Linear solvers for large algebraic systems from structural mechanics

Symposium of Advances in Contact Mechanics: a tribute to Prof J. J. Kalker

Delft / EWI
prof. dr. ir. C. Vuik
October 23, 2008
Outline

• Structural mechanics
  • Computational framework
  • Finite element discretization
• Numerical linear algebra
  • Overview numerical methods for linear systems
  • Parallel direct solver
  • Combining methods to create new solver
• Test case
• Discussion
How to compute deformation?

Force balance

- Forces should be in balance,

\[ \int_{\Omega} div(\sigma) + \mathbf{f} - \rho \mathbf{g} \, d\Omega = 0 \]

- Residual when out of balance,

\[ \mathbf{r} = div(\sigma) + \mathbf{f} - \rho \mathbf{g} \]

internal force \( \sigma \), external force \( \mathbf{f} \),
gravitation \( \mathbf{g} \)
How to compute deformation?
Virtual work

- Define virtual work,

\[ \delta W = \int \mathbf{r} \cdot \delta \mathbf{u} \, dv \]

- Equilibrium

\[ \delta W (\mathbf{X}) = \delta W_{\text{int}} (\mathbf{X}) - \delta W_{\text{ext}} (\mathbf{X}) = 0 \]
How to compute deformation?

Linearized virtual work

- First order Taylor,
\[
\delta W (X_0) + D_{\Delta u} [\delta W (X_0)] = 0
\]

- Linearized virtual work

\[
\int_V (\nabla_0 \Delta u \cdot S) : \nabla_0 \delta v dV + \int_V (\nabla_0 \Delta u : F \cdot C \cdot F^T) : \nabla_0 \delta v dV = \delta v \cdot f_{ext} - \int_V P : \nabla_0 \delta v dV
\]
How to compute deformation?
Material response

- Three material properties,
  - (Hyper) elasticity
  - Plasticity (permanent deformation)
  - Viscosity (permanent deformation)
How to compute deformation?
Non-linear material response
Finite element discretization

FE mesh
Finite element discretization
Tetrahedral elements

• Introduce local coordinate system,

\[(x, y, z) \rightarrow (\xi, \eta, \zeta)\]

• Transformation between local and global coordinates,

\[\frac{d}{d\xi} = J \frac{d}{dx}\]
Finite element discretization

Shape functions

• 1D Example with linear shape functions,

\[ x = N_1(\xi) x_1 + N_2(\xi) x_2 \]

• For 3D case,

\[ x = \sum_{i=1}^{4} N_i(\xi, \eta, \zeta) \cdot x_i \]

• For stability and accuracy higher order shape functions necessary
Finite element discretization

Numerical integration

- Use Gauss point(s) for numerical integration,

\[
\int_{\Omega} f \, d\Omega \approx f (\xi_g, \eta_g, \zeta_g) |J| \int_{\Omega} d\Omega
\]
Finite element discretization
Stiffness matrix

- Discretized, linearized virtual work,

\[ K \Delta u = f_{ext} - f_{int} \]
Finite element discretization
Stiffness matrix

- Properties of $K$,
  - Symmetric
  - Positive definite
  - Sparse
  - No specific pattern of non-zero matrix entries
  - Large differences in entry values due to material properties
  - Changes due to non-linear material properties
How to compute deformation?
Balancing of forces algorithm

MESH
Linear solver
Material Models

Apply load

Assemble, $K$

Solve,

$K\Delta u = \Delta f$

Compute internal force (Newton-Raphson)

Newmark (dynamic)
Newton-Raphson
Problems with algorithm

Scale

Number of elements: 1,890,057

Number of nodes: 307,735

Number of non zero elements in stiffness matrix: 21,296,523
Problems with algorithm
Accuracy and approximation

\[ \sigma \]

\[ \Delta f_{est}^{i+1} \]

\[ \Delta f_{est}^i \]

\[ \varepsilon \]
Numerical methods
Overview

• (Parallel) Direct solver
• Iterative solvers,
  • Preconditioning
  • CG method
• Deflation, Domain Decomposition of Multigrid?
Parallel direct solver
Definition

• Direct solver,

\[ x = A^{-1}b \]

• Matrix cannot be singular or ill conditioned, this leads to inaccurate solutions

• Large (3D) models yield large linear systems, serial direct solvers lack CPU power and memory
Parallel direct solver
MUMPS

• Solution? Go parallel! Spread work over computing nodes. Adding more nodes implies more CPU power and memory.

