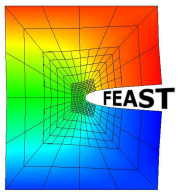
technische universität
dortmund

Efficient Multilevel Solvers and High Performance Computing Techniques for the Finite Element Simulation of Large-Scale Elasticity Problems

Hilmar Wobker[1]

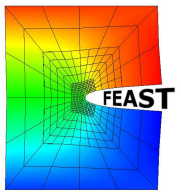
[1]Institute of Applied Mathematics and Numerics, TU Dortmund, Germany
email: hilmar.wobker@math.tu-dortmund.de

January 15, 2010

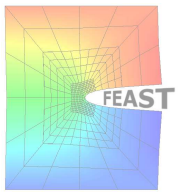


Overview

1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...



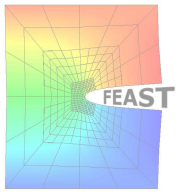
Overview



Introduction/Motivation

Efficiency of iterative linear solvers influenced by

- material parameters,
- nonlinear effects,
- the shape of the geometry,
- the size and the quality of the underlying computational mesh,
- algorithmic parameters, and
- the number of processors in a parallel computing system.



Introduction/Motivation

technische universität dortmund

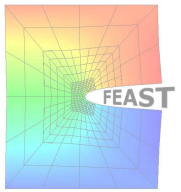
Three aspects of efficiency:

- numerical efficiency: amount of work to achieve a desired goal; convergence rate and robustness
- processor efficiency: ability to exploit the full capacity of modern hardware
- parallel efficiency: communication vs. computation, scalability

Hardware-oriented numerics:
achieve good efficiency in all three aspects

$\rightarrow$ difficult to realise due to conflicting demands

$\rightarrow$ our attempt: FEAST



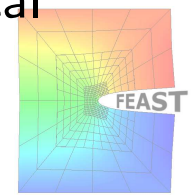
Introduction/Motivation

On the one hand:

- hardware-oriented library is tedious to develop and to maintain (multi-core architectures, vector processors, Cache-sizes, co-processors (GPUs, Cell), latency/speed of interconnects, compiler issues, ...)
- mandatory for high efficiency: a priori known data layout, especially of FE matrices

On the other hand:

- different physical applications $\Rightarrow$ different matrix structures (heat transfer, elasticity, Navier-Stokes, Fluid-Solid-Interaction, 2D / 3D, etc.)
- 'application programmers' don't want to be bothered with technical details



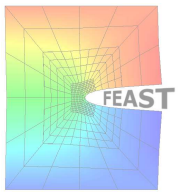
Introduction/Motivation

Remedy: Realise *multivariate* operations (operators) as a series (set) of *scalar* operations (operators)

Advantages:

- facilitates strict separation of low-level kernel/library functionalities and high-level application code
- 'kernel programmers' can concentrate on the scalar equation case, do not have to heed all the physical applications
- 'application programmers' can concentrate on the application, do not have to heed processor architectures, MPI communication, matrix fill-in patterns, ...
- efficiency of the scalar kernel routines automatically available for multivariate problems
- kernel enhancements (new finite element, GPU solvers, ...) usable without any changes of the application code

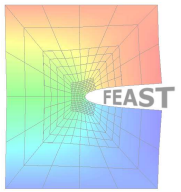
$\rightarrow$ prototypically demonstrated with FEASTsolid



Overview

technische universität dortmund

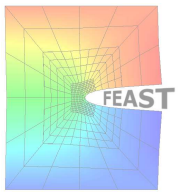
1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...



Overview

technische universität dortmund

1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...

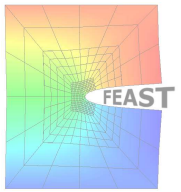


FEAST's Meshing Concept

technische universität dortmund

Memory wall problem:
'data moving $\gg$ data processing'

- improvement of hardware characteristics per year (1990–2004):
  - processor peak performance: 60 %
  - memory bandwidth: 30 %
  - memory latency: 5.5 %
- 1992: 1 memory access $\approx$ 1 FLOP
  2004: 1 memory access $\approx$ 100 FLOPs
- 'memory gap' is still broadening
- unstructured meshes $\Rightarrow$ indirect adressing
  $\Rightarrow$ expensive memory access
- plus: low arithmetic intensity (sparse matrices)
  $\Rightarrow$ poor processor efficiency (low MFLOP/s rates)

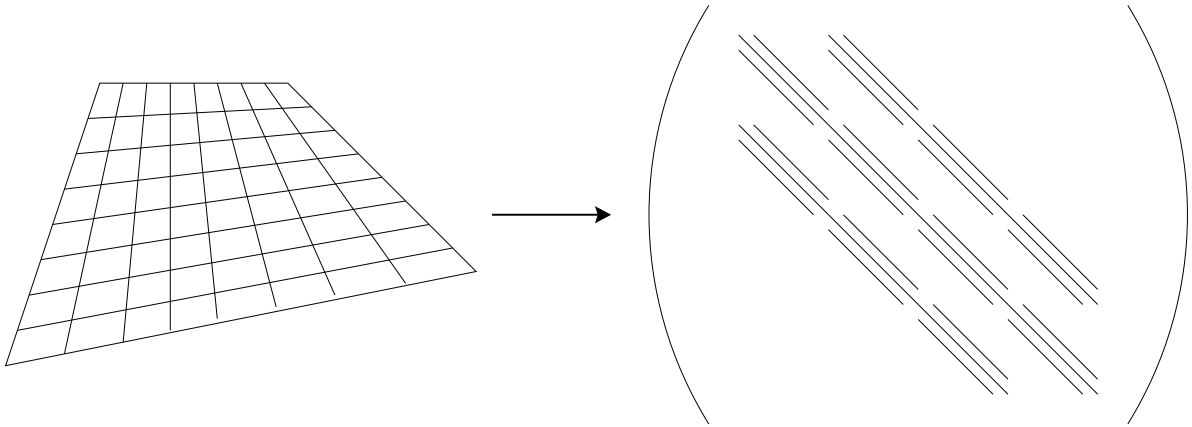


FEAST's Meshing Concept

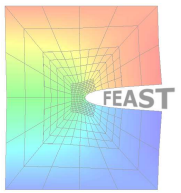
technische universität dortmund

Remedy: use structured data with high spatial and temporal locality

- FEAST uses generalised tensor product meshes



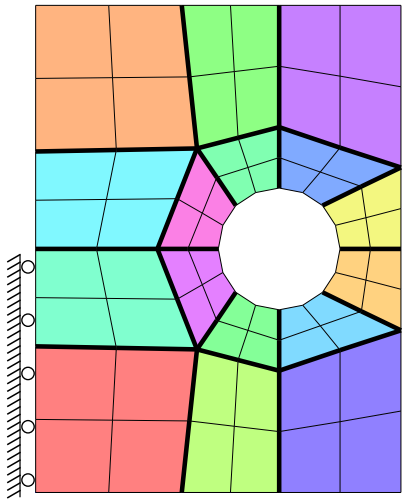
- rowwise numbering
  $\Rightarrow$ exactly 9 matrix bands for bilinear elements
- direct adressing, caching
- optimised Linear Algebra routines (SparseBandedBLAS)

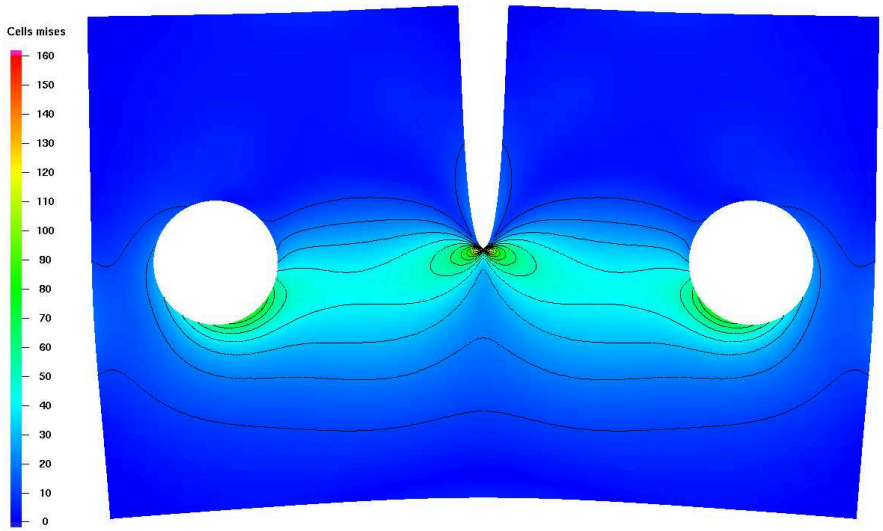


FEAST's Meshing Concept

technische universität dortmund

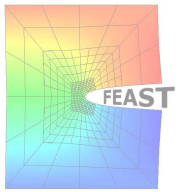
- more complex (unstructured) domains by joining several (structured) TP meshes
- local matrices + appropriate border data exchanges
  $\Rightarrow$ 'virtual' global matrix





Here:

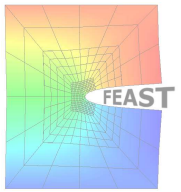
- 64 TP meshes (=64 local matrices), each refined 10x
- distributed over 16 processors
- $\Rightarrow 1.34 \cdot 10^8$ degrees of freedom in total



Overview

technische universität dortmund

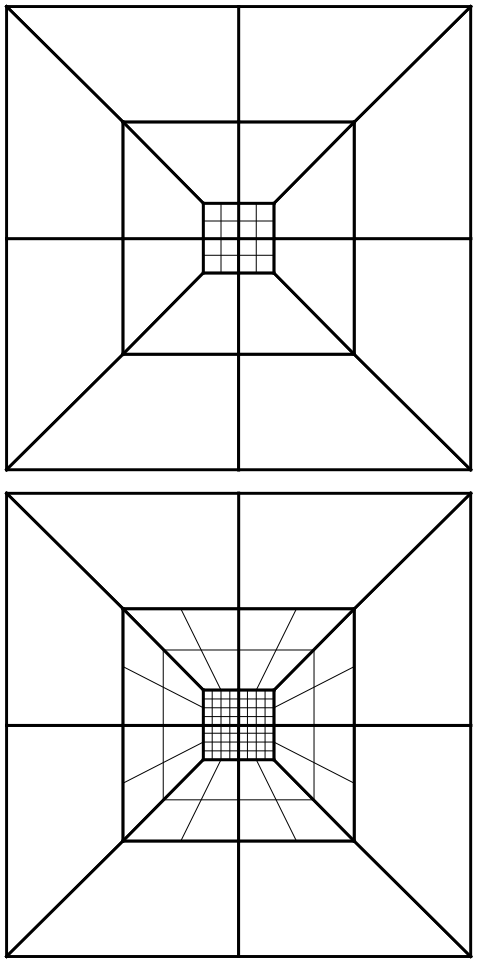
1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...



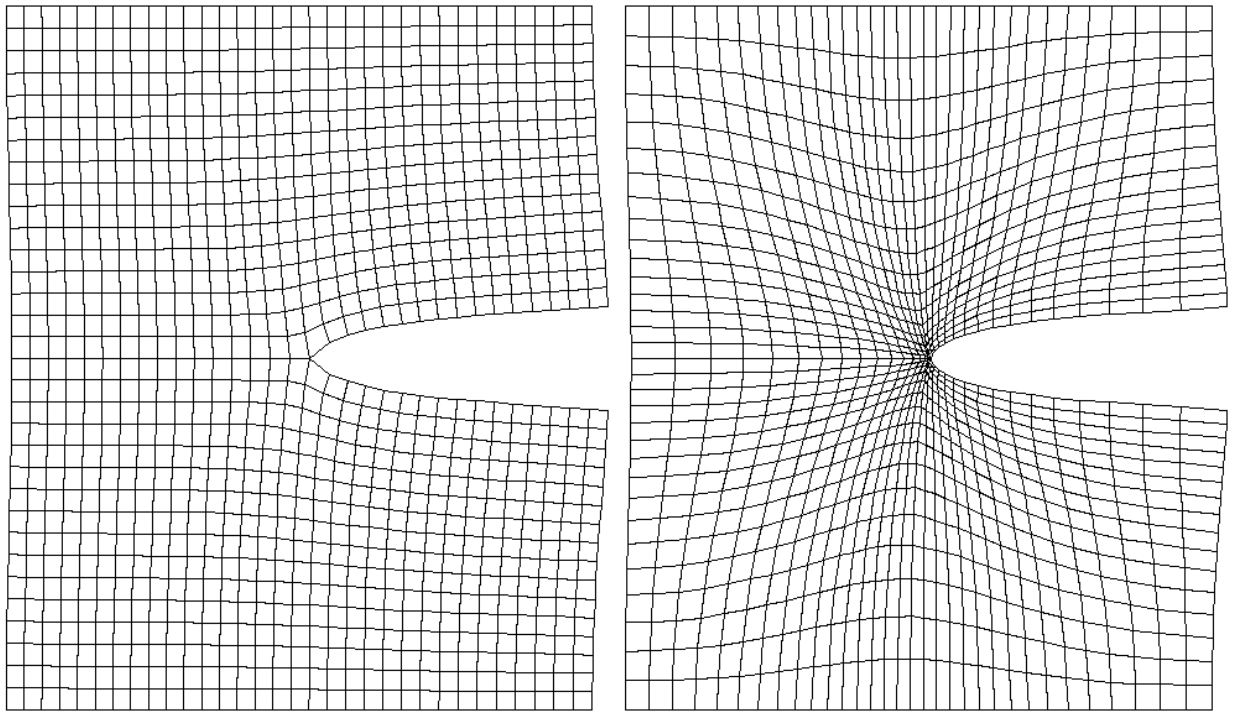
FEAST's Adaptivity Concept

technische universität dortmund

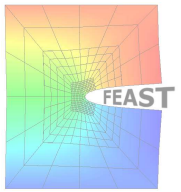
Patch-wise Hanging Nodes:



Mesh Deformation:

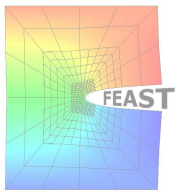


TP property fulfilled!



