Nonlinear Algebraic Multigrid for Problems with Constraints

Michael Gee
mwgee@sandia.gov

Trilinos User Group Meeting 2005

Solve efficiently in parallel large systems of nonlinear equations
from implicit transient/quasistatic nonlinear solid mechanics simulations:

→ geometric, material and contact nonlinearities

→ Analytic Jacobian (stiffness matrix) can not be given or
is of poor quality/precision

→ unstructured 3D discretizations

→ parallel, scalable and memory efficient

→ stable, reliable, user—friendly (commercial use)

→ approach:

→ nonlinear algebraic multigrid based on smoothed aggregation

→ use graph of constrained problem

→ construct Jacobian operator(s) by block colored finite differencing

Objective

The method basically consists of 4 components:

→ full approximation scheme (FAS) nonlinear V−cycle

→ algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

→ precond. nonlinear CG (nlnCG) / precond. matrixfree Newton method as nonlinear smoothers/solvers

→ block colored finite differencing scheme

Implementation in *Trilinos* framework:

→ make use of several *Trilinos* subpackages:
ML (algebraic multigrid)
NOX (nonlinear solvers, finite differencing)
AztecOO (Krylov solvers)
EpetraExt (parallel coloring)
Epetra, *Teuchos*, *...*

Objective

→ Full approximation scheme (FAS) nonlinear V−cycle

→ Algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

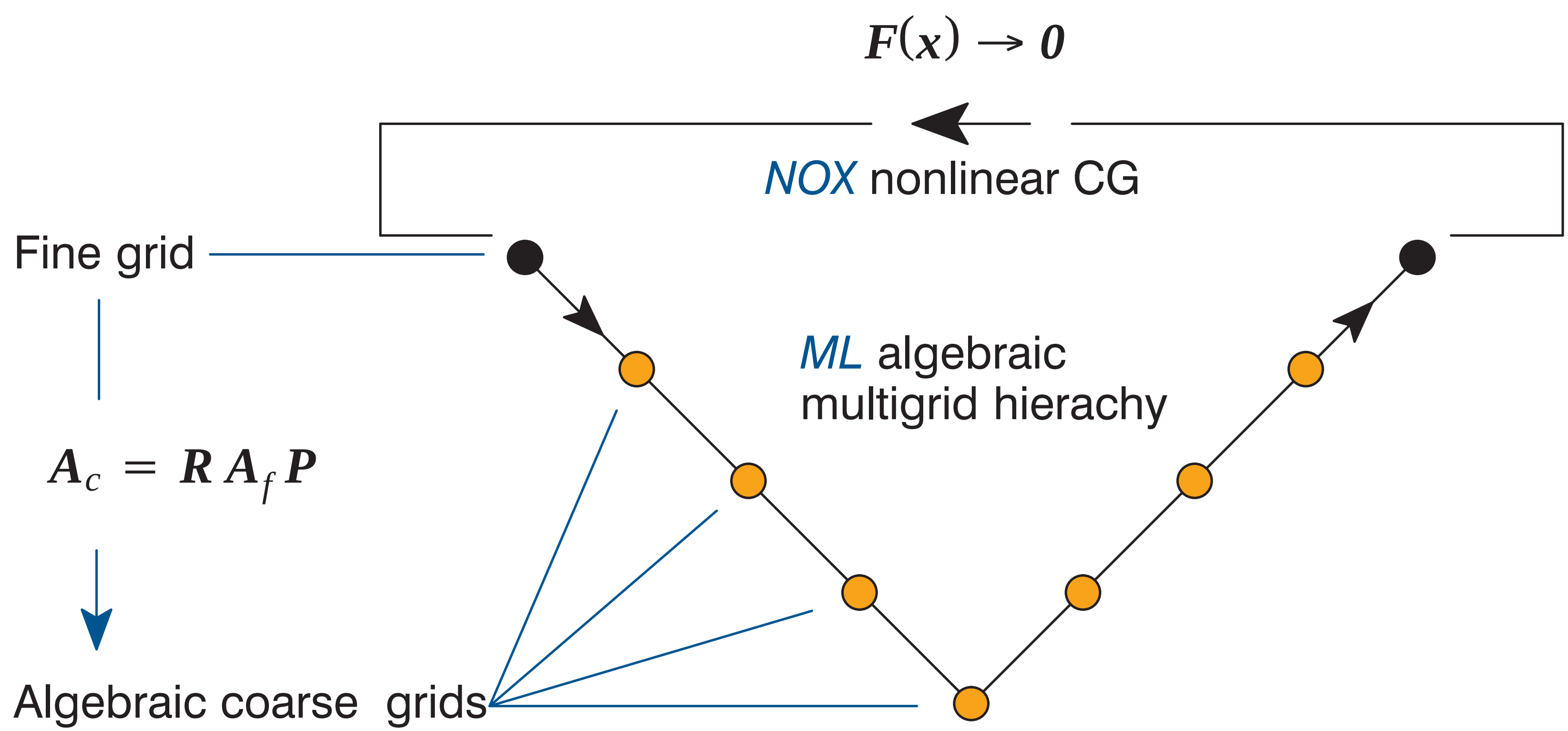
→ Preconditioned nonlinear CG / quasi−Newton method

→ Block colored finite differencing scheme

→ Constraints

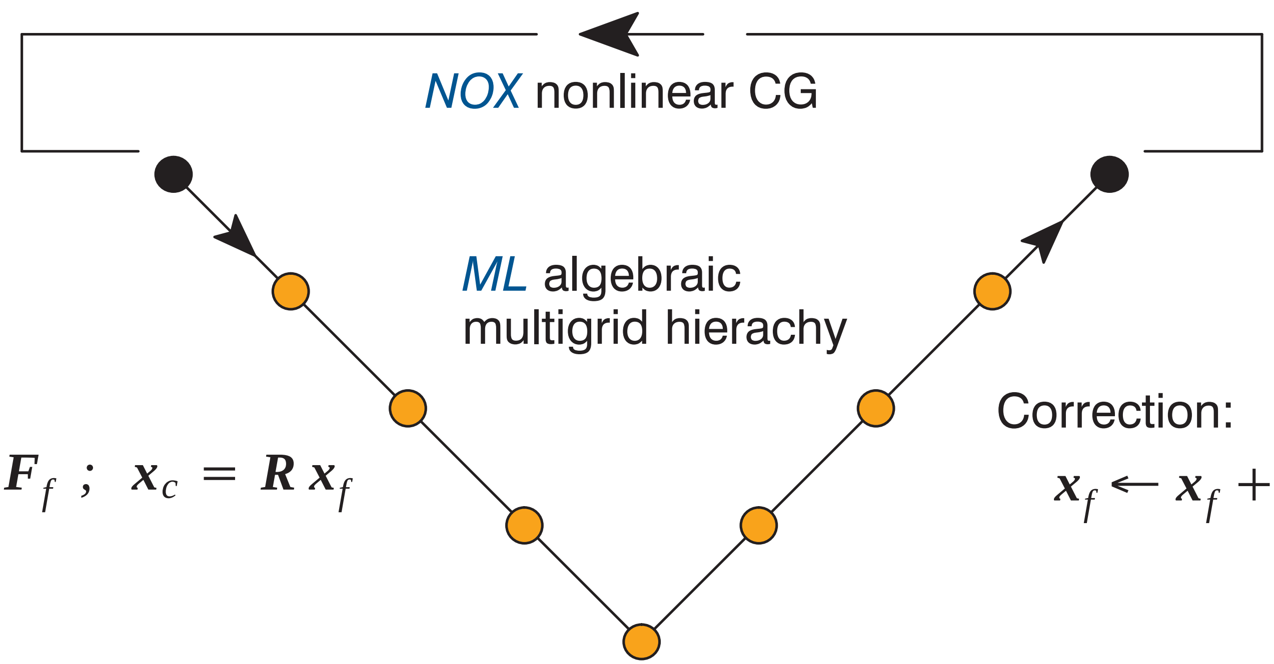
→ Example

Outline



Nonlinear V–cycle (Full Approx. Scheme – FAS)

