Thoughts on Composing Nonlinear Solvers

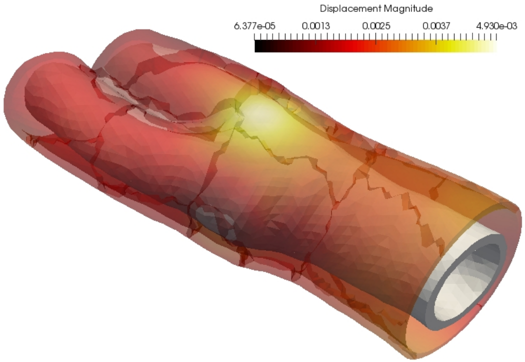
Matthew Knepley

Computer Science and Engineering
University at Buffalo

AMS Sectional Meeting
Honolulu, HI Mar 23, 2019

Pioneering Work

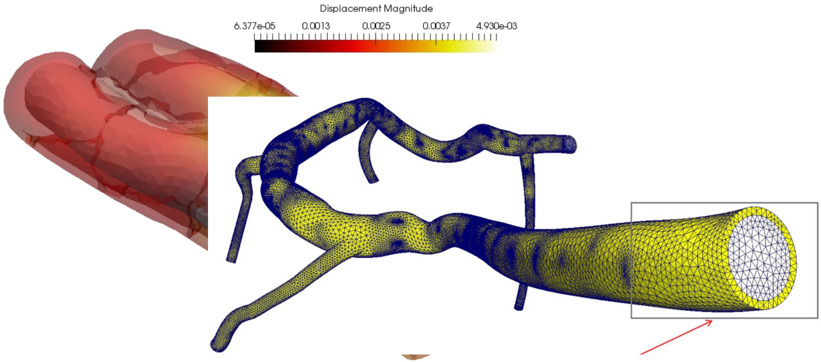
Nonlinear Elimination for Computational Hyperelasticity
Shihua Gong and Xiao-Chuan Cai



Pioneering Work

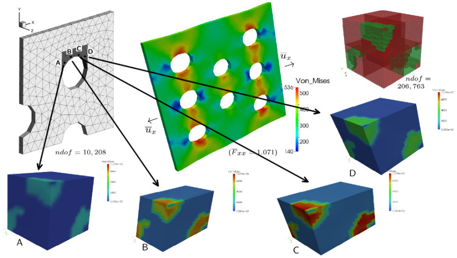
Nonlinear Elimination for Computational Hyperelasticity
Shihua Gong and Xiao-Chuan Cai

Nonlinear Elimination for Fluid-Structure Interaction
Fande Kong and Xiao-Chuan Cai



Pioneering Work

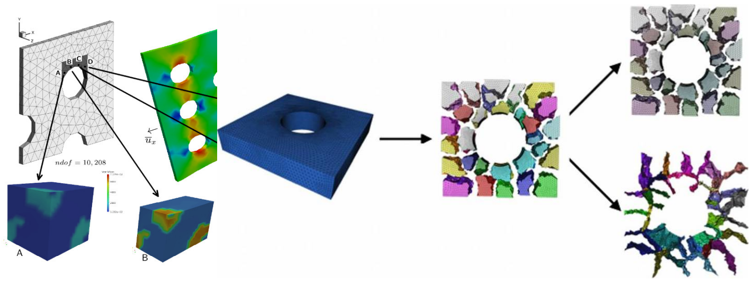
Nonlinear FETI-DP and BDDC methods: A unified framework and parallel results
Axel Klawonn, Martin Lanser, Oliver Rheinbach, Matthias Uran



Pioneering Work

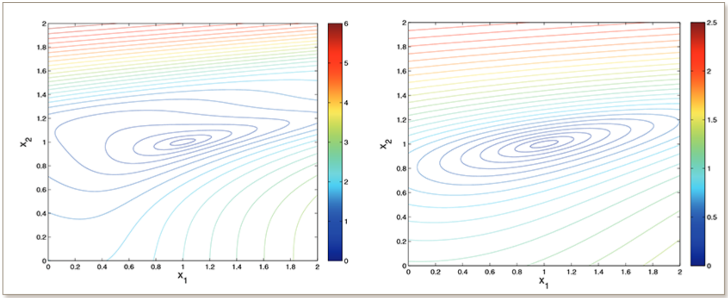
Nonlinear FETI-DP and BDDC methods: A unified framework and parallel results
Axel Klawonn, Martin Lanser, Oliver Rheinbach, Matthias Uran

A Highly Scalable Implementation of Inexact Nonlinear FETI-DP Without Sparse Direct Solvers
Axel Klawonn, Martin Lanser, Oliver Rheinbach



Pioneering Work

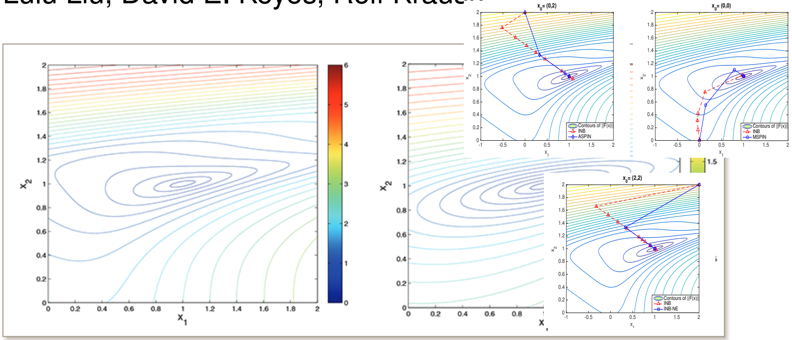
Field-split preconditioned inexact Newton algorithms
Lulu Liu and David E. Keyes



Pioneering Work

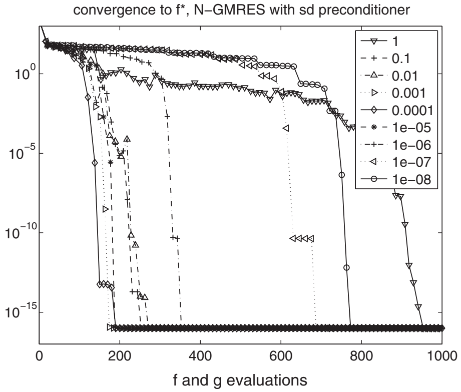
Field-split preconditioned inexact Newton algorithms
Lulu Liu and David E. Keyes

A Note on Adaptive Nonlinear Preconditioning Techniques
Lulu Liu, David E. Keyes, Rolf Krause



Pioneering Work

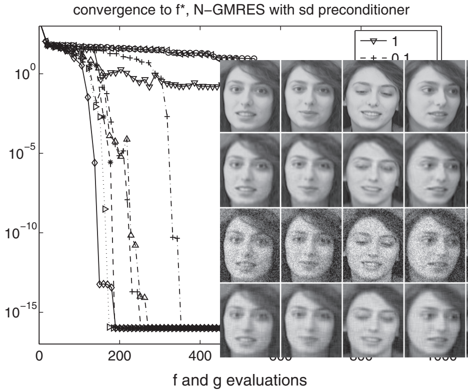
Steepest Descent Preconditioning for NGMRES Optimization
Hans De Sterck

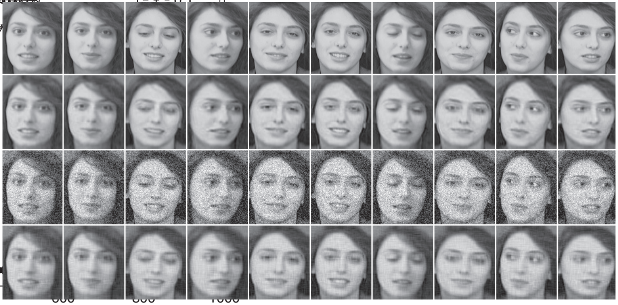


Pioneering Work

Steepest Descent Preconditioning for NGMRES Optimization
Hans De Sterck

Nonlinear Preconditioning for Tucker Decomposition
Alexander Howse and Hans De Sterck





Outline

Abstract System

Out prototypical nonlinear equation is:

$$\mathbf{F}(\mathbf{x}) = \mathbf{b}$$

and we define the residual as

$$\mathbf{r}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{b}$$

Abstract System

Out prototypical nonlinear equation is:

$$\mathbf{F}(\mathbf{x}) = \mathbf{b}$$

and we define the (linear) residual as

$$\mathbf{r}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$$

Linear Left Preconditioning

The modified equation becomes

$$P^{-1}(A\mathbf{x} - \mathbf{b}) = 0 \tag{1}$$

Linear Left Preconditioning

The modified defect correction equation becomes

$$P^{-1}\left(A\mathbf{x}_i - \mathbf{b}\right) = \mathbf{x}_{i+1} - \mathbf{x}_i \tag{2}$$

Additive Combination

The linear iteration

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\alpha P^{-1} + \beta Q^{-1})(A\mathbf{x}_i - \mathbf{b}) \qquad (3)$$

becomes the nonlinear iteration

Additive Combination

The linear iteration

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\alpha P^{-1} + \beta Q^{-1})\mathbf{r}_i \tag{4}$$

becomes the nonlinear iteration

Additive Combination

The linear iteration

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\alpha P^{-1} + \beta Q^{-1})\mathbf{r}_i \tag{4}$$

becomes the nonlinear iteration

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha(\mathcal{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i) + \beta(\mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i) \tag{5}$$

Nonlinear Left Preconditioning

From the additive combination, we have

$$P^{-1}\mathbf{r} \Longrightarrow \mathbf{x}_i - \mathcal{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) \tag{6}$$

so we define the preconditioning operation as

$$\mathbf{r}_L \equiv \mathbf{x} - \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}) \tag{7}$$

