Abstract

"Enhancing iterative solution methods for general FEM computations using rigid body modes."

Alex Sangers - June 27, 2014 at 15:30 in EEMCS LB 01.010

DIANA is a general finite element software package that can be used to analyze a wide range of problems arising in Civil engineering. The solution of one or more systems of linear equations is a computational intensive part of a finite element analysis.

Iterative solvers have proved to be efficient for solving these systems of equations. However, the convergence of iterative solvers stagnates if there are large stiffness jumps in the underlying model.

The considered remedy is based on the approximate rigid body modes of the model. To identify the approximate rigid body modes in a finite element application we propose a generally applicable method based on element stiffness matrices. These rigid body modes can be used for deflation and coarse grid correction.

TNO DIANA A TNO Company TUDelft

Enhancing iterative solvers in DIANA

Enhancing iterative solution methods for general FEM computations using rigid body modes.

Alex Sangers

Delft Institute of Applied Mathematics
TNO DIANA

June 27, 2014

Content

Finite element analysis

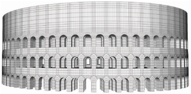
1. Real-life application

Finite element analysis

1. Real-life application

$\Rightarrow$

2. Model with elements

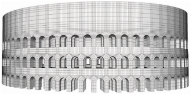


Finite element analysis

1. Real-life application

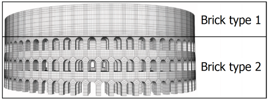
⇒

2. Model with elements



⇓

3. Assign properties

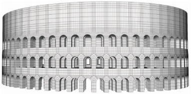


Finite element analysis

1. Real-life application

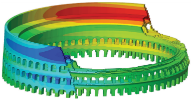
⇒

2. Model with elements



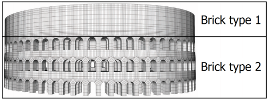
⇓

4. Analysis



⇐

3. Assign properties

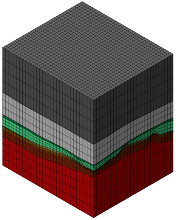


Finite elements applications

- Structures

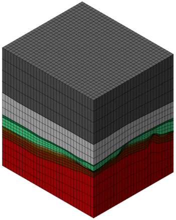
Finite elements applications

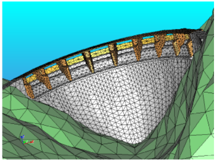
- Structures
- Geomechanics



Finite elements applications

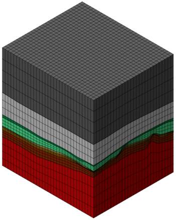
- Structures
- Geomechanics
- Dams and dikes

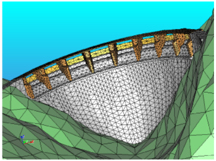


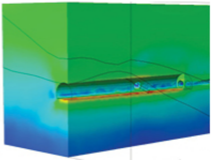


Finite elements applications

- Structures
- Geomechanics
- Dams and dikes
- Tunneling

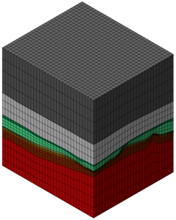
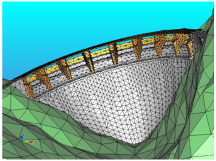
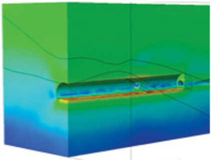
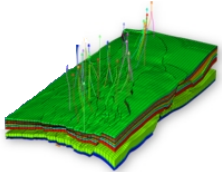






Finite elements applications

- Structures
- Geomechanics
- Dams and dikes
- Tunneling
- Oil and gas industry

One-dimensional Poisson problem

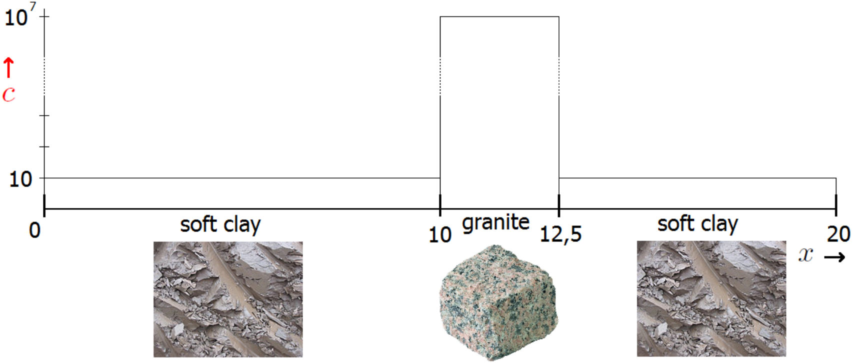
A 1D problem consisting of soft clay and granite:

$$-\frac{d}{dx}\left(c\,\frac{du}{dx}\right) = f, \quad x \in (0, 20)$$
$$u = 0, \quad x \in \{0, 20\}.$$

One-dimensional Poisson problem

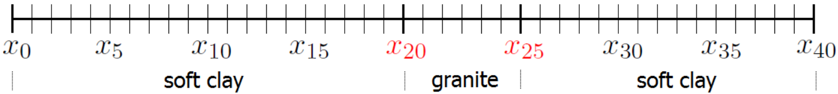
A 1D problem consisting of soft clay and granite:

$$-\frac{d}{dx}\left(c\,\frac{du}{dx}\right) = f, \quad x \in (0,20)$$
$$u = 0, \quad x \in \{0,20\}.$$



