Outline

follows the structure of the report

- Introduction
- Mesh-based Modeling of Cuts
- Finite Element Simulation for Virtual Cutting
- Numerical Solvers
  - Direct solvers
  - Iterative solvers
- Meshfree Methods
- Summary & Application Study
- Discussion & Conclusion

Numerical Solvers

- Implicit time integration leads to a linear system of equations
$$Ax = b$$
  - when using the linear strain tensor and a linear material model
- A is a sparse, symmetric, positive definite matrix
- Update of the system matrix A required …
  - due to adaptation of the finite element model (cutting)
  - in every time step, when using the corotational strain formulation
  - Requires re-initialization of the solver

Direct Solvers

- Obtain exact solution in a finite number of steps
- Matrix inversion: $b = A^{-1}x$ $(A \in \mathbb{R}^{n \times n})$
  - Computing time $O(n^3)$ (initialization) and $O(n^2)$ (solve)
  - Memory $O(n^2)$
  - Only feasible for (very) small n
  - Incremental update via Sherman-Morrison-Woodbury formulae
    - $(A - UV^T)^{-1} = A^{-1} + A^{-1}U(E - V^T A^{-1} U)^{-1} V^T A^{-1}$
    - Update can be restructured to be in $O(n)$ under certain assumptions considering the number of non-zero entries [Zhong et al. 2005]

Direct Solvers

- Cholesky factorization: $A = LL^T$ for a spd matrix A

$$L \underbrace{L^T x}_{y :=} = b$$

$$Ly = b$$

$$L^T x = y$$

  - Better constant factors than matrix inversion
  - Can also be incrementally updated [Turkiyyah et al. 2009]

Iterative Solvers

- Successively compute approximations x_m to the solution x

$$x = \lim_{m \to \infty} x_m$$

- Allows for balancing speed and accuracy
  - Monitor norm of residual $r_m = b - Ax_m$
  - Stop if residual reduction $\frac{\|r_m\|_2}{\|r_0\|_2} \leq \tau$ for given threshold τ

Iterative Solvers

- Conjugate Gradient Method

$$Ax = b \quad \Leftrightarrow \quad \underbrace{\frac{1}{2}x^T Ax - b^T x}_{F(x):=} \rightarrow min \qquad \text{for spd matrix } A$$

  - F has a single, global minimum (paraboloid)
  - Iterative search for minimum:

$$x_{m+1} = x_m + \lambda_m p_m$$

$$p_m = -\nabla F(x_m) + \sum_{j=0}^{m-1} \alpha_j p_j \qquad p_i^T A p_j = 0 \text{ for } i \neq j$$

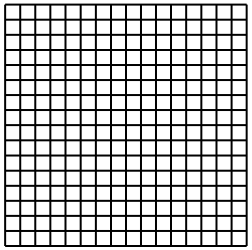
    - Problem-adapting
    - x_m minimizes F on affine subspace of continuously increasing dimension
  - Requires matrix-vector products and dot products
  - Efficient parallelization using OpenMP [Chentanez et al. 2009] or CUDA [Courtecuisse et al. 2010]

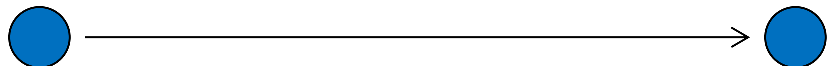
Iterative Solvers

- So far: “Blackbox” solvers
- More advanced solvers: Geometric multigrid solvers
  - Basic relaxation schemes (Jacobi, Gauss-Seidel) only reduce high-frequency error components effectively
  - Consider the problem on a hierarchy of successively coarser grids
  - Reduce lower-frequency error components on coarser grids (where they appear at a higher frequency)