Multiplicative Combination

The linear iteration

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (P^{-1} + Q^{-1} - Q^{-1}AP^{-1})\mathbf{r}_i \tag{8}$$

becomes the nonlinear iteration

Multiplicative Combination

The linear iteration

$$\mathbf{x}_{i+1/2} = \mathbf{x}_i - P^{-1}\mathbf{r}_i \tag{9}$$

$$\mathbf{x}_i = \mathbf{x}_{i+1/2} - Q^{-1}\mathbf{r}_{i+1/2} \tag{10}$$

becomes the nonlinear iteration

Multiplicative Combination

The linear iteration

$$\mathbf{x}_{i+1/2} = \mathbf{x}_i - P^{-1}\mathbf{r}_i \tag{9}$$

$$\mathbf{x}_i = \mathbf{x}_{i+1/2} - Q^{-1}\mathbf{r}_{i+1/2} \tag{10}$$

becomes the nonlinear iteration

$$\mathbf{x}_{i+1} = \mathcal{M}(\mathbf{F}, \mathcal{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}), \mathbf{b}) \tag{11}$$

Nonlinear Right Preconditioning

For the linear case, we have

$$AP^{-1}\mathbf{y} = \mathbf{b} \tag{12}$$

$$\mathbf{x} = P^{-1}\mathbf{y} \tag{13}$$

so we define the preconditioning operation as

$$\mathbf{y} = \mathcal{M}(\mathbf{F}(\mathcal{N}(\mathcal{F}, \cdot, \mathbf{b})), \mathbf{x}_i, \mathbf{b}) \tag{14}$$

$$\mathbf{x} = \mathcal{N}(\mathbf{F}, \mathbf{y}, \mathbf{b}) \tag{15}$$

Nonlinear Preconditioning

Type	Sym	Statement	Abbreviation
Additive	$+$	$\mathbf{x} + \alpha(\mathcal{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}) - \mathbf{x}) + \beta(\mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}) - \mathbf{x})$	$\mathcal{M} + \mathcal{N}$
Multiplicative	$*$	$\mathcal{M}(\mathbf{F}, \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{b})$	$\mathcal{M} * \mathcal{N}$
Left Prec.	$-_L$	$\mathcal{M}(\mathbf{x} - \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}, \mathbf{b})$	$\mathcal{M} -_L \mathcal{N}$
Right Prec.	$-_R$	$\mathcal{M}(\mathbf{F}(\mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b})), \mathbf{x}, \mathbf{b})$	$\mathcal{M} -_R \mathcal{N}$
Inner Lin. Inv.	$\backslash$	$\mathbf{y} = J(\mathbf{x})^{-1}\mathbf{r}(\mathbf{x}) = \mathrm{K}(J(\mathbf{x}), \mathbf{y}_0, \mathbf{b})$	$\mathcal{N} \backslash \mathrm{K}$

Composing Scalable Nonlinear Algebraic Solvers (Brune et al. 2015)

Outline

Outline

Nonlinear Richardson

```
1: procedure NRICH($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
2:     $\mathbf{d} := -\mathbf{r}(\mathbf{x}_i)$
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$        ▷ $\lambda$ determined by line search
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

L Adds line search to $\mathcal{N}$

R Uses $\mathcal{N}$ to improve search direction

Nonlinear Richardson

```
1: procedure NRICH($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
2:     $\mathbf{d} := -\mathbf{r}(\mathbf{x}_i)$
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$        ▷ $\lambda$ determined by line search
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

L Adds line search to $\mathcal{N}$

R Uses $\mathcal{N}$ to improve search direction

Line Search

Equivalent to NRICH $-_L \mathcal{N}$:

$$\text{NRICH} -_L \mathcal{N}$$

Line Search

Equivalent to NRICH $-_L \mathcal{N}$:

$$\begin{aligned} &\text{NRICH} -_L \mathcal{N} \\ \text{NRICH}(&\mathbf{x} - \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}, \mathbf{b}) \end{aligned}$$

Line Search

Equivalent to NRICH $-_L \mathcal{N}$:

$$
\begin{aligned}
\text{NRICH} \, -_L \, \mathcal{N} \\
\text{NRICH}(\mathbf{x} - \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}, \mathbf{b}) \\
\mathbf{x}_{i+1} = \mathbf{x}_i - \lambda \mathbf{r}_L
\end{aligned}
$$

Line Search

Equivalent to NRICH $-_L \mathcal{N}$:

$$
\begin{aligned}
&\text{NRICH} -_L \mathcal{N} \\
&\text{NRICH}(\mathbf{x} - \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}, \mathbf{b}) \\
\mathbf{x}_{i+1} &= \mathbf{x}_i - \lambda \mathbf{r}_L \\
\mathbf{x}_{i+1} &= \mathbf{x}_i + \lambda(\mathcal{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i)
\end{aligned}
$$

Line Search

Equivalent to NRICH $-_L \mathcal{N}$:

$$\begin{aligned} &\text{NRICH} -_L \mathcal{N} \\ &\text{NRICH}(\mathbf{x} - \mathcal{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}, \mathbf{b}) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i - \lambda \mathbf{r}_L \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \lambda(\mathcal{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i) \end{aligned}$$

Let R_1 be Richardson iteration with a unit step scaling (no damping). Then we have

$$\mathcal{M} -_L \mathbb{R}_1 = \mathcal{M} \qquad \mathbb{R}_1 -_L \mathcal{M} = \mathcal{M} \tag{16}$$

so that $\mathbb{R}_1$ is the identity operation for left preconditioning, whereas for right preconditioning this is just the identity map.

Outline

Newton-Krylov

```
1: procedure $\mathcal{N}\backslash\mathrm{K}(\mathbf{F}, \mathbf{x}_i, \mathbf{b})$
2:         $\mathbf{d} = J(\mathbf{x}_i)^{-1}\mathbf{r}(\mathbf{x}_i, \mathbf{b})$        ▷ solve by Krylov method
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda\mathbf{d}$        ▷ $\lambda$ determined by line search
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

Left Preconditioned Newton-Krylov

```
1: procedure $\mathcal{N}\backslash\mathrm{K}(\mathbf{x} - \mathcal{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}_i, 0)$
2:         $\mathbf{d} = \frac{\partial(\mathbf{x}_i - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))}{\partial \mathbf{x}_i}^{-1} (\mathbf{x}_i - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))$
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

Jacobian Computation

$$\frac{\partial(\mathbf{x} - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))}{\mathbf{x}_i} = I - \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b})}{\partial \mathbf{x}_i},$$

Direct differencing would require

- one inner nonlinear iteration

per **Krylov** iteration.

Jacobian Computation

$$\frac{\partial(\mathbf{x} - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))}{\mathbf{x}_i} = I - \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b})}{\partial \mathbf{x}_i},$$

Direct differencing would require

- one inner nonlinear iteration

per **Krylov** iteration.

Jacobian Computation

Impractical!

$$\frac{\partial(\mathbf{x} - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))}{\mathbf{x}_i} = I - \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b})}{\partial \mathbf{x}_i},$$

Direct differencing would require

- one inner nonlinear iteration

per **Krylov** iteration.

Jacobian Computation

Approximation for NASM

$$\frac{\partial(\mathbf{x} - \mathcal{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}))}{\partial \mathbf{x}} = \frac{\partial(\mathbf{x} - (\mathbf{x} - \sum_b J_b(\mathbf{x}_b)^{-1} \mathbf{F}_b(\mathbf{x}_b)))}{\partial \mathbf{x}}$$
$$\approx \sum_b J_b(\mathbf{x}_{b*})^{-1} J(\mathbf{x})$$

This would require

- one inner nonlinear iteration
- small number of block solves

per **outer nonlinear** iteration.

Nonlinearly preconditioned inexact Newton algorithms (X.-C. Cai and Keyes 2002)

Right Preconditioned Newton-Krylov

```
1: procedure NK($\mathbf{F}(\mathcal{M}(\mathbf{F}, \cdot, \mathbf{b})), \mathbf{y}_i, \mathbf{b}$)
2:         $\mathbf{x}_i = \mathcal{M}(\mathbf{F}, \mathbf{y}_i, \mathbf{b})$
3:         $\mathbf{d} = J(\mathbf{x})^{-1}\mathbf{r}(\mathbf{x}_i)$
4:         $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda\mathbf{d}$                ▷ $\lambda$ determined by line search
5: end procedure
6: return $\mathbf{x}_{i+1}$
```

Jacobian Computation

First-Order Approximation

Only the action of the original Jacobian is needed at first order:

$$
\begin{aligned}
\mathbf{y}_{i+1} &= \mathbf{y}_i - \lambda \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i}^{-1} J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i)) \\
\mathcal{M}(\mathbf{F}, \mathbf{y}_{i+1}) &= \mathcal{M}(\mathbf{F}, \mathbf{y}_i - \lambda \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i}^{-1} J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))) \\
&\approx \mathcal{M}(\mathbf{F}, \mathbf{y}_i) \\
&\quad - \lambda \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i} \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i}^{-1} J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))) \\
&= \mathcal{M}(\mathbf{F}, \mathbf{y}_i) - \lambda J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i)) \\
\mathbf{x}_{i+1} &= \mathbf{x}_i - \lambda J(\mathbf{x}_i)^{-1} \mathbf{F}(\mathbf{x}_i)
\end{aligned}
$$

$\mathcal{N} \backslash \mathrm{K} -_R \vec{M}$ is equivalent to $\mathcal{N} \backslash \mathrm{K} * \vec{M}$ at first order

A parallel adaptive nonlinear elimination preconditioned inexact Newton method for transonic full

