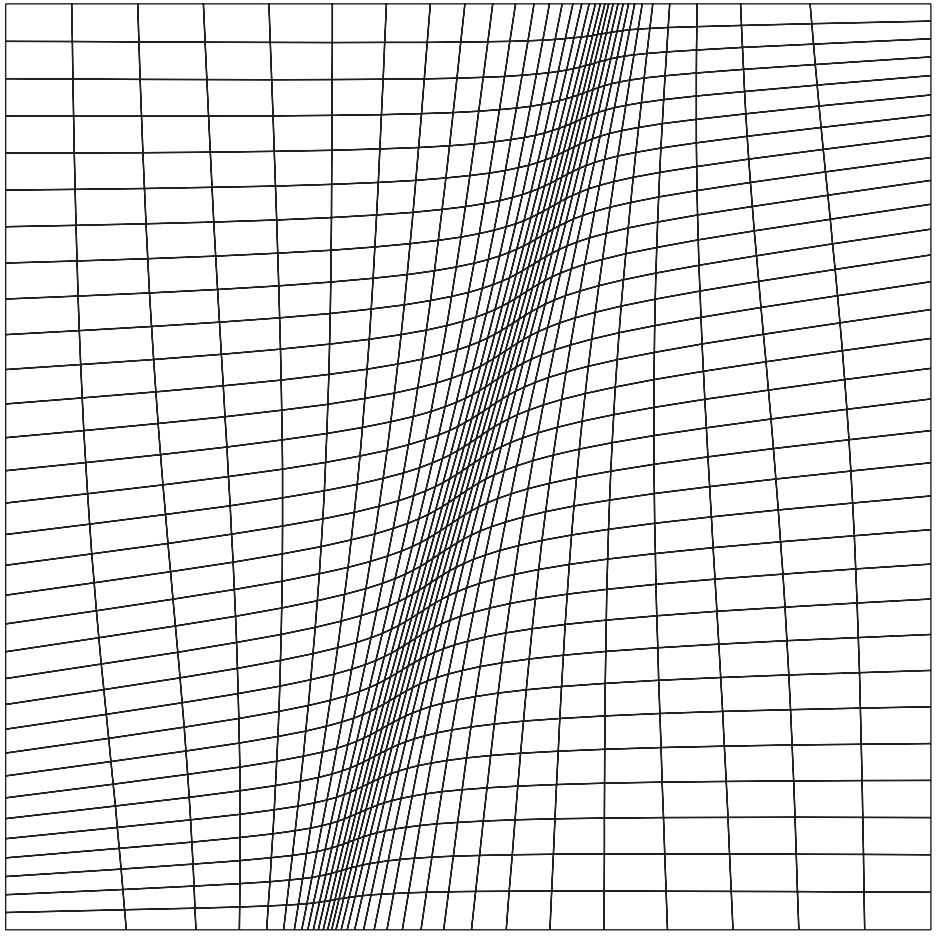
Outline

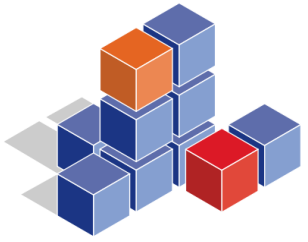
- motivation
- grid concepts for HPC
- r-adaptivity (deformation method)
- concepts of r-h-adaptivity
- numerical example



Why r-adaptivity?

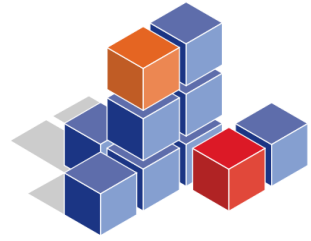
reason 1: flexibility and accuracy (e.g. mesh aligment)





Why r-adaptivity?

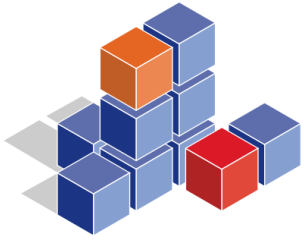
reason 2: **SPEED**



Why r-adaptivity?

reason 2: **SPEED**

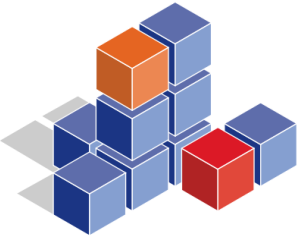
- use existing “standard hardware” optimally (caching etc.)
- use emerging powerful hardware as coprocessors



Why r-adaptivity?

reason 2: **SPEED**

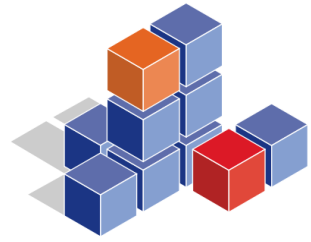
- use existing “standard hardware” optimally (caching etc.)
- use emerging powerful hardware as coprocessors