Geometric Multigrid

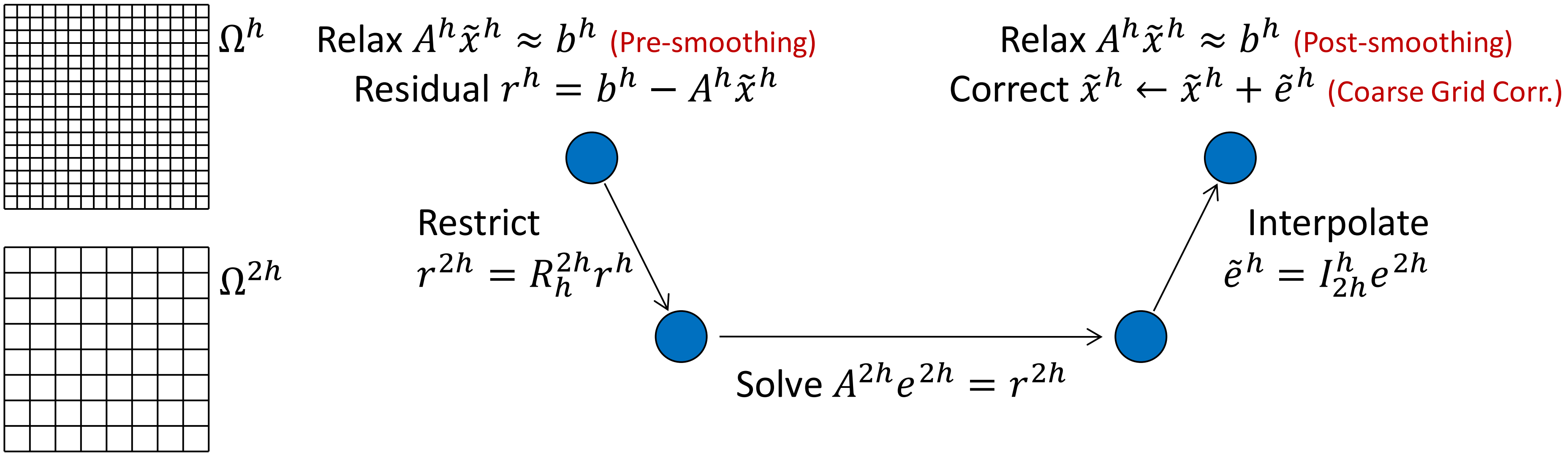
- Solve $A^h x^h = b^h$, current approximate solution $\tilde{x}^h$