Jacobian Computation

First-Order Approximation

Only the action of the original Jacobian is needed at first order:

$$\mathbf{y}_{i+1} = \mathbf{y}_i - \lambda \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i}^{-1} J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))$$

$$\begin{aligned}
\mathcal{M}(\mathbf{F}, \mathbf{y}_{i+1}) &= \mathcal{M}(\mathbf{F}, \mathbf{y}_i - \lambda \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i}^{-1} J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))) \\
&\approx \mathcal{M}(\mathbf{F}, \mathbf{y}_i) \\
&\quad - \lambda \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i} \frac{\partial \mathcal{M}(\mathbf{F}, \mathbf{y}_i)}{\partial \mathbf{y}_i}^{-1} J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))) \\
&= \mathcal{M}(\mathbf{F}, \mathbf{y}_i) - \lambda J(\mathcal{M}(\mathbf{F}, \mathbf{y}_i))^{-1} \mathbf{F}(\mathcal{M}(\mathbf{F}, \mathbf{y}_i)) \\
\mathbf{x}_{i+1} &= \mathbf{x}_i - \lambda J(\mathbf{x}_i)^{-1} \mathbf{F}(\mathbf{x}_i)
\end{aligned}$$

$\mathcal{N}\backslash K -_R \vec{M}$ is equivalent to $\mathcal{N}\backslash K * \vec{M}$ at first order

A parallel adaptive nonlinear elimination preconditioned inexact Newton method for transonic full

Jacobian Computation

Direct Approximation

$$\begin{aligned}\mathbf{F}(\mathcal{M}(\mathbf{F},\mathbf{y}_i,\mathbf{b})) &= J(\mathcal{M}(\mathbf{F},\mathbf{y}_i,\mathbf{b}))\frac{\partial\mathcal{M}(\mathbf{F},\mathbf{y}_i,\mathbf{b})}{\partial\mathbf{y}_i}(\mathbf{y}_{i+1}-\mathbf{y}_i)\\ &\approx J(\mathcal{M}(\mathbf{F},\mathbf{y}_i,\mathbf{b}))(\mathcal{M}(\mathbf{F},\mathbf{y}_i+\mathbf{d},\mathbf{b})-\mathbf{x}_i)\end{aligned}$$

- Solve for **d**
- Requires inner nonlinear solve for each Krylov iterate
- Needs FGMRES

On nonlinear preconditioners in Newton-Krylov methods for unsteady flows (Birken and Jameson 2010)

Outline

Anderson

```
1: procedure ANDERSON($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
2:     $\gamma_i = (\mathbf{F}_i^T \mathbf{F}_i)^{-1} \mathbf{F}_i^T \mathbf{r}(\mathbf{x}_i, \mathbf{b})$        ▷ solve LS by SVD
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i + \beta \mathbf{r}(\mathbf{x}_i, \mathbf{b}) - (\mathbf{x}_k + \beta \mathbf{F}_k)\gamma_k$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

Iterative procedures for nonlinear integral equations (Anderson 1965)

Generalized Broyden

```
1: procedure GB($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
2:     $\gamma_i = (\mathbf{F}_i^T \mathbf{F}_i)^{-1} \mathbf{F}_i^T \mathbf{r}(\mathbf{x}_i, \mathbf{b})$        ▷ solve LS by SVD
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i - G_0 \mathbf{r}(\mathbf{x}_i, \mathbf{b}) - (\mathbb{X}_k - G_0 \mathbf{F}_k)\gamma_k$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

Two classes of multisecant methods for nonlinear acceleration (Fang and Saad 2009)

Left Preconditioned Generalized Broyden

```
1: procedure GB($\mathbf{x} - \vec{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}_i, \mathbf{b}$)
2:     $\gamma_i = (\mathbf{F}_i^T \mathbf{F}_i)^{-1} \mathbf{F}_i^T (\mathbf{x} - \vec{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}))$        ▷ solve LS by SVD
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i - G_0(\mathbf{x} - \vec{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})) - (\mathbb{X}_k - G_0 \mathbf{F}_k)\gamma_k$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

We change the minimization problem,
since we minimize over different residuals.

Anderson acceleration for fixed-point iterations (Walker and Ni 2011)

Left Preconditioned Generalized Broyden

```
1: procedure GB($\mathbf{x} - \vec{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}_i, \mathbf{b}$)
2:     $\gamma_i = (\mathbf{F}_i^T \mathbf{F}_i)^{-1} \mathbf{F}_i^T (\mathbf{x} - \vec{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}))$    ▷ solve LS by SVD
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i - G_0(\mathbf{x} - \vec{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})) - (\mathbb{X}_k - G_0 \mathbf{F}_k)\gamma_k$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

We change the minimization problem,
since we minimize over different residuals.

Anderson acceleration for fixed-point iterations (Walker and Ni 2011)

Right Preconditioned Generalized Broyden

```
1: procedure GB($\mathbf{F}(\mathcal{M}(\mathbf{F}, \cdot, \mathbf{b})), \mathbf{x}_i, \mathbf{b}$)
2:     $\gamma_i = (\mathbf{F}_i^T \mathbf{F}_i)^{-1} \mathbf{F}_i^T \mathbf{r}(\mathcal{M}(\mathbf{x}_i), \mathbf{b})$        ▷ solve LS by SVD
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i - G_0 \mathbf{r}(\mathcal{M}(\mathbf{x}_i), \mathbf{b}) - (\mathbb{X}_k - G_0 \mathbf{F}_k)\gamma_k$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

We change the minimization problem,
since we use candidate solutions from the inner solver.

Krylov Subspace Acceleration for Nonlinear Multigrid Schemes with Application to Recirculating Flow (Washio and Oosterlee 2000)

Right Preconditioned Generalized Broyden