$$F(x) \rightarrow 0$$



Restriction:

$$F_c = R\,F_f \;;\; x_c = R\,x_f$$

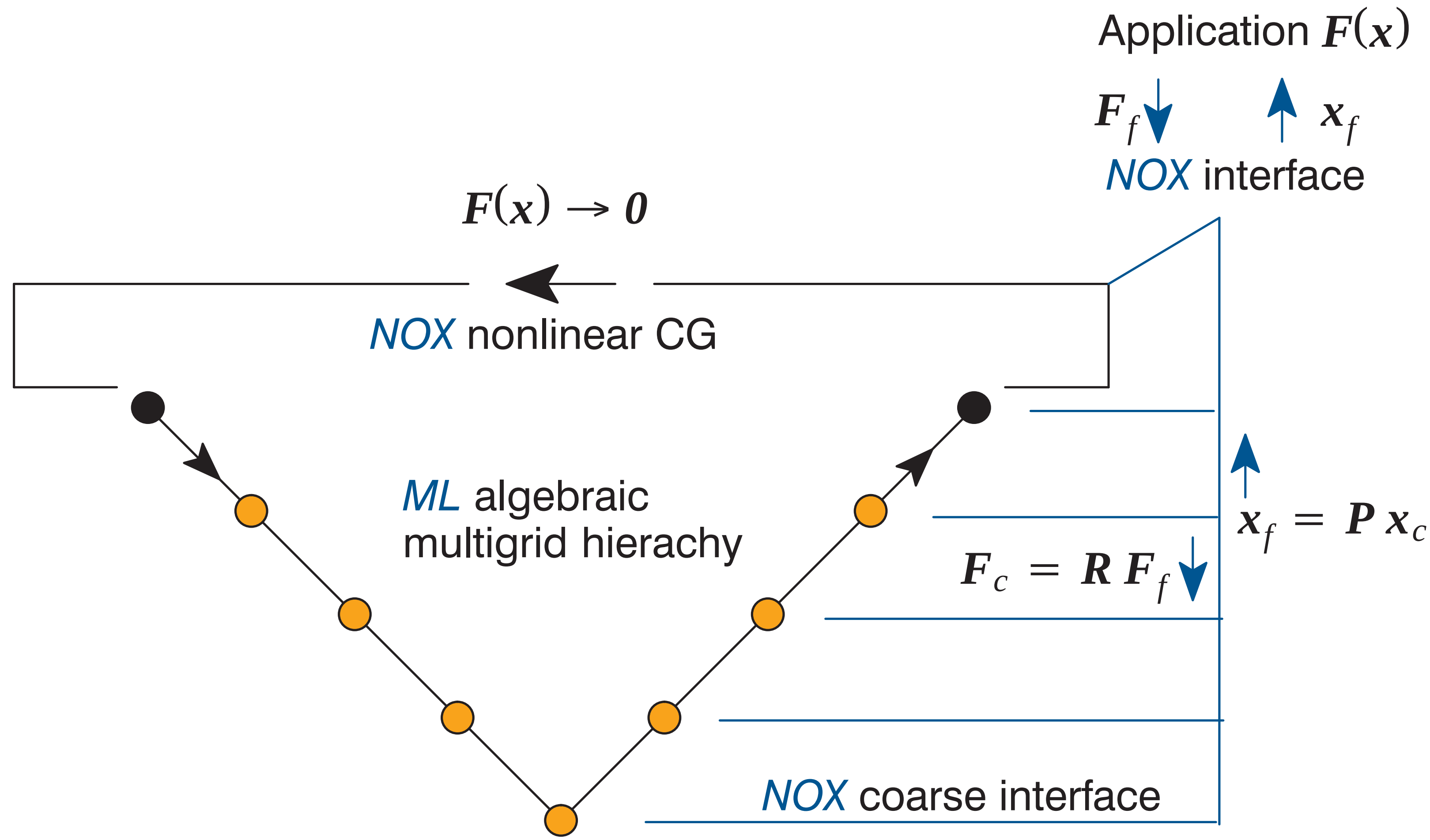
Correction:

$$x_f \leftarrow x_f + P\left(x_c - R\,x_f\right)$$

Modified coarse problem:

$$\tilde{F}_c\,(x_c) = F_c\left(R\,x_f\right) - R\,F_f\left(x_f\right) \rightarrow 0_c$$

Nonlinear V–cycle (Full Approx. Scheme – FAS)



→ fine interface implements `NOX::EpetraNew::Interface::Required/Jacobian` and `NOX::Parameter::PrePostOperator`

→ nonlinear ML preconditioner implements `Epetra_Operator` and `NOX::EpetraNew::Interface::Preconditioner`

Variational residual evaluation

FAS_Vcycle($\boldsymbol{x}_{(l)}$, $\boldsymbol{F}_{(l)}$, l)

set up FAS—system of eqns.

$$\boldsymbol{F}_1 = \boldsymbol{F}_{(l)}$$
$$\boldsymbol{F}_2 = \boldsymbol{F}\left(\boldsymbol{x}_{(l)}\right)$$
$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_{(l)}$$

coarse solve

if (level $\equiv$ coarsest level)

Solve $\boldsymbol{F}\left(\boldsymbol{x}_{(l)}\right) = \boldsymbol{F}_2 - \boldsymbol{F}_1$

$\boldsymbol{x}_{(l)} \leftarrow \boldsymbol{x}_{(l)} - \tilde{\boldsymbol{x}}$

Return

presmoothing

Relax on $\boldsymbol{F}\left(\boldsymbol{x}_{(l)}\right) = \boldsymbol{F}_2 - \boldsymbol{F}_1$

restrict, call cycle, add correction

$\boldsymbol{x}_{(l+1)} = \boldsymbol{R}\,\boldsymbol{x}_{(l)}$; $\boldsymbol{F}_{(l+1)} = \boldsymbol{R}\,\boldsymbol{F}\left(\boldsymbol{x}_{(l)}\right)$

FAS_Vcycle($\boldsymbol{x}_{(l+1)}$, $\boldsymbol{F}_{(l+1)}$, $l + 1$)

$\boldsymbol{x}_{(l)} \leftarrow \boldsymbol{x}_{(l)} + \boldsymbol{P}\,\boldsymbol{x}_{(l+1)}$

post—smoothing

Relax on $\boldsymbol{F}\left(\boldsymbol{x}_{(l)}\right) = \boldsymbol{F}_2 - \boldsymbol{F}_1$

$\boldsymbol{x}_{(l)} \leftarrow \boldsymbol{x}_{(l)} - \tilde{\boldsymbol{x}}$

Return

Note:

→ FAS—V—cycle is potentially matrix—free (operators $\boldsymbol{P}$ and $\boldsymbol{R}$ can be based on graph only)