Overview

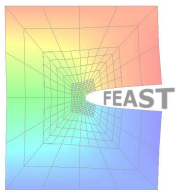
technische universität dortmund



Parallel vs. Numerical Efficiency

- improvement of hardware characteristics per year (1990–2004):
  - processor peak performance: 60 %
  - network technology (latency, bandwidth): 30 %
- physical constraint: speed of light
- 2004: 1 inter-processor communication $\approx$ 4000 FLOPs
  2020: 1 inter-processor communication $\approx$ 670 000 FLOPs
- number of processors in high-end super computers:
  2004: $\approx$ 4000
  2010: $>$ 100 000
- similar to the 'memory wall' problem
- $\Rightarrow$ parallel efficiency determined by amount of data exchange

data locality required for
parallel efficiency



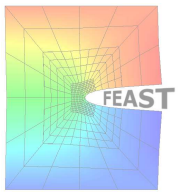
Parallel vs. Numerical Efficiency

On the other hand:

- elliptic problems: solution in a point is influenced by all boundary values
- information has to 'travel' at least once through the whole grid
- fast global data exchange necessary for good iterative convergence rates

global information required for numerical efficiency

- conflicting demands: data locality vs. fast global data exchange
- parallel and numerical efficiency hard to combine



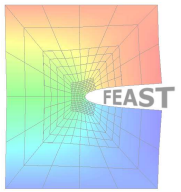
Parallel vs. Numerical Efficiency

Requirements of parallel solution methods:

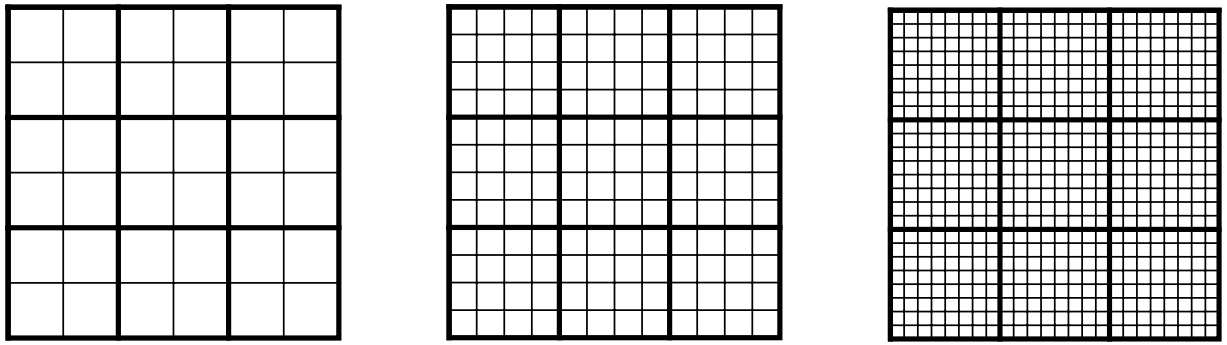
- low inter-processor communication
- high processor loads
- good scalability
- good convergence behaviour
- robustness with respect to complicated geometries, mesh refinement level, mesh irregularities, and number/size of subdomains

Suitable solver concepts for elliptic problems:

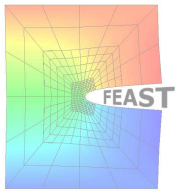
- multigrid methods (MG)
- domain decomposition methods (DD)



Multigrid



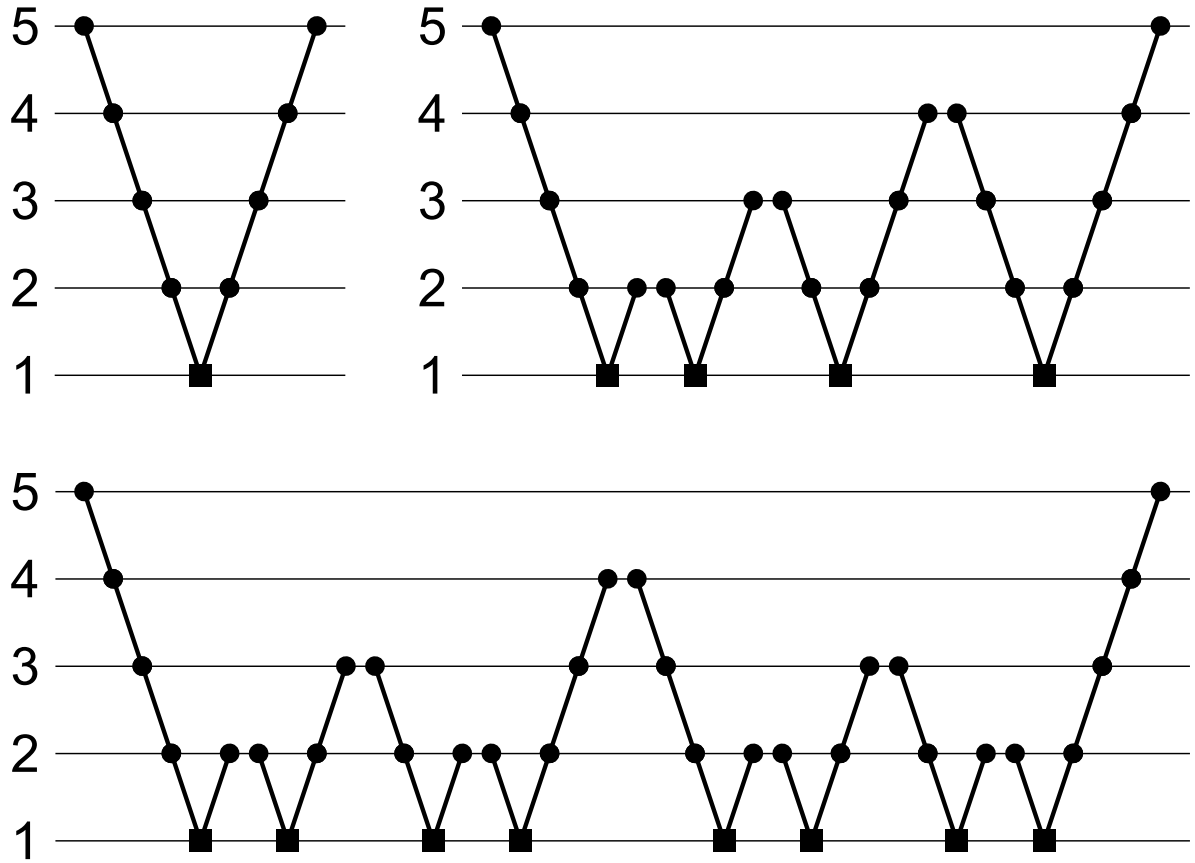
- use a hierarchy of nested grids to solve $\mathbf{Ax} = \mathbf{b}$
- basic components:
  - smoothing (level l): $\mathbf{x}^{(l)} \leftarrow \mathbf{x}^{(l)} + \omega \mathbf{S}^{(l)}(\mathbf{b}^{(l)} - \mathbf{A}^{(l)}\mathbf{x}^{(l)})$
  - grid transfer: restriction $(l \rightarrow l-1)$, prolongation $(l-1 \rightarrow l)$
  - coarse grid solving: $\mathbf{x}^{(1)} = (\mathbf{A}^{(1)})^{-1}\mathbf{b}^{(1)}$
- exploiting smoothing property of elementary methods like Jacobi or Gauß-Seidel
- convergence behaviour independent of the refinement level
- runtime $O(\#\text{DOF})$

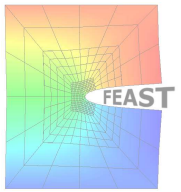


Multigrid

Possibilities to traverse the grid hierarchy:

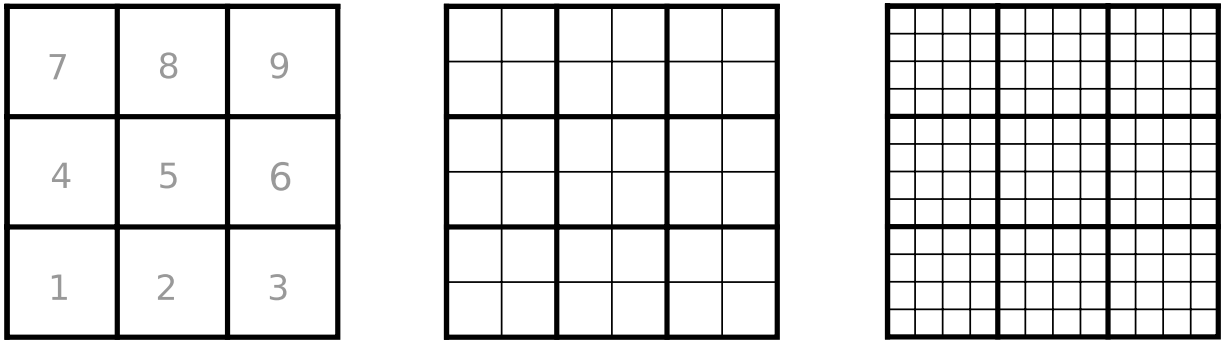
- V-, F- and W-cycle
- arithmetic costs vs. convergence speed / robustness



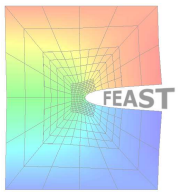


Block-smoothed Multigrid

technische universität dortmund



- smoothing operation highly recursive (Gauß-Seidel, ILU)
- bad parallelisation potential
- remedy: relax the smoothing operation by applying it *block-wise* (additively, block-Jacobi)
- use *minimal overlap* (next slide)
- number of vertices n, number of subdomains M:
  $M \to 1$: standard multigrid with selected smoother
  $M \to n$: standard multigrid with Jacobi smoother



Minimal Overlap

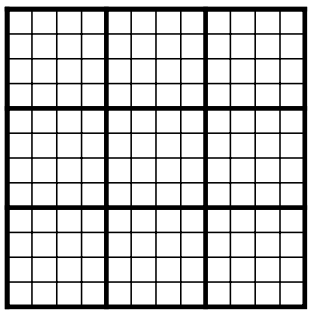
technische universität dortmund

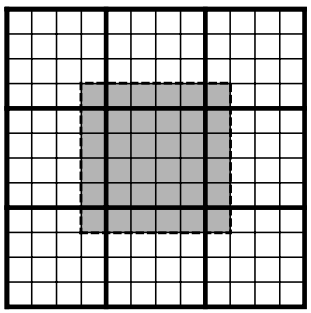
Switching between global layer and local layer:

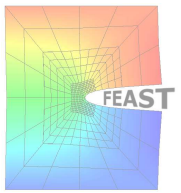
- global set of vertices on level l: $\mathcal{V}^{(l)}$
- set of vertices of i-th subdomain: $\mathcal{V}_i^{(l)}$
- local index of vertex k in subdomain i: $\mathrm{loc}_i^{(l)}(k)$
- prolongation matrix $\mathbf{P}_i^{(l)}$:

$$(\mathbf{x}^{(l)})_k = (\mathbf{P}_i^{(l)}\mathbf{x}_i^{(l)})_k := \begin{cases} (\mathbf{x}_i^{(l)})_{\mathrm{loc}_i^{(l)}(k)} & k \in \mathcal{V}_i^{(l)} \\ 0 & k \in \mathcal{V}^{(l)} \setminus \mathcal{V}_i^{(l)} \end{cases}$$

- restriction matrix: $\mathbf{R}_i^{(l)} := (\mathbf{P}_i^{(l)})^T$
- local matrix: $\mathbf{A}_i^{(l)} := \mathbf{R}_i^{(l)}\mathbf{A}^{(l)}\mathbf{P}_i^{(l)}$
- interpretation: 'ghost cells', extended Dirichlet boundaries





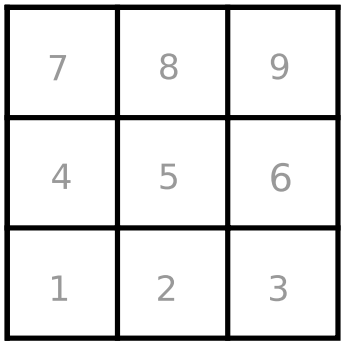


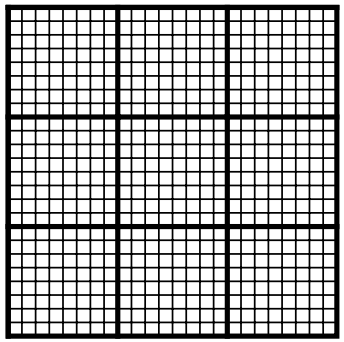
Domain Decomposition

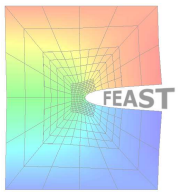
- classical 'divide & conquer' strategy:
  replace the solution of one large system (global layer) by the solution of several small systems (local layer)
- interpret DD as preconditioner $\tilde{\mathbf{A}}$:

$$\mathbf{x} \leftarrow \mathbf{x} + \omega \tilde{\mathbf{A}}(\mathbf{b} - \mathbf{A}\mathbf{x})$$

- global coupling typically via Krylov space method
- two classes:
  - non-overlapping methods (substructuring / Schur complement), extra interface problem
  - overlapping Schwarz methods



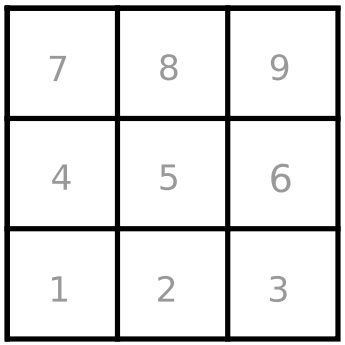


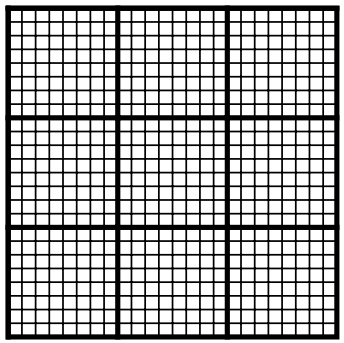


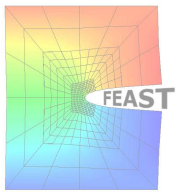
Overlapping Schwarz Methods

technische universität dortmund

- one-level Schwarz method: dependence on number of subdomains, size of the overlap
- add coarse grid problem: two-level Schwarz method
- generalisation: multilevel Schwarz methods (MLDD)
- MLDD methods can be
  - purely additive (additive within one level / additive between levels)
  - purely multiplicative (multiplicative/multiplicative)
  - hybrid (additive/multiplicative or multiplicative/additive)
- FEAST uses a hybrid MLDD method with minimal overlap that is
  - additive within one level
  - multiplicative between levels







FEAST's Solution Method

technische universität dortmund

- one iteration (multiplicative part between levels):

$$\mathbf{x}^{(L)} \leftarrow \mathbf{x}^{(L)} + \tilde{\mathbf{A}}^{(L)}(\mathbf{b}^{(L)} - \mathbf{A}^{(L)}\mathbf{x}^{(L)})$$

$$l = L-1, \ldots, 2:$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{R}^{(l)}(\mathbf{b}^{(l+1)} - \mathbf{A}^{(l+1)}\mathbf{x}^{(l+1)}), \quad \mathbf{x}^{(l)} \leftarrow \tilde{\mathbf{A}}^{(l)}\mathbf{b}^{(l)}$$

$$l = 1:$$

$$\mathbf{b}^{(1)} \leftarrow \mathbf{R}^{(2)}(\mathbf{b}^{(2)} - \mathbf{A}^{(2)}\mathbf{x}^{(2)}), \quad \mathbf{x}^{(1)} \leftarrow \left(\mathbf{A}^{(1)}\right)^{-1}\mathbf{b}^{(1)}$$

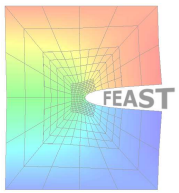
$$l = 2, \ldots, L:$$

$$\mathbf{x}^{(l)} \leftarrow \mathbf{x}^{(l)} + \mathbf{P}^{(l)}\mathbf{x}^{(l-1)}, \quad \mathbf{x}^{(l)} \leftarrow \mathbf{x}^{(l)} + \tilde{\mathbf{A}}^{(l)}(\mathbf{b}^{(l)} - \mathbf{A}^{(l)}\mathbf{x}^{(l)})$$

- connection between global and local layer (additive part within one level):

$$\tilde{\mathbf{A}}^{(l)} := \widetilde{\sum_{i=1}^{M}} \mathbf{P}_i^{(l)} \tilde{\mathbf{A}}_i^{(l)} \mathbf{R}_i^{(l)}.$$

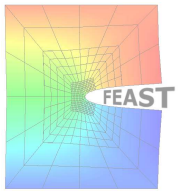
- *local preconditioner* $\tilde{\mathbf{A}}_i^{(l)}$ for the local matrix $\mathbf{A}_i^{(l)}$ (slide after next)



FEAST's Solution Method

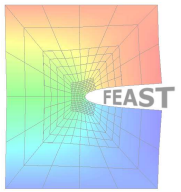
Why minimal overlap?