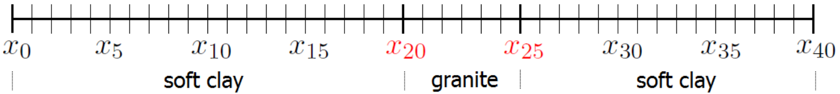
Finite elements for Poisson problem

1. Mesh the model into 40 equal-sized elements



Finite elements for Poisson problem

1. Mesh the model into 40 equal-sized elements



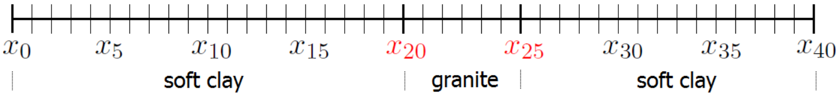
2. Compute each element stiffness matrix and vector:

$$K^{e_m} = \begin{pmatrix} c(x_m) & -c(x_{m+1}) \\ -c(x_m) & c(x_{m+1}) \end{pmatrix}, \qquad f^{e_m} = \begin{pmatrix} f(x_m) \\ f(x_{m+1}) \end{pmatrix}.$$

Jumps in $c(x_{20})$ and $c(x_{25})$.

Finite elements for Poisson problem

1. Mesh the model into 40 equal-sized elements



2. Compute each element stiffness matrix and vector:

$$K^{e_m} = \begin{pmatrix} c(x_m) & -c(x_{m+1}) \\ -c(x_m) & c(x_{m+1}) \end{pmatrix}, \qquad f^{e_m} = \begin{pmatrix} f(x_m) \\ f(x_{m+1}) \end{pmatrix}.$$

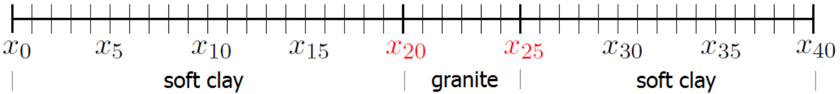
   Jumps in $c(x_{20})$ and $c(x_{25})$.

3. Assemble the stiffness matrix and vector:

$$K = \bigcup_{m=1}^{40} K^{e_m}, \qquad f = \bigcup_{m=1}^{40} f^{e_m}.$$

Finite elements for Poisson problem

1. Mesh the model into 40 equal-sized elements



2. Compute each element stiffness matrix and vector:

$$K^{e_m} = \begin{pmatrix} c(x_m) & -c(x_{m+1}) \\ -c(x_m) & c(x_{m+1}) \end{pmatrix}, \qquad f^{e_m} = \begin{pmatrix} f(x_m) \\ f(x_{m+1}) \end{pmatrix}.$$

Jumps in $c(x_{20})$ and $c(x_{25})$.

3. Assemble the stiffness matrix and vector:

$$K = \bigcup_{m=1}^{40} K^{e_m}, \qquad f = \bigcup_{m=1}^{40} f^{e_m}.$$

4. Solve $Ku = f$.

Iterative solvers

Solve $Ku = f$ step by step.

Iterative solvers

Solve $Ku = f$ step by step.

- Popular algorithms:
  - Conjugate Gradient (CG) for symmetric K
  - Restarted GMRES (GMRES(s)) for non-symmetric K

Iterative solvers

Solve $Ku = f$ step by step.

- Popular algorithms:
  - Conjugate Gradient (CG) for symmetric K
  - Restarted GMRES (GMRES(s)) for non-symmetric K
- Convergence speed depends on eigensystem of K
  - Clustered eigenvalues $\Rightarrow$ fast convergence

Iterative solvers

Solve $Ku = f$ step by step.

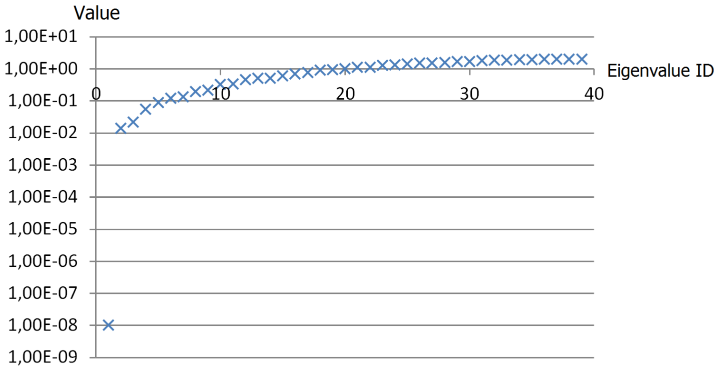
- Popular algorithms:
  - Conjugate Gradient (CG) for symmetric K
  - Restarted GMRES (GMRES(s)) for non-symmetric K
- Convergence speed depends on eigensystem of K
  - Clustered eigenvalues $\Rightarrow$ fast convergence
- *Preconditioning* may improve convergence
  - $P^{-1}Ku = P^{-1}f$
  - $P \approx K$
  - $Px = y$ is easy to solve

Solving the Poisson problem

Solve the Jacobi preconditioned system:

$$P^{-1}Ku = P^{-1}f$$

The convergence of Preconditioned CG depends on $\lambda(P^{-1}K)$:

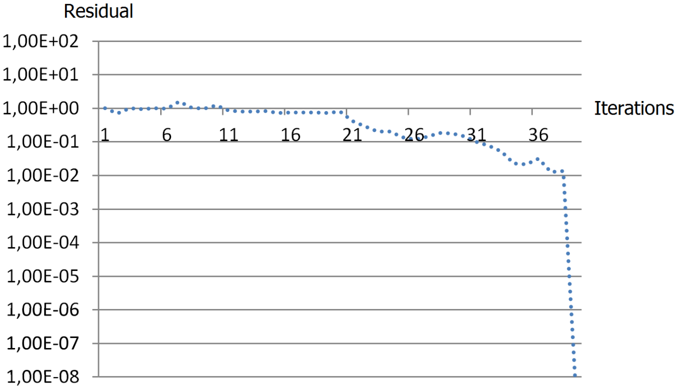


Solving the Poisson problem

Solve the Jacobi preconditioned system:

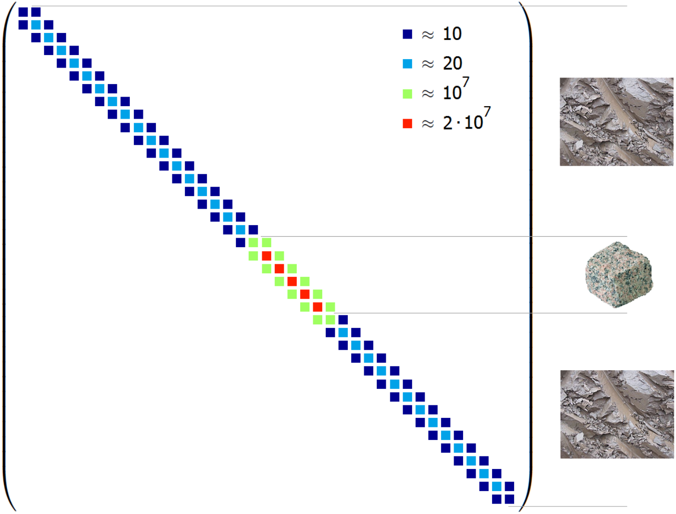
$$P^{-1}Ku = P^{-1}f$$

The convergence of Preconditioned CG:



Stiffness matrix of Poisson problem

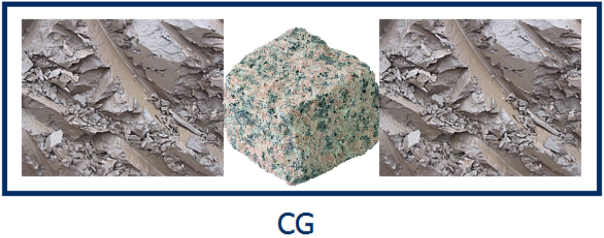
$$|K| =$$



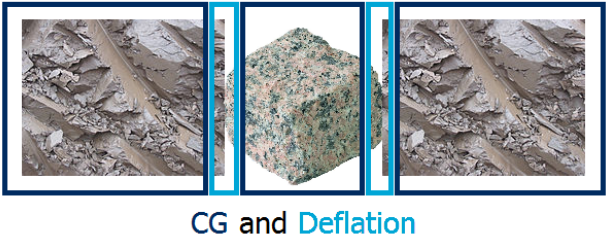
Solving the Poisson problem

CG

Solving the Poisson problem



Deflation: Apply $\Pi^{\perp}$

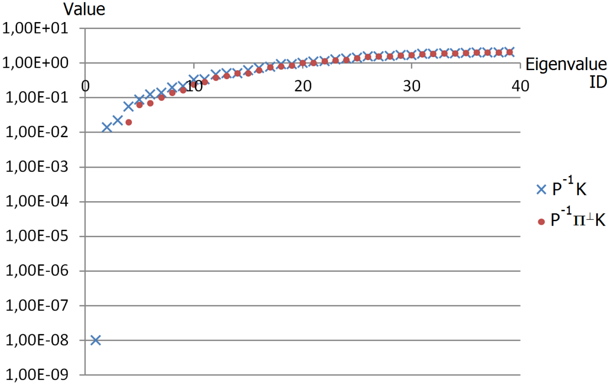


Solving the deflated Poisson problem

Solve the deflated Jacobi preconditioned system:

$$P^{-1}\Pi^{\perp}Ku = P^{-1}\Pi^{\perp}f$$

The convergence of Deflated PCG depends on $\lambda(P^{-1}\Pi^{\perp}K)$:

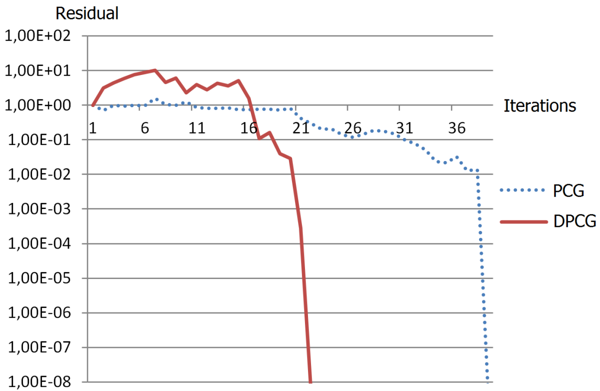


Solving the deflated Poisson problem

Solve the deflated Jacobi preconditioned system:

$$P^{-1}\Pi^{\perp}Ku = P^{-1}\Pi^{\perp}f$$

The convergence of Deflated PCG:



Approximate rigid bodies

True rigid bodies

$Kx = 0$ for some $x \neq 0$.