→ A method to Relax/Solve is not yet introduced here

Nonlinear V—cycle

Sandia National Laboratories

→ Full approximation scheme (FAS) nonlinear V−cycle

→ Algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

→ Preconditioned nonlinear CG / quasi−Newton method

→ Block colored finite differencing scheme

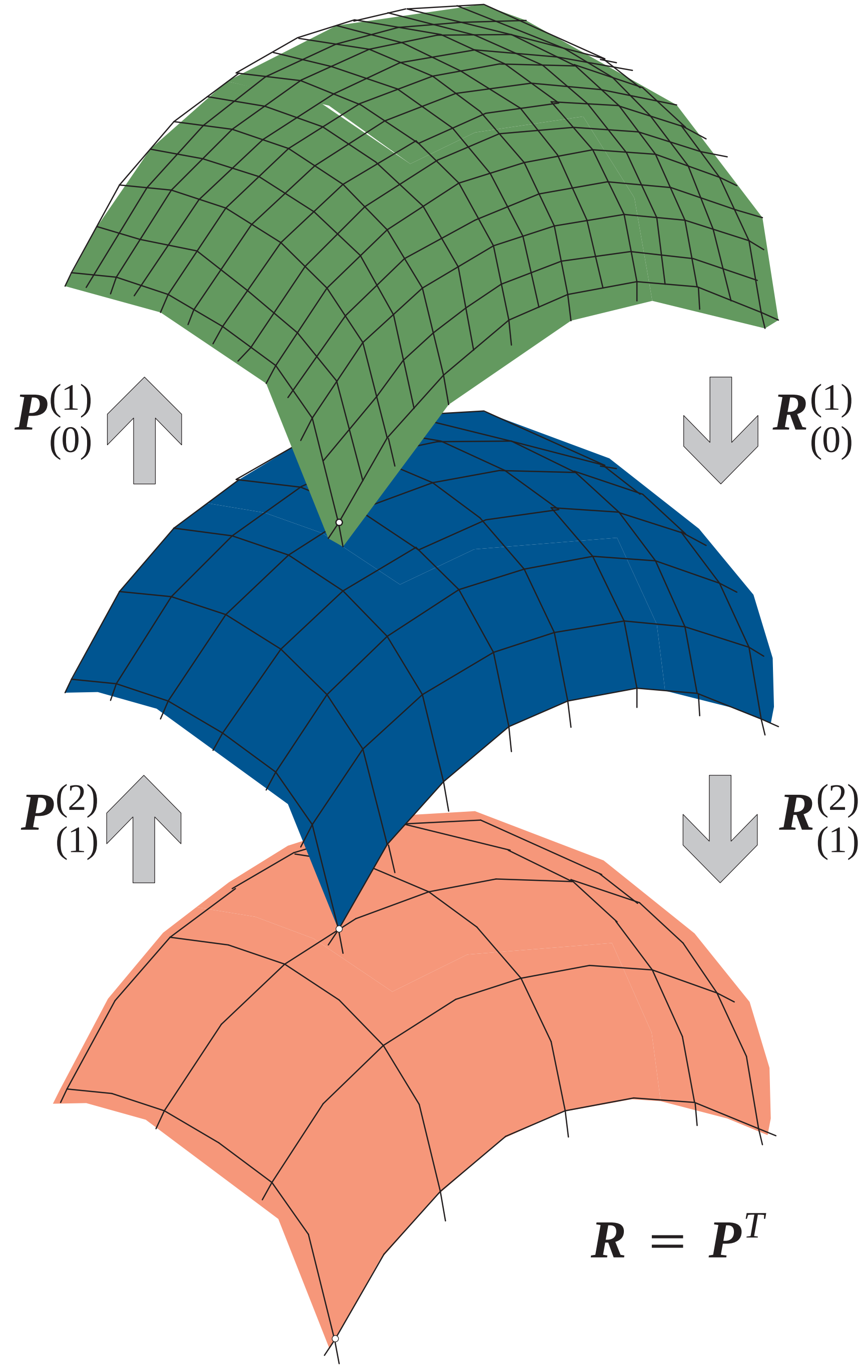
→ Constraints

→ Example

Outline

Sandia National Laboratories

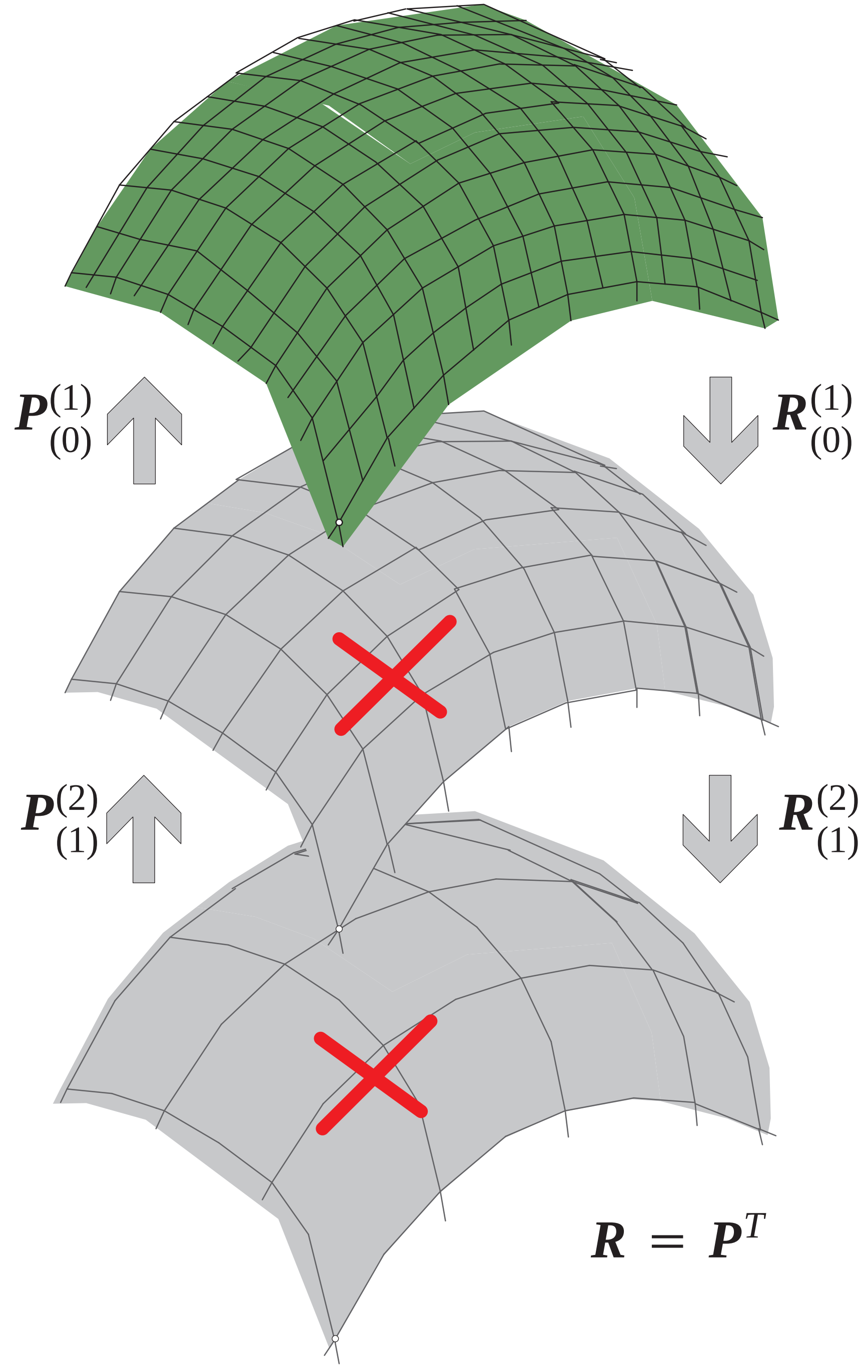
→ In geometric multigrid, transfer operators are obtained from element shape functions



Smoothed aggregation multigrid

→ In geometric multigrid, transfer operators are obtained from element shape functions

→ In smoothed aggregation multigrid, transfer operators are constructed from aggregate (patch) – wise representation of rigid body modes
[VANEK, MANDEL, BREZINA 1990–2005]