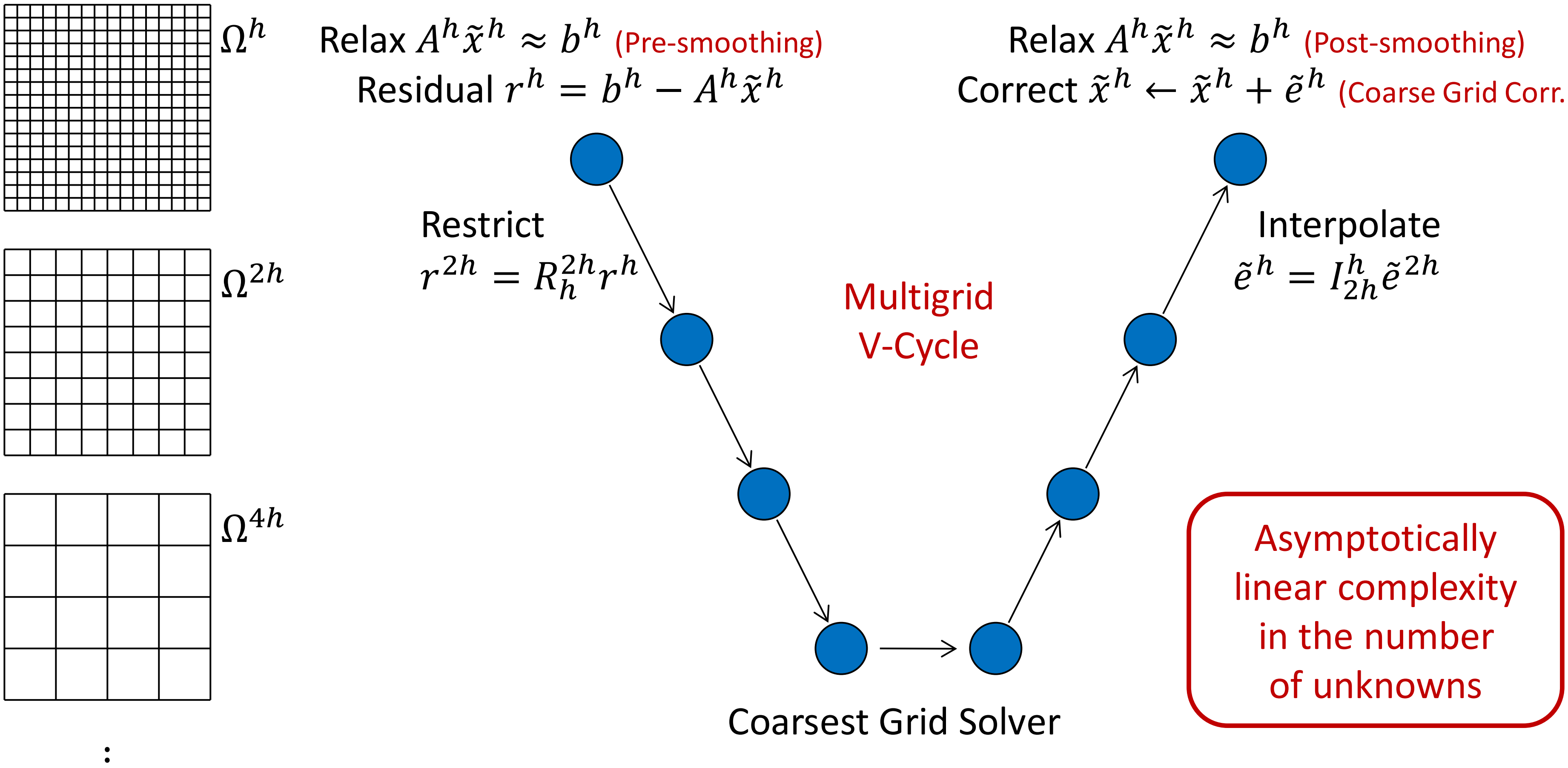
Geometric Multigrid

- Solve $A^h x^h = b^h$, current approximate solution $\tilde{x}^h$



Geometric Multigrid

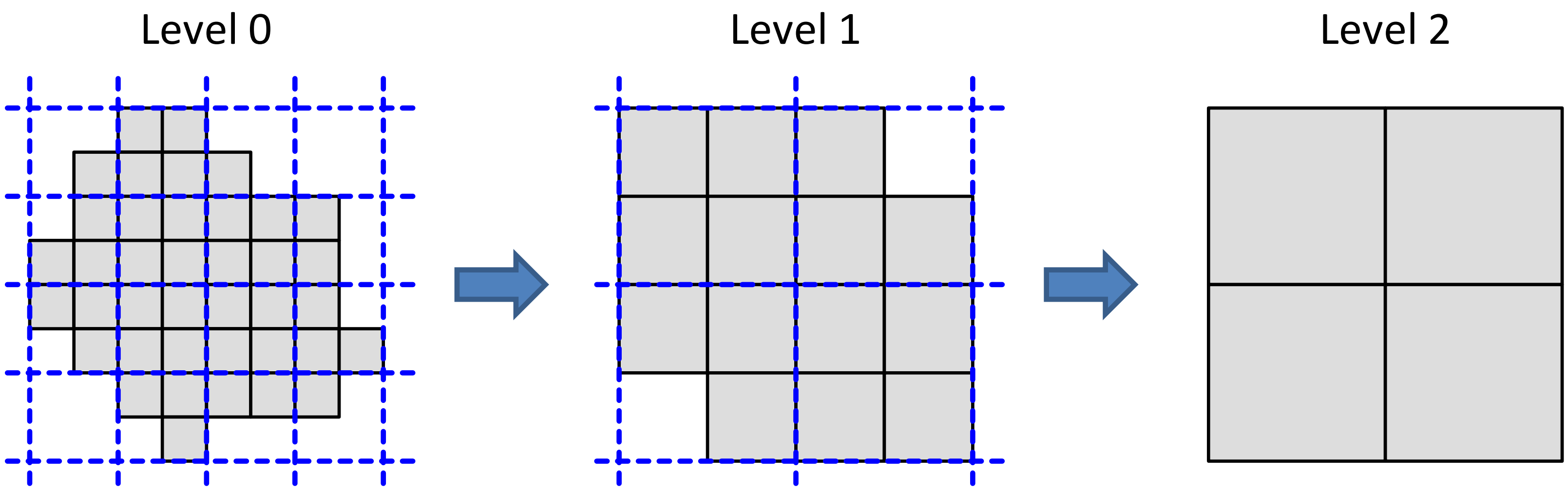
- Solve $A^h x^h = b^h$, current approximate solution v^h



