μ-Finite Element Analysis of Human Bone Structures

Peter Arbenz

Chair of Computational Science, ETH Zürich
E-mail: arbenz@inf.ethz.ch
http://people.inf.ethz.ch/arbenz/
Collaborators

- Chair of Computational Science, ETH Zurich
  - Uche Mennel
  - Marzio Sala
  - Cyril Flaig
  - Yves Ineichen
  - Sumit Paranjape

- Institute for Biomechanics, ETH Zurich
  - Harry van Lenthe
  - Andreas Wirth
  - David Christen
  - Ralph Müller

- IBM Zurich Research Lab
  - Costas Bekas
  - Alessandro Curioni
Outline of the talk

1. Introduction
2. Solving the system of equations
3. Numerical experiments
4. Conclusions
In healthy bone, there is continuous matrix remodeling of bone. Up to 10% of all bone mass may be undergoing remodeling at any point in time.

Osteoporosis is a disease characterized by an imbalance between bone resorption and bone formation.

Osteoporosis causes a low bone mass and a deterioration of the bone microarchitecture.

[From http://www.osteoswiss.ch]
In Switzerland, the risk for an osteoporotic fracture in women above 50 years is about 50%, for men the risk is about 20%.

Mostly affected bones are vertebra, femur, and radius.

Not all fractures are actually noticed, but osteoporosis means a considerable reduction in the quality of life for many affected people.

Enormous impact on individuals, society and health care systems (as health care problem second only to cardiovascular diseases).

Osteoporosis is tried to be detected by Densitometry or other imaging technologies such as Quantitative Computer Tomography (QCT). These techniques allow to estimate / measure the bone mineral density.
Introduction / Osteoporosis (cont.)

- Since global parameters like bone density do not admit to predict the fracture risk, patients have to be treated in ways that take into account the structure of individual bones.
- Patient-specific finite element (FE) models of bones based on 3D high-resolution CT images become increasingly attractive for evaluation of stiffness and strength \textit{in vivo}.
- This project is a part of the VPHOP or the Osteoporotic Virtual Physiological Human Project in FP7. One of the goals: predict the absolute risk of fracture in patients with low bone mass.
In vitro/in vivo assessment of bone strength

High-res. pQCT → Create FE → Strength

Strains (%)

pQCT: peripheral Quantitative Computed Tomography

Courtesy Harry van Lenthe
University and ETH Zurich
In vitro/in vivo assessment of bone strength (cont’d)

1. $\mu$CT / high-resolution pQCT scan

2. CT voxel information is converted directly into 8-noded hexahedral elements $\rightarrow$ large number of degrees of freedom


4. Bone tissue is assumed to be isotropic
   Elastic parameters (elasticity modulus $E = 17$ GPa, Poisson ratio $\nu = 0.3$) are taken from measurements.

5. A scalable, robust and reliable parallel solver

6. A scalable high-performance computer
The mathematical model

- Equations of **linearized 3D elasticity** (pure displacement formulation): Find displacement field \( u \) that minimizes total potential energy

\[
\int_{\Omega} \left[ \mu \varepsilon(u) : \varepsilon(u) + \frac{\lambda}{2} (\text{div } u)^2 - f^T u \right] \, d\Omega - \int_{\Gamma_N} g^T u \, d\Gamma,
\]

with Lamé’s constants \( \lambda, \mu \), volume forces \( f \), boundary tractions \( g \), symmetric strain tensor

\[
\varepsilon(u) := \frac{1}{2} (\nabla u + (\nabla u)^T).
\]

- In typical analyses: Vertical displacements at top and bottom boundary planes are (partially) prescribed (Dirichlet boundary conditions).
  Otherwise boundary is free (homogeneous Neumann b.c.)
Continuum finite element (FE) models using $\mu$FE

Size of voxels
10-100$\mu$m $\rightarrow$ $\mu$FE analysis.
Solving the system of equations

- System of equation
  \[ Kx = b \]

\( K \) is very large, sparse, symmetric positive definite.

- Original (Rietbergen 1996 [3]) approach by people of ETH Biomechanics: preconditioned conjugate gradient (PCG) algorithm
  - element-by-element (EBE) matrix multiplication

\[
K = \sum_{e=1}^{n_{el}} T_e K_e T_e^T, \tag{1}
\]

Note: all element matrices are identical!

