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

Hilmar Wobker

1Institute 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

1. Introduction/Motivation
   - 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

2. Multilevel Solvers for Compressible Elasticity Problems
   - Basic Relations of Elasticity
   - Operator Splitting
   - Solution Concept
   - Selected Numerical Examples

3. ML Saddle Point Solvers for Incompressible Elasticity Problems
   - Just a Brief 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.
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

→ difficult to realise due to conflicting demands
→ our attempt: FEAST
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
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

→ Prototypically demonstrated with FEASTsolid
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

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

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 addressing
  $\Rightarrow$ expensive memory access

- Plus: low arithmetic intensity (sparse matrices)
  $\Rightarrow$ poor processor efficiency (low MFLOP/s rates)
FEAST’s Meshing Concept

Remedy: use structured data with high spatial and temporal locality

- FEAST uses generalised tensor product meshes
- rowwise numbering
  ⇒ exactly 9 matrix bands for bilinear elements
- direct addressing, caching
- optimised Linear Algebra routines (SparseBandedBLAS)
FEAST’s Meshing Concept

- more complex (unstructured) domains by joining several (structured) TP meshes
- local matrices + appropriate border data exchanges
  ⇒ ‘virtual’ global matrix

Here:
- 64 TP meshes (=64 local matrices), each refined 10x
- distributed over 16 processors
- ⇒ 1.34 \cdot 10^8 degrees of freedom in total
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…
FEAST’s Adaptivity Concept

Patch-wise Hanging Nodes:

Mesh Deformation:

TP property fulfilled!
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...
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

\[
\text{data locality required for parallel efficiency}
\]
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
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 \(Ax = b\)
- basic components:
  - smoothing (level \(l\)): \(x^{(l)} \leftarrow x^{(l)} + \omega S^{(l)}(b^{(l)} - A^{(l)}x^{(l)})\)
  - grid transfer: restriction \((l \rightarrow l - 1)\), prolongation \((l - 1 \rightarrow l)\)
  - coarse grid solving: \(x^{(1)} = (A^{(1)})^{-1}b^{(1)}\)
- exploiting smoothing property of elementary methods like Jacobi or Gauß-Seidel
- convergence behaviour independent of the refinement level
- runtime \(O(\#DOF)\)
Multigrid

Possibilities to traverse the grid hierarchy:

- V-, F- and W-cycle
- arithmetic costs vs. convergence speed / robustness
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 $M$:
$M \rightarrow 1$: standard multigrid with selected smoother
$M \rightarrow n$: standard multigrid with Jacobi smoother
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 $i$: $\text{loc}_i^{(l)}(k)$
- prolongation matrix $\mathbf{P}_i^{(l)}$:

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

- 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 $\tilde{A}$:

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

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

- 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
one iteration (multiplicative part between levels):

\[
x^{(L)} \leftarrow x^{(L)} + \tilde{A}^{(L)}(b^{(L)} - A^{(L)} x^{(L)})
\]

\[l = L - 1, \ldots, 2:
\]

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

\[l = 1:
\]

\[
b^{(1)} \leftarrow R^{(2)}(b^{(2)} - A^{(2)} x^{(2)}), \quad x^{(1)} \leftarrow (A^{(1)})^{-1} b^{(1)}
\]

\[l = 2, \ldots, L:
\]

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

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

\[
\tilde{A}^{(l)} := \sum_{i=1}^{M} P^{(l)}_{i} \tilde{A}_{i}^{(l)} R_{i}^{(l)}.
\]

local preconditioner \(\tilde{A}_{i}^{(l)}\) for the local matrix \(A_{i}^{(l)}\) (slide after next)
Why minimal overlap?