Approximate rigid bodies

True rigid bodies

$Kx = 0$ for some $x \neq 0$.

Approximate rigid bodies

Represents “weak coupling” in the model.

Identifying approximate rigid bodies

- DIANA is general FE software
  - Wide range of elements
  - Wide range of materials

Identifying approximate rigid bodies

- DIANA is general FE software
  - Wide range of elements
  - Wide range of materials
- Element matrices!
  - Always present
  - Fair comparison

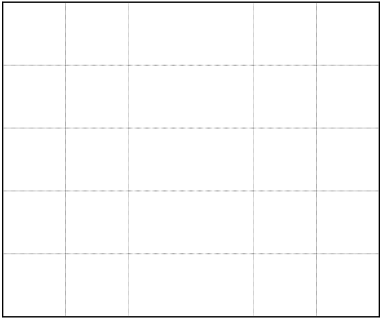
Identifying approximate rigid bodies

- DIANA is general FE software
  - Wide range of elements
  - Wide range of materials
- Element matrices!
  - Always present
  - Fair comparison

$$\frac{1}{n}\mathsf{tr}(K^{e_m}) = \frac{1}{n}\sum_{i=1}^{n} K_{ii}^{e_m}$$

Coloring algorithm

Consider this two-dimensional finite element mesh



Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Nonlinear iteration loop

- Reuse the coloring if $\frac{1}{n}\mathrm{tr}(K^{e_m})$ changes less than 50%.

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	81	2	3	3	5
91	80	19	14	1	2
2	3	26	31	3	2
4	4	21	19	2	4

Nonlinear iteration loop

- Reuse the coloring if $\frac{1}{n}\mathrm{tr}(K^{e_m})$ changes less than 50%.

Coloring algorithm

Consider this two-dimensional finite element mesh

73	77	4	2	1	3
86	62	2	4	3	5
91	80	19	14	1	2
2	3	18	31	2	2
4	4	21	19	2	4

Nonlinear iteration loop

- Reuse the coloring if $\frac{1}{n}\mathrm{tr}(K^{e_m})$ changes less than 50%.

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - KZ(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K \Pi^{\in}$.

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - KZ(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K \Pi^{\in}$.

Split u by

$$\begin{aligned} u &= u^{\perp} + u^{\in} \\ &= (I - \Pi^{\in})u + \Pi^{\in} u. \end{aligned}$$

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - KZ(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K \Pi^{\in}$.

Split u by

$$\begin{aligned} u &= u^{\perp} + u^{\in} \\ &= (I - \Pi^{\in})u + \Pi^{\in} u. \end{aligned}$$

First part $u^{\perp}$: $\quad u^{\perp} = (I - \Pi^{\in})u = Z(Z^T K Z)^{-1} Z^T f.$

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - KZ(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K \Pi^{\in}$.

Split u by

$$\begin{aligned} u &= u^{\perp} + u^{\in} \\ &= (I - \Pi^{\in})u + \Pi^{\in} u. \end{aligned}$$

First part $u^{\perp}$: $\quad u^{\perp} = (I - \Pi^{\in})u = Z(Z^T K Z)^{-1} Z^T f.$

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - KZ(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K\Pi^{\in}$.

Split u by

$$u = u^{\perp} + u^{\in}$$
$$= (I - \Pi^{\in})u + \Pi^{\in} u.$$

First part $u^{\perp}$: $\quad u^{\perp} = (I - \Pi^{\in})u = Z(Z^T K Z)^{-1} Z^T f.$

Second part $u^{\in}$: $\quad K u^{\in} = K\Pi^{\in} u = \Pi^{\perp} K u.$

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - K Z(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K \Pi^{\in}$.

Split u by

$$\begin{aligned} u &= u^{\perp} + u^{\in} \\ &= (I - \Pi^{\in})u + \Pi^{\in} u. \end{aligned}$$

First part $u^{\perp}$: $\quad u^{\perp} = (I - \Pi^{\in})u = Z(Z^T K Z)^{-1} Z^T f.$ ✓

Second part $u^{\in}$: $\quad K u^{\in} = K \Pi^{\in} u = \Pi^{\perp} K u.$

$$\Rightarrow \Pi^{\perp} K u = \Pi^{\perp} f.$$

Deflation

Define

$$\Pi^{\in} = I - Z(Z^T K Z)^{-1} Z^T K,$$
$$\Pi^{\perp} = I - K Z(Z^T K Z)^{-1} Z^T,$$

so that $\Pi^{\perp} K = K \Pi^{\in}$.