Why r-adaptivity?

example: Poisson problem, multigrid on a tensor-product-like mesh, mixed-precision iterative refinement

	CPU			GPU			
NEQ	#Iter	Time	RES	#Iter	Time	RES	Speedup
4225	8	0.11	6.61-6	5:9	0.19	6.61-6	0.58
16641	8	0.19	1.66-6	5:9	0.24	1.66-6	0.79
66049	8	0.47	4.18-7	5:9	0.33	4.18-7	1.42
263196	8	1.56	1.04-7	5:9	0.53	1.04-7	2.94
1050625	8	5.93	2.62-8	5:9	1.42	2.62-8	4.18

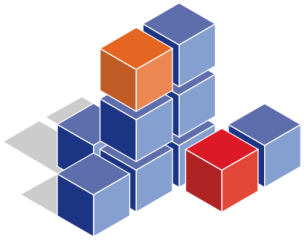


Why r-adaptivity?

example: Poisson problem, multigrid on a tensor-product-like mesh, mixed-precision iterative refinement

	CPU			GPU			
NEQ	#Iter	Time	RES	#Iter	Time	RES	Speedup
4225	8	0.11	6.61-6	5:9	0.19	6.61-6	0.58
16641	8	0.19	1.66-6	5:9	0.24	1.66-6	0.79
66049	8	0.47	4.18-7	5:9	0.33	4.18-7	1.42
263196	8	1.56	1.04-7	5:9	0.53	1.04-7	2.94
1050625	8	5.93	2.62-8	5:9	1.42	2.62-8	4.18

necessary: (local) generalised tensor product meshes



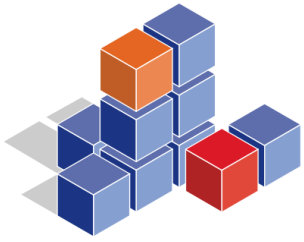
Why r-adaptivity?

example: Poisson problem, multigrid on a tensor-product-like mesh, mixed-precision iterative refinement

	CPU			GPU			
NEQ	#Iter	Time	RES	#Iter	Time	RES	Speedup
4225	8	0.11	6.61-6	5:9	0.19	6.61-6	0.58
16641	8	0.19	1.66-6	5:9	0.24	1.66-6	0.79
66049	8	0.47	4.18-7	5:9	0.33	4.18-7	1.42
263196	8	1.56	1.04-7	5:9	0.53	1.04-7	2.94
1050625	8	5.93	2.62-8	5:9	1.42	2.62-8	4.18

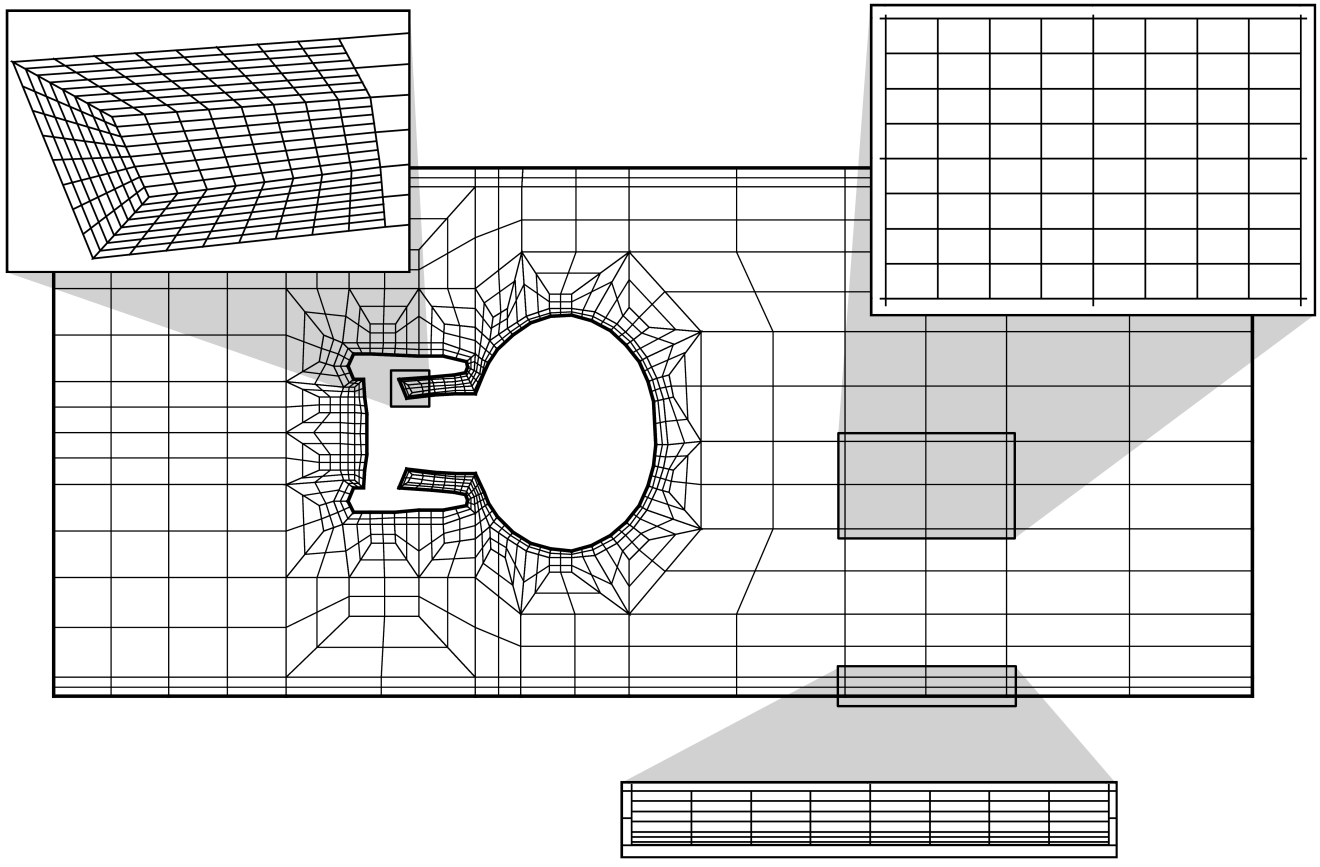
necessary: (local) generalised tensor product meshes