- sufficient for robust convergence behaviour of multilevel DD  
  → numerical efficiency
- local submeshes preserve tensor product property  
  → processor efficiency
- minimal amount of data exchange between subdomains  
  → parallel efficiency
- implementation and data structures greatly simplified
two strategies to realise the local preconditioner $\tilde{A}_i^{(l)}$:

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 ($\tilde{A}_i^{(l)}$ being the local smoother)

$\Rightarrow$ MLDD generalisation of MG
The Local Preconditioner

First strategy:

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

$$A_i = (L_i^L + L_i^C + L_i^U) + (C_i^L + C_i^C + C_i^U) + (U_i^L + U_i^C + U_i^U)$$

- Some local smoothers:

  $$\tilde{A}_i^{\text{Jacobi}} := (C_i^C)^{-1}$$
  $$\tilde{A}_i^{\text{GS}} := (L_i^L + L_i^C + L_i^U + C_i^L + C_i^C)^{-1}$$
  $$\tilde{A}_i^{\text{TriGS}} := (L_i^L + L_i^C + L_i^U + C_i^L + C_i^C + C_i^U)^{-1}$$
  $$\tilde{A}_i^{\text{MTriGS}} : \tilde{A}_i^{\text{TriGS}} \text{ applied to column-wise numbered grid}$$
  $$\tilde{A}_i^{\text{ADiTriGS}} : \text{alternating application of } \tilde{A}_i^{\text{TriGS}} \text{ and } \tilde{A}_i^{\text{MTriGS}}$$
Second strategy:

- solve local systems $A_i x_i = b_i$ (approximately) via
  - *direct solver* if the system is not too large ($\#\text{DOF} < 20\,000$)
  - *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: $\#\text{DOF} \approx 10^6$
  $\Rightarrow$ local multigrid mandatory

**Using local multigrid within global MLDD is one of the core ideas of FEAST’s solution method!**
Comparison of the two strategies (black dot = favoured strategy):

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage requirements</td>
<td>●●●●●</td>
<td></td>
</tr>
<tr>
<td>flexibility</td>
<td>○○●○</td>
<td></td>
</tr>
<tr>
<td>‘black box’ character</td>
<td>○●●○</td>
<td></td>
</tr>
<tr>
<td>arithmetic costs</td>
<td>●○○○</td>
<td></td>
</tr>
<tr>
<td>communication vs. computation</td>
<td>○○○●</td>
<td></td>
</tr>
<tr>
<td>influence on global iteration</td>
<td>○○○●</td>
<td></td>
</tr>
<tr>
<td>load balancing</td>
<td>●○○○</td>
<td></td>
</tr>
<tr>
<td>using co-processors</td>
<td>○○○●</td>
<td></td>
</tr>
</tbody>
</table>
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
The ScaRC Concept

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)
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

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...
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
Remedy:

- adaptively damp the suboptimal coarse grid correction:
  \[ x^{(l)} \leftarrow x^{(l)} + \alpha P^{(l)} x^{(l-1)} \]

(subscripts for local subdomains omitted)

- minimise error in energy norm:
  \[ (x^{(l)} - x^*)^T A^{(l)} (x^{(l)} - x^*) \]

  \( (x^* \text{ exact solution}) \)

- for symmetric \( A^{(l)} \):
  \[ \alpha = \frac{c^{(l)^T} d^{(l)}}{c^{(l)^T} A^{(l)} d^{(l)}} \]

  \( (c^{(l)} := P^{(l)} x^{(l-1)} \text{ prolonged coarse grid correction,} \)

  \( d^{(l)} := b^{(l)} - A^{(l)} x^{(l)} \text{ current defect}) \)
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:
- prolongedated 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

(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 $1e-6, MG([V|F] 22, 0.7) - JAC-D$
V-cycle (left) and F-cycle (right)
• anisotropic example
• consider top right subdomain again
• local solver MG(1e-6, [V,F]22, 0.6)–JAC–D
ACGC - Numerical Examples

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
  ⇒ not reliable
- standard W-cycle: always converges, but strange oscillations depending on number of grid levels
- W-cycle + ACGC: always converges, no oscillations
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

anisotropic subdomain, ADiTriGS smoother

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

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

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:

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

⇒ How many FLOPs are needed per DOF to gain one digit?