Split u by

$$\begin{aligned} u &= u^{\perp} + u^{\in} \\ &= (I - \Pi^{\in})u + \Pi^{\in} u. \end{aligned}$$

First part $u^{\perp}$: $\quad u^{\perp} = (I - \Pi^{\in})u = Z(Z^T K Z)^{-1} Z^T f.$

Second part $u^{\in}$: $\quad K u^{\in} = K \Pi^{\in} u = \Pi^{\perp} K u.$

$$\Rightarrow \Pi^{\perp} K u = \Pi^{\perp} f.$$

Coarse grid correction

The coarse grid correction

$$P_C = I + Z(Z^T K Z)^{-1} Z^T$$

corrects the solution during the iteration process:

$$u_{k+1} = u_k + Z(Z^T K Z)^{-1} Z^T r_0.$$

Coarse grid correction

The coarse grid correction

$$P_C = I + Z(Z^T K Z)^{-1} Z^T$$

corrects the solution during the iteration process:

$$u_{k+1} = u_k + Z(Z^T K Z)^{-1} Z^T r_0.$$

In two-level Additive Schwarz form (dual preconditioning):

$$P_{C,P^{-1}} = P^{-1} + Z(Z^T K Z)^{-1} Z^T.$$

Deflation vs. coarse grid correction

Define the coarse matrix

$$E = Z^T K Z.$$

	Deflation	**Coarse grid correction**
	$\Pi^{\perp} = I - KZE^{-1}Z^T$	$P_C = I + ZE^{-1}Z^T$

Deflation vs. coarse grid correction

Define the coarse matrix

$$E = Z^T K Z.$$

	Deflation $\Pi^{\perp} = I - KZE^{-1}Z^T$	**Coarse grid correction** $P_C = I + ZE^{-1}Z^T$
Cost per iteration	$+$	$+$

Deflation vs. coarse grid correction

Define the coarse matrix

$$E = Z^T K Z.$$

	Deflation	**Coarse grid correction**
	$\Pi^{\perp} = I - KZE^{-1}Z^T$	$P_C = I + ZE^{-1}Z^T$
Cost per iteration	$+$	$+$
Parallelizability	$-$	$+$

Deflation vs. coarse grid correction

Define the coarse matrix

$$E = Z^T K Z.$$

	Deflation $\Pi^{\perp} = I - KZE^{-1}Z^T$	**Coarse grid correction** $P_C = I + ZE^{-1}Z^T$
Cost per iteration	$+$	$+$
Parallelizability	$-$	$+$
Effectiveness	$+$	$-$

Deflation vs. coarse grid correction

Define the coarse matrix

$$E = Z^T K Z.$$

	Deflation $\Pi^{\perp} = I - KZE^{-1}Z^T$	Coarse grid correction $P_C = I + ZE^{-1}Z^T$
Cost per iteration	$+$	$+$
Parallelizability	$-$	$+$
Effectiveness	$+$	$-$
Numerical sensitivity	$-$	$+$

Deflation vs. coarse grid correction

Define the coarse matrix

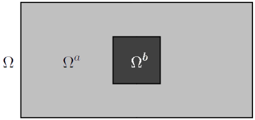
$$E = Z^T K Z.$$

	Deflation $\Pi^{\perp} = I - KZE^{-1}Z^T$	**Coarse grid correction** $P_C = I + ZE^{-1}Z^T$
Cost per iteration	$+$	$+$
Parallelizability	$-$	$+$
Effectiveness	$+$	$-$
Numerical sensitivity	$-$	$+$

Note: Z needs to have linearly independent vectors

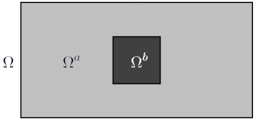
Domain decomposition

Consider model Ω with two rigid bodies Ω^a and Ω^b.



Domain decomposition

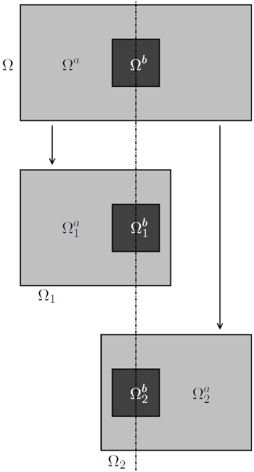
Consider model Ω with two rigid bodies Ω^a and Ω^b.



$$Z = \begin{pmatrix} Z^a & \\ & Z^b \\ Z^a & \end{pmatrix}$$

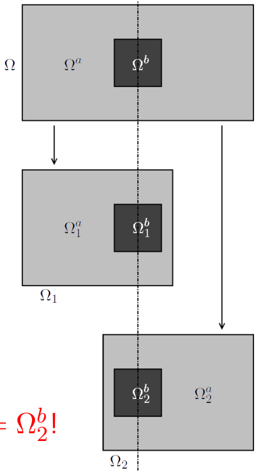
Domain decomposition

Consider model Ω with two rigid bodies Ω^a and Ω^b.



Domain decomposition

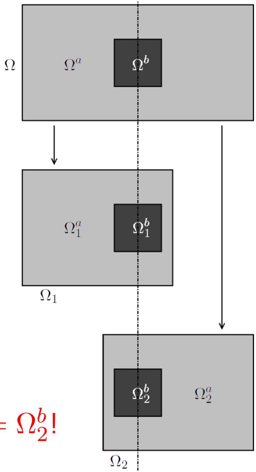
Consider model Ω with two rigid bodies Ω^a and Ω^b.



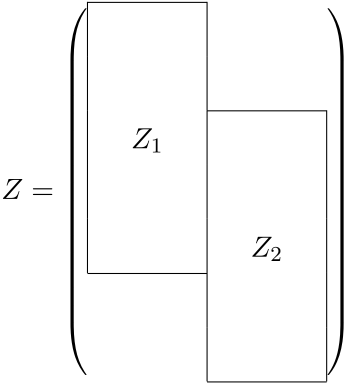
$\Omega_1^b = \Omega_2^b!$

Domain decomposition

Consider model Ω with two rigid bodies Ω^a and Ω^b.