- diagonal (Jacobi) preconditioning
- very memory economic, slow convergence as problems get big
Solving the system of equations (cont.)

- Our first approach targeted at reducing the iteration count: PCG which smoothed aggregation AMG preconditioning [4] (Similar to Adams et al. SC’04 [1])
- Requires assembling $K$ $\rightarrow$ large memory requirement.
- Needs parallelization for distributed memory machines.
- Employ software: Trilinos (Sandia NL, Albuquerque, NM)
  In particular we use
  - Distributed (multi) vectors and (sparse) matrices (Epetra).
  - Domain decomposition (load balance) with ParMETIS
  - Iterative solvers and preconditioners (AztecOO)
  - Smoothed aggregation AMG preconditioner (ML)
  - Direct solver on coarsest level (AMESOS)
Our second approach targeted at reducing memory consumption:
Want to stick with PCG and devise a scalable (multigrid) preconditioner that can be built without assembling $K$. \cite{2}
- Typical operator complexity of SA-AMG is 1.4. Hope to reduce the memory requirements by a factor

$$3.5 = \frac{1.4}{0.4}.$$  

**Smoother:** Chebyshev polynomial smoother (Adams et al. 2003)
- Determine a polynomial that is small on the ‘upper part of the spectrum’.
- Only needs matrix-vector product.
- Perfect scalability.
Constructing a multigrid preconditioner without system matrix

- Smoothed Aggregation based Algebraic MultiGrid (SA-AMG) preconditioning does not need system matrix to build multigrid hierarchy!
- The aggregates are constructed by means of the matrix graph. Each node of the grid gives rise to a $3 \times 3$ block in the system matrix. Graph is an order of magnitude smaller. We use METIS to construct aggregates. Aggregates consist of ca. 100 nodes.
- Prolongator is based on near-kernel (near-nullspace) which in linear elasticity are translations and rotations that are determined from the node coordinates.
- Our new approach: Generate the second-finest multigrid level matrix-free with unsmoothed aggregates, and the further levels in the standard way.
Setup procedure for an abstract multigrid solver

1: Define the number of levels, \( L \)
2: \textbf{for} level \( \ell = 0, \ldots, L - 1 \) \textbf{do}
3: \hspace{1em} \textbf{if} \( \ell < L - 1 \) \textbf{then}
4: \hspace{2em} Define prolongator \( P_{\ell} \);
5: \hspace{2em} Define restriction \( R_{\ell} = P_{\ell}^T \);
6: \hspace{2em} \( K_{\ell+1} = R_{\ell} K_{\ell} P_{\ell} \);
7: \hspace{2em} Define smoother \( S_{\ell} \);
8: \hspace{1em} \textbf{else}
9: \hspace{2em} Prepare for solving with \( K_{\ell} \);
10: \hspace{1em} \textbf{end if}
11: \textbf{end for}
Smoothed aggregation (SA) AMG preconditioner I

1. Build adjacency graph $G_0$ of $K_0 = K$.
   (Take 3 $\times$ 3 block structure into account.)

2. Group graph vertices into contiguous subsets, called aggregates. Each aggregate represents a coarser grid vertex.
   - Typical aggregates: 3 $\times$ 3 $\times$ 3 nodes (of the graph) up to 5 $\times$ 5 $\times$ 5 nodes (if aggressive coarsening is used)
   - ParMETIS
   - Note: The matrices $K_1, K_2, \ldots$ need much less memory space than $K_0$!
Define a grid transfer operator:

- Low-energy modes (near kernel), in our case, the rigid body modes are ‘chopped’ according to aggregation

\[
B_\ell = \begin{bmatrix}
B_1^{(\ell)} \\
\vdots \\
B_{n_{\ell+1}}^{(\ell)}
\end{bmatrix}, \quad B_j^{(\ell)} = \text{rows of } B_\ell \text{ corresponding to grid points assigned to } j^{th} \text{ aggregate.}
\]