• MUMPS project: public domain package and developed by CERFACS, graal.ens-lyon.fr/MUMPS/
Parallel direct solver
MUMPS
Iterative methods
Basics

- Solve,
  \[ Ax = b \]
- Use sequence of approximations of solution \( x \),
  \[ x_0, x_1, x_2, \ldots, x_k \]

where,
\[ x_{k+1} = x_k + M^{-1} (b - Ax_k) \]

- Choice of \( M \) defines iterative method
Iterative methods

Available methods

• Splitting based methods \( (M = N - A) \),
  • Jacobi
  • Gauss-Seidel
  • SSOR
• Krylov subspace methods
  • CG
  • GMRES
• Multigrid
Iterative methods
Preconditioning

• Condition number,
\[ \kappa_p(A) = \| A \|_p \| A^{-1} \|_p. \]

• For (symmetric) SPD matrices,
\[ \kappa_p(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}. \]

• Improve condition of matrix,
\[ M^{-1} Ax = M^{-1} b. \]
Iterative methods

Preconditioning

- Preconditioner is approximation of original matrix
- Matrix $M$ can be any constant linear solver
- Many choices,
  - Incomplete LU or Cholesky decomposition,
  - Basic iterative methods (GS, Jacobi)
  - Multigrid
  - Domain decomposition
  - Deflation
Iterative methods
Conjugate gradient (CG)

- Krylov subspace,
  \[ x_0 + \text{span} \left\{ M^{-1}r_0, M^{-1}A(M^{-1}r_0), \ldots, (M^{-1}A)^{i-1}(M^{-1}r_0) \right\} \]

- Good performance for well conditioned SPD matrices
- Slow converging components corresponds to smallest eigenvalues of A
- Preconditioner necessary for ill conditioned systems
Iterative methods

Multigrid

• Idea: Approximation of (smooth) error of the solution on coarser grids. Back propagate error to fine grid:

\[ Ax = b \rightarrow \Delta x = x - \tilde{x} \]

\[ r_h = A_h \Delta x_h \rightarrow I_h^H \rightarrow r_H = A_H \Delta x_H \]

\[ \tilde{x}_h^{k+1} = \tilde{x}_h^k + I_H^h \Delta x_H \]

• Benefit: Reduction of size the system that has to be solved with direct solver.
Iterative methods
Multigrid

• How to choose grid operators $I_h^H$, $I_h^H$?

• How to choose coarse grid cells on unstructured grids?
Iterative solvers
Domain decomposition

• Divide large problem into subdomains, divide work load and easy parallelizable.

• Rewrite original system,

\[ \Omega_i, \forall i \in \{1, 2, \ldots, s\} \]

\[ A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \text{ with } A = \begin{pmatrix} B & E \\ F & C \end{pmatrix} \]

where \( y, g \) correspond to interface nodes
Iterative solvers

Domain decomposition

• Schur complement $S$,

\[
(C - FB^{-1}E) y = g - FB^{-1}f \\
Sy = g'
\]

• Solve $y$ and obtain $x$ from,

\[
x = B^{-1} (f - Ey)
\]
Iterative solvers
Domain decomposition

• How to choose subdomains?

• How to solve on subdomains?
Iterative solvers
Deflation

- Filter out the eigenvalues that belong to the slow converging components of for e.g. the CG method
- Deflation components,

\[
\begin{align*}
Z & \in \mathbb{R}^{n \times k}, \quad k < n - d, \quad \text{deflation subspace matrix} \\
E &= Z^T A Z \in \mathbb{R}^{k \times k}, \quad \text{inversion Galerkin matrix or coarse matrix} \\
Q &= Z E^{-1} Z^T \in \mathbb{R}^{n \times n}, \quad \text{correction matrix} \\
P &= I - A Q \in \mathbb{R}^{n \times n}, \quad \text{deflation matrix} \\
PAx &= Pb \\

d, \quad \text{number of zero eigenvalues} \\
k, \quad \text{number of deflation vectors}
\end{align*}
\]
Iterative solvers

Deflation

• Preferably the deflation vectors are the eigenvectors corresponding to the smallest eigenvalues (think of condition number)
• Computation of deflation vectors is expensive, use approximations,
  • Physical: interface elements with high discontinuities
  • Analytical: use information CG, previous time steps, FE discretization etc.
Hybrid solver
Combining numerical methods

\[ \text{Solve } Mv = w \]

- Krylov subspace
- Preconditioner \((M)\)
- Multigrid
- Domain decomposition
- Deflation
- \((\text{Parallel) Direct solver})\]
Test case
Compression
Test case
Compression
Test case
Deflation + CG + preconditioning
Test case
Deflation + CG + preconditioning
Future research

People

• Civil Engineering (group Scarpas),
  • A. Scarpas
  • C. Kasbergen
• Applied Mathematics (group Vuik),
  • C. Vuik
  • M.B van Gijzen
  • T.B Jönsthövel