FEAST



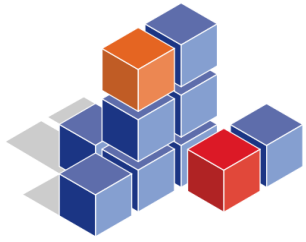
(Grid-related part of the) FEAST concept

global grid: “many” local generalised tensor product meshes (“macros”).



MFLOP-rate preserving adaptivity

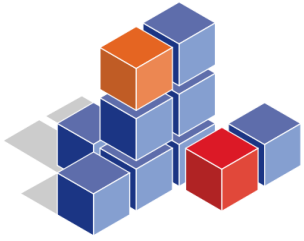
mandatory: preserve local logical tensor product structure



MFLOP-rate preserving adaptivity

mandatory: preserve local logical tensor product structure

approach 1: macrowise h-adaptivity (’hanging nodes’)

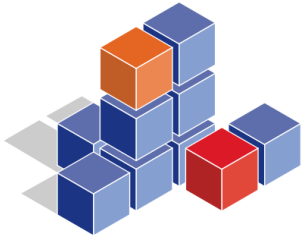


MFLOP-rate preserving adaptivity

mandatory: preserve local logical tensor product structure

approach 1: macrowise h-adaptivity (’hanging nodes’)

approach 2: r-adaptivity (preserves the logical structure of the grid)



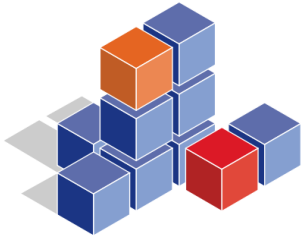
MFLOP-rate preserving adaptivity

mandatory: preserve local logical tensor product structure

approach 1: macrowise h-adaptivity (’hanging nodes’)

approach 2: r-adaptivity (preserves the logical structure of the grid)

approach 3: combine **1** and **2**



Background of our deformation method

- domain Ω
- triangulation $\mathbb{T}$ with quads T
- "monitor function" $0 < f \in \mathcal{C}^1(\bar{\Omega})$: desired area distribution
- "weighting function" $0 < g \in \mathcal{C}^1(\bar{\Omega})$: current area distribution

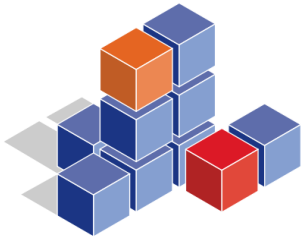
goal: construct transformation $\Phi : \Omega \to \Omega$ with

$$g(x)|J\Phi(x)| = f(\Phi(x)) \quad \forall x \in \Omega$$

and

$$\Phi : \partial\Omega \to \partial\Omega.$$

$$\mathbb{T}_{\text{new}} = \Phi(\mathbb{T})$$



Basic deformation algorithm

Deformation$(f, \mathbb{T})$

compute $\tilde{f} - \tilde{g}$, $g = g(\mathbb{T})$; $\quad \tilde{f} := c/f, \tilde{g} = C/g, \int \tilde{f} \stackrel{!}{=} \int \tilde{g}$

solve

$(\nabla w_h, \nabla \varphi_h) = (\tilde{f} - \tilde{g}, \varphi_h) \quad \forall \varphi_h \in \mathcal{Q}_1(\mathbb{T}), \quad \partial_{\mathfrak{n}} w_h|_{\partial\Omega} = 0$