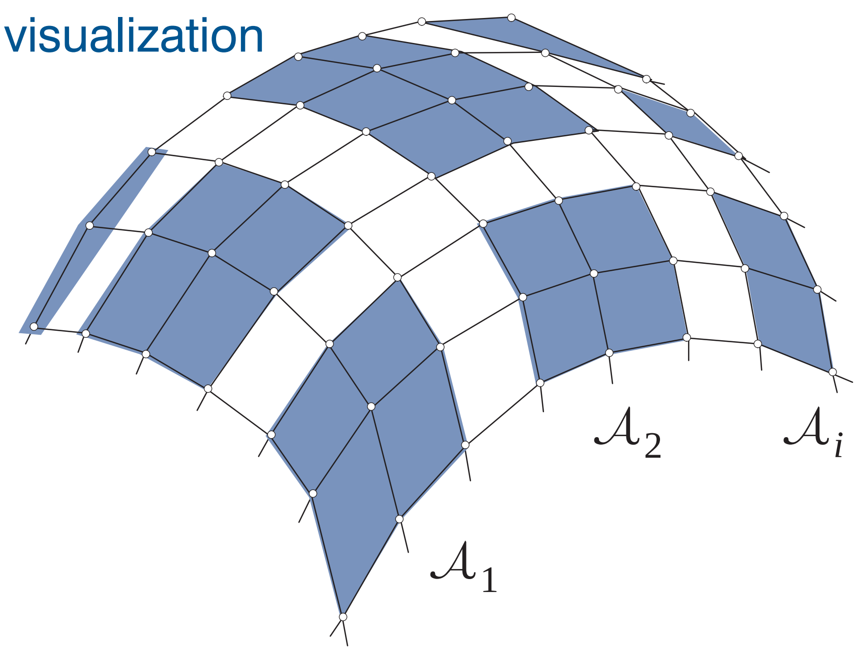


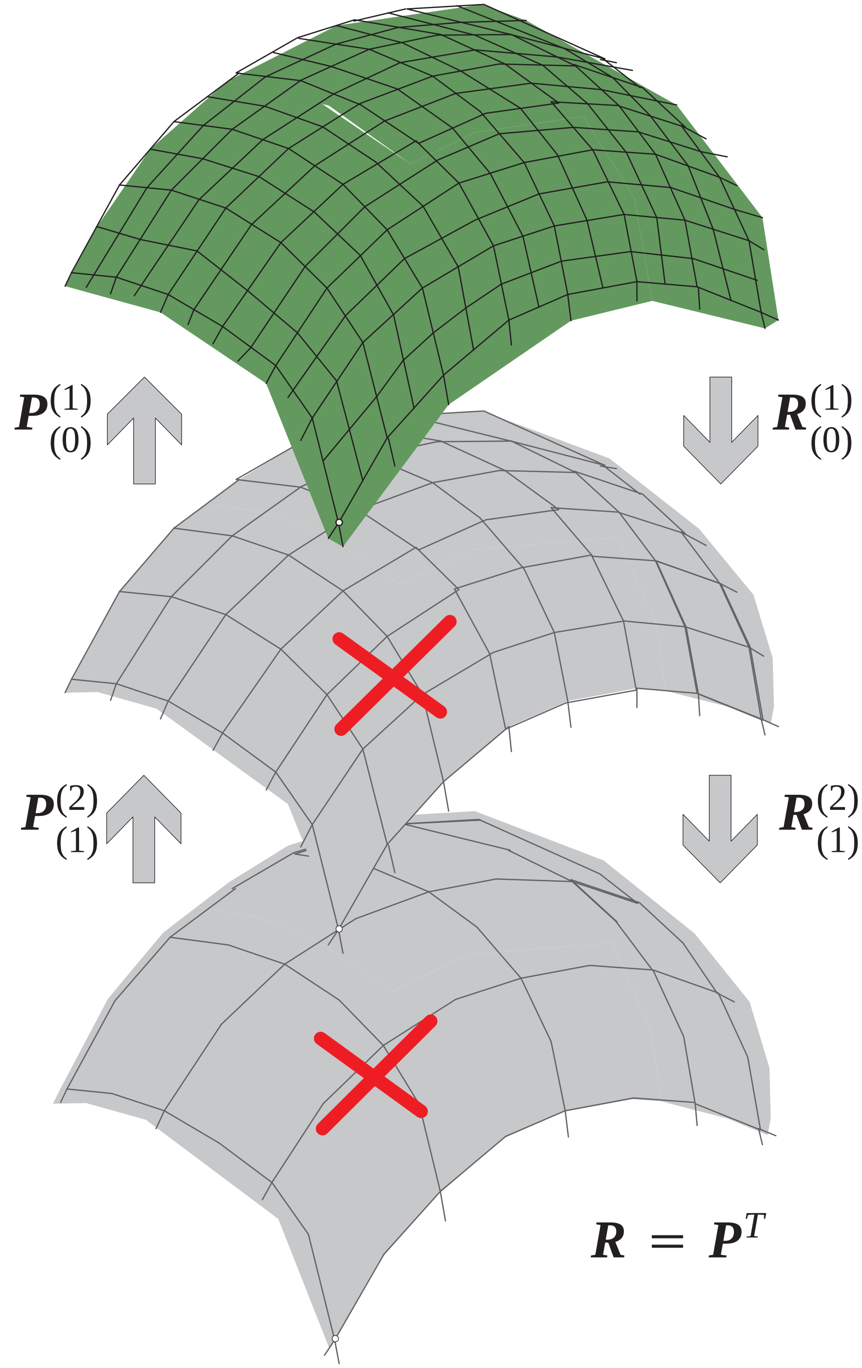
Smoothed aggregation multigrid

- → In smoothed aggregation multigrid, transfer operators are constructed from aggregate (patch) – wise representation of rigid body modes
  [VANEK, MANDEL, BREZINA 1990–2005]

- → Aggregates are constructed from the graph of the problem on some level, not from any grid

symbolic visualization only!





Smoothed aggregation multigrid

- In smoothed aggregation multigrid, transfer operators are constructed from aggregate (patch) – wise representation of rigid body modes

- Rigid body modes on fine level (0) are input:

$$\boldsymbol{B}^{(0)} = \left[\boldsymbol{b}_1^{(0)} \; \boldsymbol{b}_2^{(0)} \; \dots \; \boldsymbol{b}_6^{(0)} \right]$$

  and are represented on coarse levels exactly:

$$\text{range}(\boldsymbol{B}^{(0)}) \subset \text{range}(\tilde{\boldsymbol{P}}_{(0)}^{(l)}) \,,$$

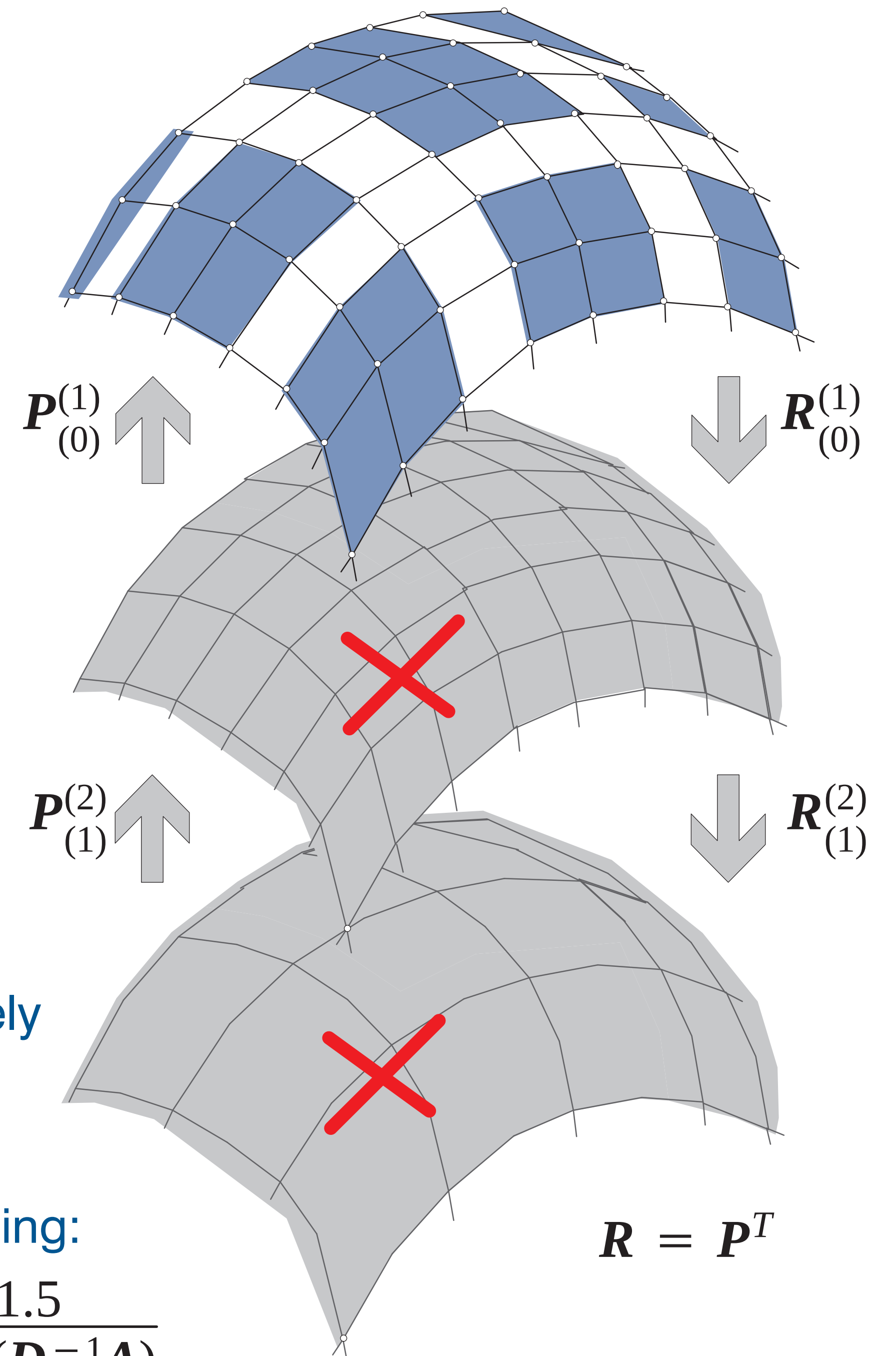
$$\tilde{\boldsymbol{P}}_{(0)}^{(l)} = \hat{\boldsymbol{P}}_{(0)}^{(1)} \hat{\boldsymbol{P}}_{(0)}^{(1)} \dots \hat{\boldsymbol{P}}_{(0)}^{(1)} \,, \quad l = 1, \dots, L$$