```
1: procedure GB($\mathbf{F}(\mathcal{M}(\mathbf{F}, \cdot, \mathbf{b})), \mathbf{x}_i, \mathbf{b}$)
2:     $\gamma_i = (\mathbf{F}_i^T \mathbf{F}_i)^{-1} \mathbf{F}_i^T \mathbf{r}(\mathcal{M}(\mathbf{x}_i), \mathbf{b})$        ▷ solve LS by SVD
3:     $\mathbf{x}_{i+1} = \mathbf{x}_i - G_0 \mathbf{r}(\mathcal{M}(\mathbf{x}_i), \mathbf{b}) - (\mathbb{X}_k - G_0 \mathbf{F}_k)\gamma_k$
4: end procedure
5: return $\mathbf{x}_{i+1}$
```

We change the minimization problem,
since we use candidate solutions from the inner solver.

Krylov Subspace Acceleration for Nonlinear Multigrid Schemes with Application to Recirculating Flow (Washio and Oosterlee 2000)

Outline

I ran NPC on some problem and it worked.

Outline

Outline

Rate of Convergence

What should be a Rate of Convergence? [Ptak, 1977]:

1. It should relate quantities which may be measured or estimated during the actual process
2. It should describe accurately in particular the initial stage of the process, not only its asymptotic behavior . . .

$$\|x_{n+1} - x^*\| \leq c\|x_n - x^*\|^q$$

Rate of Convergence

What should be a Rate of Convergence? [Ptak, 1977]:

1. It should relate quantities which may be measured or estimated during the actual process
2. It should describe accurately in particular the initial stage of the process, not only its asymptotic behavior . . .

$$\|x_{n+1} - x_n\| \leq c\|x_n - x_{n-1}\|^q$$

Rate of Convergence

What should be a Rate of Convergence? [Ptak, 1977]:

1. It should relate quantities which may be measured or estimated during the actual process
2. It should describe accurately in particular the initial stage of the process, not only its asymptotic behavior . . .

$$\|x_{n+1} - x_n\| \leq \omega(\|x_n - x_{n-1}\|)$$

where we have for all $r \in (0, R]$

$$\sigma(r) = \sum_{n=0}^{\infty} \omega^{(n)}(r) < \infty$$

Nondiscrete Induction

Define an approximate set $Z(r)$, where $x^* \in Z(0)$ implies $f(x^*) = 0$.

Nondiscrete Induction

Define an approximate set $Z(r)$, where $x^* \in Z(0)$ implies $f(x^*) = 0$.

For Newton's method, we use

$$Z(r) = \left\{ x \middle| \|f'(x)^{-1} f(x)\| \leq r, d(f'(x)) \geq h(r), \|x - x_0\| \leq g(r) \right\},$$

where

$$d(A) = \inf_{\|x\| \geq 1} \|Ax\|,$$

and $h(r)$ and $g(r)$ are positive functions.

Nondiscrete Induction

Define an approximate set $Z(r)$, where $x^* \in Z(0)$ implies $f(x^*) = 0$.

For $r \in (0, R]$,

$$Z(r) \subset U(Z(\omega(r)), r)$$

implies

$$Z(r) \subset U(Z(0), \sigma(r)).$$

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = G x_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = Gx_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

$$\begin{aligned} x^* &\in Z(0) \\ x_n &\in Z(\omega^{(n)}(r_0)) \end{aligned}$$

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = G x_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

$$\begin{aligned} \|x_{n+1} - x_n\| &\leq \omega^{(n)}(r_0) \\ \|x_n - x^*\| &\leq \sigma(\omega^{(n)}(r_0)) \end{aligned}$$

Nondiscrete Induction

For the fixed point iteration

$$x_{n+1} = G x_n,$$

if I have

$$x_0 \in Z(r_0)$$

and for $x \in Z(r)$,

$$\begin{aligned} \|Gx - x\| &\leq r \\ Gx &\in Z(\omega(r)) \end{aligned}$$

then

$$\begin{aligned} \|x_n - x^*\| &\leq \sigma(\omega(\|x_n - x_{n-1}\|)) \\ &= \sigma(\|x_n - x_{n-1}\|) - \|x_n - x_{n-1}\| \end{aligned}$$

Newton's Method

$$\omega_{\mathcal{N}}(r) = cr^2$$

Newton's Method

$$\omega_{\mathcal{N}}(r) = \frac{r^2}{2\sqrt{r^2 + a^2}}$$
$$\sigma_{\mathcal{N}}(r) = r + \sqrt{r^2 + a^2} - a$$

where

$$a = \frac{1}{k_0}\sqrt{1 - 2k_0 r_0},$$

k_0 is the (scaled) Lipschitz constant for f', and
r_0 is the (scaled) initial residual.

Newton's Method

$$\omega_{\mathcal{N}}(r) = \frac{r^2}{2\sqrt{r^2 + a^2}}$$
$$\sigma_{\mathcal{N}}(r) = r + \sqrt{r^2 + a^2} - a$$

This estimate is *tight* in that the bounds hold with equality for some function f,

$$f(x) = x^2 - a^2$$

using initial guess

$$x_0 = \frac{1}{k_0}.$$

Also, if equality is attained for some n_0, this holds for all $n \geq n_0$.

Newton's Method

$$\omega_{\mathcal{N}}(r) = \frac{r^2}{2\sqrt{r^2 + a^2}}$$
$$\sigma_{\mathcal{N}}(r) = r + \sqrt{r^2 + a^2} - a$$

If $r \gg a$, meaning we have an inaccurate guess,

$$\omega_{\mathcal{N}}(r) \approx \frac{1}{2} r,$$

whereas if $r \ll a$, meaning we are close to the solution,

$$\omega_{\mathcal{N}}(r) \approx \frac{1}{2a} r^2.$$

Chord Method

If we define the approximate set

$$Z(r) = \{u \in X; \|u - u_0\| \leq \sigma_{\mathcal{R}}(r_0) - \sigma_{\mathcal{R}}(r), \|f'^{-1}(u_0) f(u)\| \leq r\}, \quad (17)$$

we get a rate of convergence

$$\omega_{\mathcal{R}}(r) = \frac{1}{2} k r^2 + r \left(1 - \sqrt{1 - 2k(r_0 - r)}\right) \quad (18)$$

and corresponding estimate function

$$\sigma_{\mathcal{R}}(r) = \frac{1}{k} \sqrt{a^2 + 2kr} - a. \quad (19)$$

Chord Method

For rate of convergence,

$$\omega_{\mathcal{R}}(r) = \frac{1}{2}kr^2 + r\left(1 - \sqrt{1 - 2k(r_0 - r)}\right) \tag{20}$$

For $r \approx r_0$,

$$\omega_{\mathcal{R}}(r) \approx kr_0 r - \frac{1}{2}kr^2 \tag{21}$$

Chord Method

For rate of convergence,

$$\omega_{\mathcal{R}}(r) = \frac{1}{2}kr^2 + r\left(1 - \sqrt{1 - 2k(r_0 - r)}\right) \tag{20}$$

For $r \approx r_0$,

$$\omega_{\mathcal{R}}(r) \approx kr_0 r - \frac{1}{2}kr^2 \tag{22}$$

For $r \ll r_0$,

$$\omega_{\mathcal{R}}(r) \approx (1 - ka)r + \frac{1}{2}kr^2. \tag{23}$$

Chord Method

For rate of convergence,

$$\omega_{\mathcal{R}}(r) = \frac{1}{2}kr^2 + r\left(1 - \sqrt{1 - 2k(r_0 - r)}\right) \tag{20}$$

The later constant is greater than the former,

$$1 - \sqrt{1 - 2kr_0} > kr_0 \tag{24}$$
$$1 - kr_0 > \sqrt{1 - 2kr_0} \tag{25}$$
$$1 - 2kr_0 + k^2r_0^2 > 1 - 2kr_0 \tag{26}$$
$$k^2r_0^2 > 0 \tag{27}$$

Chord Method

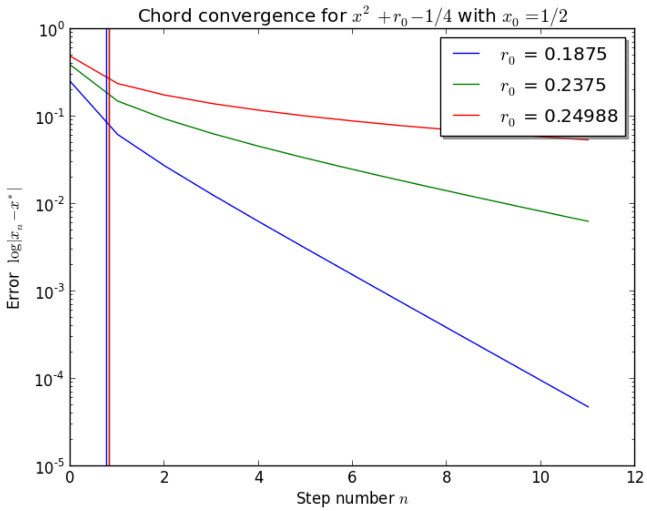
For rate of convergence,

$$\omega_{\mathcal{R}}(r) = \frac{1}{2}kr^2 + r\left(1 - \sqrt{1 - 2k(r_0 - r)}\right) \tag{20}$$

Convergence decelerates nearer to the solution.

Chord Method

Vertical lines at onset of asymptotic convergence ($\|x_{n+1} - x_n\| \approx a/e$)



Outline

Left vs. Right

Left:

$$\mathcal{F}(x) \implies x - \mathcal{N}(\mathcal{F}, x, b)$$

Right:

$$x \implies y = \mathcal{N}(\mathcal{F}, x, b)$$

Heisenberg vs. Schrödinger Picture

Left vs. Right

Left:

$$\mathcal{F}(x) \implies x - \mathcal{N}(\mathcal{F}, x, b)$$

Right:

$$x \implies y = \mathcal{N}(\mathcal{F}, x, b)$$

Heisenberg vs. Schrödinger Picture

$\mathcal{M} -_R \mathcal{N}$

We start with $x \in Z(r)$, apply $\mathcal{N}$ so that

$$y \in Z(\omega_{\mathcal{N}}(r)),$$

and then apply $\mathcal{M}$ so that

$$x' \in Z(\omega_{\mathcal{M}}(\omega_{\mathcal{N}}(r))).$$

Thus we have

$$\omega_{\mathcal{M} -_R \mathcal{N}} = \omega_{\mathcal{M}} \circ \omega_{\mathcal{N}}$$

Non-Abelian

$\mathcal{N} -_R$ NRICH

$$
\begin{aligned}
\omega_{\mathcal{N}} \circ \omega_{\text{NRICH}} &= \frac{1}{2}\frac{r^2}{\sqrt{r^2+a^2}} \circ cr, \\
&= \frac{1}{2}\frac{c^2 r^2}{\sqrt{c^2 r^2+a^2}}, \\
&= \frac{1}{2}\frac{cr^2}{\sqrt{r^2+(a/c)^2}}, \\
&= \frac{1}{2}c\frac{r^2}{\sqrt{r^2+\tilde{a}^2}},
\end{aligned}
$$

Non-Abelian

$$\mathcal{N} -_R \text{NRICH: } \tfrac{1}{2} c \frac{r^2}{\sqrt{r^2+\tilde{a}^2}}$$

NRICH $-_R \mathcal{N}$

$$\begin{aligned}
\omega_{\text{NRICH}} \circ \omega_{\mathcal{N}} &= cr \circ \frac{1}{2} \frac{r^2}{\sqrt{r^2+a^2}}, \\
&= \frac{1}{2} c \frac{r^2}{\sqrt{r^2+a^2}}, \\
&= \frac{1}{2} c \frac{r^2}{\sqrt{r^2+a^2}}.
\end{aligned}$$

Non-Abelian

$$\mathcal{N} -_R \text{NRICH: } \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + \tilde{a}^2}}$$

$$\text{NRICH} -_R \mathcal{N}\text{: } \frac{1}{2} c \frac{r^2}{\sqrt{r^2 + a^2}}$$

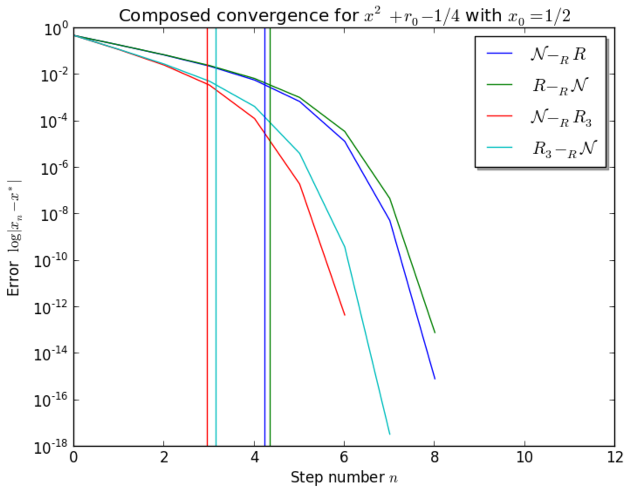
The first method also changes the onset of second order convergence.

Example

$$f(x) = x^2 + (0.0894427)^2$$

n	$\|x_{n+1} - x_n\|$	$\|x_{n+1} - x_n\| - w^{(n)}(r_0)$	$\|x_n - x^*\| - s(w^{(n)}(r_0))$
0	1.9990e+00	$< 10^{-16}$	$< 10^{-16}$
1	9.9850e-01	$< 10^{-16}$	$< 10^{-16}$
2	4.9726e-01	$< 10^{-16}$	$< 10^{-16}$
3	2.4470e-01	$< 10^{-16}$	$< 10^{-16}$
4	1.1492e-01	$< 10^{-16}$	$< 10^{-16}$
5	4.5342e-02	$< 10^{-16}$	$< 10^{-16}$
6	1.0251e-02	$< 10^{-16}$	$< 10^{-16}$
7	5.8360e-04	$< 10^{-16}$	$< 10^{-16}$
8	1.9039e-06	$< 10^{-16}$	$< 10^{-16}$
9	2.0264e-11	$< 10^{-16}$	$< 10^{-16}$
10	0.0000e+00	$< 10^{-16}$	$< 10^{-16}$

Example



Matrix iterations also 1D scalar once you diagonalize

Pták's nondiscrete induction and its application to matrix iterations, Liesen, IMA J. Num. Anal., 2014.

Outline

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

First we show that

$$\omega(s) \leq \frac{s}{r}\omega(r),$$

which means that convex rates of convergence are non-decreasing.

This implies that compositions of convex rates of convergence are also convex and non-decreasing.

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

Then we show that

$$\omega(r) < r \qquad \forall r \in (0, R)$$

by contradiction.

Composed Rates of Convergence

Theorem

If ω_1 and ω_2 are convex rates of convergence, then $\omega = \omega_1 \circ \omega_2$ is a rate of convergence.

This is enough to show that

$$\omega_1(\omega_2(r)) < \omega_1(r),$$

and in fact

$$(\omega_1 \circ \omega_2)^{(n)}(r) < \omega_1^{(n)}(r).$$

Multidimensional Induction Theorem

Preconditions

Theorem

Let

- p *(1 for our case) and* m *(2 for our case) be two positive integers,*
- X *be a complete metric space and* $D \subset X^p$*,*
- $G : D \rightarrow X^p$ *and* $F : D \rightarrow X^{p+1}$ *be defined by* $Fu = (u, Gu)$*,*
- $F_k = P_k F$*,* $-p + 1 \leq k \leq m$*, the components of* F*,*
- $P = P_m$*,*
- $Z(r) \subset D$ *for each* $r \in T^p$*,*
- ω *be a rate of convergence of type* (p, m) *on* T*,*
- $u_0 \in D$ *and* $r_0 \in T^p$*.*

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$u_0 \in Z(r_0),$$
$$PFZ(r) \subset Z(\tilde\omega(r)),$$
$$\|F_k u - F_{k+1} u\| \leq \omega_k(r),$$

for all $r \in T^p$, $u \in Z(r)$, and $k = 0, \ldots, m-1$, then

1. *u_0 is admissible, and $\exists x^* \in X$ such that $(P_k u_n)_{n\geq 0} \to x^*$,*
2. *and the following relations hold for $n > 1$,*

$$Pu_n \in Z(\tilde\omega(r_0)),$$
$$\|P_k u_n - P_{k+1} u_n\| \leq \omega_k^{(n)}(r_0), \qquad 0 \leq k \leq m-1,$$
$$\|P_k u_n - x^*\| \leq \sigma_k(\tilde\omega(r_0)), \qquad 0 \leq k \leq m;$$

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$u_0 \in Z(r_0),$$
$$PFZ(r) \subset Z(\tilde{\omega}(r)),$$
$$\|F_k u - F_{k+1} u\| \leq \omega_k(r),$$

for all $r \in T^p$, $u \in Z(r)$, and $k = 0, \ldots, m-1$, then

1. *u_0 is admissible, and $\exists x^* \in X$ such that $(P_k u_n)_{n \geq 0} \to x^*$,*
2. *and the following relations hold for $n > 1$,*

$$\|P_k u_n - x^*\| \leq \sigma_k(r_n), \qquad 0 \leq k \leq m.$$

where $r_n \in T^p$ and $Pu_{n-1} \in Z(r_n)$.

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$
\begin{aligned}
u_0 &\in Z(r_0), \\
PFZ(r) &\subset Z(\tilde{\omega}(r)), \\
\|F_k u - F_{k+1} u\| &\le \omega_k(r),
\end{aligned}
$$

for all $r \in T^p$*,* $u \in Z(r)$*, and* $k = 0, \ldots, m-1$*, then*

1. u_0 *is admissible, and* $\exists x^* \in X$ *such that* $(P_k u_n)_{n \ge 0} \to x^*$*,*
2. *and the following relations hold for* $n > 1$*,*

$$
\begin{aligned}
P u_n &\in Z(\tilde{\omega}(r_0)), \\
\|P_k u_n - P_{k+1} u_n\| &\le \omega_k^{(n)}(r_0), \qquad 0 \le k \le m-1, \\
\|P_k u_n - x^*\| &\le \sigma_k(\tilde{\omega}(r_0)), \qquad 0 \le k \le m;
\end{aligned}
$$

Multidimensional Induction Theorem

Theorem

If the following conditions hold

$$u_0 \in Z(r_0),$$
$$PFZ(r) \subset Z(\omega \circ \psi(r)),$$
$$\|F_0 u - F_1 u\| \leq r,$$
$$\|F_1 u - F_2 u\| \leq \psi(r),$$

for all $r \in T^p$, $u \in Z(r)$, and $k = 0, \ldots, m-1$, then

1. *u_0 is admissible, and $\exists x^* \in X$ such that $(P_k u_n)_{n \geq 0} \to x^*$,*
2. *and the following relations hold for $n > 1$,*

$$Pu_n \in Z(\tilde\omega(r_0)),$$
$$\|P_k u_n - P_{k+1} u_n\| \leq \omega_k^{(n)}(r_0), \qquad 0 \leq k \leq m-1,$$
$$\|P_k u_n - x^*\| \leq \sigma_k(\tilde\omega(r_0)), \qquad 0 \leq k \leq m;$$

Composed Newton Methods

Theorem

Suppose that we have two nonlinear solvers

- $\mathcal{M}$, Z_1, ω,
- $\mathcal{N}$, Z_0, ψ,

and consider $\mathcal{M} -_R \mathcal{N}$, meaning a single step of $\mathcal{N}$ for each step of $\mathcal{M}$.

Concretely, take $\mathcal{M}$ to be the Newton iteration, and $\mathcal{N}$ the Chord method. Then the assumptions of the theorem above are satisfied using $Z = Z_1$ and

$$\omega(r) = \{\psi(r), \omega \circ \psi(r)\},$$

giving us the existence of a solution, and both a priori and a posteriori bounds on the error.

Outline

Driven Cavity Problem

SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_monitor_short -snes_converged_reason -snes_view
```