- Let \( B_j^{(\ell)} = Q_j^{(\ell)} R_j^{(\ell)} \) be QR factorization of \( B_j^{(\ell)} \) then

\[
B_\ell = \tilde{P}_\ell B_{\ell+1}, \quad \tilde{P}_\ell^T \tilde{P}_\ell = I_{n_{\ell+1}},
\]

with

\[
\tilde{P}_\ell = \text{diag}(Q_1^{(\ell)}, \ldots, Q_{n_{\ell+1}}^{(\ell)}) \quad \text{and} \quad B_{\ell+1} = \begin{bmatrix}
R_1^{(\ell)} \\
\vdots \\
R_{n_{\ell+1}}^{(\ell)}
\end{bmatrix}.
\]

Columns of \( B_{\ell+1} \) span the near kernel of \( K_{\ell+1} \).

- Notice: matrices \( K_\ell \) are \textit{not} used in constructing tentative prolongators \( \tilde{P}_\ell \), near kernels \( B_\ell \), and graphs \( G_\ell \).
Smoothed aggregation (SA) AMG preconditioner III

For elliptic problems, it is advisable to perform an additional step, to obtain smoothed aggregation (SA).

\[ P_\ell = (I_\ell - \omega_\ell D_\ell^{-1} K_\ell) \tilde{P}_\ell, \quad \omega_\ell = \frac{4/3}{\lambda_{\text{max}}(D_\ell^{-1} K_\ell)}, \]

smoothed prolongator

In non-smoothed aggregation: \( P_\ell = \tilde{P}_\ell \)
Computing $K_1 = P_0^T K_0 P_0$

- $P_0$ maps the degrees of freedom of the finest level (nodes) to the degrees of freedom of the second finest level (aggregates).
- In (block) column $i$ those rows are nonzero that correspond to nodes of aggregate $i$. 
Computing $K_1 = P_0^T K_0 P_0$ (cont.)

- Multiplying $K_0$ with the $i$-th column of $P_0$ introduces nonzeros at nodes directly connected to aggregate $i$. 

![Diagram showing the effect of multiplying $K_0$ with the $i$-th column of $P_0$]
Computing $K_1 = P_0^T K_0 P_0$ (cont.)

- Want to collapse different columns of $P_0$ in one column of $\hat{P}_0$ such that nonzeros in one column of $K_0 \hat{P}_0$ do not interfere with each other. Aggregates have to be sufficiently far apart.
- Since other neighbors of a neighbor can interfere with the original aggregate, neighbors of neighbors cannot be in equal columns.
- Can use graph coloring algorithm of the graph of $K_1$: aggregates that are two edges apart must have different colors (distance-2 coloring).
‘Matrix-free’ multigrid

- We do NOT form $K = K_0$ but do an element-by-element (EBE) matrix multiplication

$$K = \sum_{e=1}^{n_{el}} T_e K_e T_e^T$$

- We do NOT smooth the first prolongator $P_0$.
- Matrices $K_2, K_3, \ldots$ are formed with smoothed prolongators from $K_1, K_2, \ldots$
- All graphs, including $G_0$ are constructed.
- Memory savings (crude approximation): $\sim 4$.
- Needs clever construction of $K_1 = P_0^T K_0 P_0$ since forming $K_0 P_0 = \sum_{e=1}^{n_{el}} T_e K_e T_e^T P_0$ is infeasible.
Mesh partitioning

- **Load balance**: Each processor gets the same number of nodes
- **Minimize solver communication**: Minimize the surface area of the interprocessor interfaces
- **Crucial for efficient parallel execution**

Initial partition based on node coordinates

ParMETIS/recursive coordinate bisection (RCB) repartition
The Trilinos software framework

- The Trilinos Project is an effort to develop parallel solver algorithms and libraries within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific applications.


- We used the following packages
  - Epetra (distributed linear algebra objects)
  - AztecOO (PCG)
  - AMESOS (coarsest level solver)
  - IFPACK (Chebyshev)
  - ML (multilevel preconditioner, includes matrix-free version)
  - (Par)METIS (repartitioning, aggregation)
  - Isorropia (RCB repartitioning)
Numerical experiments

- Weak scalability test with an artificial bone. Number of degrees of freedom ranges from 300’000 to 2 billion.

- Strong scalability test with a large real bone with $1.3 \cdot 10^9$ degrees of freedom.