- Tentative prolongations are constructed recursively using the aggregates:

$$\boldsymbol{B}^{(l-1)} = \boldsymbol{P}_{(l-1)}^{(l)} \boldsymbol{B}^{(l)} \quad , \quad l = 1, \dots, L$$

- Final prolongations are obtained through smoothing:

$$\boldsymbol{P}_{(l-1)}^{(l)} = \left(\boldsymbol{I} - \omega \, \boldsymbol{D}^{-1} \boldsymbol{A}_{(l)} \right) \hat{\boldsymbol{P}}_{(l-1)}^{(l)} \quad , \quad \omega = \frac{1.5}{\lambda_{\max}(\boldsymbol{D}^{-1}\boldsymbol{A})}$$



Smoothed aggregation multigrid

→ In smoothed aggregation multigrid, transfer operators are constructed from aggregate (patch) – wise representation of rigid body modes

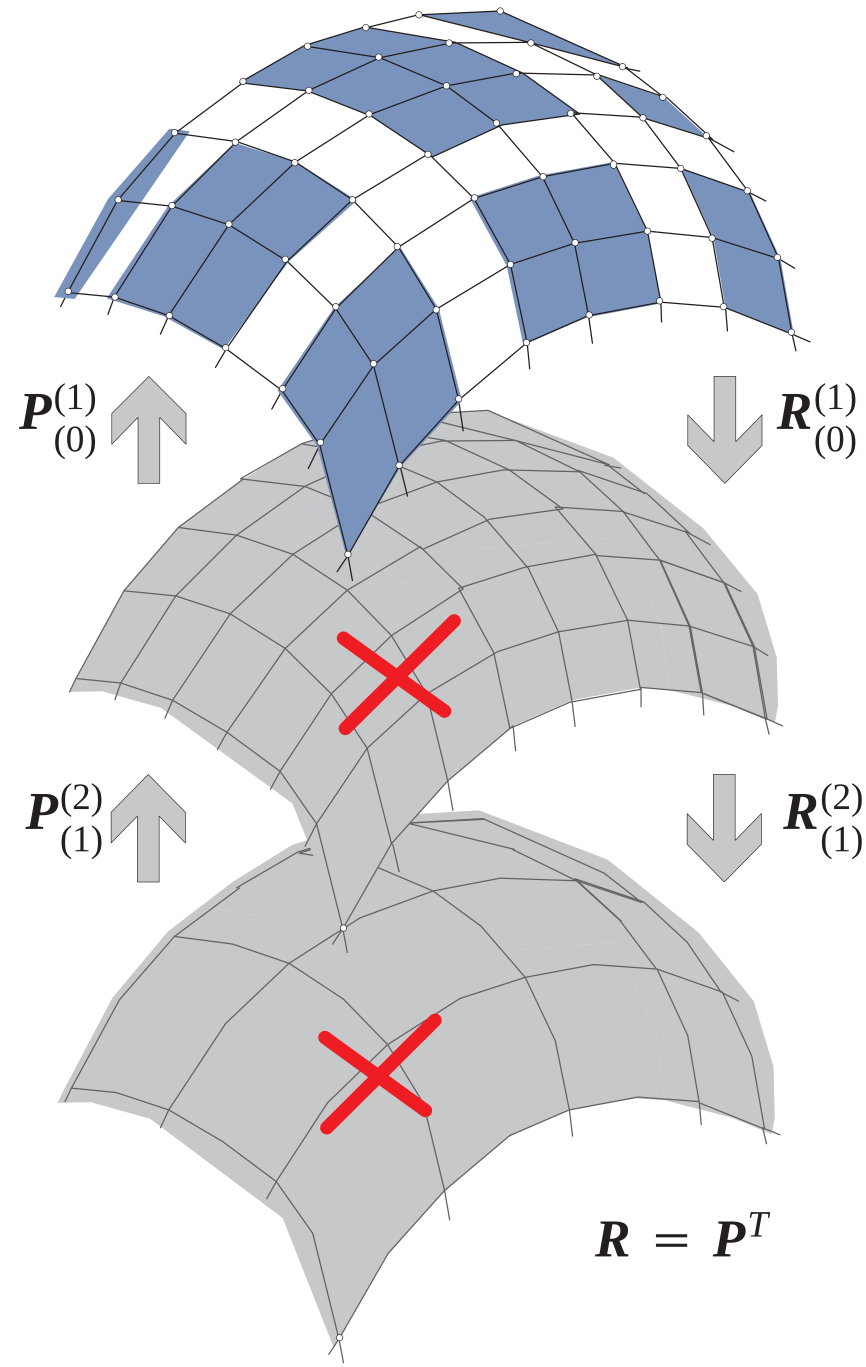
→ Rigid body modes on fine level (0) are input:

$$B^{(0)} = \left[b_1^{(0)} \; b_2^{(0)} \; \ldots \; b_6^{(0)} \right]$$

→ Additional near–nullspace components not captured well by the existing MG–preconditioner can be computed and added to the set of functions to be represented exactly (adaptive SA [BREZINA ET AL. 2005]):

$$B^{(0)} = \left[b_1^{(0)} \; b_2^{(0)} \; \ldots \; b_6^{(0)} \; p_1^{(0)} \; p_2^{(0)} \; \ldots \right]$$

→ Rebuilding MG–hierarchy results in improved convergence (and higher cost in setup and per iteration)



Smoothed aggregation multigrid

→ Full approximation scheme (FAS) nonlinear V−cycle

→ Algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

→ Preconditioned nonlinear CG / quasi−Newton method

→ Block colored finite differencing scheme

→ Constraints

→ Example

Outline

Compute the residual:

$$\boldsymbol{F}_j = \boldsymbol{F}(\boldsymbol{x}_j) \quad \text{on level } (l)$$

Compute the conjugate search direction:

$$\boldsymbol{s}_{j+1} = \boldsymbol{M}^{-1}\boldsymbol{F}_j + \beta_j \boldsymbol{s}_j$$

$$\beta_j = \begin{cases} \max\left[\dfrac{\boldsymbol{F}_j^T \boldsymbol{M}^{-1}(\boldsymbol{F}_j - \boldsymbol{F}_{j-1})}{\boldsymbol{F}_{j-1}^T \boldsymbol{M}^{-1}\boldsymbol{F}_{j-1}}, 0\right] & , \; j > 0 \\ 0 & , \; j = 0 \end{cases}$$

