Fast Solvers for Large-Scale Systems of Finite Element Equations

Ulrich Langer

Radon Institute for Computational and Applied Mathematics
Austrian Academy of Sciences

Institute of Computational Mathematics
Johannes Kepler University Linz

Special Research Program SFB F013 on
Numerical and Symbolic Scientific Computing

5. LS-DYNA Forum 2006, Ulm, 12. und 13. Oktober 2006

Outline

Where do we need Fast Solvers ?

Implicite time discretization of dynamic problems like

$$\rho \frac{\partial^2 u}{\partial^2 t} + c \frac{\partial u}{\partial t} - \frac{E}{2(1+\nu)} \left(\Delta u + \frac{1}{1-2\nu} \nabla(\nabla \cdot u) \right) = f(x,t) \quad (1)$$

leads to the solution of large linear systems of FE equations

$$\boxed{A \, \underline{u} \; = \; \underline{b} \quad \text{in } \mathbb{R}^n} \qquad (2)$$

at each time step, where the system matrix A has the structure

$$A = M + \alpha\tau D + \beta\tau^2 K \qquad (3)$$

with the time step $\tau = \Delta t$.

Where do we need Fast Solvers ? (cont.)

Similarly, **harmonic excitations**

$$f(x,t) = \hat{f}(x)\exp(i\omega t), \ldots \tag{4}$$

and **static** or **quasistatic** boundary value problems also lead to the solution of large linear systems of FE equations of the form

$$\boxed{A\,\underline{u} \;=\; \underline{b} \quad \text{in } \mathbb{R}^n}$$

with system matrices of the form (without damping)

$$A = K - \omega^2 M \tag{5}$$

and

$$A = K, \tag{6}$$

respectively.

Outline

Gaussian Elimination: Ax=b

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}.$$

Gaussian Elimination: Ax=b

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}.$$

Gaussian Elimination: Ax=b

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}.$$

Gaussian Elimination: Ax=b

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}.$$

Gaussian Elimination: Ax=b

Gaussian Elimination = LU - Decomposition:

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}.$$

$$U\underline{x} = \underline{c} = L^{-1}\underline{b} \quad \Longrightarrow \quad LU\underline{x} = \underline{b} \quad \Longrightarrow \quad A = LU$$

Complexity Estimate:

- ops $\approx BW^2 n = n^{\frac{2d-2}{d}} n = n^{\frac{3d-2}{d}}$
- Memory $\approx BWn = n^{\frac{d-1}{d}} n = n^{\frac{2d-1}{d}}$

Gaussian Elimination: Ax=b

Gaussian Elimination = LU - Decomposition:

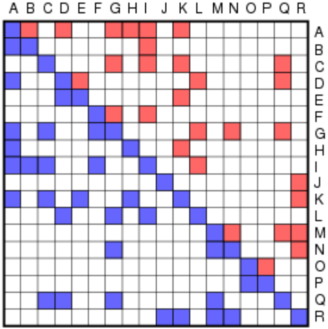
$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}.$$

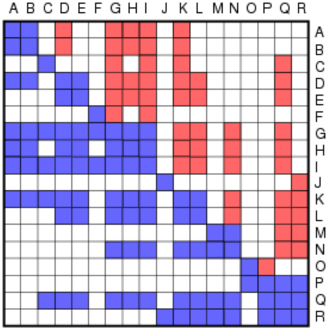
$$U\underline{x} = \underline{c} = L^{-1}\underline{b} \quad \Longrightarrow \quad LU\underline{x} = \underline{b} \quad \Longrightarrow \quad A = LU$$

Complexity Estimate:

- ops $\approx BW^2 n = n^{\frac{2d-2}{d}} n = n^{\frac{3d-2}{d}}$
- Memory $\approx BWn = n^{\frac{d-1}{d}} n = n^{\frac{2d-1}{d}}$

Sparse Direct Methods

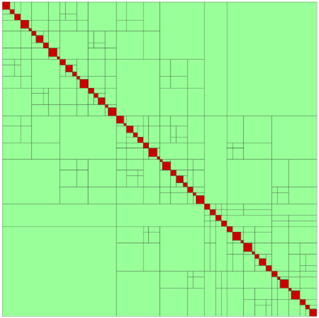


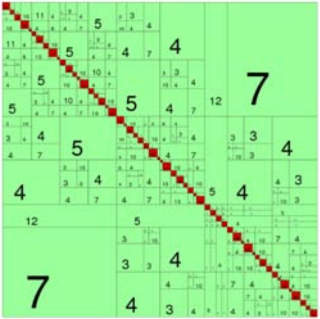


Complexity Estimate:

- Factorization: ops $\approx n^{3/2}$ for $d = 2$ and n^2 for $d = 3$
- Solution: ops $\approx n \log(n)$ for $d = 2$ and $n^{4/3}$ for $d = 3$

H-Matrix Technology





- $A \quad \Longrightarrow \quad \mathcal{A}\colon \|A - \mathcal{A}\| \le \varepsilon \|A\| \quad \Longrightarrow \quad \tilde{\mathcal{A}} = \mathcal{L}\mathcal{U}$
- $\varepsilon \approx$ = discretisation error $\Longrightarrow$ Solver !
- $\varepsilon = 10^{-1} \ldots 10^{-2} \Longrightarrow$ Preconditioner $C = \mathcal{L}\mathcal{U}$!
- Complexity $\approx n$ up to a polylogarithmical factor !!

Outline

Gauss' Idea for an Iteration

„Faſt jeden Abend mache ich eine neue Auflage des Tableau, wo immer leicht nachzuhelfen ist. Bei der Einförmigkeit des Meſſungsgeschäfts gibt dies immer eine angenehme Unterhaltung; man ſieht daran auch immer gleich, ob etwas Zweifelhaftes eingeſchlichen iſt, was noch wünſchenswert bleibt uſw. Ich empfehle Ihnen dieſen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigſtens nicht, wenn Sie mehr als zwei Unbekannte haben. Das indirecte Verfahren läßt ſich halb im Schlafe ausführen oder man kann während desſelben an andere Dinge denken."

C. F. Gauss in [2]

Jacobi and Gauss-Seidel Iterations

Our system of FE equations

$$\underline{A}\,\underline{u} = \underline{b} \qquad \Longleftrightarrow \qquad \sum_{j=1}^{n} a_{ij} u_j = f_i, \quad i = 1, \cdots, n$$

can be written in the fixed point form as follows

$$u_i = -\frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} u_j + \sum_{j=i+1}^{n} a_{ij} u_j \right) + \frac{1}{a_{ii}} f_i$$

Jacobi:

$$u_i^{k+1} = -\frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} u_j^k + \sum_{j=i+1}^{n} a_{ij} u_j^k \right) + \frac{1}{a_{ii}} f_i$$

Jacobi and Gauss-Seidel Iterations

Our system of FE equations

$$\boxed{A\,\underline{u} = \underline{b}} \qquad \Longleftrightarrow \qquad \boxed{\sum_{j=1}^{n} a_{ij}u_j = f_i, \quad i = 1,\cdots,n}$$

can be written in the fixed point form as follows

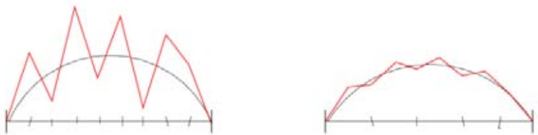
$$\boxed{u_i = -\frac{1}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}u_j + \sum_{j=i+1}^{n} a_{ij}u_j\right) + \frac{1}{a_{ii}} f_i}$$

Gauss-Seidel: $$u_i^{k+1} = -\frac{1}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}u_j^{k+1} + \sum_{j=i+1}^{n} a_{ij}u_j^{k}\right) + \frac{1}{a_{ii}} f_i$$

Properties of GSI for FE Matrices like A=K

$$u_i^{k+1} = -\frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} u_j^{k+1} + \sum_{j=i+1}^{n} a_{ij} u_j^k \right) + \frac{1}{a_{ii}} f_i$$