- Typical production sized problem of $15 \cdot 10^6$ degrees of freedom.

- All numbers have been obtained on the Cray XT5 at the Swiss Supercomputing Center CSCS.

![CSCS](image)

The machine has AMD Istanbul nodes with 6 cores and 8 GB RAM each.
Weak scalability of matrix-free preconditioning

Problem size scales with the number of processors.
Execution times should stay constant.
Weak scalability: problem sizes

We identify the artificial bones by $C_x$ where $x = 1, \ldots, 19$.

Bone $C_x$ has

<table>
<thead>
<tr>
<th>#elements</th>
<th>$60'482 \times 3^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>#nodes</td>
<td>$\sim 100'000 \times 3^3$</td>
</tr>
<tr>
<td>#matrix rows</td>
<td>$\sim 290'000 \times 3^3$</td>
</tr>
</tbody>
</table>

Number of degrees of freedom
### Weak scalability: ParMETIS

<table>
<thead>
<tr>
<th>name</th>
<th>#cores</th>
<th>#it_steps</th>
<th>solver time</th>
<th>prec setup time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>45</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>67</td>
<td>169</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>71</td>
<td>285</td>
<td>72</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>66</td>
<td>270</td>
<td>81</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>58</td>
<td>252</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td>81</td>
<td>71</td>
<td>306</td>
<td>90</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>68</td>
<td>303</td>
<td>94</td>
</tr>
<tr>
<td>8</td>
<td>192</td>
<td>61</td>
<td>278</td>
<td>98</td>
</tr>
<tr>
<td>9</td>
<td>273</td>
<td>54</td>
<td>250</td>
<td>103</td>
</tr>
<tr>
<td>10</td>
<td>375</td>
<td>62</td>
<td>329</td>
<td>109</td>
</tr>
<tr>
<td>12</td>
<td>648</td>
<td>62</td>
<td>445</td>
<td>127</td>
</tr>
<tr>
<td>14</td>
<td>1029</td>
<td>63</td>
<td>664</td>
<td>169</td>
</tr>
<tr>
<td>15</td>
<td>1265</td>
<td>57</td>
<td>562</td>
<td>170</td>
</tr>
<tr>
<td>16</td>
<td>1536</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>17</td>
<td>1842</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>18</td>
<td>2187</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>19</td>
<td>2572</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Weak scalability: Recursive Coordinate Bisection

<table>
<thead>
<tr>
<th>name</th>
<th>#cores</th>
<th>#it_steps</th>
<th>solver time</th>
<th>prec setup time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>45</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>53</td>
<td>137</td>
<td>67</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>66</td>
<td>264</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>66</td>
<td>273</td>
<td>108</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>74</td>
<td>338</td>
<td>123</td>
</tr>
<tr>
<td>6</td>
<td>81</td>
<td>73</td>
<td>332</td>
<td>152</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>68</td>
<td>329</td>
<td>200</td>
</tr>
<tr>
<td>8</td>
<td>192</td>
<td>65</td>
<td>298</td>
<td>137</td>
</tr>
<tr>
<td>9</td>
<td>273</td>
<td>76</td>
<td>386</td>
<td>264</td>
</tr>
<tr>
<td>10</td>
<td>375</td>
<td>72</td>
<td>442</td>
<td>248</td>
</tr>
<tr>
<td>12</td>
<td>648</td>
<td>70</td>
<td>451</td>
<td>227</td>
</tr>
<tr>
<td>14</td>
<td>1029</td>
<td>69</td>
<td>527</td>
<td>443</td>
</tr>
<tr>
<td>15</td>
<td>1265</td>
<td>62</td>
<td>530</td>
<td>578</td>
</tr>
<tr>
<td>16</td>
<td>1542</td>
<td>64</td>
<td>1042</td>
<td>528</td>
</tr>
<tr>
<td>17</td>
<td>1842</td>
<td>66</td>
<td>1009</td>
<td>733</td>
</tr>
<tr>
<td>18</td>
<td>2187</td>
<td>70</td>
<td>1070</td>
<td>897</td>
</tr>
<tr>
<td>19</td>
<td>2572</td>
<td>68</td>
<td>1444</td>
<td>1370</td>
</tr>
</tbody>
</table>
Weak scalability: ParMETIS vs. RCB
Weak scalability: ParMETIS vs. RCB (Discussion)