$v_h :=$ recovered_gradient(w_h)

DO FORALL $x \in \mathbb{T}$

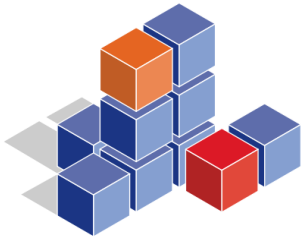
solve

$\partial_t \varphi(x,t) = \frac{v_h(\varphi(x,t),t)}{t\tilde{f}(\varphi(x,t)) + (1-t)\tilde{g}(\varphi(x,t))}, \quad 0 \le t \le 1, \ \varphi(x,0) = x$

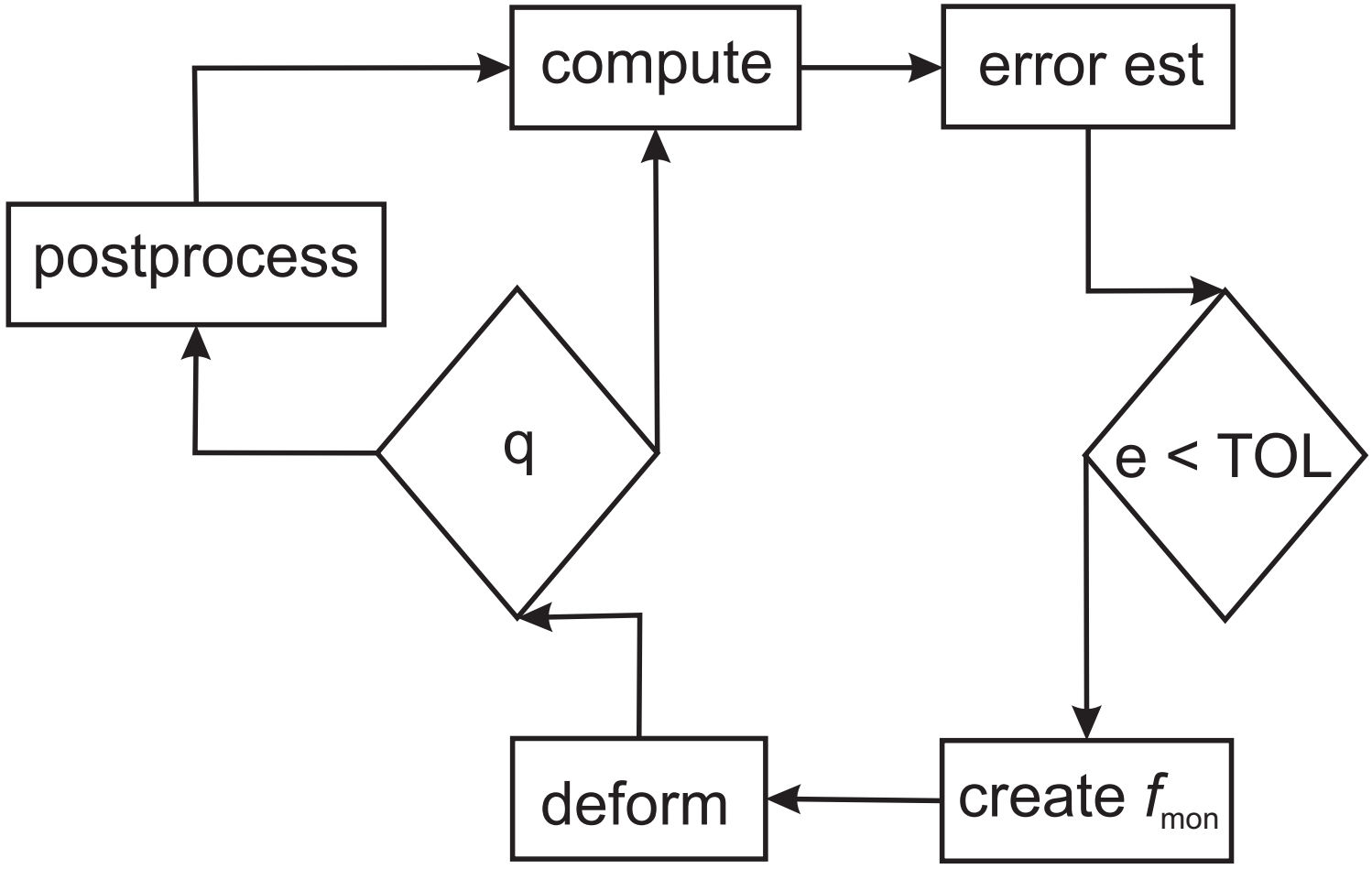
$\Phi(x) = \varphi(x,1)$

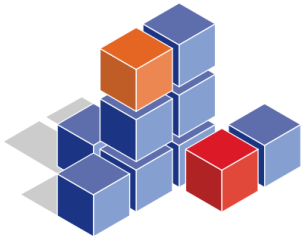
ENDDO

END Deformation