- 2-layer-ScaRC solver with Jacobi smoother:
  \( \text{MG}(1e^{-6}, V11) \_\_ \text{MG}(1e^{-1}, V22, 0.7) - \text{JAC-D}\_\_\_\_\_D \)

- 2-layer-ScaRC solver with ADiTriGS smoother:
  \( \text{MG}(1e^{-6}, F22) \_\_ \text{MG}(1e^{-1}, V22) - \text{ADI-D}\_\_\_D \)

- only vary maximum MG level, fix coarse grid level to 1
ACGC - Numerical Examples

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

isotropic configuration (left), anisotropic configuration (right)
ACGC - Numerical Examples

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

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...
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

CROSSOVER

0.0  0.80  1.6  2.4
1-layer-ScaRC vs. 2-layer-ScaRC

ASMO

0.0  0.028  0.057  0.085

u
Crossover-ISO (left), Crossover-Aniso (right), ASMO (bottom)
1-layer-ScaRC vs. 2-layer-ScaRC

Number of global smoothing steps
(⇒ amount of communication)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>88</td>
<td>352</td>
<td>776</td>
<td>32</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>96</td>
<td>464</td>
<td>1016</td>
<td>32</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>104</td>
<td>552</td>
<td>1296</td>
<td>32</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>8</td>
<td>112</td>
<td>624</td>
<td>1504</td>
<td>32</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>9</td>
<td>120</td>
<td>680</td>
<td>1688</td>
<td>32</td>
<td>32</td>
<td>56</td>
</tr>
</tbody>
</table>

- 2-layer-ScaRC successfully hides the local irregularities (anisotropies) from the global solver
- ⇒ strongly favoured in a massively parallel computation
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
  ⇒ there is ‘nothing to hide’ within a 2-layer-ScaRC solver
- future attempts: massively parallel computations, more complicated physical equations, 3D
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

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…
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)
solid body $\bar{\Omega} \subset \mathbb{R}^3$
boundary $\Gamma := \partial \Omega$, $\Gamma = \Gamma_D \cup \Gamma_N$
deformation mapping $\Phi : \bar{\Omega} \rightarrow \mathbb{R}^3$ with $\det(\nabla \Phi) > 0$
displacements $u(x) = (u_1(x), u_2(x), u_3(x))^T$ of material point $x \in \bar{\Omega}$, $\Phi = \text{id} + u$
kinematic relation between displacements and strains:
linearised strain tensor $\varepsilon = \frac{1}{2}(\nabla u + \nabla u^\top)$
Stress principle of Euler and Cauchy: existence of a vector field $t: \bar{\Omega} \times S_1 \rightarrow \mathbb{R}^3$ (Cauchy stress vector) with:

- for arbitrary $V \subset \bar{\Omega}$ (g applied surface force):
  $$t(x, n) = g(x), \quad x \in \Gamma_N \cap \partial V$$

- axiom of force balance ($f$ applied body force):
  $$\int_V f(x) \, dx + \int_{\partial V} t(x, n) \, da = 0$$

- axiom of balance of angular momenta:
  $$\int_V \mathbf{x} \times f(x) \, dx + \int_{\partial V} \mathbf{x} \times t(x, n) \, da = 0$$
Cauchy's theorem:
existence of a symmetric tensor field \( \sigma : \bar{\Omega} \to \mathbb{M}^3 \) (Cauchy stress tensor) that satisfies

- \( t(x, n) = \sigma(x)n, \quad x \in \bar{\Omega}, n \in S_1 \),
- the PDE
  \[ -\text{div}(\sigma(x)) = f(x), \quad x \in \Omega, \]
- and the boundary conditions
  \[ \sigma(x)n = g(x), \quad x \in \Gamma_N. \]
Basic Relations of Elasticity

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

\[ \sigma = 2\mu \varepsilon + \lambda \text{tr}(\varepsilon)I \]

- Lamé constants \( \mu \) and \( \lambda \)
- relation to Young’s modulus \( E \) and Poisson ratio \( \nu \):

\[ \mu = \frac{E}{2(1 + \nu)}, \quad \lambda = \frac{E\nu}{(1 + \nu)(1 - 2\nu)} \]