- sufficient for robust convergence behaviour of multilevel DD
  $\rightarrow$ numerical efficiency
- local submeshes preserve tensor product property
  $\rightarrow$ processor efficiency
- minimal amount of data exchange between subdomains
  $\rightarrow$ parallel efficiency
- implementation and data structures greatly simplified



The Local Preconditioner

- two strategies to realise the local preconditioner $\tilde{\mathbf{A}}_i^{(l)}$:
  1. apply one step of an elementary iterative scheme (Jacobi, Gauß-Seidel, ...)
  2. apply some iterative or direct solution method to (approximately) solve the local system
- first strategy: MLDD coincides with block-smoothed MG ($\tilde{\mathbf{A}}_i^{(l)}$ being the local smoother)
- $\Rightarrow$ MLDD generalisation of MG

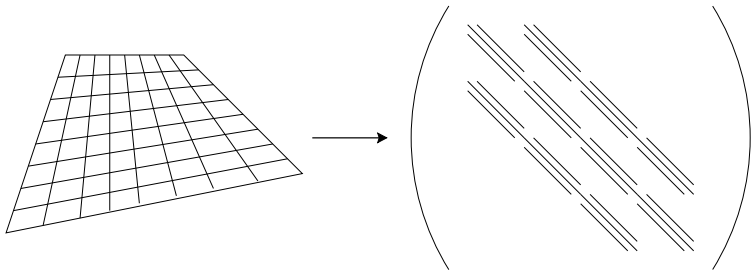


The Local Preconditioner

First strategy:

- Notation of the local 9-band matrix (omitting superscript l):

$$\mathbf{A}_i = (\mathbf{L}_i^L + \mathbf{L}_i^C + \mathbf{L}_i^U) + (\mathbf{C}_i^L + \mathbf{C}_i^C + \mathbf{C}_i^U) + (\mathbf{U}_i^L + \mathbf{U}_i^C + \mathbf{U}_i^U)$$



- some local smoothers:

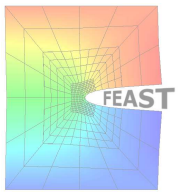
$\tilde{\mathbf{A}}_i^{\text{Jacobi}} := (\mathbf{C}_i^C)^{-1}$

$\tilde{\mathbf{A}}_i^{\text{GS}} := (\mathbf{L}_i^L + \mathbf{L}_i^C + \mathbf{L}_i^U + \mathbf{C}_i^L + \mathbf{C}_i^C)^{-1}$

$\tilde{\mathbf{A}}_i^{\text{TriGS}} := (\mathbf{L}_i^L + \mathbf{L}_i^C + \mathbf{L}_i^U + \mathbf{C}_i^L + \mathbf{C}_i^C + \mathbf{C}_i^U)^{-1}$

$\tilde{\mathbf{A}}_i^{\text{MTriGS}}$: $\tilde{\mathbf{A}}_i^{\text{TriGS}}$ applied to column-wise numbered grid

$\tilde{\mathbf{A}}_i^{\text{ADiTriGS}}$: alternating application of $\tilde{\mathbf{A}}_i^{\text{TriGS}}$ *and* $\tilde{\mathbf{A}}_i^{\text{MTriGS}}$



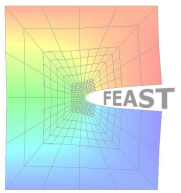
The Local Preconditioner

technische universität dortmund

Second strategy:

- solve local systems $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}_i$ (approximately) via
  - *direct solver* if the system is not too large (#DOF$< 20\,000$)
  - *multigrid method*, otherwise
- local MG uses the same smoothers as block-smoothed (global) MG (see previous slide)
- 'automatically toggle' between MG and direct solver via *truncated multigrid method*
- typical local problem size: #DOF$\approx 10^6$
  $\Rightarrow$ local multigrid mandatory

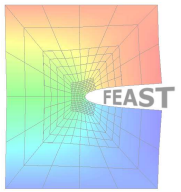
Using local multigrid within global MLDD is one of the core ideas of FEAST's solution method!



The Local Preconditioner

Comparison of the two strategies (black dot = favoured strategy):

storage requirements	1	○ ● ○ ○	2
flexibility	1	○ ○ ● ○	2
'black box'character	1	○ ● ○ ○	2
arithmetic costs	1	● ○ ○ ○	2
communication vs. computation	1	○ ○ ○ ●	2
influence on global iteration	1	○ ○ ○ ●	2
load balancing	1	● ○ ○ ○	2
using co-processors	1	○ ○ ○ ●	2



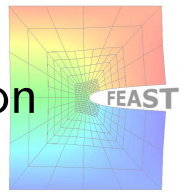
The ScaRC Concept

Basic idea of FEAST's solver concept ScaRC (Scalable Recursive Clustering):

- allow both strategies at the same time
- choose local solver components adaptively according to the 'local situation'

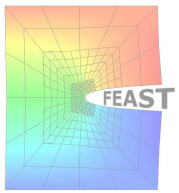
Advantages of ScaRC:

- convergence rates independent of refinement level
- convergence rates independent of number of subdomains (in case of only mild macro anisotropies)
- 'simple' local solvers/smoothers on 'simple' subdomains, 'strong' ones on 'difficult' subdomains
- good balance of computational costs and convergence behaviour
- hide local irregularities from the global solver
- minimise number of global iterations and amount of communication
- high processor loads due to local tensor product grids



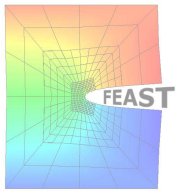
Disadvantages of ScaRC:

- convergence rates dependent on number of subdomains in case of stronger macro anisotropies (block-Jacobi character)
  - can be alleviated by enhancing the global multilevel solver with Krylov space methods
  - further idea: 'Recursive Clustering' (3-layer-ScaRC, merging subdomains)
- bad ratio computation/communication on coarser grid levels (inherent to multilevel approaches!)
- difficult to realise: 'automatic adaptation system' for local solver components
- even more difficult: dynamic load balancing
- local multigrid solvers are nonconforming (next slides)



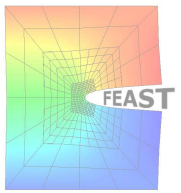
Notations

- first strategy: 1-layer-ScaRC
  (multigrid scheme only on global layer)
- second strategy: 2-layer-ScaRC
  (multigrid schemes on global *and* local layer)
- short notation
  - 1-layer-ScaRC:
    `MG__JAC__D`
    `MG(1e-6,V44,0.7)__JAC__D`
  - 2-layer-ScaRC:
    `MG__MG-ADI-D__D`
    `MG(1e-6,F22,0.7)__MG(1e-1,V22)-ADI-D__D`
  - '`__`': layer change
    '`D`': direct coarse grid solver
    '`1e-6`': relative stopping criterion ('gain 6 digits')
    '`V22`': cycle type and pre-/postsmoothing steps
    '`0.7`': damping parameter

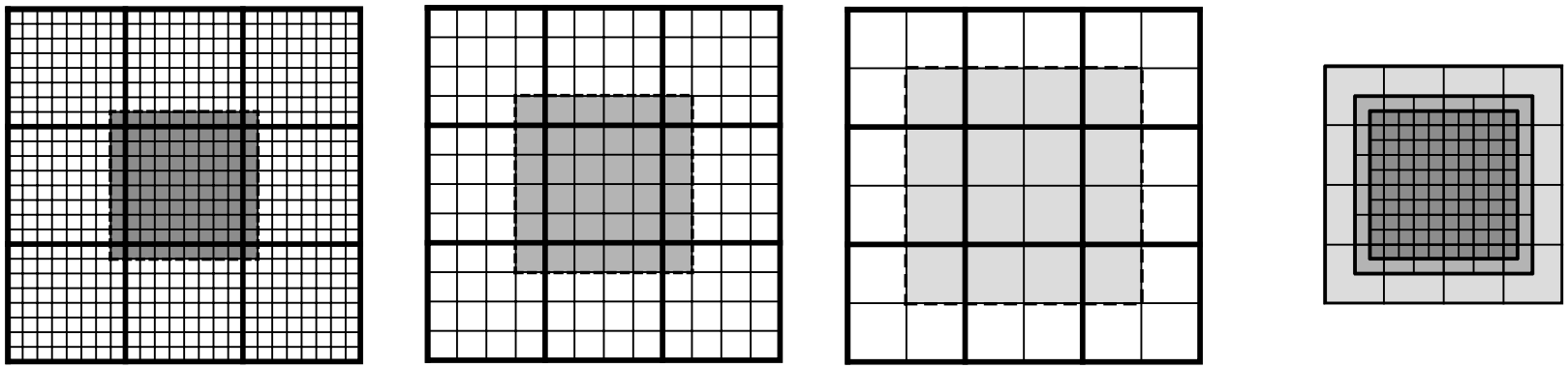


Overview

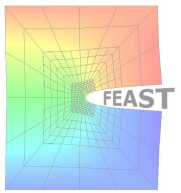
technische universität dortmund



Nonconforming Local MG



- minimal overlap $\Rightarrow$ extended Dirichlet boundaries
- local domain size increases with coarser grid levels
- nonnested local grids $\Rightarrow$ nonconforming local multigrid method
- coarse grid correction not optimal
- (massive) convergence problems of standard multigrid schemes
- loss of level independency



Adaptive Coarse Grid Correction

Remedy:

- adaptively damp the suboptimal coarse grid correction:

$$\mathbf{x}^{(l)} \leftarrow \mathbf{x}^{(l)} + \alpha \mathbf{P}^{(l)} \mathbf{x}^{(l-1)}$$

  (subscripts for local subdomains omitted)

- minimise error in energy norm:

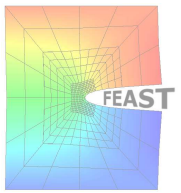
$$(\mathbf{x}^{(l)} - \mathbf{x}^*)^{\mathsf{T}} \mathbf{A}^{(l)} (\mathbf{x}^{(l)} - \mathbf{x}^*)$$

  ($\mathbf{x}^*$ exact solution)

- for symmetric $\mathbf{A}^{(l)}$:

$$\alpha = \frac{{\mathbf{c}^{(l)}}^{\mathsf{T}} \mathbf{d}^{(l)}}{{\mathbf{c}^{(l)}}^{\mathsf{T}} \mathbf{A}^{(l)} \mathbf{d}^{(l)}}$$

  ($\mathbf{c}^{(l)} := \mathbf{P}^{(l)} \mathbf{x}^{(l-1)}$ prolongated coarse grid correction,
  $\mathbf{d}^{(l)} := \mathbf{b}^{(l)} - \mathbf{A}^{(l)} \mathbf{x}^{(l)}$ current defect)



Adaptive Coarse Grid Correction

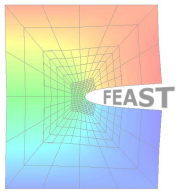
Advantages:

- established technique
- easy to implement
- relatively low computational costs (one matrix vector multiplication, two scalar products)
- damping parameter computed per subdomain:
  - $\alpha \ll 1$ for subdomains that suffer strongly from ext. Dirichl. bound.
  - $\alpha \approx 1$ otherwise

  $\Rightarrow$ overall solution process not unnecessarily impaired in general

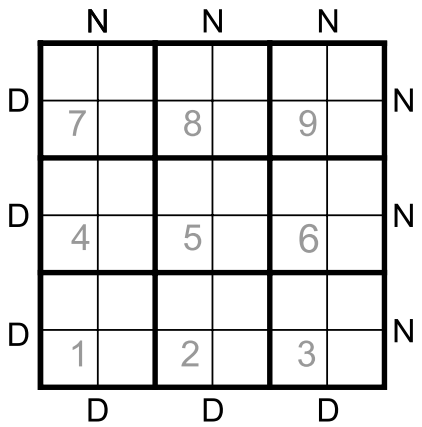
Heuristic used in FEAST before ACGC:

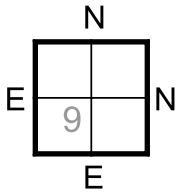
- prolongated values corresponding to subdomain boundary nodes were not fully added (divided by number of incident subdomains)
- equally affects all local solves
- fails in some cases

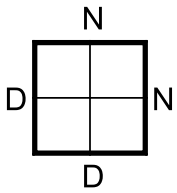


ACGC - Numerical Examples

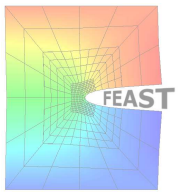
technische universität dortmund







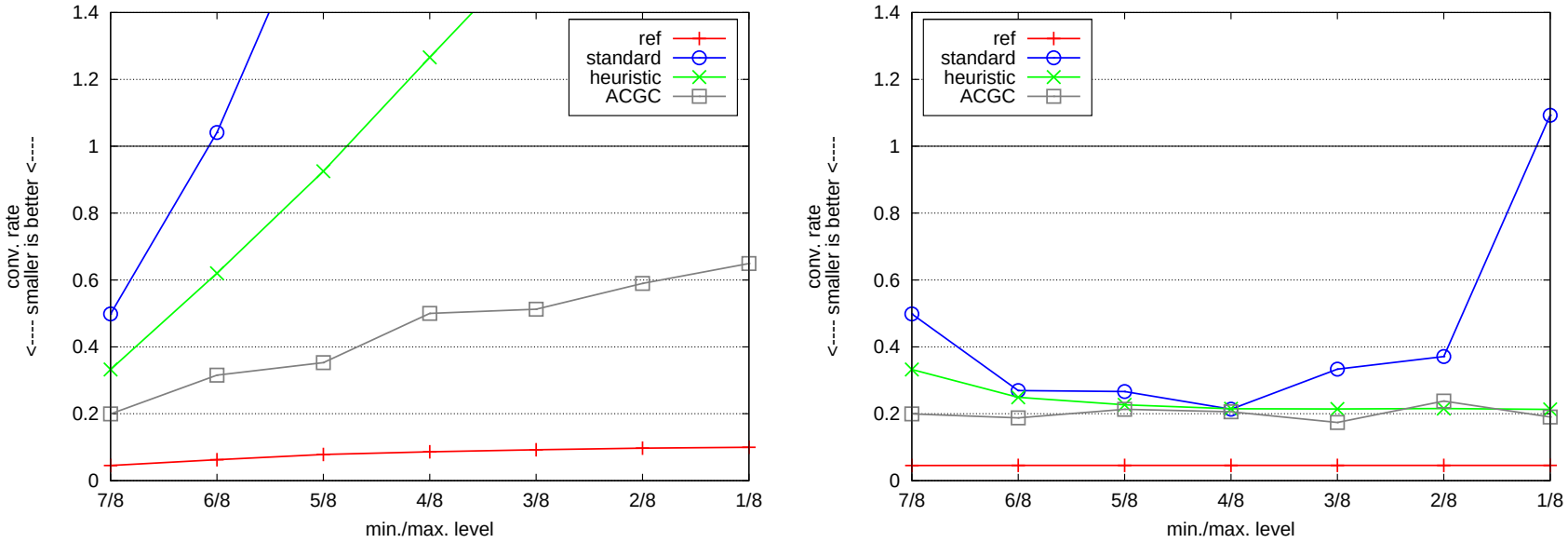
- (outer) geometric boundary conditions: Neumann ('N'), Dirichlet ('D')
- (inner) extended Dirichlet boundaries ('E')
- only consider local solve on top right subdomain (center figure)
- reference computation with standard Dirichlet instead of extended Dirichlet boundary conditions (right figure)
- solve standard Poisson problem $-\Delta u = 1$
- local solver `1e-6,MG([V|F]22,0.7)-JAC-D`