Asymptotically linear complexity in the number of unknowns

Multigrid Hierarchy Construction

- (Semi-)Regular hexahedral grids

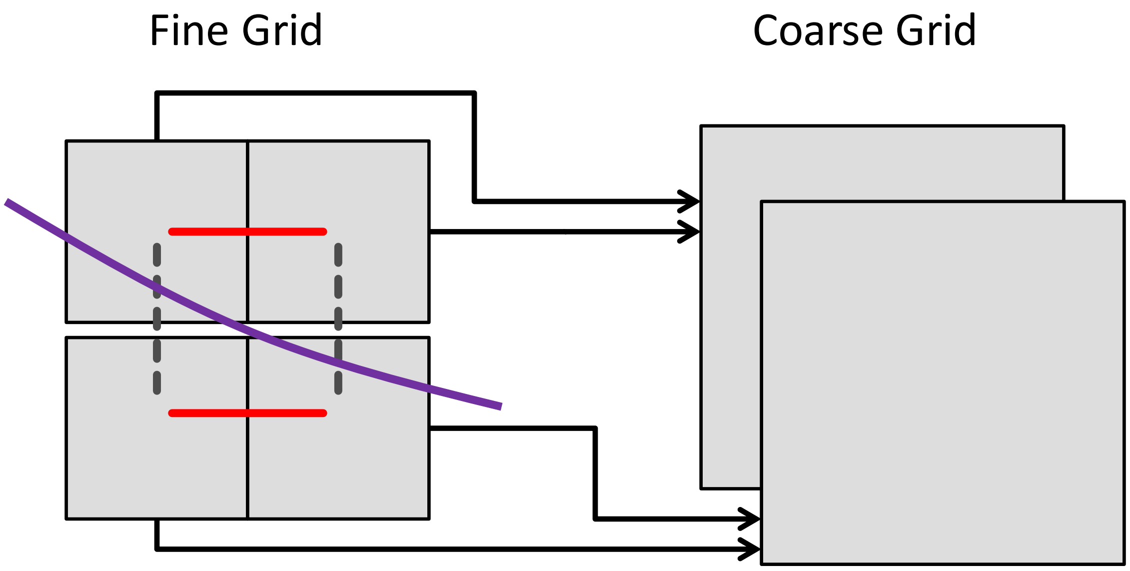


  - Blocks of 2^3 cells are merged into coarse grid cells of double size
  - A cell is created if it covers at least one cell on the finer level
    - Coarser cells might be only partially filled [Liehr et al. 2009]