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

<table>
<thead>
<tr>
<th>Material</th>
<th>( E ) [ N/m(^2)]</th>
<th>( \nu ) [-]</th>
<th>( \mu ) [ N/m(^2)]</th>
<th>( \lambda ) [ N/m(^2)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steel/Iron</td>
<td>( 2.1 \cdot 10^{11} )</td>
<td>0.29</td>
<td>( 8.14 \cdot 10^{10} )</td>
<td>( 1.1 \cdot 10^{11} )</td>
</tr>
<tr>
<td>Lead</td>
<td>( 1.6 \cdot 10^{10} )</td>
<td>0.44</td>
<td>( 5.6 \cdot 10^{9} )</td>
<td>( 4.1 \cdot 10^{10} )</td>
</tr>
<tr>
<td>Rubber</td>
<td>( 2.5 \cdot 10^{7} )</td>
<td>0.499–0.5</td>
<td>( 8.2 \cdot 10^{6} )</td>
<td>( 4.1 \cdot 10^{9} )</td>
</tr>
</tbody>
</table>
Basic Relations of Elasticity

resulting boundary value problem - the Lamé Equation:

\[-2\mu \, \text{div}(\varepsilon) - \lambda \, \nabla \, \text{div}(u) = f \quad \text{in } \Omega\]
\[u = \bar{u} \quad \text{on } \Gamma_D\]
\[\sigma n = g \quad \text{on } \Gamma_N\]

bilinear form:

\[k(u, v) := \int_{\Omega} 2\mu \varepsilon(u) : \varepsilon(v) + \lambda \, \text{div}(u) \, \text{div}(v) \, dx\]

weak formulation: Find \(u - \bar{u} \in V := \{v \in H^1(\Omega)^3 \mid v = 0 \text{ on } \Gamma_D\}\) such that

\[k(u, v) = \int_{\Omega} f \cdot v \, dx + \int_{\Gamma_N} g \cdot v \, da, \quad v \in V\]
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…
Operator Splitting

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

\[
- 2\mu \text{ div}(\varepsilon) - \lambda \nabla \text{ div}(u)
\]

\[
= \begin{pmatrix} (2\mu + \lambda)\partial_{11} + \mu\partial_{22} & (\mu + \lambda)\partial_{12} \\ (\mu + \lambda)\partial_{21} & \mu\partial_{11} + (2\mu + \lambda)\partial_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}
\]

- rewrite left hand side of weak form:

\[
k(u, v) = \int_{\Omega} (2\mu + \lambda)\partial_1 u_1 \partial_1 v_1 + \mu \partial_2 u_1 \partial_2 v_1 \, dx
\]

+ ... (mixed terms)

+ \int_{\Omega} \mu \partial_1 u_2 \partial_1 v_2 + (2\mu + \lambda)\partial_2 u_2 \partial_2 v_2 \, dx

= ... (next slide)
Operator Splitting

- rewrite left hand side of weak form:

\[
k(u, v) = \int_{\Omega} \left[ \begin{pmatrix} 2\mu + \lambda & 0 \\ 0 & \mu \end{pmatrix} \nabla u_1 \right] \cdot \nabla v_1 \, dx + k_{12}(u_2, v_1)
\]

\[=: k_{11}(u_1, v_1)\]

\[+ k_{21}(u_1, v_2) + \int_{\Omega} \left[ \begin{pmatrix} \mu & 0 \\ 0 & 2\mu + \lambda \end{pmatrix} \nabla u_2 \right] \cdot \nabla v_2 \, dx
\]

\[=: k_{22}(u_2, v_2)\]

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

\[
\begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}
\]

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

- $K_{11}$ and $K_{22}$ correspond to scalar elliptic operators

(‘anisotropic Laplace operator’)
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...
Solution Concept

- basic iteration - block preconditioned Richardson method:

\[ u^{k+1} = u^k + \tilde{K}^{-1}(f - Ku^k) \]

- defect computation:

\[ f - Ku^k = \begin{pmatrix} f_1 - K_{11}u_1^k - K_{12}u_2^k \\ f_2 - K_{21}u_1^k - K_{22}u_2^k \end{pmatrix} \]

- e.g., block Jacobi preconditioner:

\[ \tilde{K}^{-1}_{\text{BJac}} = \begin{pmatrix} K_{11}^{-1} & 0 \\ 0 & K_{22}^{-1} \end{pmatrix} \]

- reduction to scalar components ⇒ exploiting FEAST concepts!