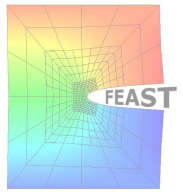
ACGC - Numerical Examples

technische universität dortmund

isotropic subdomain, Jacobi smoother

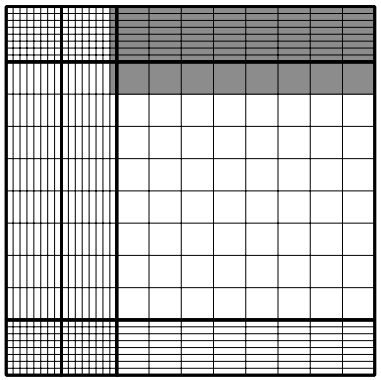
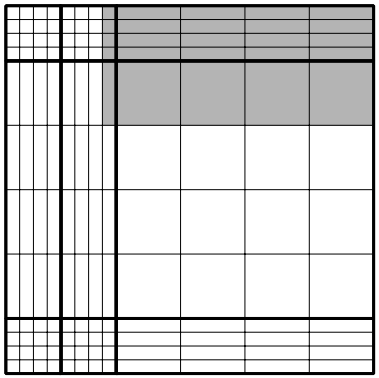
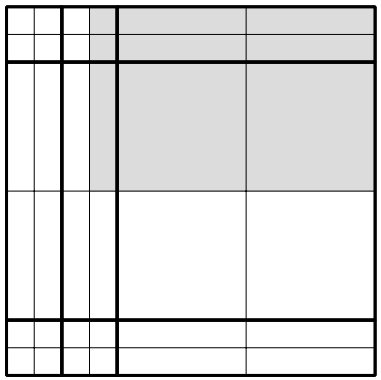
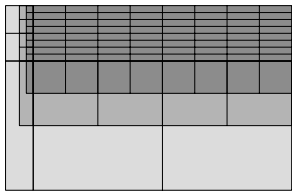


V-cycle (left) and F-cycle (right)

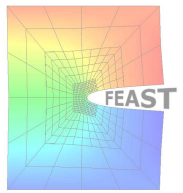


ACGC - Numerical Examples

technische universität dortmund

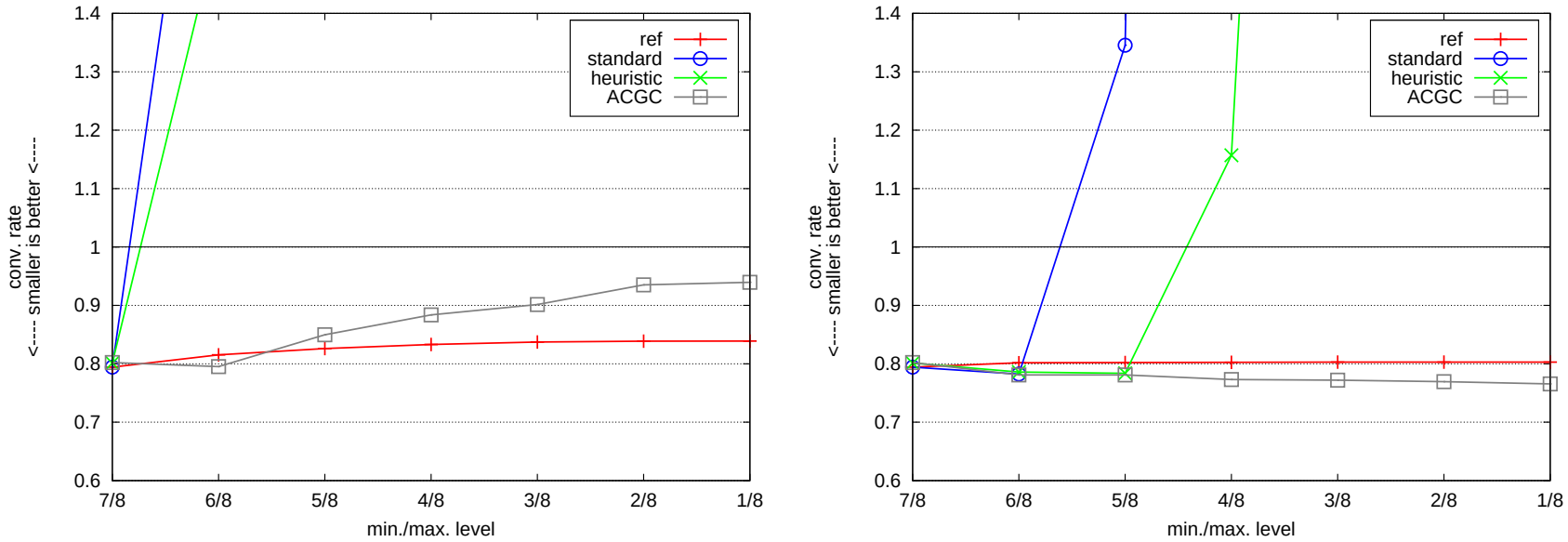
- anisotropic example
- consider top right subdomain again
- local solver `MG(1e-6,[V|F]22,0.6)-JAC-D`



ACGC - Numerical Examples

technische universität dortmund

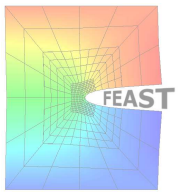
anisotropic subdomain, Jacobi smoother



V-cycle (left) and F-cycle (right)

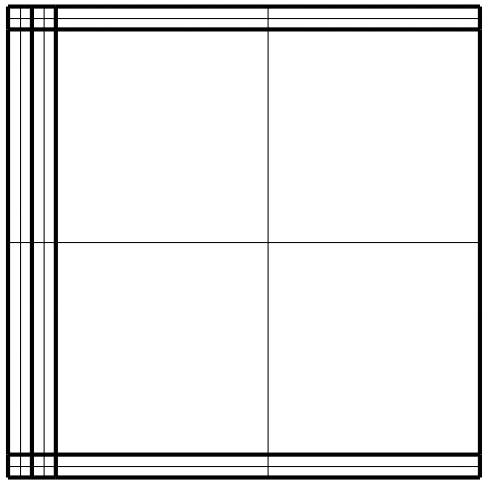
Summary for W-cycle:

- W-cycle with heuristic: converges in most, but not in all cases $\Rightarrow$ not reliable
- standard W-cycle: always converges, but strange oscillations depending on number of grid levels
- W-cycle + ACGC: always converges, no oscillations

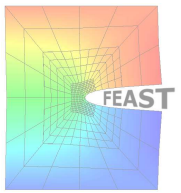


ACGC - Numerical Examples

technische universität dortmund



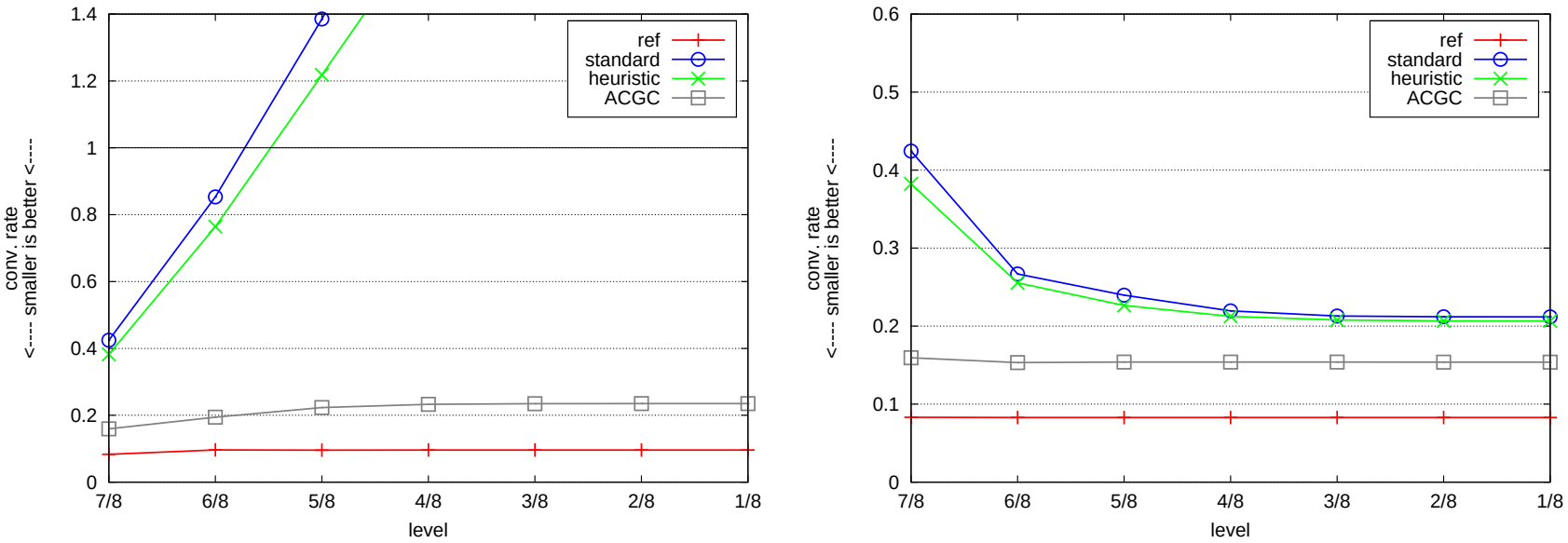
- now: use ADiTriGS instead of Jacobi as local smoother
- increase anisotropies
- consider top right subdomain again
- local solver `MG(1e-6,[V|F]22,1.0)-ADI-D`
- solve standard Poisson problem $-\Delta u = 1$



ACGC - Numerical Examples

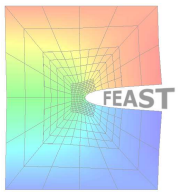
technische universität dortmund

anisotropic subdomain, ADiTriGS smoother



V-cycle (left) and F-cycle (right)

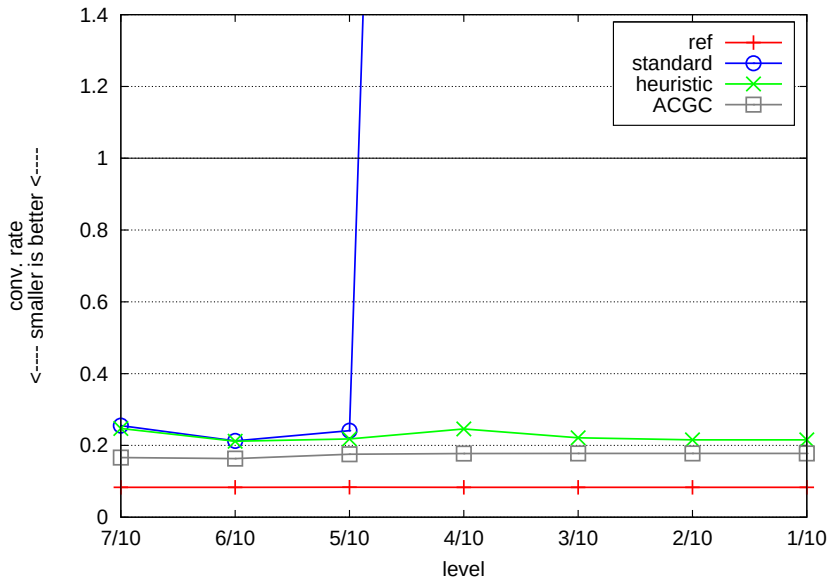
- hence, standard/heuristic F-cycle reliable?
- no! (next slide)

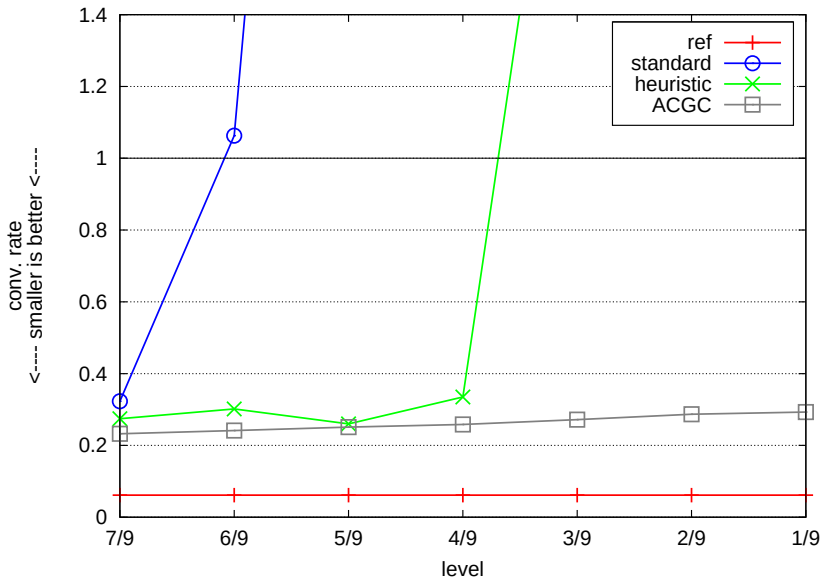


ACGC - Numerical Examples

technische universität dortmund

anisotropic subdomain, ADiTriGS smoother, F-cycle

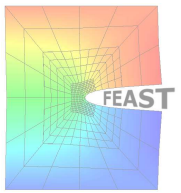




isotr. operator $-\Delta u$ (left), anisotr. operator $-10\partial_{xx}u - \partial_{yy}u$ (right)

Summary for W-cycle:

- standard W-cycle and W-cycle with heuristic: always converge, but strange oscillations depending on number of grid levels
- W-cycle + ACGC: always converges, no oscillations



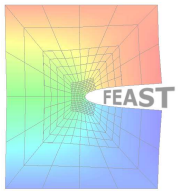
Now: consider arithmetic costs of the global solver scheme

- use total arithmetic efficiency:

$$\text{TAE} = -\frac{\#\text{FLOPs}}{\#\text{DOF} \times \#\text{iter} \times \log_{10}(c)}$$

  $\Rightarrow$ How many FLOPs are needed per DOF to gain one digit?