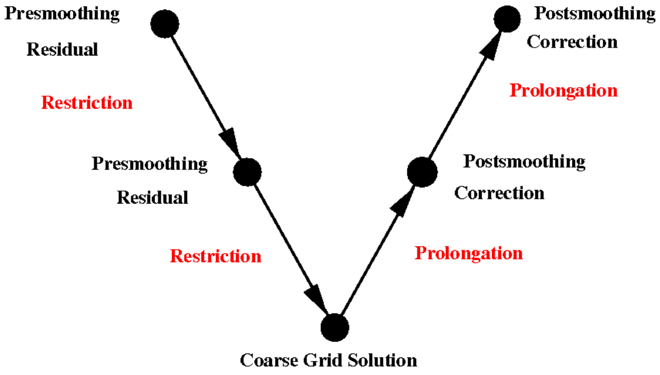
- Slow convergence, d.h. convergence rate $q = 1 - O(h^2)$
- Fast smoothing of $\underline{e}^k = \underline{u} - \underline{u}^k$ rsp. $\underline{r}^k = A\underline{e}^k = \underline{b} - A\underline{u}^k$



$\Rightarrow$ Combine SMOOTHING with COARSE-GRID-CORRECTION

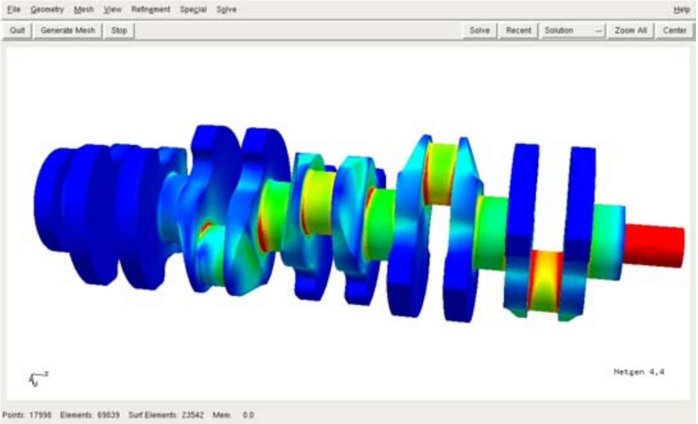
Geometrical Multigrid Methods (MGM)

$\underline{u}_l^{old}$ $\quad A_l\, \underline{u}_l = \underline{b}_l \quad \text{in } \mathbb{R}^{n_l}$ $\quad \underline{u}_l^{new}$



Result: Linear Complexity: ops = $O(n_l \ln(\varepsilon^{-1}))$, M = $O(n_l)$

Netgen/NGSolve: Von-Mises Stress in a Crank-shaft



69839 tets, p = 3, 1,105,983 dof, 34 min on 2.4 GHz PC 1.2 GB

From Geometric to Algebraic MGM

In Practice, only the fine grid information is usually available:

- the mesh $\tau_h = \tau_l$ and the set $\omega_h = \omega_l$ of nodes,
- the system matrix $A_h = A_l$ and the rhs $\underline{b}_h = \underline{b}_l$.

Then we want to construct the coarse "grid" components from the given fine grid information:

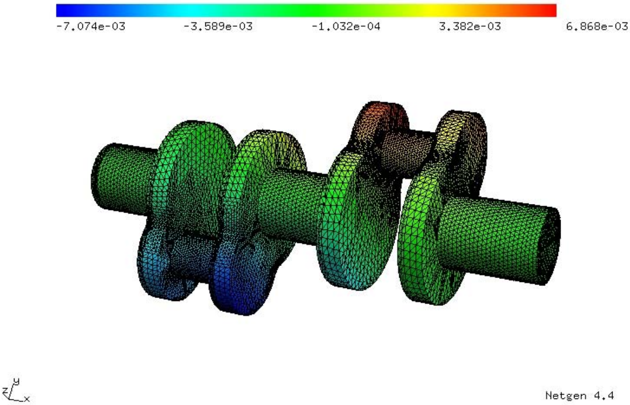
- $\omega_{j-1} = (\omega_j)_C$, where ω_j is split into $(\omega_j)_C$ and $(\omega_j)_F$ on the basis the matrix graph,
- prolongation P^j_{j-1} is definded by interpolation.

Once P^j_{j-1} is defined, we easily get the

- restriction $R^j_{j-1} = (P^j_{j-1})^T$,
- coarse grid matrix $A_{j-1} = P^j_{j-1} A_j R^j_{j-1}$,

where j is runing from l to 2.