$$
\begin{aligned}
-\Delta U - \partial_y \Omega &= 0 \\
-\Delta V + \partial_x \Omega &= 0 \\
-\Delta \Omega + \nabla \cdot ([U\Omega, V\Omega]) - \mathrm{Gr}\, \partial_x T &= 0 \\
-\Delta T + \mathrm{Pr}\, \nabla \cdot ([UT, VT]) &= 0
\end{aligned}
$$

Driven Cavity Problem

SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_monitor_short -snes_converged_reason -snes_view

lid velocity = 100, prandtl # = 1, grashof # = 100
  0 SNES Function norm 768.116
  1 SNES Function norm 658.288
  2 SNES Function norm 529.404
  3 SNES Function norm 377.51
  4 SNES Function norm 304.723
  5 SNES Function norm 2.59998
  6 SNES Function norm 0.00942733
  7 SNES Function norm 5.20667e-08
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 7
```

Driven Cavity Problem

SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e4
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_monitor_short -snes_converged_reason -snes_view
```

Driven Cavity Problem

SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e4
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_monitor_short -snes_converged_reason -snes_view

lid velocity = 100, prandtl # = 1, grashof # = 10000
  0 SNES Function norm 785.404
  1 SNES Function norm 663.055
  2 SNES Function norm 519.583
  3 SNES Function norm 360.87
  4 SNES Function norm 245.893
  5 SNES Function norm 1.8117
  6 SNES Function norm 0.00468828
  7 SNES Function norm 4.417e-08
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 7
```

Driven Cavity Problem

SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e5
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_monitor_short -snes_converged_reason -snes_view
```

Driven Cavity Problem

SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e5
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_monitor_short -snes_converged_reason -snes_view

lid velocity = 100, prandtl # = 1, grashof # = 100000
  0 SNES Function norm 1809.96
Nonlinear solve did not converge due to DIVERGED_LINEAR_SOLVE iterations 0
```

Driven Cavity Problem

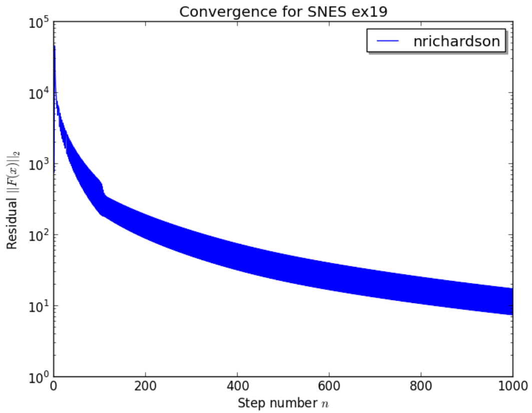
SNES ex19.c

```
./ex19 -lidvelocity 100 -grashof 1e5
  -da_grid_x 16 -da_grid_y 16 -da_refine 2 -pc_type lu
  -snes_monitor_short -snes_converged_reason -snes_view
```