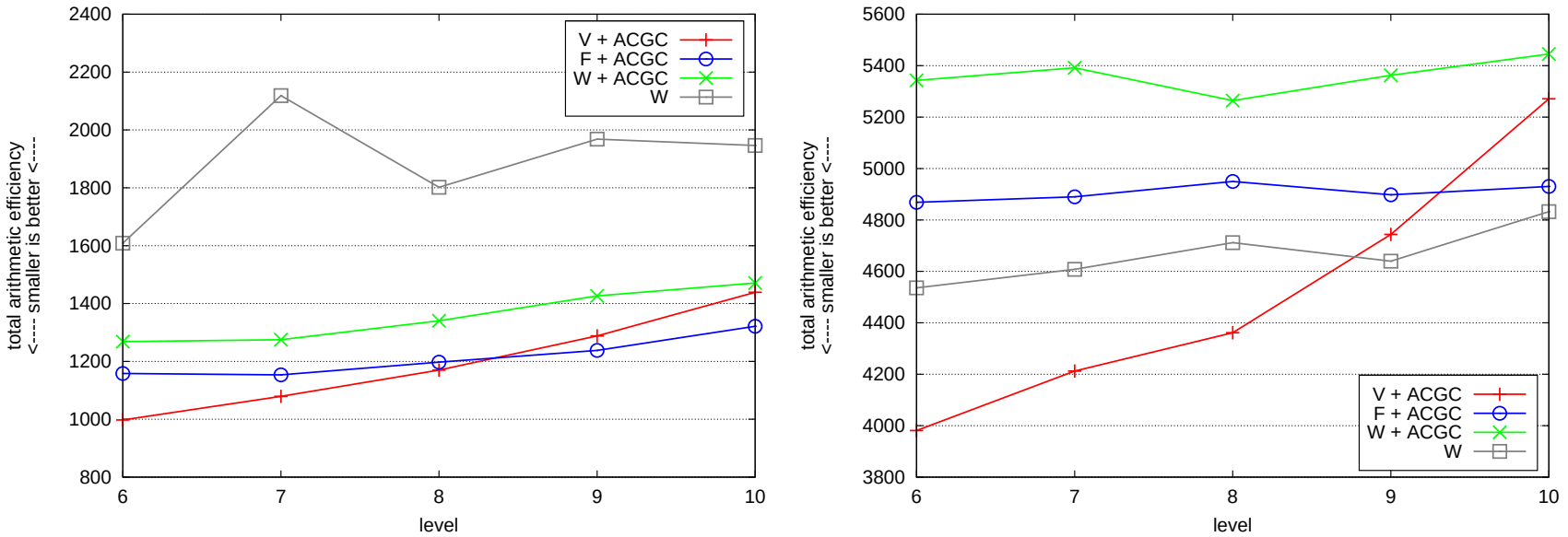
- 2-layer-ScaRC solver with Jacobi smoother:
  `MG(1e-6,V11)__MG(1e-1,V22,0.7)-JAC-D__D`
- 2-layer-ScaRC solver with ADiTriGS smoother:
  `MG(1e-6,F22)__MG(1e-1,V22)-ADI-D__D`
- only vary maximum MG level, fix coarse grid level to 1



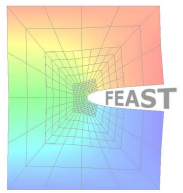
ACGC - Numerical Examples

technische universität dortmund

2-layer-ScaRC with local Jacobi smoother, isotropic operator



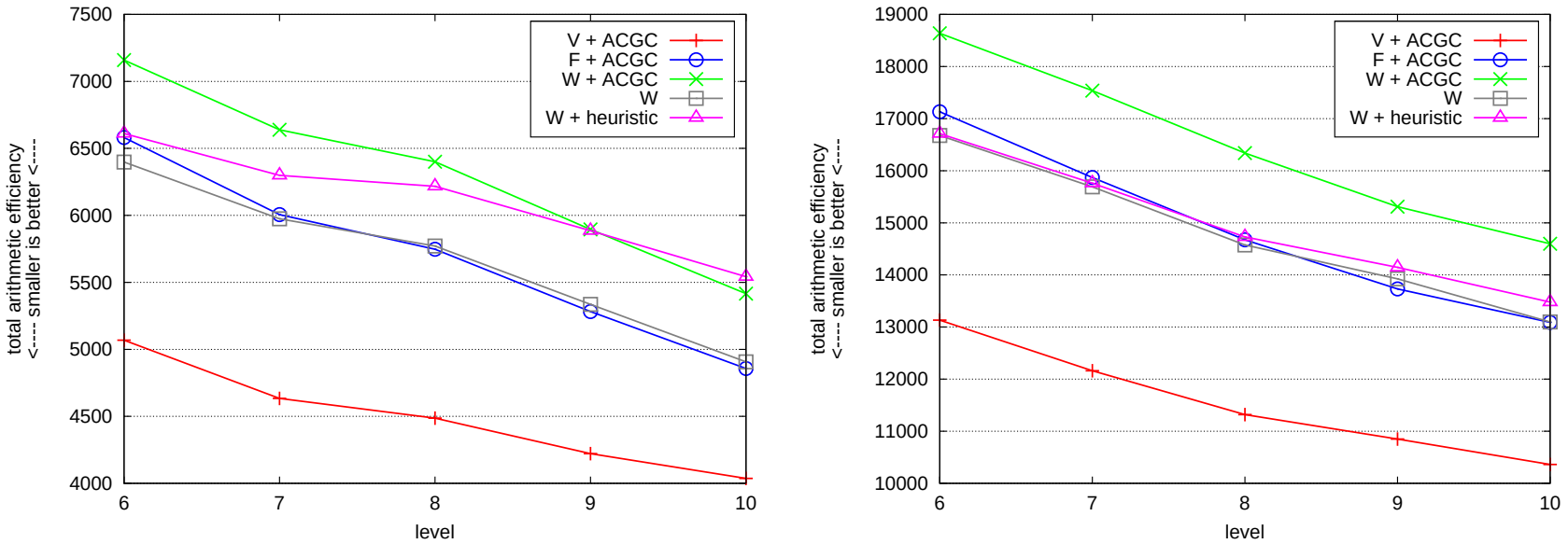
isotropic configuration (left), anisotropic configuration (right)



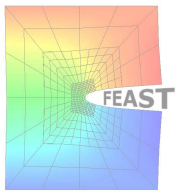
ACGC - Numerical Examples

technische universität dortmund

2-layer-ScaRC with local ADiTriGS smoother, anisotropic configuration



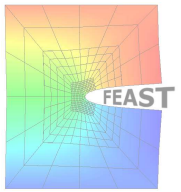
isotropic operator (left), anisotropic operator (right)



ACGC - Summary

Adaptive coarse grid correction:

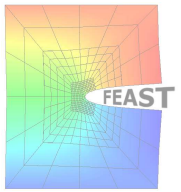
- method to alleviate the negative effects of the extended Dirichlet boundary conditions ('increasing subdomain size')
- easy to implement
- low arithmetic costs
- facilitates the use of V- and F-cycle multigrid
- smoothes oscillatory convergence behaviour of W-cycle
- often less expensive than standard W-cycle (GPUs!)
- and: can also improve standard (conforming) multigrid methods



Overview

technische universität dortmund

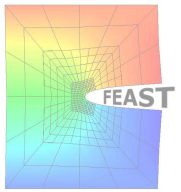
1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...



1-layer-ScaRC vs. 2-layer-ScaRC

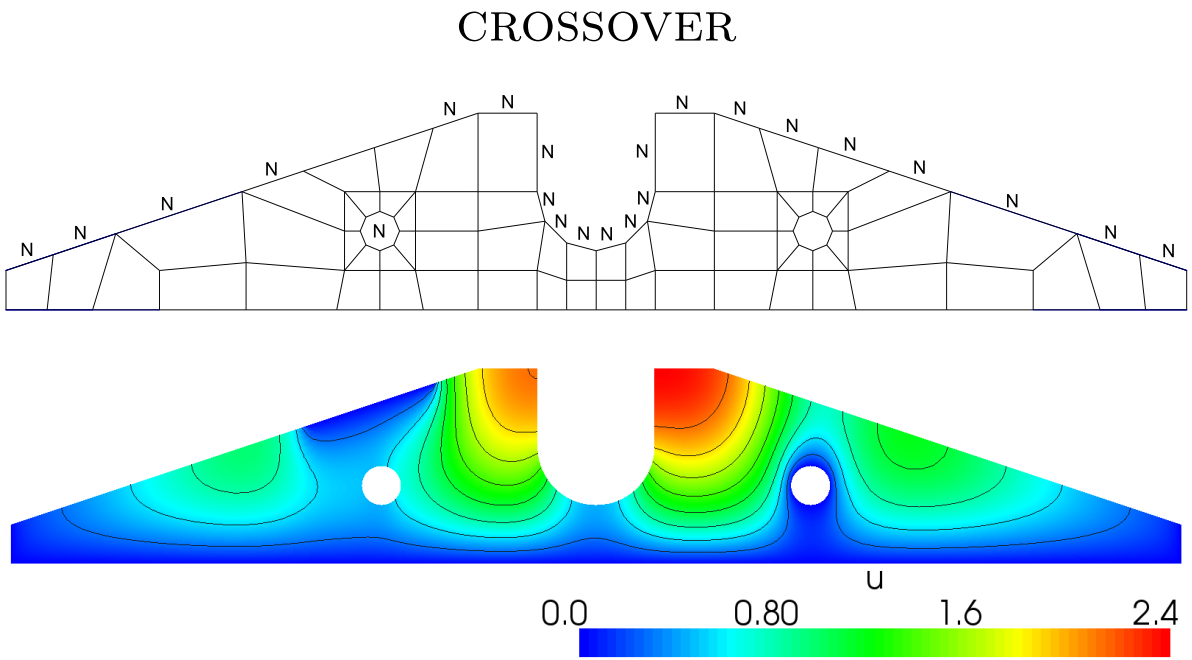
Is 2-layer-ScaRC really superior to 1-layer-ScaRC?

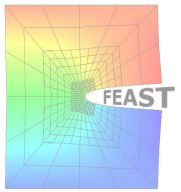
- use slightly anisotropic operator $-\partial_{xx} u - 4\partial_{yy} u = 1$
- use multigrid-Krylov solvers:
  - 1-layer-ScaRC: `MG-FGMRES4__JAC__D`
  - 2-layer-ScaRC: `MG-FGMRES4__MG(T7)-BICG-JAC-D__D`
- use more complex grids (see next slides):
  - CROSSOVER-ISO: 64 subdomains, 16.8M DOF (level 9), aspect ratio 2.91
  - CROSSOVER-ANISO: aspect ratio 20.4
  - ASMO: 70 subdomains, 18.4M DOF (level 9), aspect ratio 18.2



1-layer-ScaRC vs. 2-layer-ScaRC

technische universität dortmund

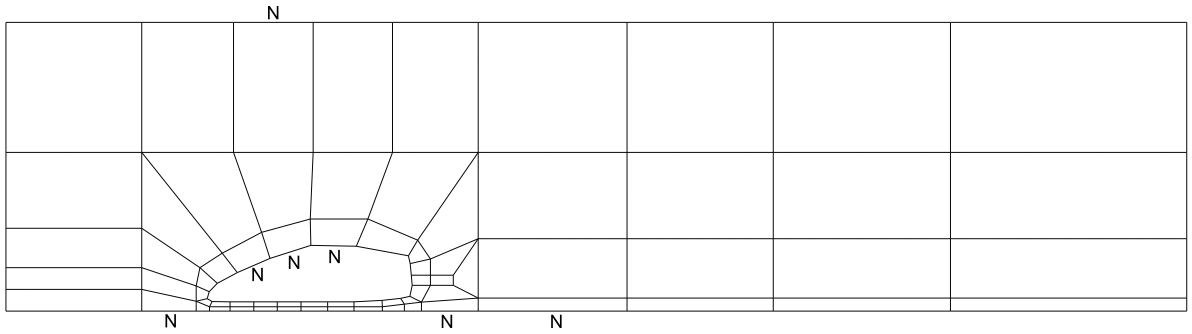


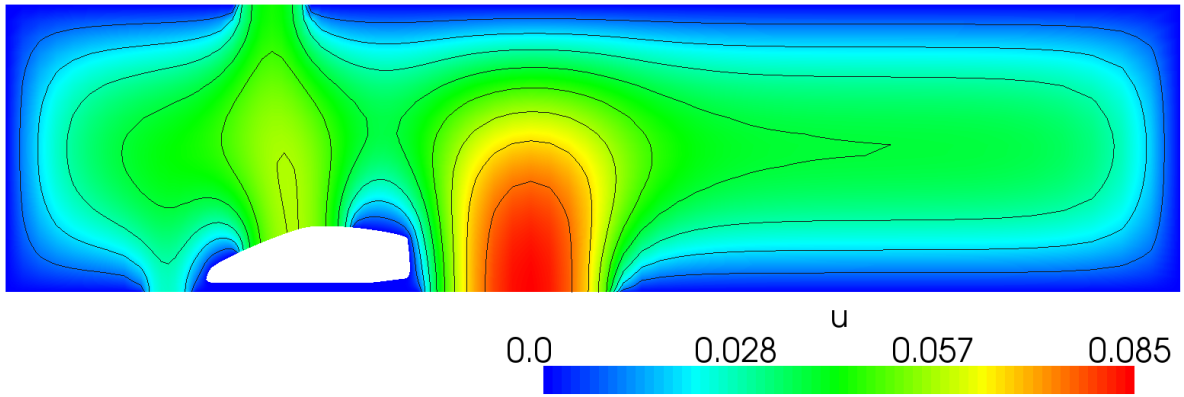


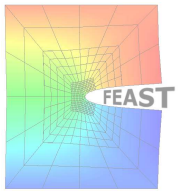
1-layer-ScaRC vs. 2-layer-ScaRC

technische universität dortmund

ASMO



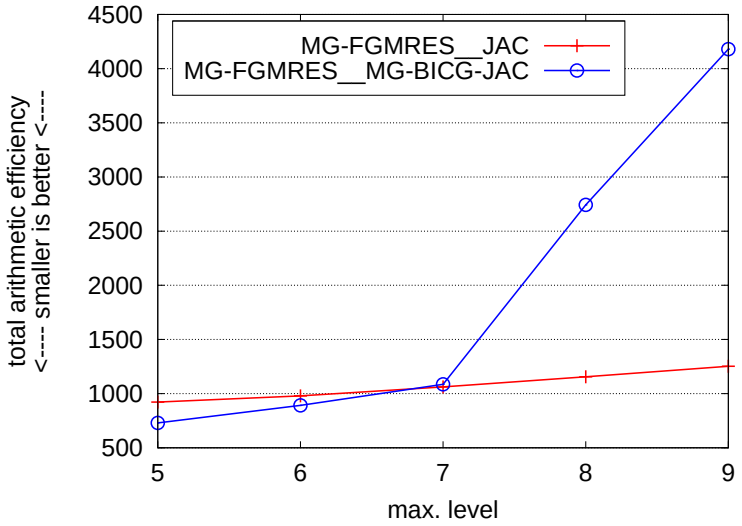


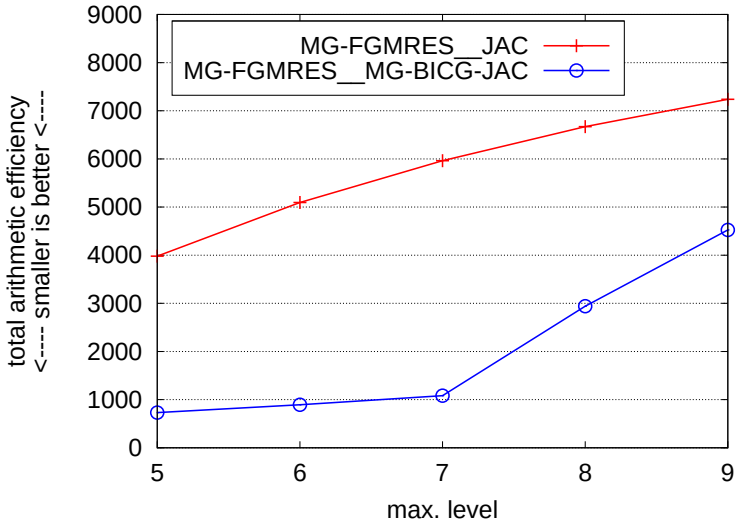


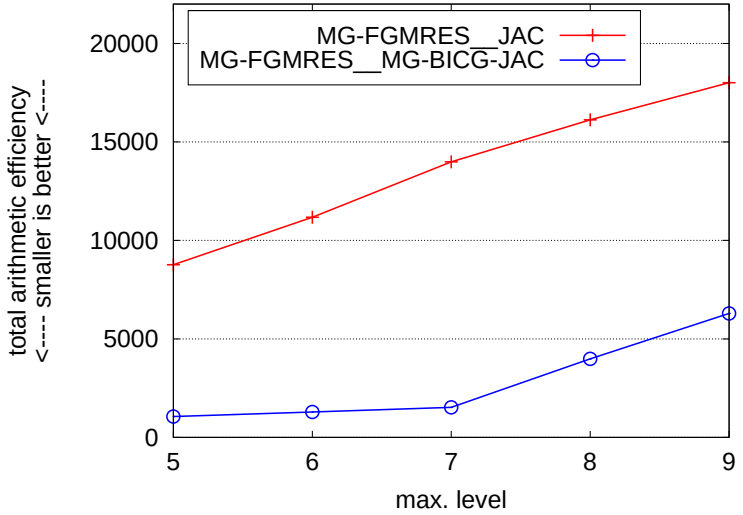
1-layer-ScaRC vs. 2-layer-ScaRC

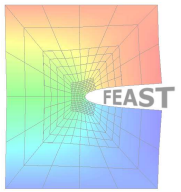
technische universität dortmund

CROSSOVER-ISO (left), CROSSOVER-ANISO (right), ASMO (bottom)