Overall algorithm



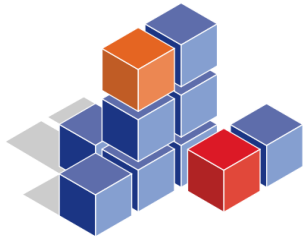


Practical aspects

monitor function:

- smooth $e(x)$
- $f_{\mathrm{mon}}(x) := -\ln(e(x))$

(topic of current research)



Practical aspects

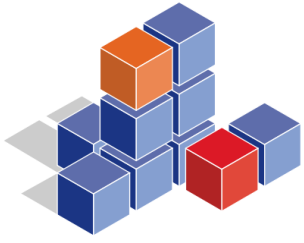
monitor function:

- smooth $e(x)$
- $f_{\mathrm{mon}}(x) := -\ln(e(x))$

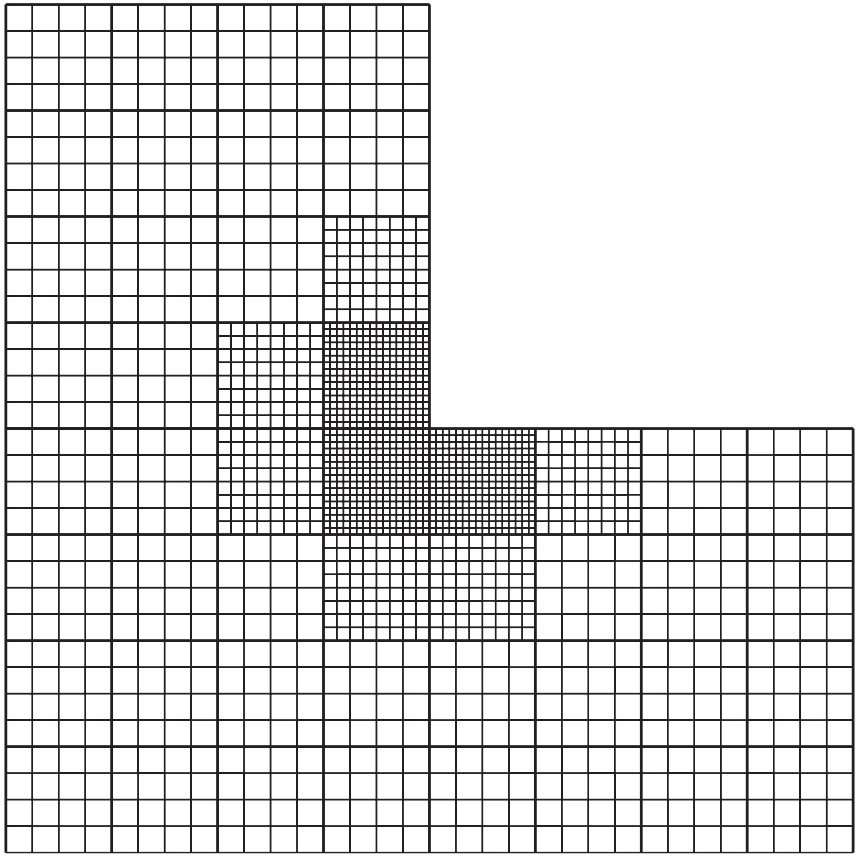
(topic of current research)

postprocessing:

- grid smoothing (Laplace, Umbrella)
- local grid optimisation



Test problem: L-domain, 48 macros

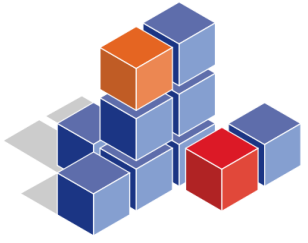


Poisson equation

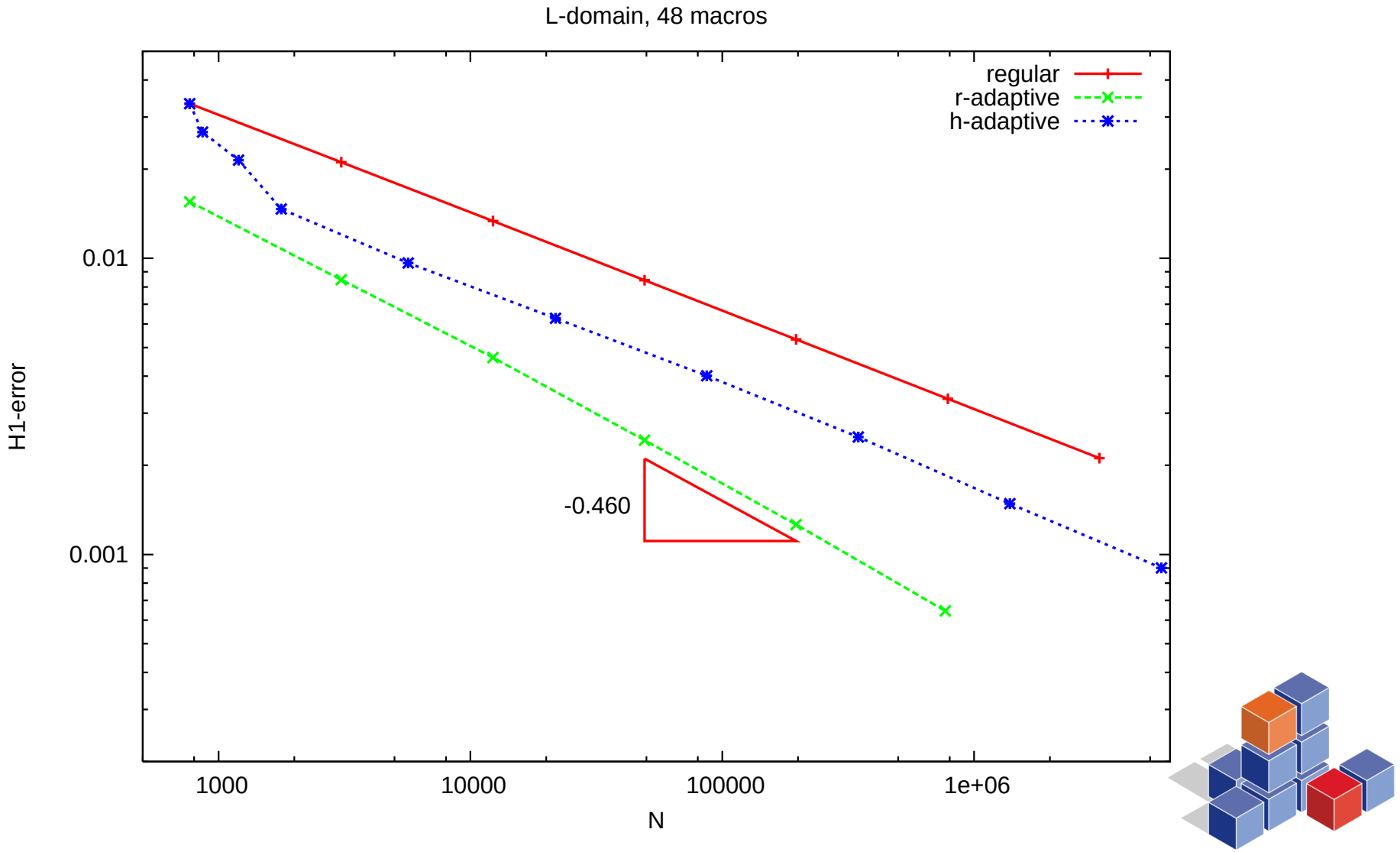
$$\Omega = [-0.5, 0.5]^2 / [0, 0.5]^2$$

$$u(r, \varphi) = r^{2/3} \sin(2/3\varphi)$$

gradient error by SPR



Test problem: r -vs. h-adaptivity

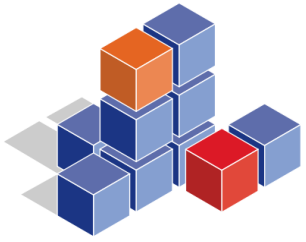


r- vs. h-adaptivity

h-adaptivity

suboptimal meshes

computation “cheap”



r- vs. h-adaptivity

h-adaptivity

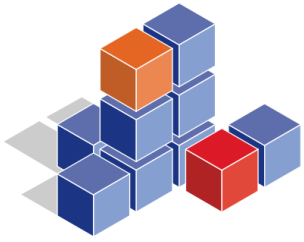
suboptimal meshes

computation “cheap”