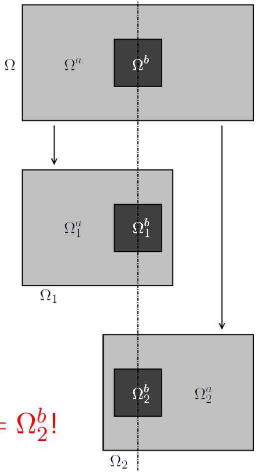


$\Omega_1^b = \Omega_2^b$!

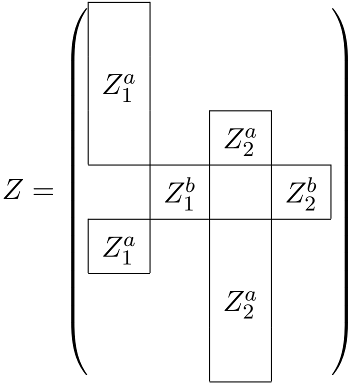


Domain decomposition

Consider model Ω with two rigid bodies Ω^a and Ω^b.

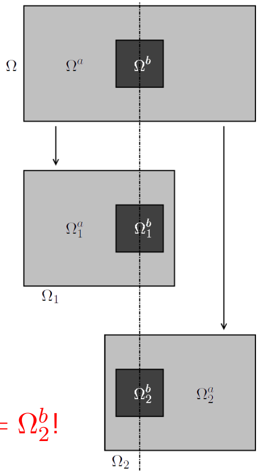


$\Omega_1^b = \Omega_2^b$!



Domain decomposition

Consider model Ω with two rigid bodies Ω^a and Ω^b.



$\Omega_1^b = \Omega_2^b$!

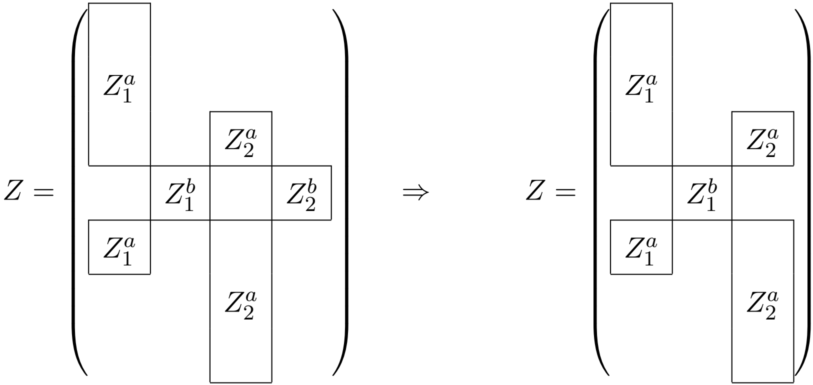
$$Z = \begin{pmatrix} Z_1^a & & Z_2^a & \\ & Z_1^b & & Z_2^b \\ Z_1^a & & Z_2^a & \end{pmatrix}$$

Linear dependent columns Z_1^b and Z_2^b!

Measures concerning the coarse matrix

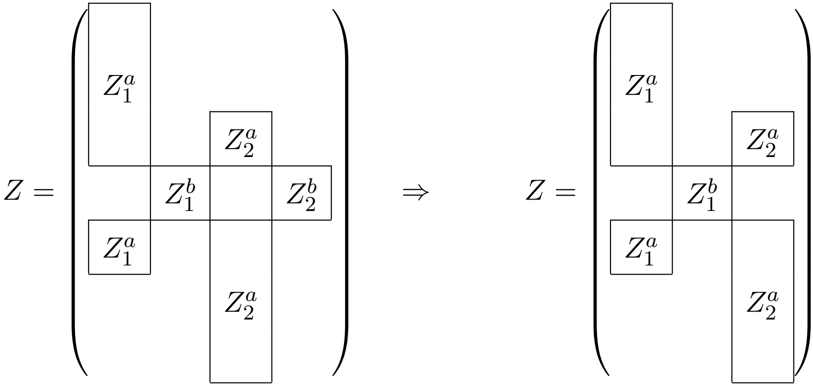
Measures concerning the coarse matrix

- Remove small overlapping bodies



Measures concerning the coarse matrix

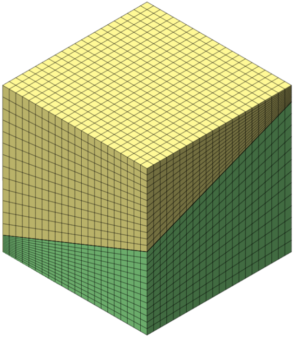
- Remove small overlapping bodies



- Switch from deflation to coarse grid correction if

$$\kappa(E) \gg 1.$$

The SplittedCube case



- Layer of interface elements splits a cube $\Rightarrow$ two bodies
- 161.711 degrees of freedom
- Constraints at three planes
- Uniform load on top