Compute new solution

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{s}_{j+1}$$

with a secant line search parameter:

$$\alpha_j = \frac{\boldsymbol{F}_j^T \boldsymbol{s}_j}{\boldsymbol{F}^T(\boldsymbol{x}_j + \boldsymbol{s}_{j+1})\boldsymbol{s}_j - \boldsymbol{F}_j^T \boldsymbol{s}_j}$$

Choose some convergence criteria, e.g.

$$\|\boldsymbol{F}\|_2 < \epsilon \qquad \text{or} \qquad \langle \boldsymbol{F}, \boldsymbol{M}^{-1}\boldsymbol{F}\rangle < \epsilon$$

Note:

→ nonlinear CG is matrixfree

→ the only operator needed is $\boldsymbol{M}^{-1}$

Preconditioned nonlinear CG

→ Preconditioned Newton−CG or Newton−GMRES (NOX/AztecOO) as a nonlinear smoother/solver

→ Preconditioner currently can be any solution/smoothing method with an existing interface to ML (MLS/SGS/Chebychev/Amesos_KLU/Ifpack)

→ Operator for the Krylov method can be matrixfree or Jacobian

→ Max. # Krylov iterations and Newton steps can be prescribed

→ Can be Newton on some, preconditioned nlnCG on other grids

Preconditioned matrixfree Newton−Krylov

Sandia National Laboratories

→ Full approximation scheme (FAS) nonlinear V−cycle

→ Algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

→ Preconditioned nonlinear CG / quasi−Newton method

→ Block colored finite differencing scheme

→ Constraints

→ Example

Outline

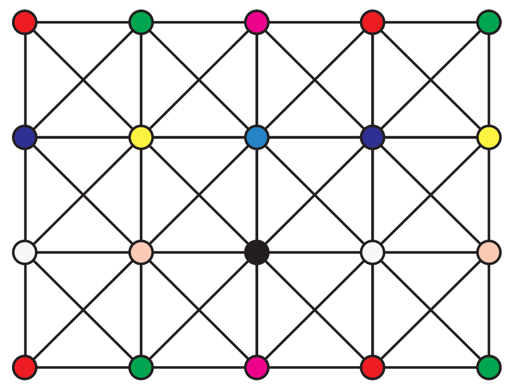
Sandia National Laboratories

Parallel graph coloring (*Trilinos–ML/EpetraExt*)

→ parallel distance–2 graph coloring for finite differencing of the Jacobian

→ as distance–2 coloring is *very* expensive, collapse graph to nodal block graph and expand obtained colors to original graph

(about $c\,\mathcal{O}\left(\frac{1}{n^3}\right)$ cheaper then scalar coloring with n dofs/node)



Colored parallel finite difference scheme (*Trilinos–NOX*)

$$\boldsymbol{A}_{ij} = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{x}_j} = \frac{\boldsymbol{F}_i\left(\boldsymbol{x} + \delta \boldsymbol{e}_j\right) - \boldsymbol{F}_i(\boldsymbol{x})}{\delta} \quad , \quad \delta = \alpha \left|\boldsymbol{x}_j\right| + \beta$$

→ $\boldsymbol{A}_{ij}$ belonging to the same color are computed simultanously

→ Number of evaluations of F is governed by the bandwith, not by the size of the problem.

Parallel colored finite differencing

→ Full approximation scheme (FAS) nonlinear V−cycle

→ Algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

→ Preconditioned nonlinear CG / quasi−Newton method

→ Block colored finite differencing scheme

→ Constraints

→ Example

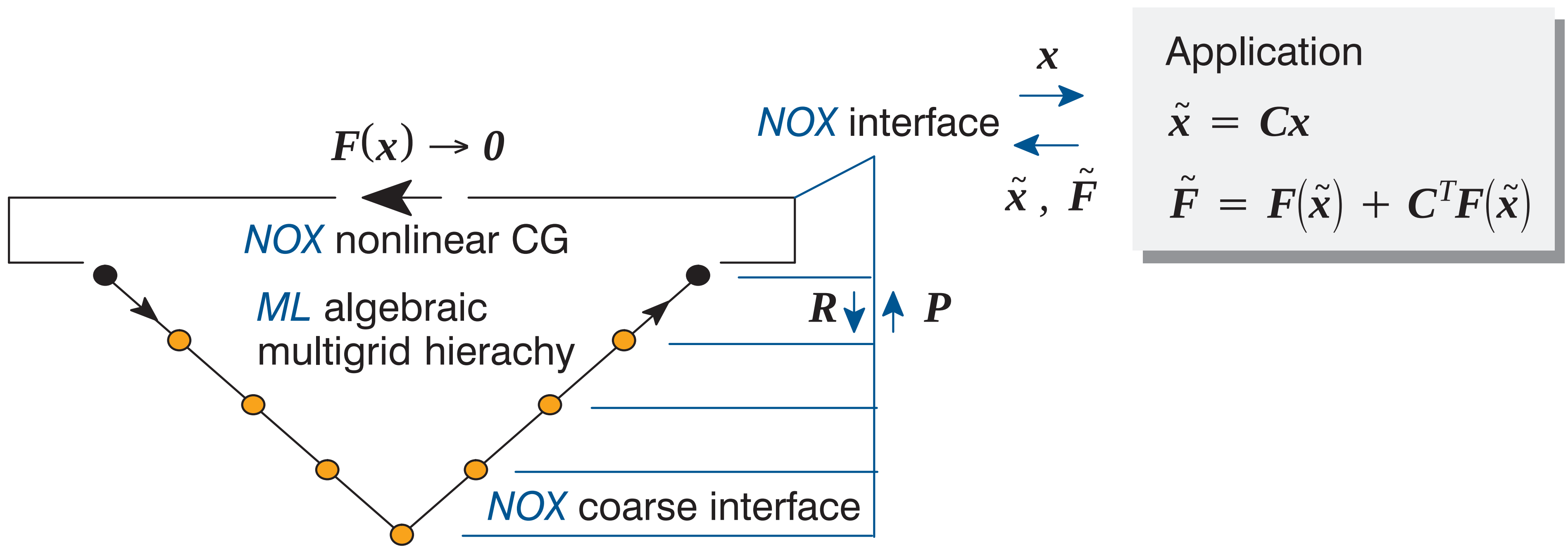
Outline

Enforcement of constraints

$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

→ The nonlinear preconditioner operates on A / x only

→ Constraints are enforced by the underlying application



Constraints

Enforcement of constraints

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{C}^T \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}$$

→ The nonlinear preconditioner operates on $\boldsymbol{A}$ / $\boldsymbol{x}$ only

→ Constraints are enforced by the underlying application

→ To obtain a good multigrid preconditioner to the constrained problem, the MG−hierachy has to build upon a modified $\boldsymbol{A}$

$$\tilde{\boldsymbol{A}}_{ij} = \frac{\partial \tilde{\boldsymbol{F}}_i}{\partial \boldsymbol{x}_j} = \frac{\tilde{\boldsymbol{F}}_i\left(\boldsymbol{x} + \delta \boldsymbol{e}_j\right) - \tilde{\boldsymbol{F}}_i(\boldsymbol{x})}{\delta}$$

NOX interface $\underset{\tilde{\boldsymbol{x}}\,,\ \tilde{\boldsymbol{F}}}{\overset{\boldsymbol{x}}{\rightleftarrows}}$

Application

$\tilde{\boldsymbol{x}} = \boldsymbol{C}\boldsymbol{x}$

$\tilde{\boldsymbol{F}} = \boldsymbol{F}\left(\tilde{\boldsymbol{x}}\right) + \boldsymbol{C}^T\boldsymbol{F}\left(\tilde{\boldsymbol{x}}\right)$

(...skipping some details and problems e.g. with frictionless contact)