- Difficult for unstructured grids

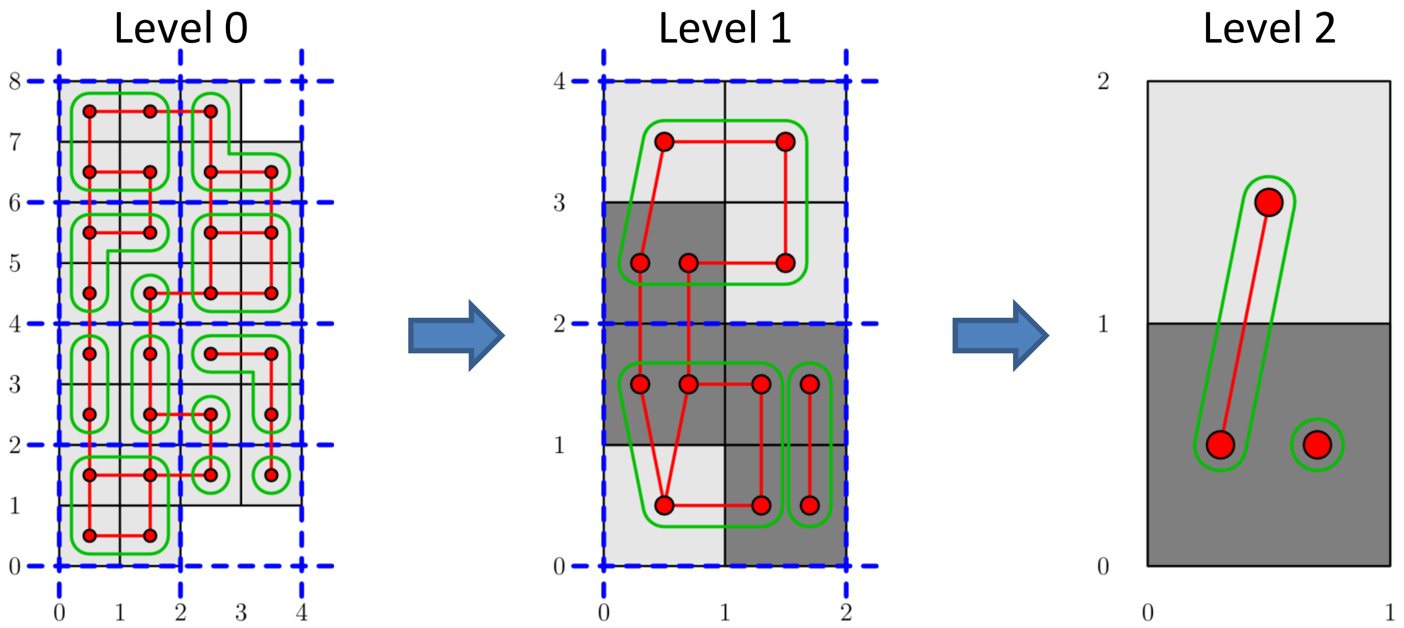
Multigrid with Cuts

- Representation of complicated topologies on the coarse grids
  - Physically disconnected parts should be represented by individual coarse grid cells
  - Duplication of cells on the coarse grids [Aftosmis et al. 2000]
  - Graph-based hierarchy construction analogous to composite elements