AMG: Crank-shaft



171,264 tets, p = 1, 107,625 dof, $\varepsilon = 10^{-8}$, 34 its, 120 sec

Preconditioned Conjugate Gradient (PCG) Method

PCG: $\underline{u}^{new} \Leftarrow \underline{u}^{old}$

{Initialization step}
$\underline{u} \Leftarrow \underline{u}^{old}$ ...

while $\delta > \varepsilon^2 \delta^0$ **do**

$\alpha \Leftarrow \delta / (A\underline{s}, \underline{s})$

$\underline{u} \leftarrow \underline{u} + \alpha \underline{s}$

$\underline{r} \leftarrow \underline{r} + \alpha A \underline{s}$

$\underline{w} \Leftarrow C^{-1} \underline{r}$

$\hat{\delta} \Leftarrow (\underline{w}, \underline{r})$

$\beta \leftarrow \hat{\delta} / \delta$

$\delta \leftarrow \hat{\delta}$

$\underline{s} \leftarrow \underline{w} + \beta \underline{s}$

end while

Preconditioning step: $\underline{w} = C^{-1} \underline{r}$

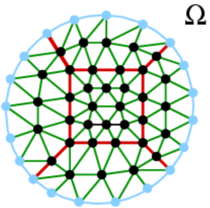
- **AMG:** $\underline{w} = C^{-1}\underline{r} = (I - E)A^{-1}\underline{r}$ means the application of 1 V-cycle to $A\underline{w} = \underline{r}$ with the initial guess $\underline{w}^{ini} = 0$.
- $C = \mathcal{L}\mathcal{U}$ is a crude ($\epsilon = 10^{-1}$) $\mathcal{H}$**-LU-Factorization** of A, i.e. $\underline{w}$ solves the system $\mathcal{L}\mathcal{U}\underline{w} = \underline{r}$.

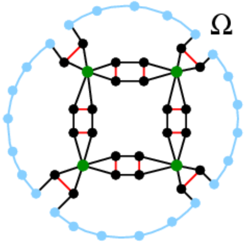
Result: Linear Complexity Solvers !

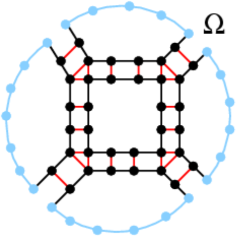
Outline

Primal-, DP-, Dual Iterative Substructuring Methods

Primal IS $\longrightarrow$ FETI-DP $\longleftarrow$ FETI







Primal Iterative FE Substructuring Methods

$$\begin{pmatrix} K_C & K_{CI} \\ K_{IC} & K_I \end{pmatrix} \begin{pmatrix} \underline{u}_C \\ \underline{u}_I \end{pmatrix} = \begin{pmatrix} \underline{f}_C \\ \underline{f}_I \end{pmatrix} \quad (7) \qquad \Leftrightarrow S_C \underline{u}_C = \underline{g}_C \quad (8)$$

- **Schur-Complement-PCG:** = PCG applied to (8) with the Schur-Complement Preconditioner (SCPC) $C_C \simeq S_C$
- **Inexact Solvers:** = PCG applied to (7) with the PC

$$C = \begin{pmatrix} I_C & E_{CI} \\ 0 & I_I \end{pmatrix} \begin{pmatrix} C_C & 0 \\ 0 & C_I \end{pmatrix} \begin{pmatrix} I_C & 0 \\ E_{IC} & I_I \end{pmatrix} \qquad (9)$$

where $C_I \simeq K_I$ and $E_{IC} = E_{CI}^T$ = stable discrete extension !

Finite Element Tearing and Interconnecting – Overview

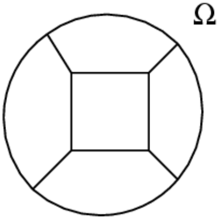
Farhat and Roux (1991)

- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity $\rightarrow$ Lagrange multipliers
- Elimination $\rightarrow$ dual problem
- PCG sub-space iteration

Finite Element Tearing and Interconnecting – Overview

Farhat and Roux (1991)

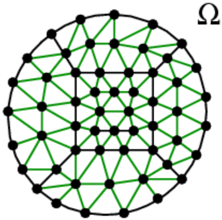
- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity → Lagrange multipliers
- Elimination → dual problem
- PCG sub-space iteration



Finite Element Tearing and Interconnecting – Overview

Farhat and Roux (1991)