```
lid velocity = 100, prandtl # = 1, grashof # = 100000
  0 SNES Function norm 1809.96
  1 SNES Function norm 1678.37
  2 SNES Function norm 1643.76
  3 SNES Function norm 1559.34
  4 SNES Function norm 1557.6
  5 SNES Function norm 1510.71
  6 SNES Function norm 1500.47
  7 SNES Function norm 1498.93
  8 SNES Function norm 1498.44
  9 SNES Function norm 1498.27
 10 SNES Function norm 1498.18
 11 SNES Function norm 1498.12
 12 SNES Function norm 1498.11
 13 SNES Function norm 1498.11
 14 SNES Function norm 1498.11
 ...
```

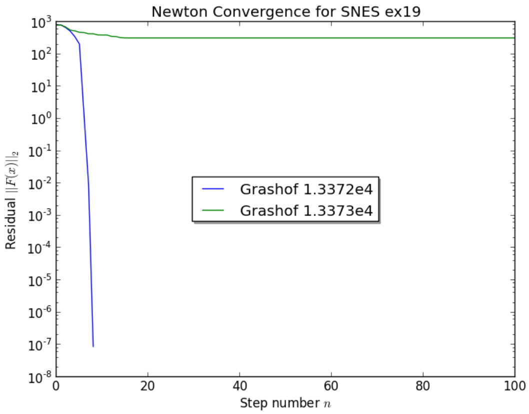
Deceleration of Convergence

```
./ex19 -lidvelocity 100 -grashof 1.3372e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type nrichardson -snes_linesearch_type cp -snes_max_it 10000
```



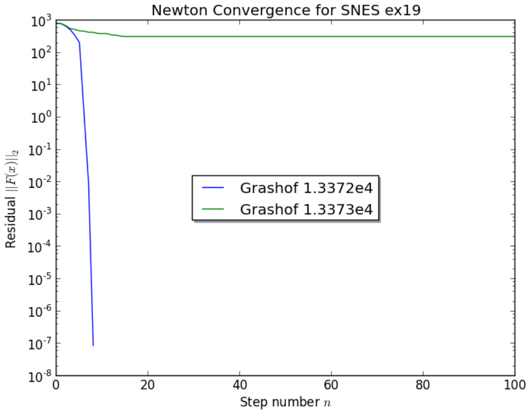
Stagnation of Newton

```
./ex19 -lidvelocity 100 -grashof 1.3372e4
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type newtonls -snes_max_it 100 -pc_type lu
```



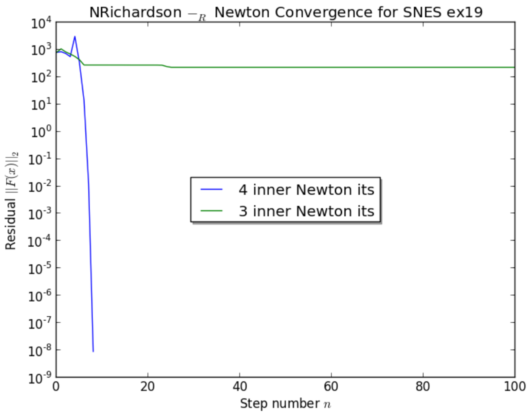
Stagnation of Newton

```
./ex19 -lidvelocity 100 -grashof 1.3373e4
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type newtonls -snes_max_it 100 -pc_type lu
```



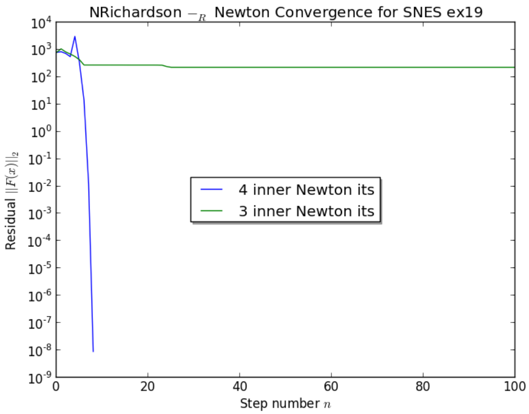
Preconditioning NRichardson with Newton

```
./ex19 -lidvelocity 100 -grashof 1.3373e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type nrichardson -snes_max_it 200
  -npc_snes_type newtonls -npc_snes_max_it 3 -npc_pc_type lu
```



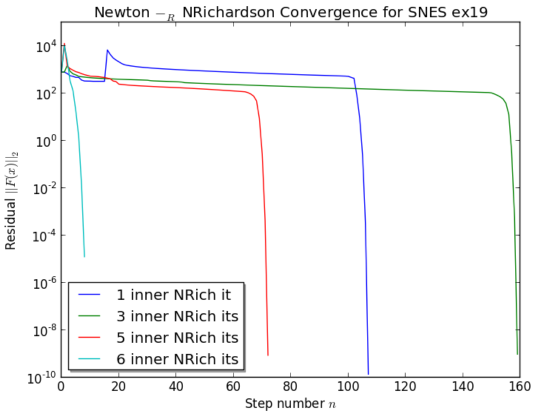
Preconditioning NRichardson with Newton

```
./ex19 -lidvelocity 100 -grashof 1.3373e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type nrichardson -snes_max_it 200
  -npc_snes_type newtonls -npc_snes_max_it 4 -npc_pc_type lu
```



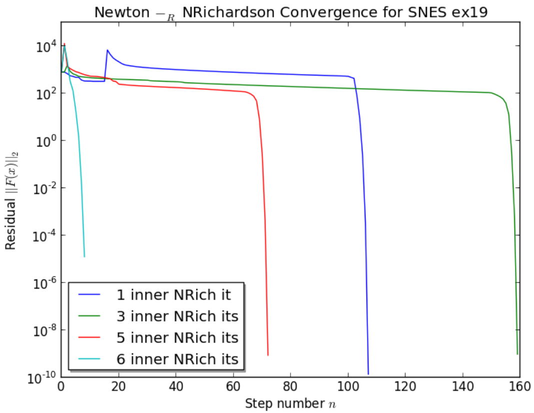
Preconditioning Newton with NRichardson

```
./ex19 -lidvelocity 100 -grashof 1.3373e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type newtonls -snes_max_it 1000 -pc_type lu
  -npc_snes_type nrichardson -npc_snes_max_it 1
```



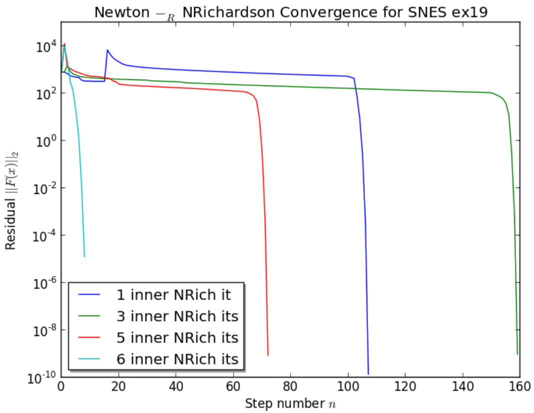
Preconditioning Newton with NRichardson

```
./ex19 -lidvelocity 100 -grashof 1.3373e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type newtonls -snes_max_it 1000 -pc_type lu
  -npc_snes_type nrichardson -npc_snes_max_it 3
```



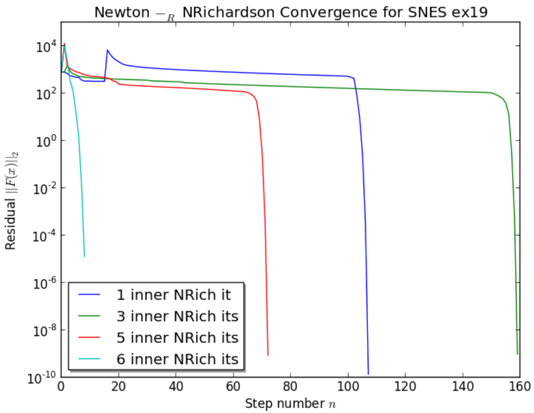
Preconditioning Newton with NRichardson

```
./ex19 -lidvelocity 100 -grashof 1.3373e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type newtonls -snes_max_it 1000 -pc_type lu
  -npc_snes_type nrichardson -npc_snes_max_it 5
```



Preconditioning Newton with NRichardson