- When it works, ParMETIS performs better than RCB.
- Repartitioning by ParMETIS breaks down above $\sim 1250$ cores.
- Reason: above this limit the program crashes with *unexpected message buffer overflows*. (Messages have arrived but corresponding receives have not been issued (yet).)
- This problem has been resolved by C. Bekas for the IBM Blue Gene by introducing collective communication.
- RCB run times increase above $\sim 1250$ as well.
- Reason: On coarse grids matrices get dense. This entails
  - sparse matrix techniques for almost dense matrices
  - all-to-all communication in the aggregation phase
- Remedy: Limit levels and iterate on the (then) coarsest level (Tuminaro & Tong, SC’00)
Weak scalability: RCB full vs. RCB reduced levels
Weak scalability: RCB full / reduced levels (Discussion)

- RCB repartitioning combined with a limitation of AMG levels (here 4) provides a scalable solver.
- Iterative solver on the coarsest level: PCG with Chebyshev polynomial preconditioner (here of degree 15).
- Problem sizes are limited by order $2^{31} \approx 2.15 \cdot 10^9$ as Trilinos (Epetra) does not use 64-bit integers.
Strong scalability: very large bone

Effective strains with zooms.
(Image by Jean Favre, CSCS)
Fixed problem size: \( n = 1.342 \cdot 10^9 \) dof’s

<table>
<thead>
<tr>
<th># cores</th>
<th># iterations</th>
<th>solution time</th>
<th>precond. constr. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1800</td>
<td>140</td>
<td>981</td>
<td>489</td>
</tr>
<tr>
<td>2000</td>
<td>152</td>
<td>892</td>
<td>441</td>
</tr>
<tr>
<td>3000</td>
<td>190</td>
<td>870</td>
<td>394</td>
</tr>
<tr>
<td>3588</td>
<td>153</td>
<td>521</td>
<td>314</td>
</tr>
<tr>
<td>4000</td>
<td>140</td>
<td>581</td>
<td>292</td>
</tr>
<tr>
<td>5400</td>
<td>162</td>
<td>605</td>
<td>313</td>
</tr>
<tr>
<td>6000</td>
<td>157</td>
<td>560</td>
<td>327</td>
</tr>
</tbody>
</table>
Strong scalability: very large bone (cont.)

Execution times for solution and construction of preconditioner

Relative efficiencies for solution and construction of preconditioner
Human bone problems

Distal part (20% of the length) of the radius in a human forearm.
**Human bone problems (cont.)**

Fixed problem size $n = 14'523'162$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>12</th>
<th>20</th>
<th>40</th>
<th>58</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>†</td>
<td>†</td>
<td>†</td>
<td>110.4</td>
<td>116.2</td>
<td>82.7</td>
<td>70.2</td>
<td></td>
</tr>
<tr>
<td>951.6</td>
<td>699.6</td>
<td>311.3</td>
<td>182.8</td>
<td>185.3</td>
<td>163.1</td>
<td>125.2</td>
<td></td>
</tr>
</tbody>
</table>

Total CPU time in seconds required to solve the problem using matrix-ready (top) and matrix-free preconditioners (bottom) on $p$ processors. The symbol † indicates failure to run because of lack of memory.
Human bone problems (cont.)
Biomechanics problem: Stability of bone-implant constructs

Bone grafts and biomaterials are often used to aid the repair of complicated fractures.
Biomechanics problem: Stability of bone-implant constructs (cont.)

![Graph showing walltime comparison between EBE and ParFE with 256 CPU]
Conclusions

- Our C++ code, ParFE, is a parallel highly scalable FE solver for bone structure analysis based on PCG with aggregation multilevel preconditioners, see http://parfe.sourceforge.net/
- The algorithm scales quite well on the Cray XT5.
- For processor numbers $> 1000$ partitioning becomes a problem with ParMETIS.
- RCB comes to our rescue.
- Dense coarse problems must be avoided. Limitation of the number of levels helps. The coarsest problem must be solved iteratively.
- With sufficient hardware, problems with hundreds of millions of degrees of freedom can be solved in a few ($\sim 10$) minutes.
References


References (cont.)