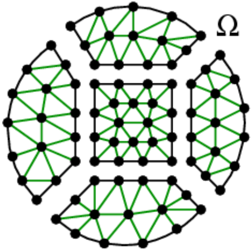
- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity → Lagrange multipliers
- Elimination → dual problem
- PCG sub-space iteration



Finite Element Tearing and Interconnecting – Overview

Farhat and Roux (1991)

- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity → Lagrange multipliers
- Elimination → dual problem
- PCG sub-space iteration

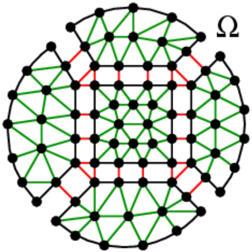


Tearing

Finite Element Tearing and Interconnecting – Overview

Farhat and Roux (1991)

- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity $\longrightarrow$ Lagrange multipliers
- Elimination $\longrightarrow$ dual problem
- PCG sub-space iteration

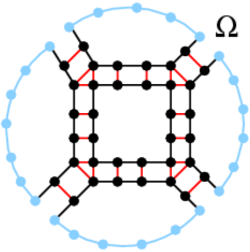


Interconnecting

Finite Element Tearing and Interconnecting – Overview

Farhat and Roux (1991)

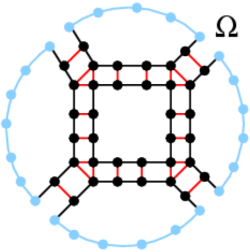
- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity → Lagrange multipliers
- Elimination → dual problem
- PCG sub-space iteration



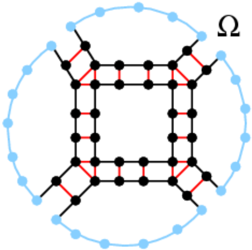
Finite Element Tearing and Interconnecting – Overview

Farhat and Roux (1991)

- Domain Decomposition
- Conformal mesh
- Separate d.o.f.
- Continuity $\longrightarrow$ Lagrange multipliers
- Elimination $\longrightarrow$ dual problem
- PCG sub-space iteration



Finite Element Tearing and Interconnecting – Formulas



The unconstraint minimization problem (**??**) is obviously equivalent to the constraint MP

$$\min_{\underline{v}_C} \sum_{i=1}^{p} \left(\frac{1}{2} (S_{C,i} \underline{v}_{C,i}, \underline{v}_{C,i}) - (\underline{g}_{C,i}, \underline{v}_{C,i}) \right) \quad (10)$$

subject to $B\underline{v} = 0$, with $\underline{v} = (\underline{v}_{C,1}, \ldots, \underline{v}_{C,p})$.

The constraint MP (10) is equivalent to the SPP

$$\begin{pmatrix} S_{C,1} & & & B_1^\top \\ & \ddots & & \vdots \\ & & S_{C,p} & B_p^\top \\ B_1 & \ldots & B_p & 0 \end{pmatrix} \begin{pmatrix} \underline{u}_{C,1} \\ \vdots \\ \underline{u}_{C,p} \\ \underline{\lambda} \end{pmatrix} = \begin{pmatrix} \underline{g}_{C,1} \\ \vdots \\ \underline{g}_{C,p} \\ \underline{0} \end{pmatrix} \Longleftrightarrow F\underline{\lambda} = \underline{d}.$$

Finite Element Tearing and Interconnecting – Features

- PCG iteration and preconditioning via local Neumann and Dirichlet solvers
- Allows massive parallelization
- Spectral Condition number $\text{cond}_2(C^{-1}K) = \mathcal{O}((1 + \log(H/h))^2)$
- Robust w.r.t. coefficient jumps

Mandel/Tezaur, 1996
Klawonn/Widlund, 2001
Brenner, 2002

Finite Element Tearing and Interconnecting – News

New versions:

- Dual-Primal FETI (FETI-DP):
  Farhat et al. (2000), Klawonn/Widlund/Dryja (2002),...
- Balanced Domain Decomposition by Constraints (BDDC):
  Dohrmann (2003), Mandel/Dohrmann (2003),...
- Inexact Versions (avoid elimination !):
  FETI: Klawonn/Widlund (2000)
  FETI-DP: Klawonn/Rheinbach (2005)
  BDDC: Dohrmann (2005)

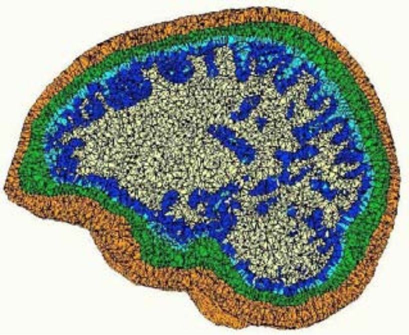
New Applications:

- Structural Mechanics, Contact, Helmholtz, Maxwell etc.