Constraints

Sandia National Laboratories

Enforcement of constraints

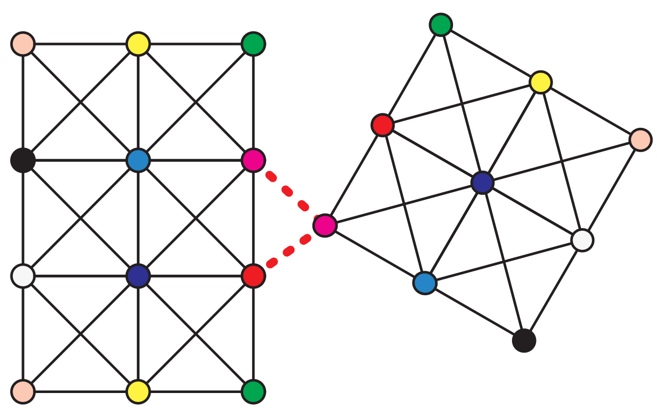
$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

→ The nonlinear preconditioner operates on A / x only

→ Constraints are enforced by the underlying application

→ To obtain a good multigrid approximation of the constrained problem, the MG−hierachy has to build upon a modified A

→ Coloring for finite differencing of A has to be based on a constraint−modified graph



Constraints

Enforcement of constraints

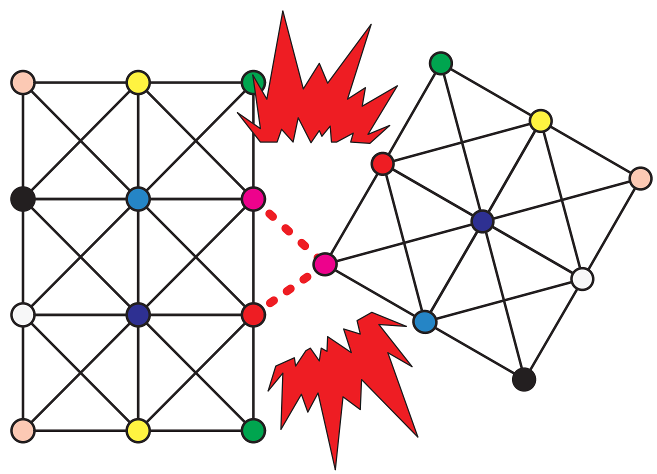
$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

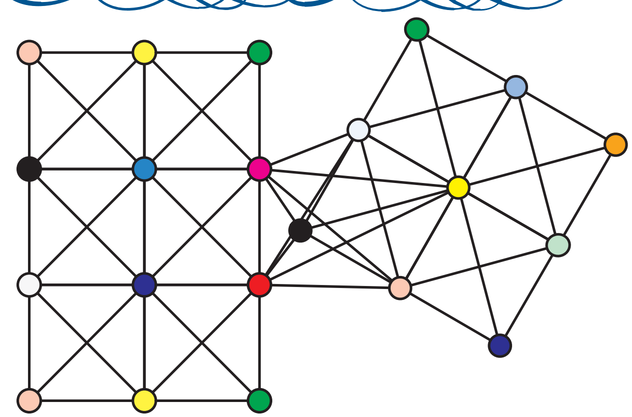
→ The nonlinear preconditioner operates on A / x only

→ Constraints are enforced by the underlying application

→ To obtain a good multigrid approximation of the constrained problem, the MG−hierachy has to build upon a modified A

→ Coloring for finite differencing of A has to be based on a constraint−modified graph





Constraints

→ Full approximation scheme (FAS) nonlinear V−cycle

→ Algebraic multigrid hierarchy (smoothed/plain aggregation multigrid)

→ Preconditioned nonlinear CG / quasi−Newton method

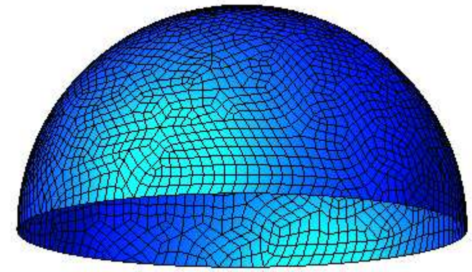
→ Block colored finite differencing scheme

→ Constraints

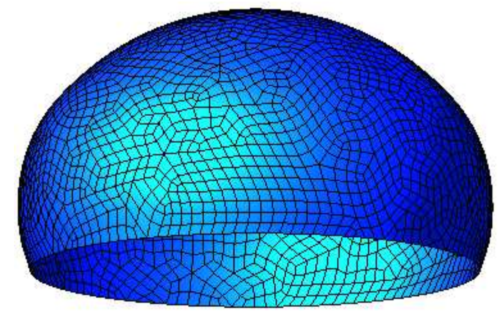
→ Example

Outline

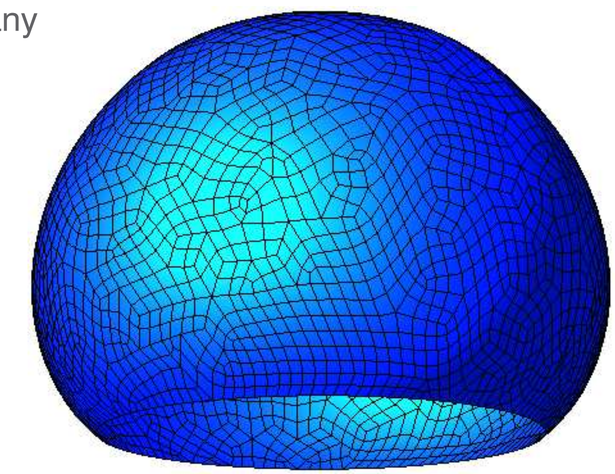
- → shell discretization with thickness change
- → Radius/thickness = 100
- → Ogden hyperelastic material
- → Hydrostatic internal pressure load
- → 19926 equations
- → implicit nonlinear dynamics (gen$-\alpha$)
- → 193 load steps, $\Delta t = 0.01$ s