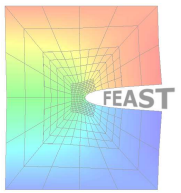


1-layer-ScaRC vs. 2-layer-ScaRC

Number of global smoothing steps
(⇒ amount of communication)

	MG-FGMRES4			MG-FGMRES4__MG		
lev	CR-I	CR-A1	AS-I	CR-I	CR-A1	AS-I
5	88	352	776	32	32	48
6	96	464	1016	32	32	48
7	104	552	1296	32	32	48
8	112	624	1504	32	32	48
9	120	680	1688	32	32	56

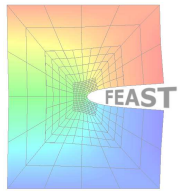
- 2-layer-ScaRC successfully hides the local irregularities (anisotropies) from the global solver
- ⇒ strongly favoured in a massively parallel computation



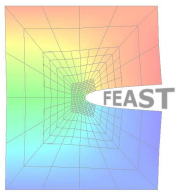
1-layer-ScaRC vs. 2-layer-ScaRC

But:

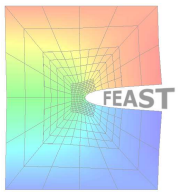
- for ADiTriGS as local smoother, examples to show superiority of 2-layer-ScaRC not found yet
- current favourite: 1-layer-ScaRC solver `BiCG-MG__ADI__D`
- reason: for ADiTriGS, mesh anisotropies are not really irregularities $\Rightarrow$ there is 'nothing to hide' within a 2-layer-ScaRC solver
- future attempts: massively parallel computations, more complicated physical equations, 3D



Overview



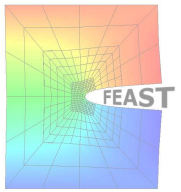
Overview



Basic Relations of Elasticity

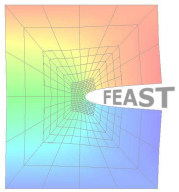
Elasticity:

- continuum mechanical discipline about deformation of solid elastic bodies
- solid and deformable: forces change the body's shape, but not its continuous coherence
- elastic: deformation process is reversible
- assumptions:
  - material is homogeneous and isotropic
  - only small deformations (linearised elasticity)



Basic Relations of Elasticity

- solid body $\bar{\Omega} \subset \mathbb{R}^3$
- boundary $\Gamma := \partial\Omega$, $\Gamma = \Gamma_D \cup \Gamma_N$
- deformation mapping $\boldsymbol{\Phi} : \bar{\Omega} \to \mathbb{R}^3$ with $\det(\nabla\boldsymbol{\Phi}) > 0$
- displacements $\mathbf{u}(\mathbf{x}) = \big(u_1(\mathbf{x}), u_2(\mathbf{x}), u_3(\mathbf{x})\big)^T$ of material point $\mathbf{x} \in \bar{\Omega}$, $\boldsymbol{\Phi} = \mathbf{id} + \mathbf{u}$
- kinematic relation between displacements and strains: linearised strain tensor $\varepsilon = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$



Basic Relations of Elasticity

Stress principle of Euler and Cauchy:
existence of a vector field $\mathbf{t} : \bar{\Omega} \times S_1 \to \mathbb{R}^3$ (Cauchy stress vector) with:

- for arbitrary $V \subset \bar{\Omega}$ ($\mathbf{g}$ applied surface force):

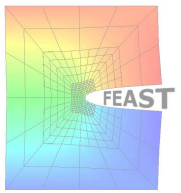
$$\mathbf{t}(\mathbf{x}, \mathbf{n}) = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Gamma_N \cap \partial V$$

- axiom of force balance ($\mathbf{f}$ applied body force):

$$\int_V \mathbf{f}(\mathbf{x}) \, \mathrm{d}x + \int_{\partial V} \mathbf{t}(\mathbf{x}, \mathbf{n}) \, \mathrm{d}a = \mathbf{0}$$

- axiom of balance of angular momenta:

$$\int_V \mathbf{x} \times \mathbf{f}(\mathbf{x}) \, \mathrm{d}x + \int_{\partial V} \mathbf{x} \times \mathbf{t}(\mathbf{x}, \mathbf{n}) \, \mathrm{d}a = \mathbf{0}$$



Basic Relations of Elasticity

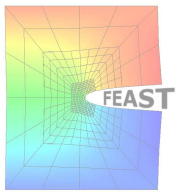
Cauchy's theorem:
existence of a symmetric tensor field $\boldsymbol{\sigma} : \bar{\Omega} \to \mathbb{M}^3$ (Cauchy stress tensor) that satisfies

- $\mathbf{t}(\mathbf{x}, \mathbf{n}) = \boldsymbol{\sigma}(\mathbf{x})\mathbf{n}, \quad \mathbf{x} \in \bar{\Omega}, \mathbf{n} \in S_1,$
- the PDE

$$-\mathbf{div}\big(\boldsymbol{\sigma}(\mathbf{x})\big) = \mathbf{f}(\mathbf{x}), \qquad \mathbf{x} \in \Omega,$$

- and the boundary conditions

$$\boldsymbol{\sigma}(\mathbf{x})\mathbf{n} = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Gamma_N.$$



Basic Relations of Elasticity

- constitutive law: relation between strains and stresses
- Hooke's law for isotropic material:

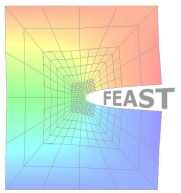
$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I}$$

- Lamé constants μ and λ
- relation to Young's modulus E and Poisson ratio ν:

$$\mu = \frac{E}{2(1+\nu)}, \qquad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

- $\nu \to 0.5$ ($\lambda \to \infty$): compressibility of material decreases

Material	E [$\mathrm{N/m^2}$]	ν [-]	μ [$\mathrm{N/m^2}$]	λ [$\mathrm{N/m^2}$]
Steel/Iron	$2.1 \cdot 10^{11}$	0.29	$8.14 \cdot 10^{10}$	$1.1 \cdot 10^{11}$
Lead	$1.6 \cdot 10^{10}$	0.44	$5.6 \cdot 10^{9}$	$4.1 \cdot 10^{10}$
Rubber	$2.5 \cdot 10^{7}$	0.499–0.5	$8.2 \cdot 10^{6}$	$4.1 \cdot 10^{9}$



Basic Relations of Elasticity

- resulting boundary value problem - the Lamé Equation:

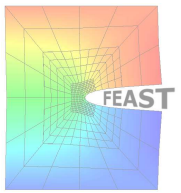
$$\begin{aligned} -2\mu\ \mathbf{div}(\varepsilon) - \lambda\,\boldsymbol{\nabla}\,\mathrm{div}(\mathbf{u}) &= \mathbf{f} && \text{in } \Omega \\ \mathbf{u} &= \bar{\mathbf{u}} && \text{on } \Gamma_D \\ \boldsymbol{\sigma}\mathbf{n} &= \mathbf{g} && \text{on } \Gamma_N \end{aligned}$$

- bilinear form:

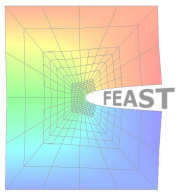
$$k(\mathbf{u},\mathbf{v}) := \int_\Omega 2\mu\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) + \lambda\,\mathrm{div}(\mathbf{u})\,\mathrm{div}(\mathbf{v})\,\mathrm{d}x$$

- weak formulation: Find $\mathbf{u} - \bar{\mathbf{u}} \in \mathbf{V} := \left\{\mathbf{v} \in H^1(\Omega)^3 \;\middle|\; \mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\right\}$ such that

$$k(\mathbf{u},\mathbf{v}) = \int_\Omega \mathbf{f}\cdot\mathbf{v}\,\mathrm{d}x + \int_{\Gamma_N} \mathbf{g}\cdot\mathbf{v}\,\mathrm{d}a, \qquad \mathbf{v} \in \mathbf{V}$$



Overview



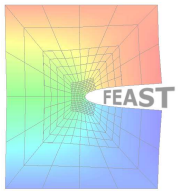
Operator Splitting

- motivation: reduction to scalar components
- rewrite left hand side of BVP (now 2D):

$$
\begin{aligned}
&-2\mu\ \mathbf{div}(\varepsilon) - \lambda\,\nabla\,\mathrm{div}(\mathbf{u}) \\
&= \begin{pmatrix} (2\mu+\lambda)\partial_{11} + \mu\partial_{22} & (\mu+\lambda)\partial_{12} \\ (\mu+\lambda)\partial_{21} & \mu\partial_{11} + (2\mu+\lambda)\partial_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}
\end{aligned}
$$

- rewrite left hand side of weak form:

$$
\begin{aligned}
k(\mathbf{u},\mathbf{v}) = &\int_\Omega (2\mu+\lambda)\partial_1 u_1 \partial_1 v_1 + \mu\partial_2 u_1 \partial_2 v_1 \ \mathrm{d}x \\
+ &\ \text{... (mixed terms)} \\
+ &\int_\Omega \mu\partial_1 u_2 \partial_1 v_2 + (2\mu+\lambda)\partial_2 u_2 \partial_2 v_2 \ \mathrm{d}x \\
= &\ \text{... (next slide)}
\end{aligned}
$$



Operator Splitting

technische universität dortmund

- rewrite left hand side of weak form:

$$k(\mathbf{u},\mathbf{v}) = \underbrace{\int_\Omega \left[\begin{pmatrix} 2\mu+\lambda & 0 \\ 0 & \mu \end{pmatrix} \nabla u_1\right] \cdot \nabla v_1 \, \mathrm{d}x}_{=:k_{11}(u_1,v_1)} + k_{12}(u_2,v_1)$$

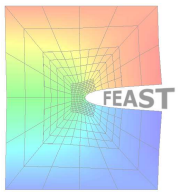
$$+ k_{21}(u_1,v_2) + \underbrace{\int_\Omega \left[\begin{pmatrix} \mu & 0 \\ 0 & 2\mu+\lambda \end{pmatrix} \nabla u_2\right] \cdot \nabla v_2 \, \mathrm{d}x}_{=:k_{22}(u_2,v_2)}$$

- FE discretisation (Q_1) $\Rightarrow$ block structured linear equation system:

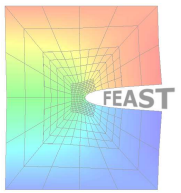
$$\begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}$$

  ('separate displacement ordering'; not used, e. g., in FEAP)

- $\mathbf{K}_{11}$ and $\mathbf{K}_{22}$ correspond to scalar elliptic operators ('anisotropic Laplace operator')



Overview



Solution Concept

- basic iteration - block preconditioned Richardson method:

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \tilde{\mathbf{K}}^{-1}(\mathbf{f} - \mathbf{K}\mathbf{u}^k)$$

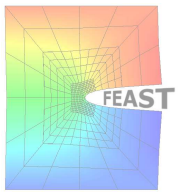
- defect computation:

$$\mathbf{f} - \mathbf{K}\mathbf{u}^k = \begin{pmatrix} \mathbf{f}_1 - \mathbf{K}_{11}\mathbf{u}_1^k - \mathbf{K}_{12}\mathbf{u}_2^k \\ \mathbf{f}_2 - \mathbf{K}_{21}\mathbf{u}_1^k - \mathbf{K}_{22}\mathbf{u}_2^k \end{pmatrix}$$

- e. g., block Jacobi preconditioner:

$$\tilde{\mathbf{K}}_{\mathrm{BJac}}^{-1} = \begin{pmatrix} \mathbf{K}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{22}^{-1} \end{pmatrix}$$

- reduction to scalar components $\Rightarrow$ exploit FEAST concepts!
- note: each global matrix $\mathbf{K}_{ij}$ is represented by local 9-band matrices corresponding to TP subdomains

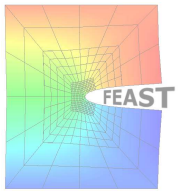


Solution Concept

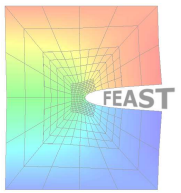
- condition number of the preconditioned system:

$$\kappa := \kappa(\tilde{\mathbf{K}}_{\mathrm{BJac}}^{-1}\mathbf{K}) \leqslant \frac{1}{c_{\mathrm{K}}}\left(1 + \frac{1}{1-2\nu}\right)$$

- three observations:
  1. κ does not depend on mesh size parameter h
     $\Rightarrow$ single-grid outer solver (i. e., Krylov method) can be sufficient
  2. κ depends on Korn's constant $c_{\mathrm{K}} = c_{\mathrm{K}}(\Omega, \Gamma_{\mathrm{D}})$
     $\Rightarrow$ convergence depends on boundary conditions / geometry (e. g., cantilever beam)
     (Korn's inequality: $\exists\, c_{\mathrm{K}} = c_{\mathrm{K}}(\Omega, \Gamma_{\mathrm{D}}) > 0 : \|\boldsymbol{\varepsilon}(\mathbf{v})\|_0^2 \geqslant c_{\mathrm{K}}\|\mathbf{v}\|_1^2, \mathbf{v} \in \mathbf{V}$)
  3. κ depends on Poisson ratio ν
     $\Rightarrow$ condition of the system increases with incompressibility of the material ($\nu \to 0.5$)
- block Gauß-Seidel or block SOR instead of block Jacobi
  $\Rightarrow$ no such estimate, but more efficient in most cases

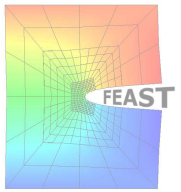


Overview



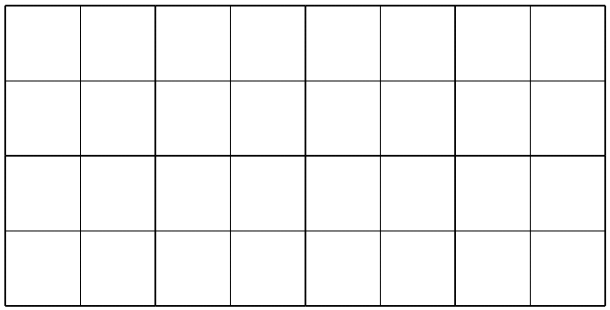
Selected Numerical Examples

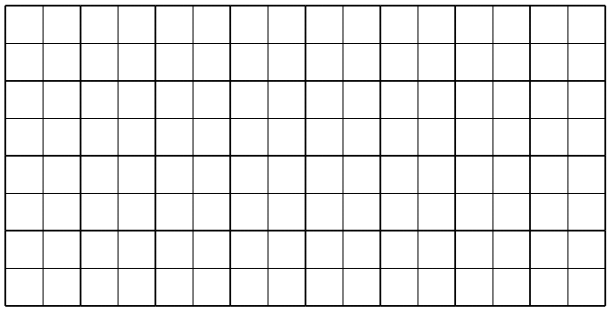
- dependence on mesh anisotropies
- dependence on geometry
- 1-layer-ScaRC vs. 2-layer-ScaRC
- parallel efficiency

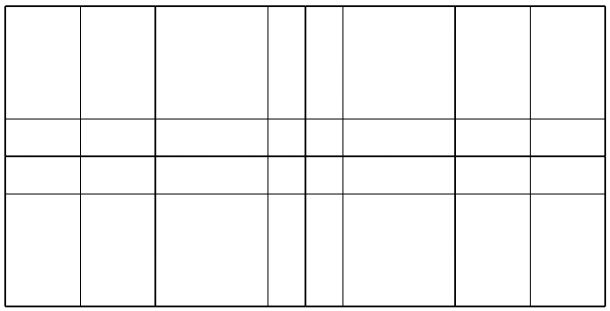


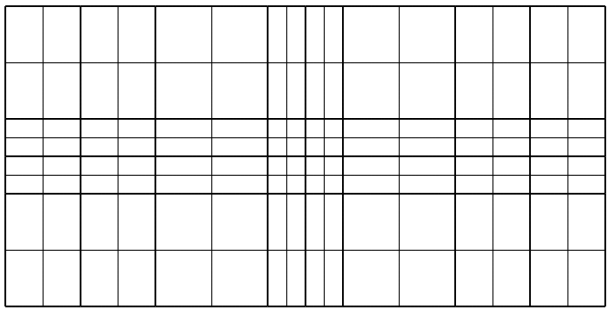
Dependence on Mesh Anisotr.