```
./ex19 -lidvelocity 100 -grashof 1.3373e2
  -da_grid_x 16 -da_grid_y 16 -da_refine 2
  -snes_type newtonls -snes_max_it 1000 -pc_type lu
  -npc_snes_type nrichardson -npc_snes_max_it 6
```



Outline

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type newtonls -snes_converged_reason
 -pc_type lu
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
  1 SNES Function norm 1132.29
  2 SNES Function norm 1026.17
  3 SNES Function norm 925.717
  4 SNES Function norm 924.778
  5 SNES Function norm 836.867
  ⋮
 21 SNES Function norm 585.143
 22 SNES Function norm 585.142
 23 SNES Function norm 585.142
 24 SNES Function norm 585.142
  ⋮
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type fas -snes_converged_reason
 -fas_levels_snes_type gs -fas_levels_snes_max_it 6
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
  1 SNES Function norm 574.793
  2 SNES Function norm 513.02
  3 SNES Function norm 216.721
  4 SNES Function norm 85.949
Nonlinear solve did not converge due to DIVERGED_INNER iterations 4
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type fas -snes_converged_reason
 -fas_levels_snes_type gs -fas_levels_snes_max_it 6
  -fas_coarse_snes_converged_reason
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 12
  1 SNES Function norm 574.793
    Nonlinear solve did not converge due to DIVERGED_MAX_IT its 50
  2 SNES Function norm 513.02
    Nonlinear solve did not converge due to DIVERGED_MAX_IT its 50
  3 SNES Function norm 216.721
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 22
  4 SNES Function norm 85.949
    Nonlinear solve did not converge due to DIVERGED_LINE_SEARCH its 42
Nonlinear solve did not converge due to DIVERGED_INNER iterations 4
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type fas -snes_converged_reason
 -fas_levels_snes_type gs -fas_levels_snes_max_it 6
  -fas_coarse_snes_linesearch_type basic
  -fas_coarse_snes_converged_reason
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 6
  ⋮
 47 SNES Function norm 78.8401
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 5
 48 SNES Function norm 73.1185
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 6
 49 SNES Function norm 78.834
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 5
 50 SNES Function norm 73.1176
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 6
  ⋮
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type nrichardson -npc_snes_max_it 1 -snes_converged_reason
 -npc_snes_type fas -npc_fas_coarse_snes_converged_reason
  -npc_fas_levels_snes_type gs -npc_fas_levels_snes_max_it 6
  -npc_fas_coarse_snes_linesearch_type basic
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 6
  1 SNES Function norm 552.271
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 27
  2 SNES Function norm 173.45
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 45
  ⋮
 43 SNES Function norm 3.45407e-05
    Nonlinear solve converged due to CONVERGED_SNORM_RELATIVE its 2
 44 SNES Function norm 1.6141e-05
    Nonlinear solve converged due to CONVERGED_SNORM_RELATIVE its 2
 45 SNES Function norm 9.13386e-06
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 45
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type ngmres -npc_snes_max_it 1 -snes_converged_reason
 -npc_snes_type fas -npc_fas_coarse_snes_converged_reason
  -npc_fas_levels_snes_type gs -npc_fas_levels_snes_max_it 6
  -npc_fas_coarse_snes_linesearch_type basic
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 6
  1 SNES Function norm 538.605
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 13
  2 SNES Function norm 178.005
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 24
  ⋮
 27 SNES Function norm 0.000102487
    Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE its 2
 28 SNES Function norm 4.2744e-05
    Nonlinear solve converged due to CONVERGED_SNORM_RELATIVE its 2
 29 SNES Function norm 1.01621e-05
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 29
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type ngmres -npc_snes_max_it 1 -snes_converged_reason
 -npc_snes_type fas -npc_fas_coarse_snes_converged_reason
 -npc_fas_levels_snes_type newtonls -npc_fas_levels_snes_max_it 6
  -npc_fas_levels_snes_linesearch_type basic
  -npc_fas_levels_snes_max_linear_solve_fail 30
  -npc_fas_levels_ksp_max_it 20 -npc_fas_levels_snes_converged_reason
  -npc_fas_coarse_snes_linesearch_type basic
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
    Nonlinear solve did not converge due to DIVERGED_MAX_IT its 6
    ⋮
        Nonlinear solve converged due to CONVERGED_SNORM_RELATIVE its 1
    ⋮
  1 SNES Function norm 0.1935
  2 SNES Function norm 0.0179938
  3 SNES Function norm 0.00223698
  4 SNES Function norm 0.000190461
  5 SNES Function norm 1.6946e-06
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 5
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type composite -snes_composite_type additiveoptimal
 -snes_composite_sneses fas,newtonls -snes_converged_reason
 -sub_0_fas_levels_snes_type gs -sub_0_fas_levels_snes_max_it 6
   -sub_0_fas_coarse_snes_linesearch_type basic
 -sub_1_snes_linesearch_type basic -sub_1_pc_type mg
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
  1 SNES Function norm 541.462
  2 SNES Function norm 162.92
  3 SNES Function norm 48.8138
  4 SNES Function norm 11.1822
  5 SNES Function norm 0.181469
  6 SNES Function norm 0.00170909
  7 SNES Function norm 3.24991e-08
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 7
```

Nonlinear Preconditioning

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short
 -snes_type composite -snes_composite_type multiplicative
 -snes_composite_sneses fas,newtonls -snes_converged_reason
 -sub_0_fas_levels_snes_type gs -sub_0_fas_levels_snes_max_it 6
   -sub_0_fas_coarse_snes_linesearch_type basic
 -sub_1_snes_linesearch_type basic -sub_1_pc_type mg
```

```
lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
  1 SNES Function norm 544.404
  2 SNES Function norm 18.2513
  3 SNES Function norm 0.488689
  4 SNES Function norm 0.000108712
  5 SNES Function norm 5.68497e-08
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 5
```

Nonlinear Preconditioning

Solver	T	N. It	L. It	Func	Jac	PC	NPC
$(\mathcal{N}\backslash K - MG)$	9.83	17	352	34	85	370	–
NGMRES $-_R$ $(\mathcal{N}\backslash K - MG)$	7.48	10	220	21	50	231	10
FAS	6.23	162	0	2382	377	754	–
FAS $+ (\mathcal{N}\backslash K - MG)$	8.07	10	197	232	90	288	–
FAS $* (\mathcal{N}\backslash K - MG)$	4.01	5	80	103	45	125	–
NRICH $-_L$ FAS	3.20	50	0	1180	192	384	50
NGMRES $-_R$ FAS	1.91	24	0	447	83	166	24

Nonlinear Preconditioning

See discussion in:

Composing Scalable Nonlinear Algebraic Solvers,
Peter Brune, Matthew Knepley, Barry Smith, and Xuemin Tu,

SIAM Review, **57**(4), 535–565, 2015.

http://www.mcs.anl.gov/uploads/cels/papers/P2010-0112.pdf

Outline

Further Questions

- What is ω for NASM? Nonlinear FETI-DP?
- What are the Rules of Thumb for NPC?
- Can a composed iteration have a larger region of convergence?
- What API makes sense for simulations?

Programming with Options

ex55: Allen-Cahn problem in 2D

- constant mobility
- triangular elements

Geometric multigrid method for saddle point variational inequalities:

```
./ex55 -ksp_type fgmres -pc_type mg -mg_levels_ksp_type fgmres
-mg_levels_pc_type fieldsplit -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_pc_fieldsplit_type schur -da_grid_x 65 -da_grid_y 65
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition user
-mg_levels_fieldsplit_1_ksp_type gmres -mg_coarse_ksp_type preonly
-mg_levels_fieldsplit_1_pc_type none   -mg_coarse_pc_type svd
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor    -pc_mg_levels 5
-mg_levels_fieldsplit_0_pc_sor_forward -pc_mg_galerkin
-snes_vi_monitor -ksp_monitor_true_residual -snes_atol 1.e-11
-mg_levels_ksp_monitor -mg_levels_fieldsplit_ksp_monitor
-mg_levels_ksp_max_it 2 -mg_levels_fieldsplit_ksp_max_it 5
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Run flexible GMRES with 5 levels of multigrid as the preconditioner

```
./ex55 -ksp_type fgmres -pc_type mg -pc_mg_levels 5
  -da_grid_x 65 -da_grid_y 65
```

Use the Galerkin process to compute the coarse grid operators

```
-pc_mg_galerkin
```

Use SVD as the coarse grid saddle point solver

```
-mg_coarse_ksp_type preonly -mg_coarse_pc_type svd
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Programming with Options

ex55: Allen-Cahn problem in 2D

Smoother: Flexible GMRES (2 iterates) with a Schur complement PC

```
-mg_levels_ksp_type fgmres -mg_levels_pc_fieldsplit_detect_saddle_point
-mg_levels_ksp_max_it 2 -mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_type schur
-mg_levels_pc_fieldsplit_factorization_type full
-mg_levels_pc_fieldsplit_schur_precondition diag
```

Schur complement solver: GMRES (5 iterates) with no preconditioner

```
-mg_levels_fieldsplit_1_ksp_type gmres
-mg_levels_fieldsplit_1_pc_type none -mg_levels_fieldsplit_ksp_max_it 5
```

Schur complement action: Use only the lower diagonal part of A00

```
-mg_levels_fieldsplit_0_ksp_type preonly
-mg_levels_fieldsplit_0_pc_type sor
-mg_levels_fieldsplit_0_pc_sor_forward
```

Magma FAS Options

Top level

```
-snes_monitor -snes_converged_reason
-snes_type fas -snes_fas_type full -snes_fas_levels 4
 -fas_levels_3_snes_monitor -fas_levels_3_snes_converged_reason
  -fas_levels_3_snes_atol 1.0e-9 -fas_levels_3_snes_max_it 2
  -fas_levels_3_snes_type newtonls -fas_levels_3_snes_linesearch_type bt
  -fas_levels_3_snes_fd_color -fas_levels_3_snes_fd_color_use_mat
  -fas_levels_3_ksp_rtol 1.0e-10 -mat_coloring_type greedy
  -fas_levels_3_ksp_gmres_restart 50 -fas_levels_3_ksp_max_it 200
   -fas_levels_3_pc_type fieldsplit
    -fas_levels_3_pc_fieldsplit_0_fields 0,2
    -fas_levels_3_pc_fieldsplit_1_fields 1
    -fas_levels_3_pc_fieldsplit_type schur
     -fas_levels_3_pc_fieldsplit_schur_precondition selfp
     -fas_levels_3_pc_fieldsplit_schur_factorization_type full
      -fas_levels_3_fieldsplit_0_pc_type lu
      -fas_levels_3_fieldsplit_pressure_ksp_rtol 1.0e-9
       -fas_levels_3_fieldsplit_pressure_pc_type gamg
       -fas_levels_3_fieldsplit_pressure_ksp_gmres_restart 100
       -fas_levels_3_fieldsplit_pressure_ksp_max_it 200
```

Magma FAS Options

2nd level