Multigrid with Cuts

- Construction of multigrid hierarchy using an undirected graph representation

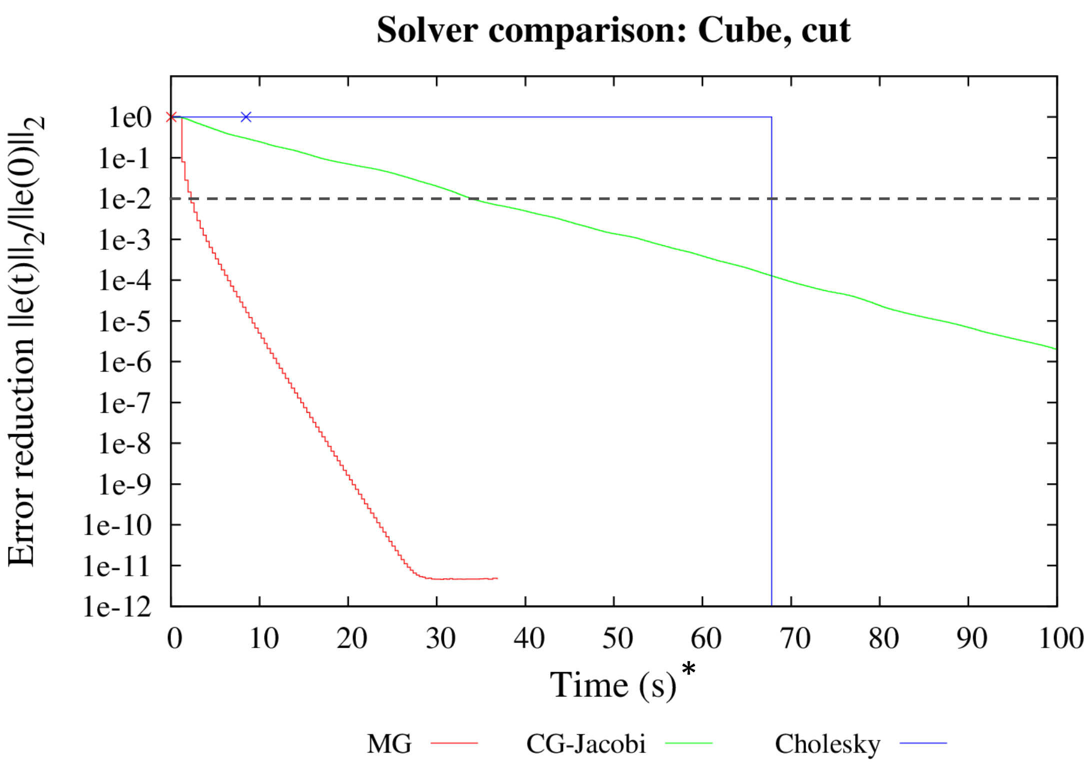


[Dick et al. 2011]

  - Works equally well for an adaptive octree grid

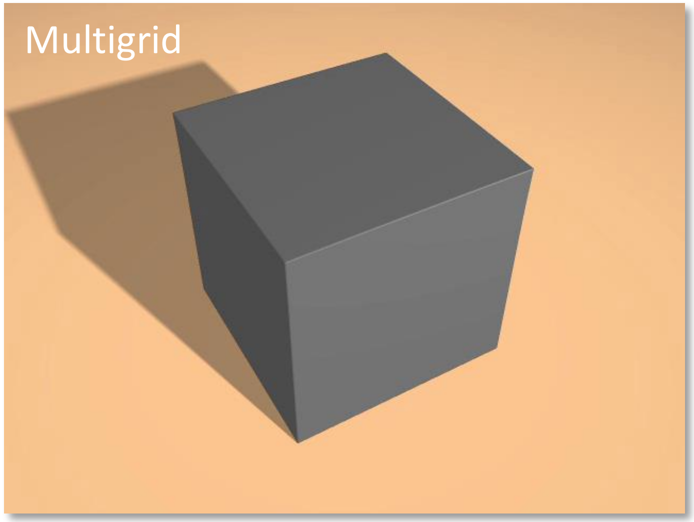
Solver Comparison

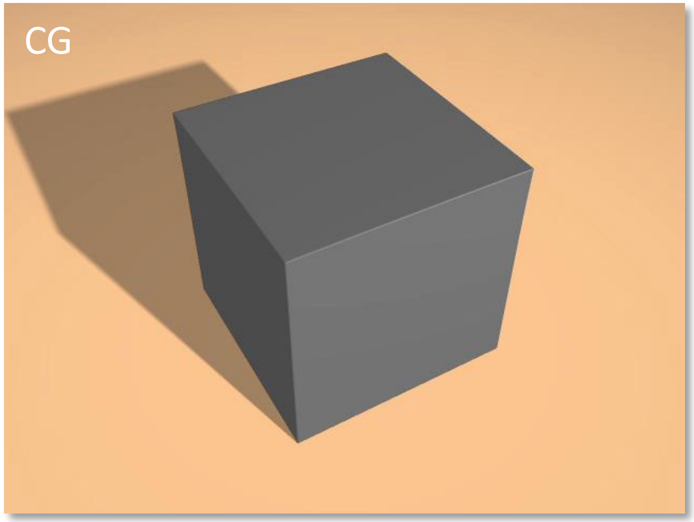
- Comparison wrt run-time (230k elements)