Results for SplittedCube case

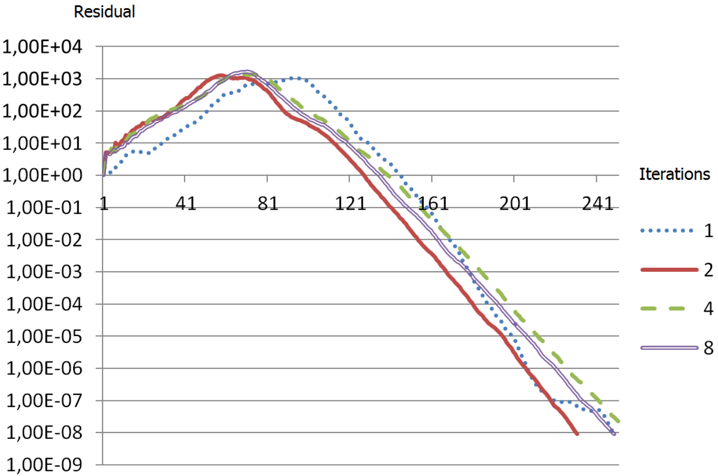
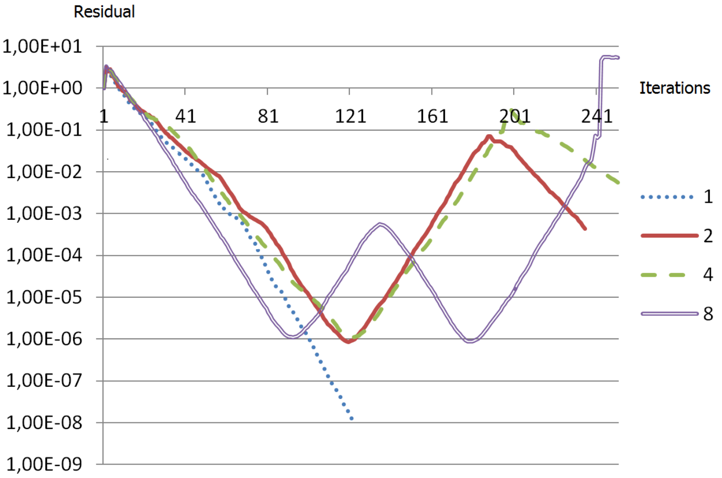


Figure: Preconditioned Conjugate Gradient (PCG)

Results for SplittedCube case



$\kappa(E)$
$3 \cdot 10^8$
$6 \cdot 10^8$
$9 \cdot 10^8$
$3 \cdot 10^{19}$

Figure: Deflated PCG

Results for SplittedCube case

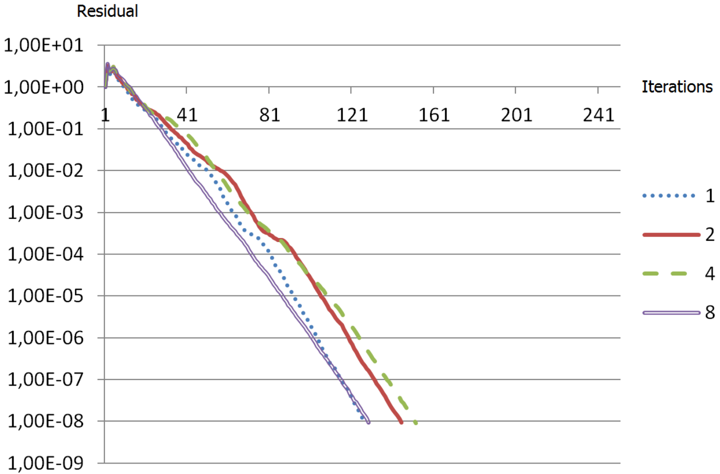
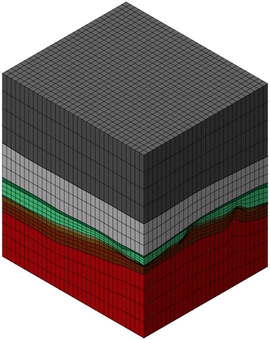


Figure: Coarse grid correction PCG

The Geo case



- Eight layers of different materials $\Rightarrow$ two bodies
- 73.336 degrees of freedom
- Constraints at two planes and all edges
- Pressure load at center

Results for Geo case

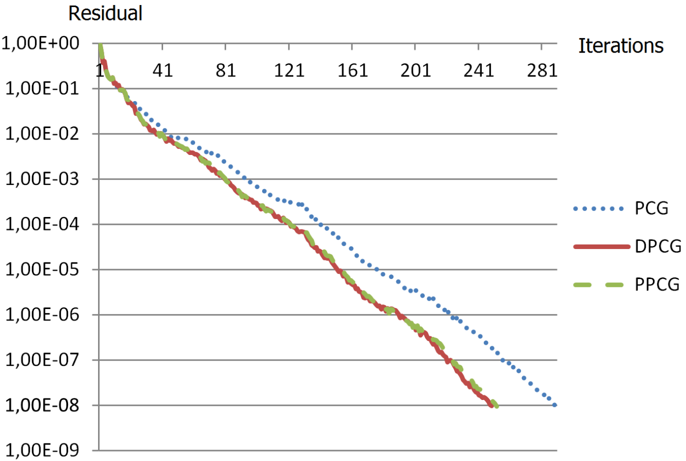


Figure: Computation with one domain

Conclusions

Enhancements

- Identify rigid bodies based on element matrices
- Reuse of rigid bodies
- Remove significantly overlapping rigid bodies
- Switch from deflation to coarse grid correction
- Tested on 2572 test problems
- Great advantage for stiffness jumps of 10^3 or larger

Conclusions

Enhancements

- Identify rigid bodies based on element matrices
- Reuse of rigid bodies
- Remove significantly overlapping rigid bodies
- Switch from deflation to coarse grid correction
- Tested on 2572 test problems
- Great advantage for stiffness jumps of 10^3 or larger

Future research

- (Physics-based) preconditioner for elements with scalar degrees of freedom
- Identify rigid bodies before domain decomposition
- Alternative non-symmetric solver: IDR(s)

Enhancing iterative solvers in DIANA

Enhancing iterative solution methods for general FEM computations using rigid body modes.