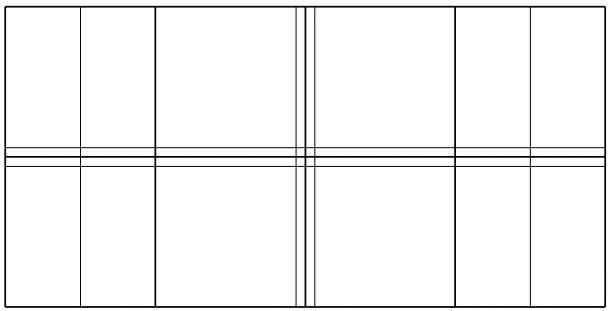
4×2 subdomains (left), 8×4 subdomains (right)

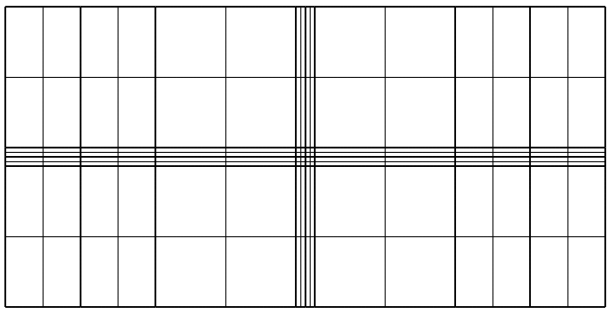


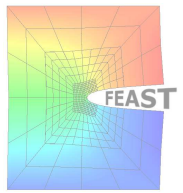








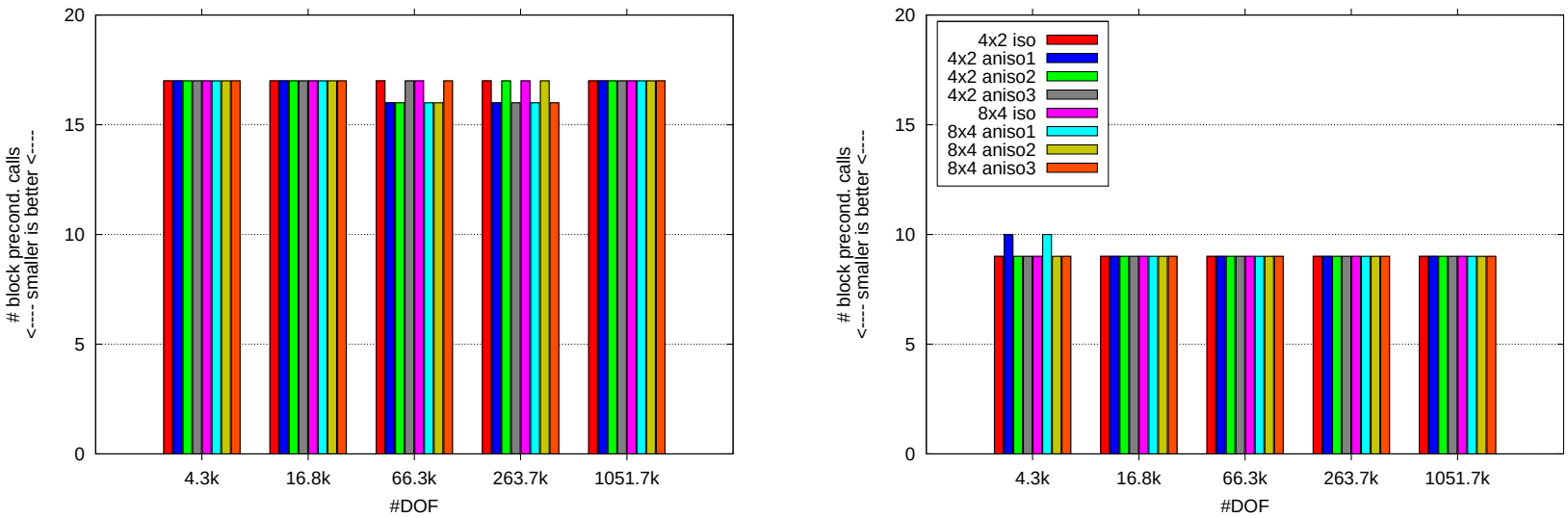




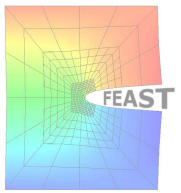
Dependence on Mesh Anisotr.

technische universität dortmund

`BiCG-BJac` (left), `BiCG-BSor` (right)

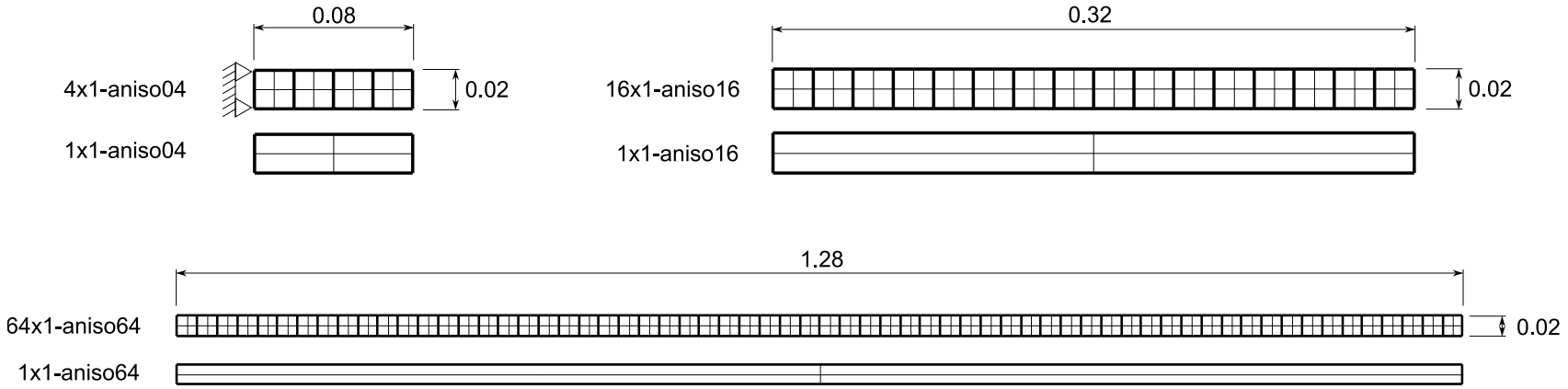


- convergence independent of mesh refinement level (outer single-grid Krylov solver!)
- BSor roughly halves number of iterations compared to BJac
- anisotropies are fully ‘absorbed’ by the scalar subsolves

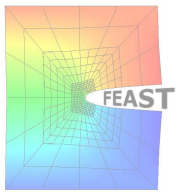


Dependence on Geometry

technische universität dortmund



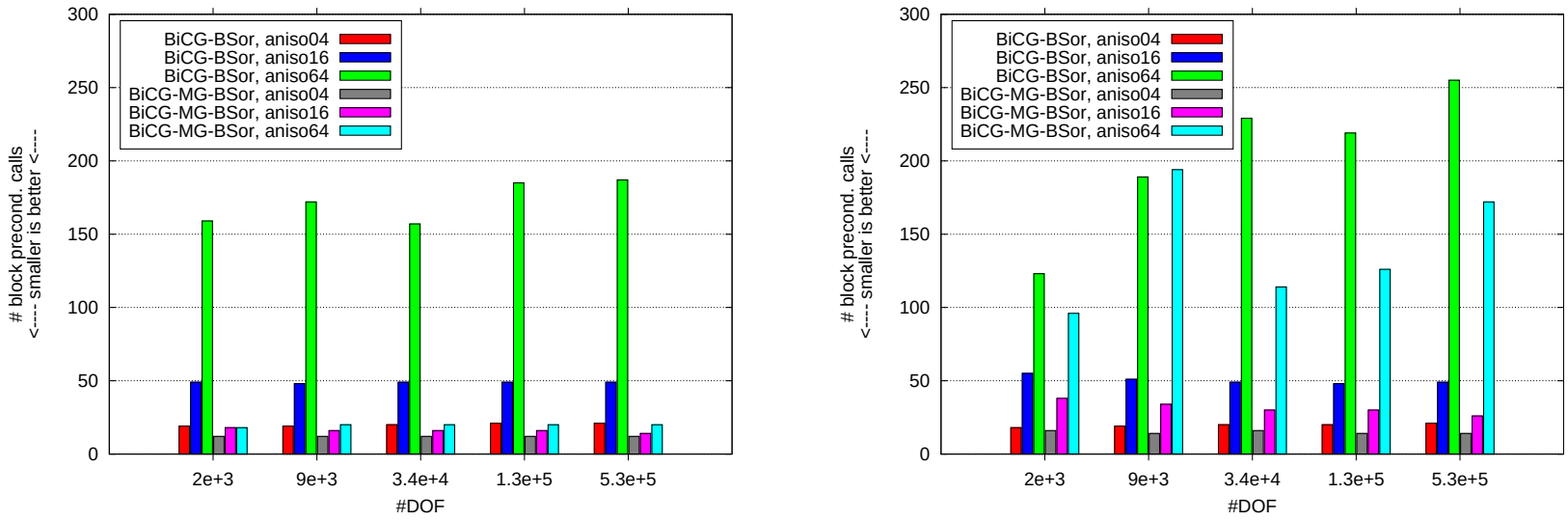
- typical cantilever beam configuration:
  left side fixed, other sides free, vertical body force applied
- increasing global anisotropy: L/H = 4, 16, 64
- two meshings:
  4/16/64 isotropic subdomains vs. 1 anisotropic subdomain
- two solvers:
  - `BiCG-BSor`: outer single-grid Krylov solver
  - `BiCG-MG-BSor`: multigrid (applied to block system!) as preconditioner for Krylov solver



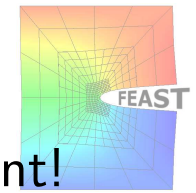
Dependence on Geometry

technische universität dortmund

4/16/64 iso subdomains (left), 1 aniso subdomain (right)

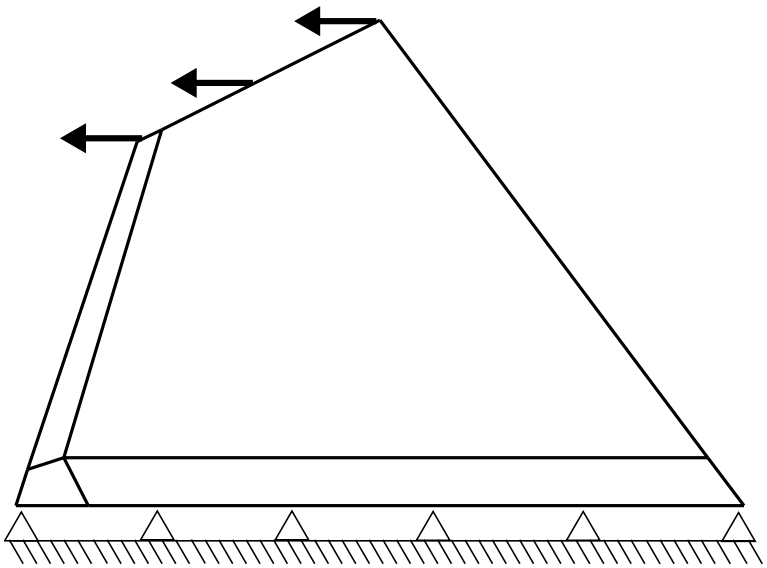


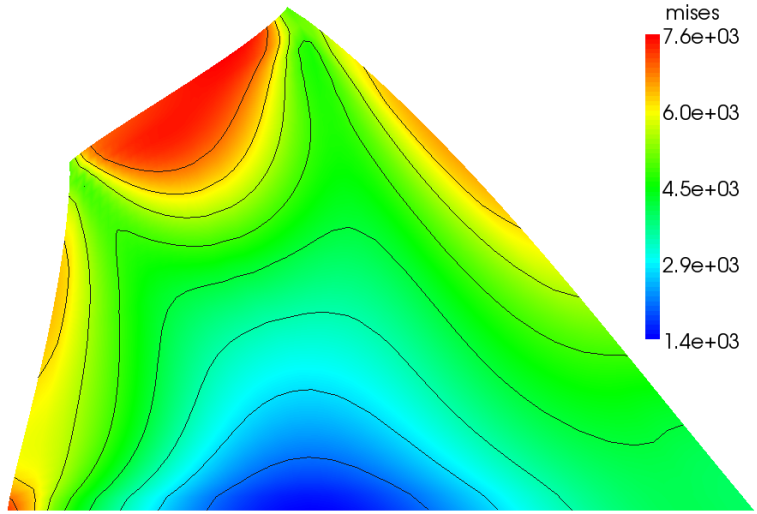
- single-grid Krylov solver *not* sufficient: convergence (more or less) independent of refinement level, but strongly dependent on global anisotropy
- additional multigrid greatly weakens this dependency (at least in case of isotropic subdomains)
- applying multigrid scheme *only to scalar subsystems* is not sufficient!