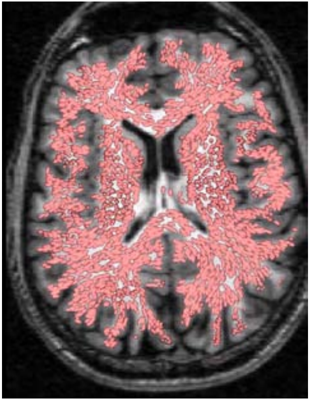
Outline

Medical Source Reconstraction: Problem Describtion

Lead field basis approach to source reconstruction problems developed by Grasedyck, Hackbusch and Wolters (MPI Leipzig):



5 layer head model
Triangulation is given, $n = 147287$



conductivity $\sigma : \Omega \to \mathbb{R}^{3\times 3}$ is given

Medical Source Reconstraction: Model and Results

- Submodel in the SRP: Neumann BVP

$$-\mathrm{div}(\sigma \nabla u) = f \text{ in } \Omega \subset \mathbb{R}^3 \text{ and } \partial_n u = 0 \text{ on } \partial\Omega \qquad (11)$$

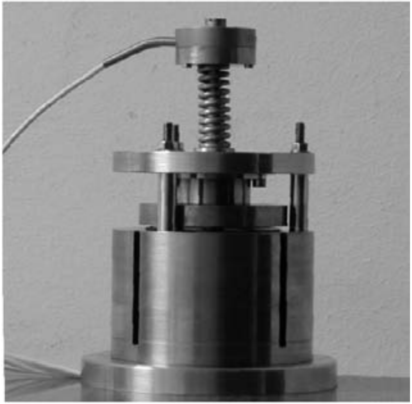
- Finite Element discretisation $\rightsquigarrow$ $Ax = b$
- The system has to be solved for $\approx \mathbf{r} = 400$ right-hand sides
- Stopping criterion: $\|Ax - b\| \leq 10^{-8}\|b\|$
- Machine: SUNFire, **900 MHz**, **single processor**

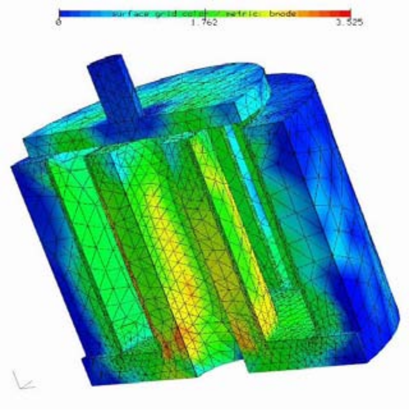
	Pardiso	$\mathcal{H}$-LU($\varepsilon = 10^{-6}$)	PEBBLES
Setup	237	468	13
Solve	2.4	1.0	10
Total	1197	868	4013

Pardiso (Gärtner/Schenk) multiple rhs optimisation
$\mathcal{H}$-LU (Grasedyck/LeBorne/Kriemann) multiple rhs optimisation
PEBBLES (Langer/Haase/Reitzinger) multiple rhs optimisation

Outline

Magnetic Valve: Model and FE Model





Joint work with M. Kaltenbacher, R. Lerch, M. Schinnerl and J. Schöberl !