$t = 0.0s$



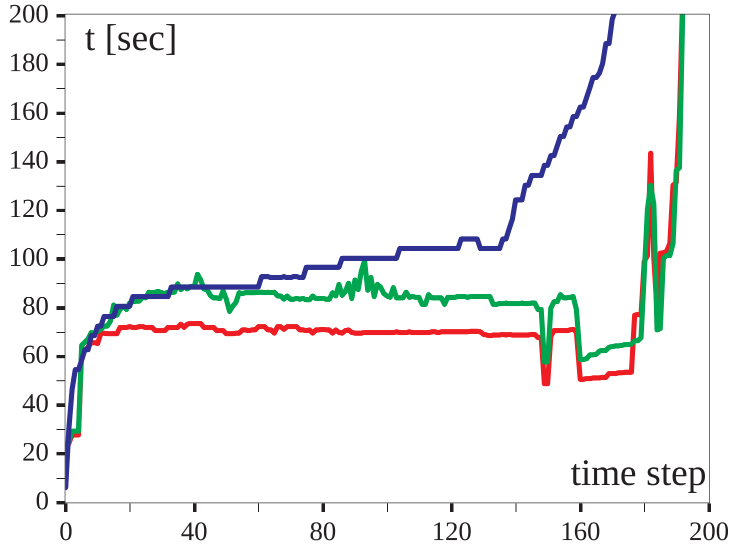
$t = 1.0s$

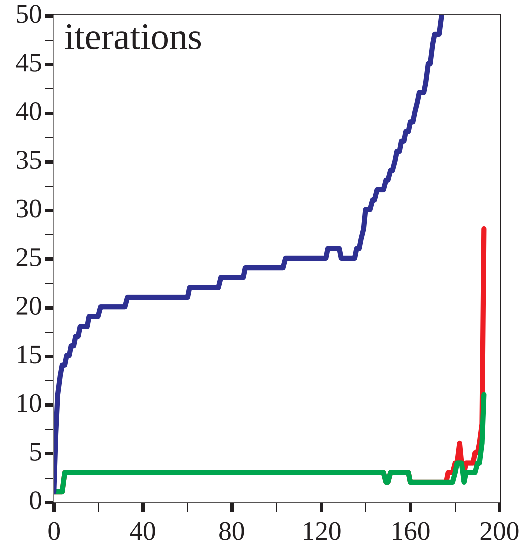


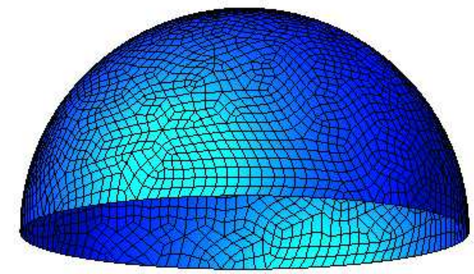
$t = 1.93s$

Inflated half sphere 1

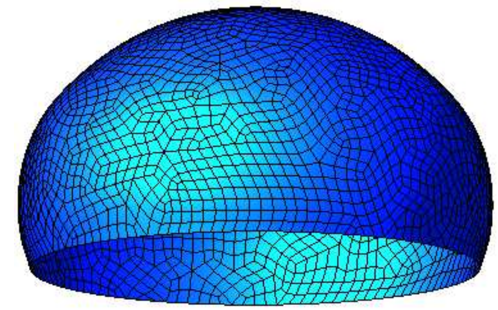
- 3–grid linear MG preconditioned nlnCG
  - Chebychev on fine grid
  - DD–SGS on medium grid
  - LU on coarse grid
- 3–grid nonlinear MG
  - preconditioned nlnCG on all grids
  - smoothers as before
- 3–grid nonlinear MG
  - preconditioned matrixfree Newton–Krylov
  - smoothers as before



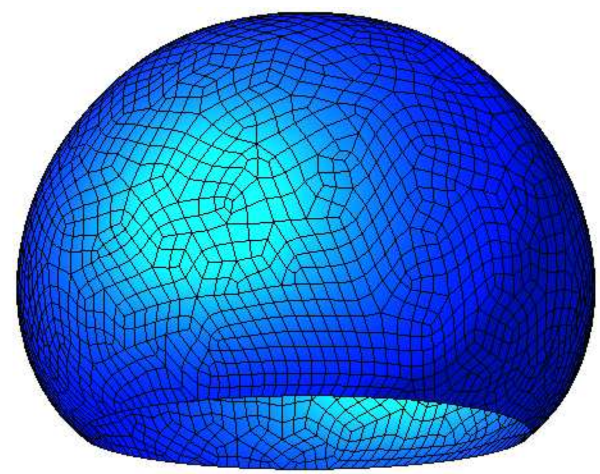




$t = 0.0s$



$t = 1.0s$



$t = 1.93s$

Inflated half sphere 1

→ nonlinear quasistatics,123 load steps

→ 3–grid nonlinear MG

preconditioned nlnCG on all grids

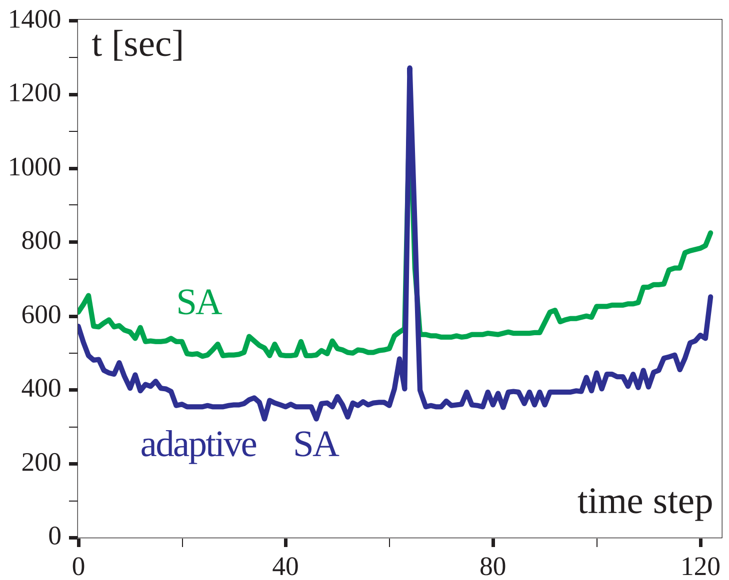
smoothers as before

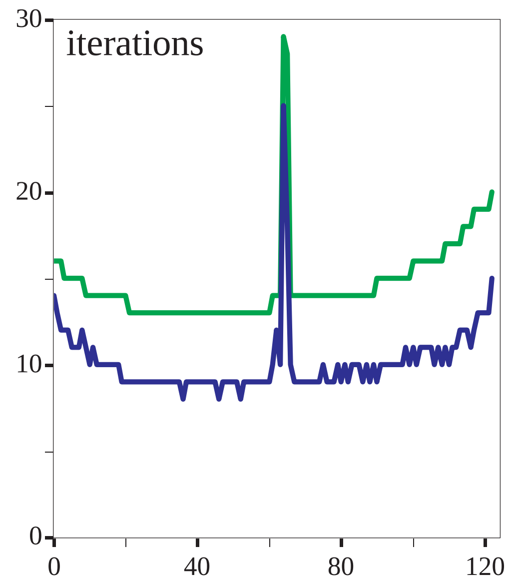
→ 3–grid nonlinear MG

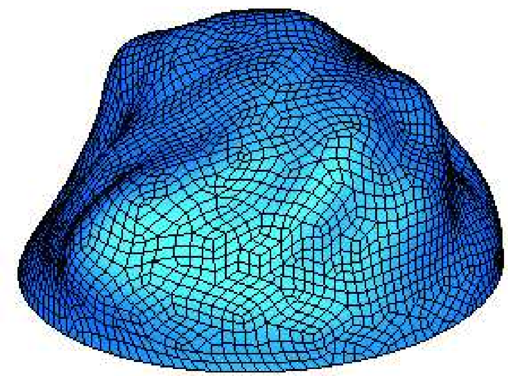
preconditioned nlnCG on all grids

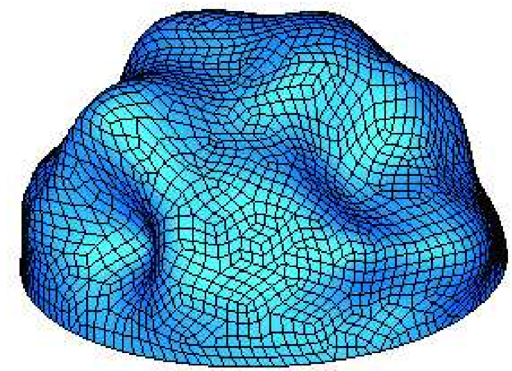
smoothers as before

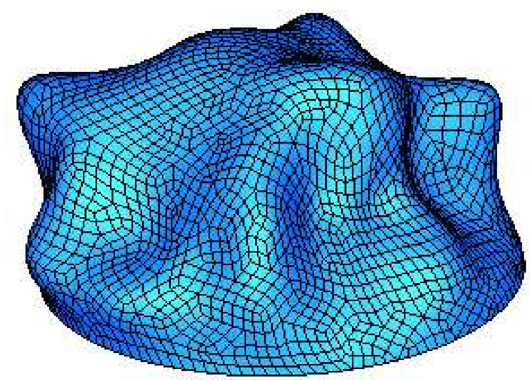
<u>3 adaptive near–nullspace components</u>











Inflated half sphere 1

- → An algebraic nonlinear multigrid preconditioner is studied
- → Evaluation of nonlinear residual on coarse grids is variational
- → nonlinear CG / Newton as nonlinear smoothers
- → Jacobian can be obtained by parallel block colored finite differencing
- → Handling of constraints (some open issues)
- → Algorithm is excellent for implicit methods in formerly explicit FE−code, as such codes normally do not supply Jacobians but have a very fast evaluation of the nonlinear function $\boldsymbol{F}$
- → Use adaptive smoothed aggregation MG [BREZINA ET AL. 2005]

Thanks to Ray Tuminaro, Kendall Pierson, Alan Williams, Russell Hooper

Conclusions

Sandia National Laboratories