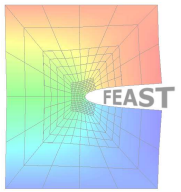
1-layer-ScaRC vs. 2-layer-ScaRC

technische universität dortmund





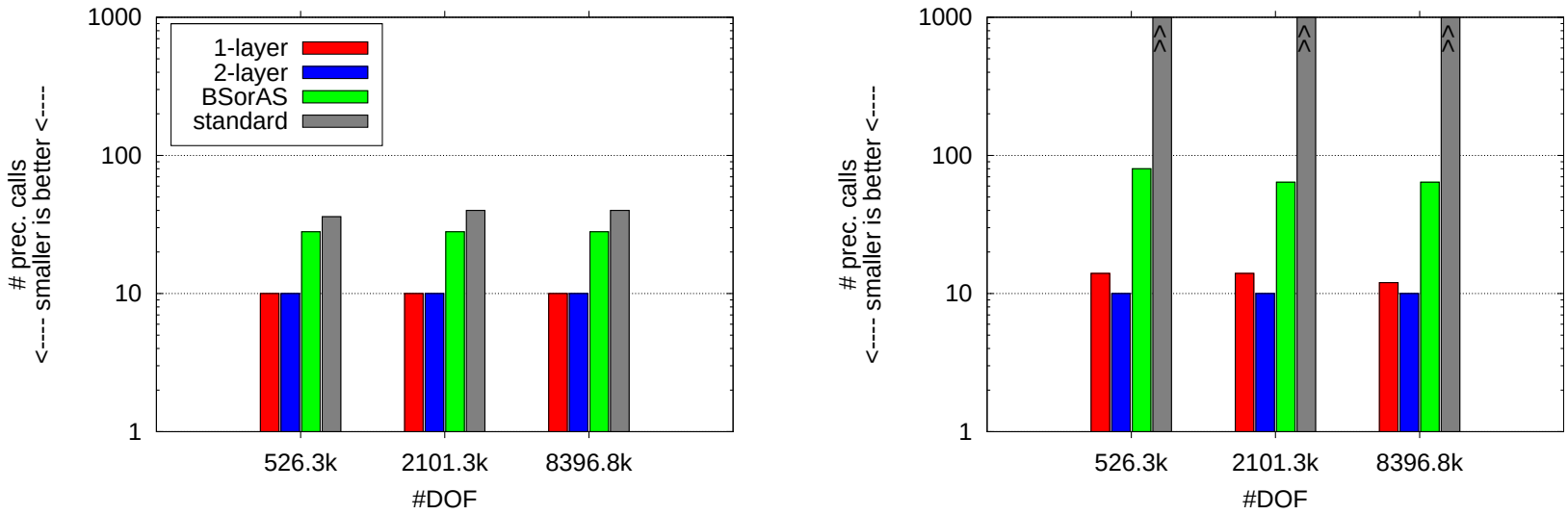
- two variants: ISO (max. aspect ratio 2.2) and ANISO (max. aspect ratio 63.6) (only ANISO shown)
- compare three solvers:
  - using 1-layer-ScaRC:
    `BiCG-MG(1,V11)-BSor[MG(1e-1)-FGMRES4__JAC__D]`
  - using 2-layer-ScaRC:
    `BiCG-MG(1,V11)-BSor[MG(1e-1)-FGMRES4__MG-BICG-JAC-D__D]`
  - 'standard solver': `MG(F11)-BiCG(2)-Jac` (using point Jacobi smoother, disregarding the block structure)



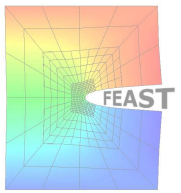
1-layer-ScaRC vs. 2-layer-ScaRC

technische universität dortmund

ISO (left), ANISO (right)

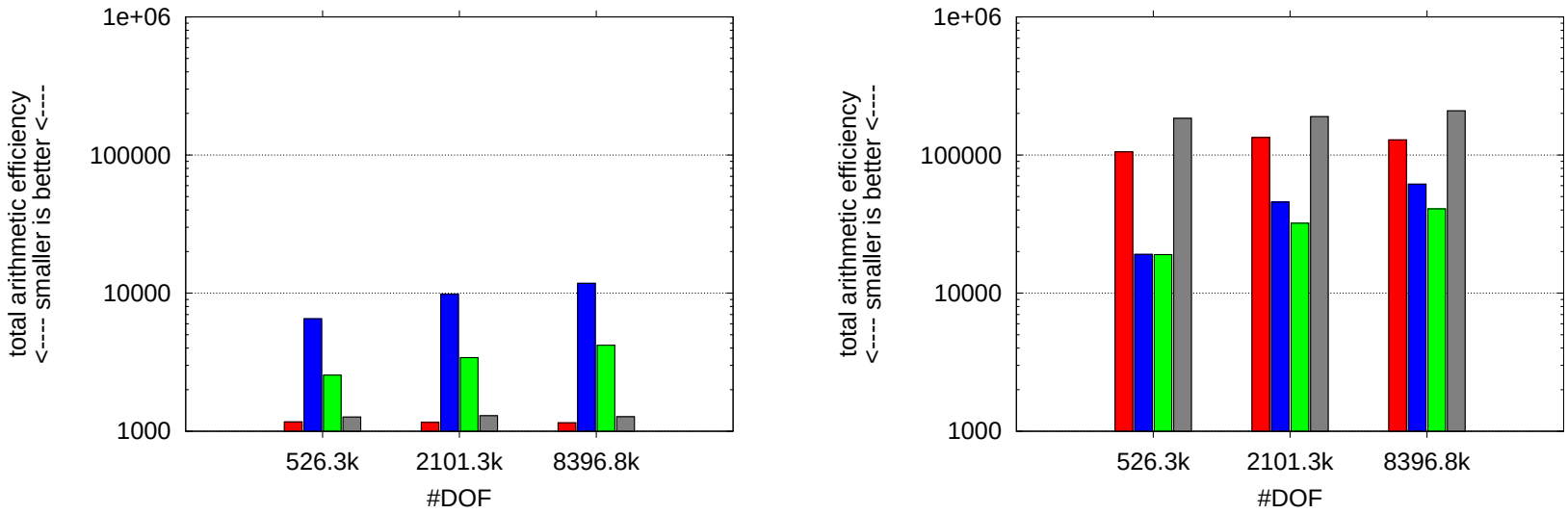


- convergence behaviour of 1-layer-ScaRC and 2-layer-ScaRC variants (nearly) independent of the configuration
- the standard solver suffers signficantly
- 2-layer-ScaRC always needs 1 iteration per call,
  1-layer-ScaRC between 6 (iso) and 70 (aniso) iterations
  $\Rightarrow$ communication amount of 2-layer-ScaRC variant much smaller!

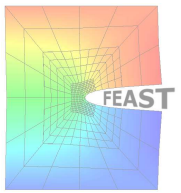


1-layer-ScaRC vs. 2-layer-ScaRC

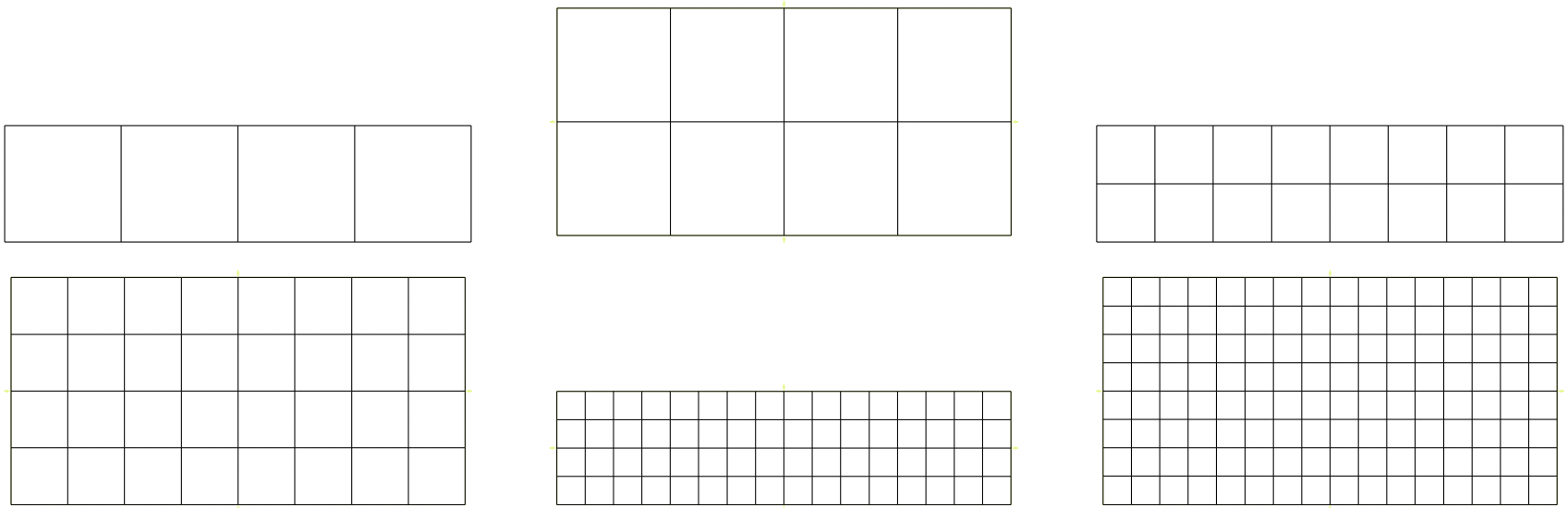
ISO (left), ANISO (right)



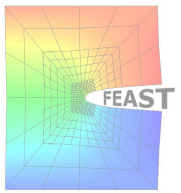
- arithmetic costs of 2-layer-ScaRC very high on ISO configuration
- arithmetic costs of 1-layer-ScaRC and the standard solver increase drastically on the ANISO configuration $\Rightarrow$ 2-layer-ScaRC superior
- just 'proof of concept'! (with ADiTriGS: TAE $\approx$ 1000)



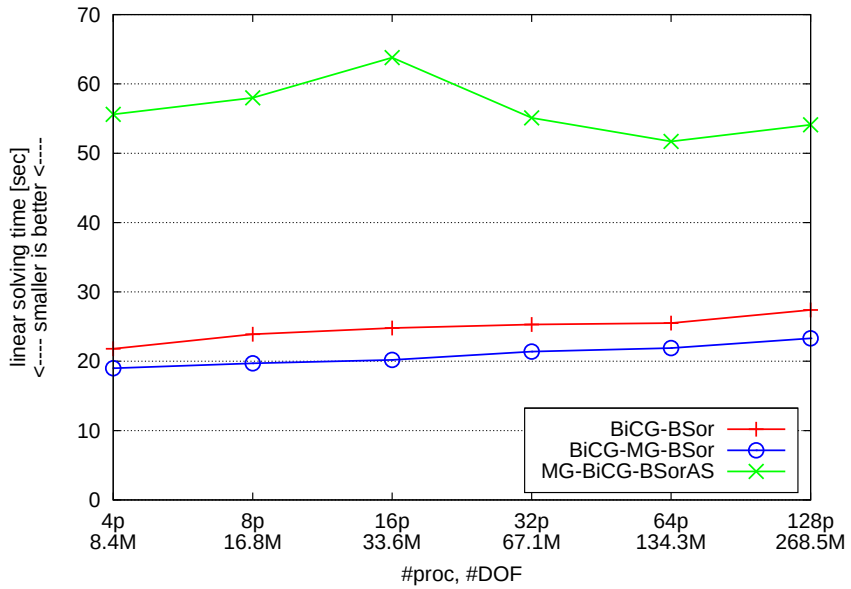
Parallel Efficiency



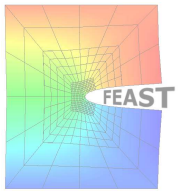
- weak scalability: increase problem size and resources by the same factor
- 'perfect weak scalability': runtime remains constant
- smallest problem: 4×1 subdomains, 4.2 M vertices, 4 processors
- largest problem: 16×8 subdomains, 134.2 M vertices, 128 proc.



Parallel Efficiency



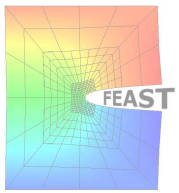
- good weak scalability
- comparable to scalar FEAST solvers
- parallel efficiency of the scalar FEAST library fully transfers to elasticity solvers



Overview

technische universität dortmund

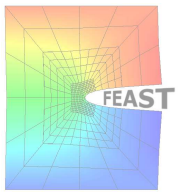
1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...



Overview

technische universität dortmund

1. Introduction/Motivation
2. Multilevel Solvers for Scalar Elliptic Equations
   - FEAST's Meshing Concept
   - FEAST's Adaptivity Concept
   - FEAST's Solver Concept
   - Local Multigrid Solvers
   - 1-layer-ScaRC vs. 2-layer-ScaRC
3. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples
4. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief Overview...

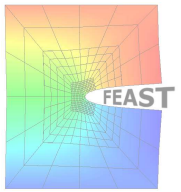


Incompressible Elast. Problems

- rubber-like materials are (nearly) incompressible, important for many industrial applications
- basic problem: $\nu \to 0.5 \Rightarrow \lambda \to \infty$
- pure displacement formulation: 'volume locking'
- significant deterioration of FE approximation and solver behaviour
- remedy: mixed displacement-pressure $\mathbf{u}/p$ formulation
- leads to Stokes-like saddle point problem $\mathcal{A}\mathbf{x} = \mathbf{b}$ with

$$\mathcal{A} := \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix}, \qquad \mathbf{x} := \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix}, \qquad \mathbf{b} := \begin{pmatrix} \mathbf{f} \\ \mathbf{0} \end{pmatrix}$$

- $\mathbf{C}$ contains compressibility and, if necessary, stabilisation terms (e. g., for Q_1/Q_1)
- distinguish segregated (uncoupled) and coupled solution methods



Saddle Point Solvers

Segregated methods:

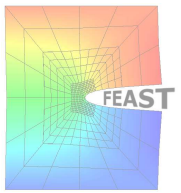
- solve subsystems for $\mathbf{u}$ and p
- Uzawa / pressure Schur complement methods:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \tilde{\mathbf{S}}^{-1}(\mathbf{B}^T\mathbf{A}^{-1}\mathbf{f} - (\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} - \mathbf{C})\mathbf{p}_k),$$

  $\tilde{\mathbf{S}}^{-1}$ preconditioner for Schur complement $\mathbf{S} := \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} - \mathbf{C}$
- block-triangular preconditioners:

$$\begin{pmatrix}\mathbf{u}_{k+1}\\ \mathbf{p}_{k+1}\end{pmatrix} = \begin{pmatrix}\mathbf{u}_k\\ \mathbf{p}_k\end{pmatrix} + \begin{pmatrix}\tilde{\mathbf{A}} & \mathbf{0}\\ \mathbf{B}^T & -\tilde{\mathbf{S}}\end{pmatrix}^{-1} \begin{pmatrix}\mathbf{f} - \mathbf{A}\mathbf{u}_k - \mathbf{B}\mathbf{p}_k\\ \mathbf{0} - \mathbf{B}^T\mathbf{u}_k - \mathbf{C}\mathbf{p}_k\end{pmatrix},$$

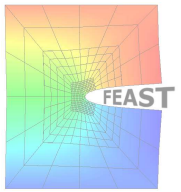
- essential for both variants:
  a good Schur complement preconditioner



Saddle Point Solvers

Segregated methods (continued):

- Schur complement preconditioning often involves solution of scalar systems
  $\Rightarrow$ scalar ScaRC solvers!
- solve $\tilde{\mathbf{A}}$-systems with techniques from compressible elasticity
  $\Rightarrow$ scalar ScaRC solvers!
- hence: also the solution of saddle point systems can essentially be brought down to the solution of scalar systems
- multigrid can be applied to the whole saddle point system
  $\Rightarrow$ overall solver potentially contains four (!) nested MG schemes
- BEAM configurations: three nested MG schemes can be beneficial in terms of numerical efficiency (with corresponding bad parallel efficiency)

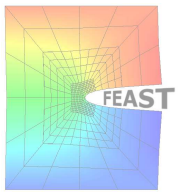


Saddle Point Solvers

Coupled methods:

- multigrid with Vanka smoothers
- solve whole saddle point system at once, but reduction to small subdomains (one element, patch of few elements)
- combine local solutions in multiplicative fashion (block Gauß-Seidel)
- mesh anisotropies need special treatment
- no Schur complement preconditioner needed
- no reduction to scalar subsystems, ScaRC solvers not applicable

In the context of my work:
segregated methods clearly superior to coupled Vanka methods (up to 30x - 40x faster)



Thank you
for your attention!