Alex Sangers

Delft Institute of Applied Mathematics
TNO DIANA

June 27, 2014

Additional slides

Approximate rigid bodies

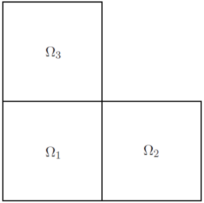
- Stiffness jumps in the underlying model

Approximate rigid bodies

- Stiffness jumps in the underlying model

- Domain decomposition

  1. Divide domain Ω into subdomains Ω_i.
  2. Compute the local solutions of subdomains Ω_i.
  3. Compute the global solution.



Choosing Z

- Eigenvectors:

$$Z = \begin{pmatrix} v_1 & \dots & v_k \end{pmatrix}.$$

- Subdomains:

$$Z_{ij} = \begin{cases} 1 & \text{if } i \in \Omega_j, \\ 0 & \text{otherwise.} \end{cases}$$

- Rigid body modes:
  Z is the rigid body modes of the approximate rigid bodies.

Rigid body modes

Consider a one-element rigid body with nodes $\underline{x}_1 = (x_1, y_1, z_1)$ and $\underline{x}_2 = (x_2, y_2, z_2)$.

$$\begin{array}{c} \\ \\ x_1 \\ x_2 \\ y_1 \\ y_2 \\ z_1 \\ z_2 \\ \\ \\ \end{array} \qquad Z = \begin{pmatrix} & & & \emptyset & & \\ & & & & & \\ 1 & 0 & 0 & 0 & -z_1 & y_1 \\ 1 & 0 & 0 & 0 & -z_2 & y_2 \\ 0 & 1 & 0 & z_1 & 0 & -x_1 \\ 0 & 1 & 0 & z_2 & 0 & -x_2 \\ 0 & 0 & 1 & -y_1 & x_1 & 0 \\ 0 & 0 & 1 & -y_2 & x_2 & 0 \\ & & & & & \\ & & & \emptyset & & \end{pmatrix}$$

The coarse matrix

Recall the coarse matrix

$$E = Z^T K Z.$$

- Deflation: $\Pi^{\perp} = I - KZE^{-1}Z^T,$
- Coarse grid correction: $P_C = I + ZE^{-1}Z^T.$

The condition (quality) of E is

$$\kappa(E) = \|E\| \cdot \|E^{-1}\|.$$

- $\kappa(E) \approx 1$ ✓
- $\kappa(E) \approx 10^{16}$ ✗

- Z needs to have linearly independent vectors

Condition number of the coarse matrix

The inverse of E is computed by a QR-decomposition:

$$E^{-1} = R^{-1}Q^T,$$

where R^{-1} is explicitly computed.

The condition number $\kappa_2(E)$ is bounded by:

$$\kappa_2(E) \leq \kappa_F(E),$$

with

$$\begin{aligned}\kappa_F(E) &= \|E\|_F \|E^{-1}\|_F \\ &= \|QR\|_F \|R^{-1}Q^T\|_F \\ &= \|R\|_F \|R^{-1}\|_F.\end{aligned}$$

Results for SplittedCube case

	$\kappa(E)$	PCG		PARDISO		DPCG		PPCG	
		iter	CPU(s)	iter	CPU(s)	iter	CPU(s)	iter	CPU(s)
1	$3 \cdot 10^8$	248	66.2	1	132.0	122	37.8	126	38.7
2	$6 \cdot 10^8$	229	43.1	1	78.0	n/a	n/a	144	30.7
4	$9 \cdot 10^8$	255	35.4	1	47.1	n/a	n/a	151	23.3
8	$3 \cdot 10^{19}$	248	37.5	1	47.9	n/a	n/a	128	23.4

Results for Geo case

	$\kappa(E)$	PCG		PARDISO		DPCG		PPCG	
		iter	CPU(s)	iter	CPU(s)	iter	CPU(s)	iter	CPU(s)
1	$2 \cdot 10^4$	289	17.5	1	18.6	248	16.6	251	16.7
2	$7 \cdot 10^4$	255	10.3	1	11.1	255	10.9	258	10.8
4	$1 \cdot 10^5$	284	9.2	1	7.1	279	10.1	283	9.8
8	$4 \cdot 10^5$	256	9.2	1	6.5	251	10.6	255	9.9

Future research

- Identify rigid bodies before the partitioning
- Specialize the rigid body modes
- Better reuse information in nonlinear iteration loop
  - Eigenvector deflation
  - Optimize the parameter for rigid body reuse
- (Physics-based) preconditioner for models with temperature or pressure
- The non-symmetric iterative solver IDR(s)