LAMÉ-NAVIER coupled with MAXWELL

- LAMÉ-NAVIER's Equations + BC + IC:

$$\rho \frac{\partial^2 d}{\partial^2 t} + c \frac{\partial d}{\partial t} - \frac{E}{2(1+\nu)} \left(\Delta d + \frac{1}{1-2\nu} \nabla(\nabla \cdot d) \right) = f_V(A)$$

- MAXWELL's Equations + BC + IC:

$$\sigma \frac{\partial A}{\partial t} + \mathrm{curl}\left(\frac{1}{\mu(|\mathrm{curl}(A)|)} \mathrm{curl}(A)\right) + \sigma \frac{\partial d}{\partial t} \times \mathrm{curl}(A) = S$$

- Coupling terms:
  - LAMÉ: (volume) Lorentz-forces + (surface) interface forces
  - MAXWELL: electromotive force + $\Omega_{mag} = \Omega_{mag}(d)$

Implicite Time Integration of the FE Equations

Mechanical nodal FE Mesh

1. **Mechanical Predictor**
   $\tilde{d} = d_n + \Delta t v_n + 0.5\Delta t^2(1-2\beta)a_n$
   $\tilde{v} = v_n + (1-\gamma)\Delta t a_n$
5. **Mechanical Solver**
   Multi-Grid-Solver:

   $$M^* a_{n+1} = f^*_{n+1}$$

   with $M^* = M + \gamma\Delta t C + \beta\Delta t^2 K$
   $f^*_{n+1} = f_{n+1} - K\tilde{d} - C\tilde{v}$
   Corrector:
   $d_{n+1} = \tilde{d} + \beta\Delta t^2 a_{n+1}$
   $v_{n+1} = \tilde{v} + \gamma\Delta t a_{n+1}$
6. **Convergence Test**

Magnetic edge FE Mesh

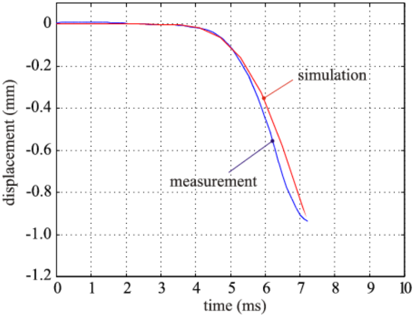
2. **Update the magnetic quantities**
3. **Magnetic Solver**
   Predictor: $\tilde{A} = A_n + (1-\alpha)\Delta t R_n$
   Multi-Grid-Solver:

   $$L^* R_{n+1} = Q^*_{n+1}$$

   with $L^* = L + \alpha\Delta t P$
   $Q^*_{n+1} = Q_{n+1} - P\tilde{A}$
   Corrector: $A_{n+1} = \tilde{A} + \alpha\Delta t R_{n+1}$
4. **Calculate the induction and the magnetic forces**

Simulation Results

Simulation vs Measurments



Parallel Processing

P	1	4	16	16
m	2	2	2	30
[sec]	453	99	31	458
Sp	1.0	4.6	14.5	0,93

$\Longrightarrow$ Valve Movie

$\Longrightarrow$ Cave Movie

Outline

Summary I

- Sparse Direct Methods
- Fast Iterative Methods
- Preconditioners and Krylow-Space-Iterations
- DDM as Parallelization Technology

Summary II

Efficient solvers are hybrid methods which exploit the best properties of both worlds, the world of direct and itereative methods:

- $\mathcal{H}$-matrix techniques:
  - From Solver to Preconditioner:
  - $\mathcal{H}$-LU($\varepsilon = 10^{-6}$) to C=$\mathcal{H}$-LU($\varepsilon = 10^{-1}$)
- Algebraic Multigrid
  - iterative methods as smoothers combined with
  - sparse direct solvers for the systems on the coarsest grid
- Domain Decomposition Methods
  - sparse direct solvers on the subdomains combined with
  - iterative solvers for the interface problems

Thanks to

- Dr. K. Gärtner (WIAS, Berlin) for PARDISO
- Prof.Dr. W. Hackbusch and his colleagues (MPI, Leipzig) for $\mathcal{H}$-matrix-results and the hlib
- Prof.Dr. J. Schöberl (RWTH Aachen, RICAM Linz) for NGSolve results
- my colleagues and my former colleagues from Linz

References

- Sparse Direct Methods: e.g. PARDISO: `www.computational.unibas.ch/cs/scicomp`
- $\mathcal{H}$-Matrix-Software: `www.hlib.org`
- NGSolve: `www.femworks.at`
- PEBBLES: `www.numa.uni-linz.ac.at/Research/Projects/pebbles.html`
- Multigrid Methods: `www.mgnet.org`
- AMG (Stüben): `www.scai.fraunhofer.de/samg.html`
- Boomer-AMG: `www.llnl.gov/CASC`
- Domain Decomposition Methods: `www.ddm.org`