r-adaptivity

(almost) optimal meshes

computation “expensive”: all iteration steps on fine grid



r-h-adaptivity 1

rh1

Computation + error estimation

DO $i = 1, n_{\max}$

deformation + postprocessing

computation + error estimation

if ($e_{n-1} < c\, e_n$) EXIT LOOP

END DO

DO

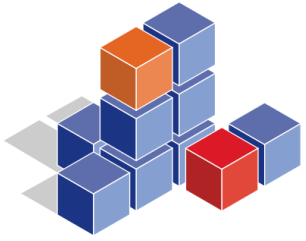
mark macros (fixed fraction strategy)

refine macros

computation + error estimation

END DO

END rh1



r-h-adaptivity 2

rh2

computation + error estimation

DO

DO $i = 1, n_{\max}$

deformation + postprocessing

computation + error estimation

if ($e_{n-1} < c\, e_n$) EXIT LOOP

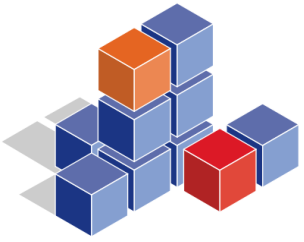
END DO

refine all macros

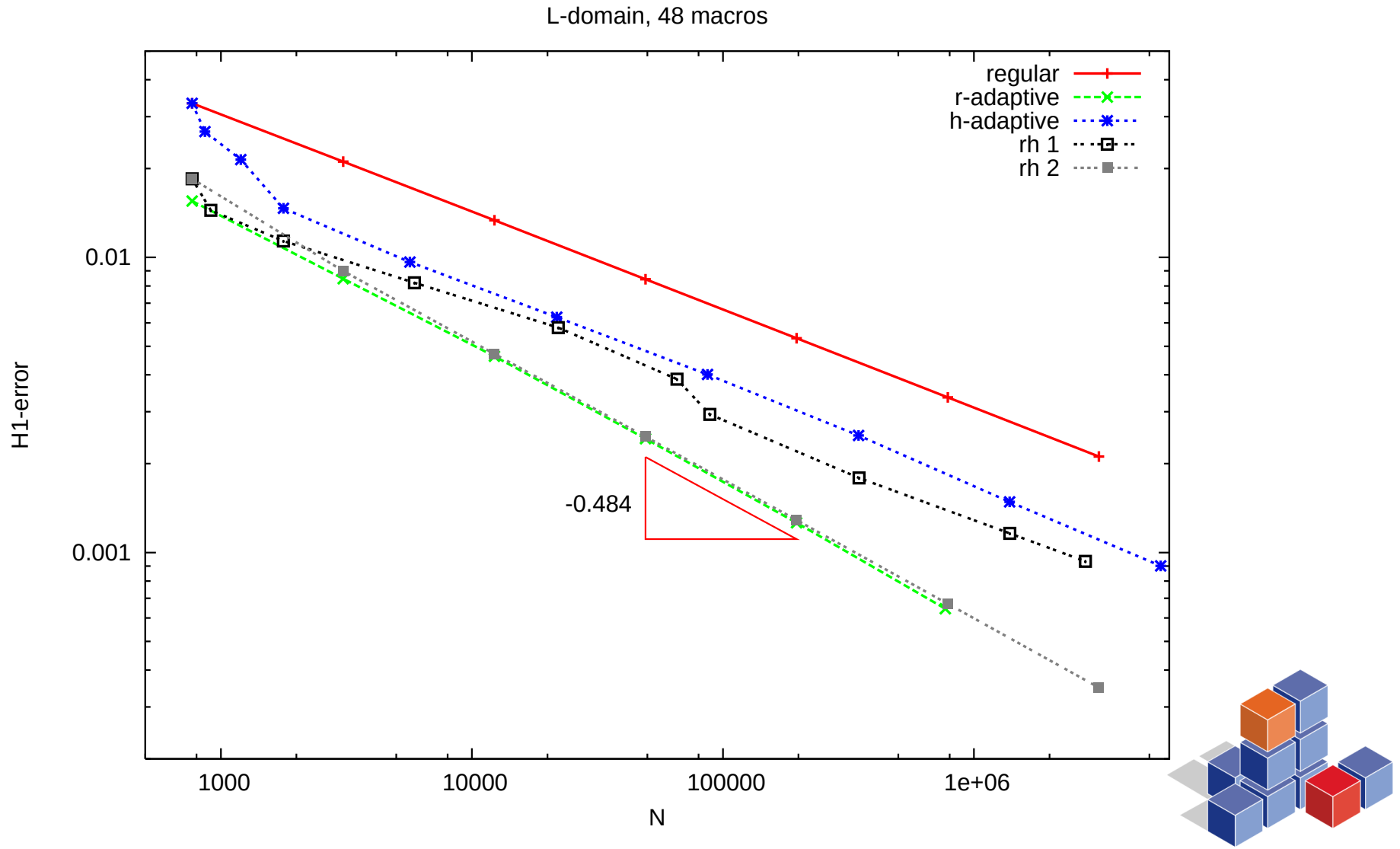
computation + error estimation

END DO

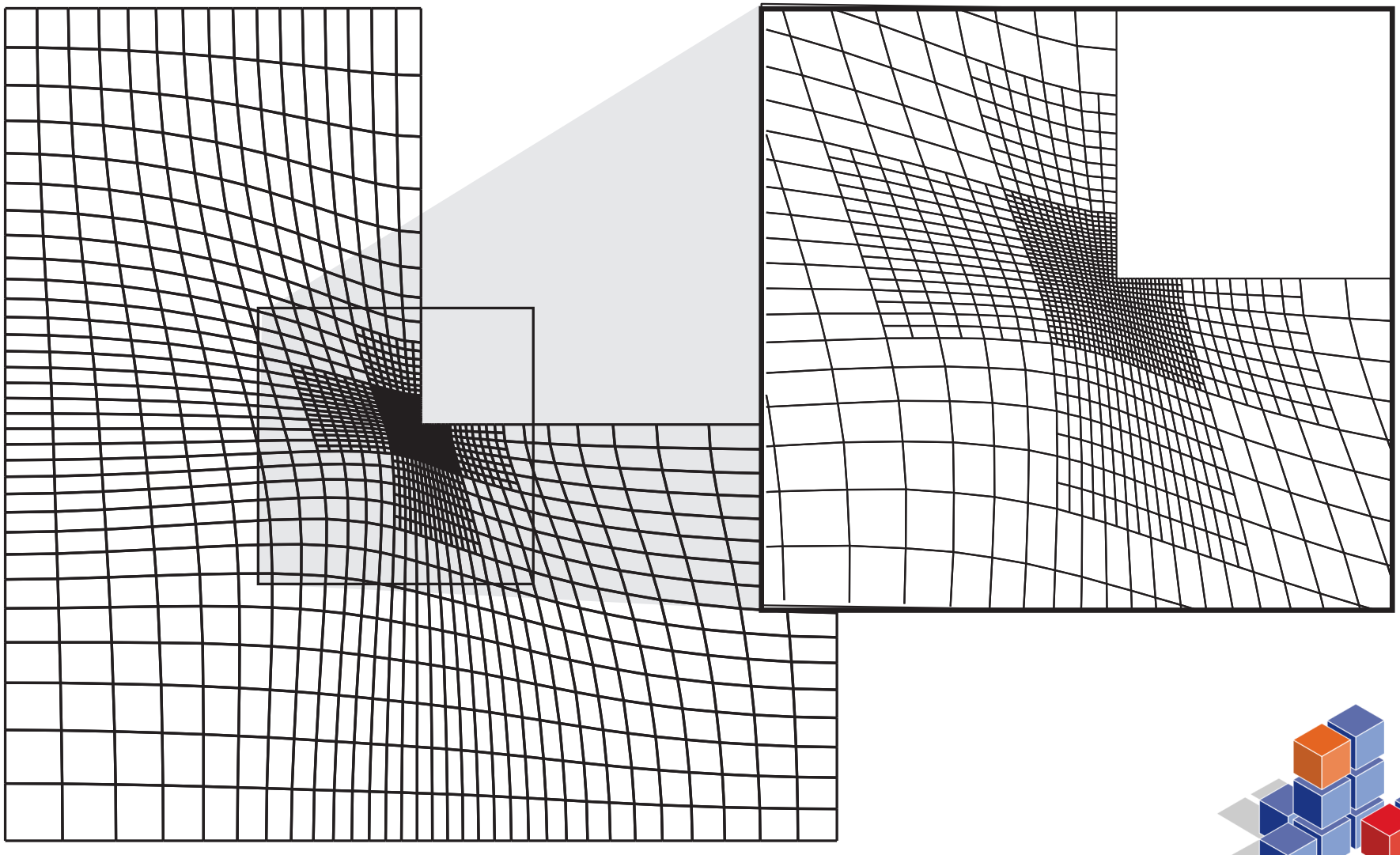
END rh2



Test problem: r-h-adaptivity

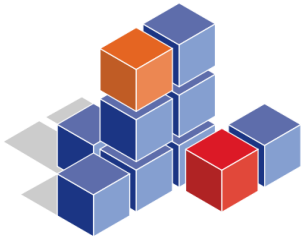


Test problem: r-h-adaptivity



Conclusion and discussion

- HPC: locally structured meshes for hardware-oriented numerical software
- HPC: locally structured meshes for emerging fast hardware
- fast and robust deformation method
- h- and r-adaptivity for HPC
- combination: r-h-adaptivity
- test example: L-domain



Thank you for your attention!