*CPU, Single Core

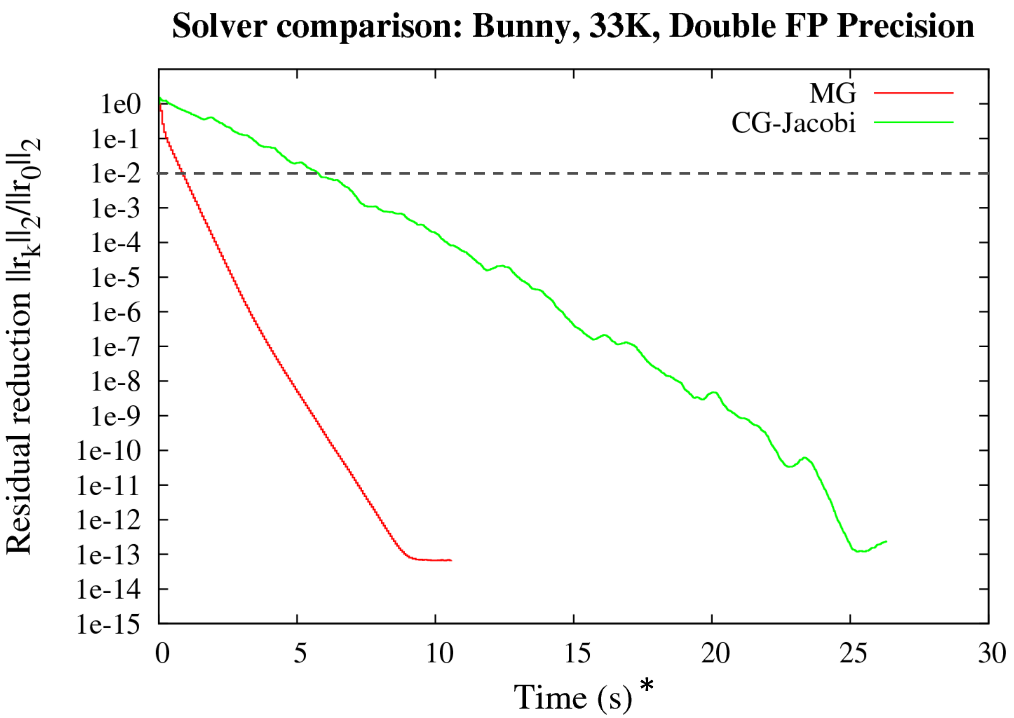
[Dick et al. 2011]

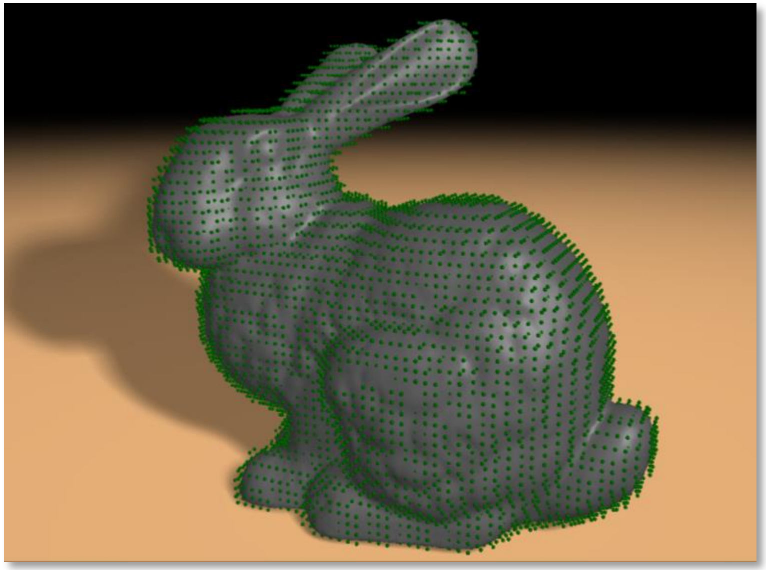




Solver Comparison

- Comparison wrt run-time (33k elements)





*CPU, Single Core

[Dick et al. 2011]

Numerical Solvers

- Discussion
  - Direct vs. iterative solvers
  - Blackbox vs. application-specific solvers
  - Speed vs. implementation effort