- note: each global matrix \( K_{ij} \) is represented by
local 9-band matrices corresponding to TP subdomains
Solution Concept

- Condition number of the preconditioned system:

\[ \kappa := \kappa(\tilde{K}^{-1}_{B\text{Jac}} K) \leq \frac{1}{c_K} \left(1 + \frac{1}{1 - 2\nu}\right) \]

- Three observations:
  1. \( \kappa \) does not depend on mesh size parameter \( h \)
     \( \Rightarrow \) single-grid outer solver (i.e., Krylov method) can be sufficient
  2. \( \kappa \) depends on Korn's constant \( c_K = c_K(\Omega, \Gamma_D) \)
     \( \Rightarrow \) convergence depends on boundary conditions / geometry (e.g., cantilever beam)
     (Korn's inequality: \( \exists c_K = c_K(\Omega, \Gamma_D) > 0 : \|\varepsilon(v)\|_0^2 \geq c_K \|v\|_1^2, v \in V \))
  3. \( \kappa \) depends on Poisson ratio \( \nu \)
     \( \Rightarrow \) 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

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...
Selected Numerical Examples

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

4 × 2 subdomains (left), 8 × 4 subdomains (right)
dependence on mesh anisotropy.

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

- **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

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!
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

ISO (left), ANISO (right)

- Convergence behaviour of 1-layer-ScaRC and 2-layer-ScaRC variants (nearly) independent of the configuration
- The standard solver suffers significantly
- 2-layer-ScaRC always needs 1 iteration per call, 1-layer-ScaRC between 6 (iso) and 70 (aniso) iterations
  ⇒ Communication amount of 2-layer-ScaRC variant much smaller!
arithmetic costs of 2-layer-ScaRC very high on \textbf{ISO} configuration

- arithmetic costs of 1-layer-ScaRC and the standard solver increase drastically on the \textbf{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: $4 \times 1$ subdomains, $4.2 \times 10^6$ vertices, 4 processors
- largest problem: $16 \times 8$ subdomains, $134.2 \times 10^6$ vertices, 128 processors
good weak scalability

comparable to scalar FEAST solvers

parallel efficiency of the scalar FEAST library fully transfers to elasticity solvers
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

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…
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 $u/p$ formulation

leads to Stokes-like saddle point problem $Ax = b$ with

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

$C$ contains compressibility and, if necessary, stabilisation terms (e.g., for $Q_1/Q_1$)

distinguish segregated (uncoupled) and coupled solution methods
Segregated methods:

- solve subsystems for $u$ and $p$
- Uzawa / pressure Schur complement methods:

$$p_{k+1} = p_k + \tilde{S}^{-1}(B^T A^{-1} f - (B^T A^{-1} B - C)p_k),$$

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

- block-triangular preconditioners:

$$
\begin{pmatrix}
  u_{k+1} \\
  p_{k+1}
\end{pmatrix} =
\begin{pmatrix}
  u_k \\
  p_k
\end{pmatrix} +
\begin{pmatrix}
  \tilde{A} & 0 \\
  B^T & -\tilde{S}
\end{pmatrix}^{-1}
\begin{pmatrix}
  f - Au_k - B p_k \\
  0 - B^T u_k - C p_k
\end{pmatrix},
$$

- essential for both variants:
  - a good Schur complement preconditioner
Segregated methods (continued):

- Schur complement preconditioning often involves solution of scalar systems
  ⇒ scalar ScaRC solvers!
- solve $\tilde{A}$-systems with techniques from compressible elasticity
  ⇒ 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
  ⇒ 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)
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!