```
-fas_levels_2_snes_monitor -fas_levels_2_snes_converged_reason
 -fas_levels_2_snes_atol 1.0e-9 -fas_levels_2_snes_max_it 2
 -fas_levels_2_snes_type newtonls -fas_levels_2_snes_linesearch_type bt
 -fas_levels_2_snes_fd_color -fas_levels_2_snes_fd_color_use_mat
 -fas_levels_2_ksp_rtol 1.0e-10 -fas_levels_2_ksp_gmres_restart 50
 -fas_levels_2_pc_type fieldsplit
  -fas_levels_2_pc_fieldsplit_0_fields 0,2
  -fas_levels_2_pc_fieldsplit_1_fields 1
  -fas_levels_2_pc_fieldsplit_type schur
   -fas_levels_2_pc_fieldsplit_schur_precondition selfp
   -fas_levels_2_pc_fieldsplit_schur_factorization_type full
    -fas_levels_2_fieldsplit_0_pc_type lu
    -fas_levels_2_fieldsplit_pressure_ksp_rtol 1.0e-9
     -fas_levels_2_fieldsplit_pressure_pc_type gamg
     -fas_levels_2_fieldsplit_pressure_ksp_gmres_restart 100
     -fas_levels_2_fieldsplit_pressure_ksp_max_it 200
```

Magma FAS Options

1st level

```
-fas_levels_1_snes_monitor -fas_levels_1_snes_converged_reason
 -fas_levels_1_snes_atol 1.0e-9
 -fas_levels_1_snes_type newtonls -fas_levels_1_snes_linesearch_type bt
 -fas_levels_1_snes_fd_color -fas_levels_1_snes_fd_color_use_mat
 -fas_levels_1_ksp_rtol 1.0e-10 -fas_levels_1_ksp_gmres_restart 50
 -fas_levels_1_pc_type fieldsplit
  -fas_levels_1_pc_fieldsplit_0_fields 0,2
  -fas_levels_1_pc_fieldsplit_1_fields 1
  -fas_levels_1_pc_fieldsplit_type schur
   -fas_levels_1_pc_fieldsplit_schur_precondition selfp
   -fas_levels_1_pc_fieldsplit_schur_factorization_type full
    -fas_levels_1_fieldsplit_0_pc_type lu
    -fas_levels_1_fieldsplit_pressure_ksp_rtol 1.0e-9
    -fas_levels_1_fieldsplit_pressure_pc_type gamg
```

Magma FAS Options

Coarse level

```
-fas_coarse_snes_monitor -fas_coarse_snes_converged_reason
 -fas_coarse_snes_atol 1.0e-9
 -fas_coarse_snes_type newtonls -fas_coarse_snes_linesearch_type bt
 -fas_coarse_snes_fd_color -fas_coarse_snes_fd_color_use_mat
 -fas_coarse_ksp_rtol 1.0e-10 -fas_coarse_ksp_gmres_restart 50
 -fas_coarse_pc_type fieldsplit
  -fas_coarse_pc_fieldsplit_0_fields 0,2
  -fas_coarse_pc_fieldsplit_1_fields 1
  -fas_coarse_pc_fieldsplit_type schur
   -fas_coarse_pc_fieldsplit_schur_precondition selfp
   -fas_coarse_pc_fieldsplit_schur_factorization_type full
    -fas_coarse_fieldsplit_0_pc_type lu
    -fas_coarse_fieldsplit_pressure_ksp_rtol 1.0e-9
    -fas_coarse_fieldsplit_pressure_pc_type gamg
```

Additive Composition

We can represent the additive update rule

$$\vec{x}_{i+1} = \vec{x}_i + \alpha(\mathcal{M}(\mathcal{F}, \vec{x}_i,) - \vec{x}_i) + \beta(\mathcal{N}(\mathcal{F}, \vec{x}_i,) - \vec{x}_i)$$

as

$$\vec{x}_{i+1} = (\mathcal{M} + \mathcal{N})(\mathcal{F}, \vec{x}_i,)$$

Additive Composition

We can represent the additive update rule

$$\vec{x}_{i+1} = \vec{x}_i + \alpha(\mathcal{M}(\mathcal{F}, \vec{x}_i,) - \vec{x}_i) + \beta(\mathcal{N}(\mathcal{F}, \vec{x}_i,) - \vec{x}_i)$$

as

$$\vec{x}_{i+1} = (\mathcal{M} + \mathcal{N})(\mathcal{F}, \vec{x}_i,)$$

Additive Composition

If $\alpha = \beta = 1$, this has an identity operation 0 (the identity map),

$$\begin{aligned}\vec{x}_{i+1} &= \vec{x}_i + \alpha(\mathcal{M}(\mathcal{F}, \vec{x}_i,) - \vec{x}_i) + \beta(0(\mathcal{F}, \vec{x}_i,) - \vec{x}_i) \\ &= \vec{x}_i + (\mathcal{M}(\mathcal{F}, \vec{x}_i,) - \vec{x}_i) + \left(\vec{x}_i - \vec{x}_i\right) \\ &= \mathcal{M}(\mathcal{F}, \vec{x}_i,)\end{aligned}$$

so that $(\mathcal{M}, +)$ is an abelian group.

Multiplicative Composition

We can represent the multiplicative update rule

$$\vec{x}_{i+1} = \mathcal{M}(\mathcal{F}, \mathcal{N}(\mathcal{F}, \vec{x}_i,),)$$

as

$$\vec{x}_{i+1} = (\mathcal{M} * \mathcal{N})(\mathcal{F}, \vec{x}_i,)$$

which is clearly associative.

Multiplicative Composition

We can represent the multiplicative update rule

$$\vec{x}_{i+1} = \mathcal{M}(\mathcal{F}, \mathcal{N}(\mathcal{F}, \vec{x}_i,),)$$

as

$$\vec{x}_{i+1} = (\mathcal{M} * \mathcal{N})(\mathcal{F}, \vec{x}_i,)$$

which is clearly associative.

Algebraic Structure

If we look at the distributive case,

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) * \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

we get the update rule

$$\begin{aligned}\vec{x}_{i+1} = \vec{x}_i &+ \alpha(\mathcal{M}(\mathcal{F}, \mathcal{Q}(\mathcal{F}, \vec{x}_i,),) - \vec{x}_i) \\ &+ \beta(\mathcal{N}(\mathcal{F}, \mathcal{Q}(\mathcal{F}, \vec{x}_i,),) - \vec{x}_i)\end{aligned}$$

Algebraic Structure

If we look at the distributive case,

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) * \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

which we can write as

$$\vec{x}_{i+1} = (\mathcal{M} * \mathcal{Q} + \mathcal{N} * \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

Algebraic Structure

If we look at the distributive case,

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) * \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

which we can write as

$$\vec{x}_{i+1} = (\mathcal{M} * \mathcal{Q} + \mathcal{N} * \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

Note however that

$$\mathcal{Q} * (\mathcal{M} + \mathcal{N}) \neq \mathcal{Q} * \mathcal{M} + \mathcal{Q} * \mathcal{N}$$

which means $(\mathcal{M}, +, *)$ is a near ring.

Algebraic Structure

If we combine it using our left NPC operation

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) -_L \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

we get the update rule

$$\begin{aligned}\vec{x}_{i+1} = \vec{x}_i &+ \alpha(\mathcal{M}(\vec{x} - \mathcal{Q}(\mathcal{F}, \vec{x}_i,), \vec{x}_i,) - \vec{x}_i) \\ &+ \beta(\mathcal{N}(\vec{x} - \mathcal{Q}(\mathcal{F}, \vec{x}_i,), \vec{x}_i,) - \vec{x}_i)\end{aligned}$$

Algebraic Structure

If we combine it using our left NPC operation

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) -_L \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

which we can write as

$$\vec{x}_{i+1} = (\mathcal{M} -_L \mathcal{Q} + \mathcal{N} -_L \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

we we again have a near ring.

Algebraic Structure

In the same way, we can combine it with our right NPC operation

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) -_R \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

and get the update rule

$$\begin{aligned}\vec{x}_{i+1} = \vec{x}_i &+ \alpha(\mathcal{M}(\mathcal{F}(\mathcal{Q}(\mathcal{F}, \vec{x}_i,)), \vec{x}_i,) - \vec{x}_i) \\ &+ \beta(\mathcal{N}(\mathcal{F}(\mathcal{Q}(\mathcal{F}, \vec{x}_i,)), \vec{x}_i,) - \vec{x}_i)\end{aligned}$$

Algebraic Structure

In the same way, we can combine it with our right NPC operation

$$\vec{x}_{i+1} = ((\mathcal{M} + \mathcal{N}) -_R \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

which we can write as

$$\vec{x}_{i+1} = (\mathcal{M} -_R \mathcal{Q} + \mathcal{N} -_R \mathcal{Q})(\mathcal{F}, \vec{x}_i,)$$

we we again have a near ring.

Polynomial solution through decomposition

Let us solve

$$x^8 - 4x^6 - 11x^4 + 30x^2 + 56 = 0$$

which can be decomposed

$$\left(x^4 - 4x^3 - 11x^2 + 30x + 56\right) \circ x^2 = 0$$
$$\left(x^2 - 15x + 56\right) \circ \left(x^2 - 2x\right) \circ x^2 = 0.$$

Polynomial solution through decomposition

Let us solve

$$x^8 - 4x^6 - 11x^4 + 30x^2 + 56 = 0$$

which can be decomposed

$$\left(x^4 - 4x^3 - 11x^2 + 30x + 56\right) \circ x^2 = 0$$
$$\left(x^2 - 15x + 56\right) \circ \left(x^2 - 2x\right) \circ x^2 = 0.$$

Polynomial solution through decomposition

Let us solve

$$x^8 - 4x^6 - 11x^4 + 30x^2 + 56 = 0$$

which can be decomposed

$$\left(x^4 - 4x^3 - 11x^2 + 30x + 56\right) \circ x^2 = 0$$
$$\left(x^2 - 15x + 56\right) \circ \left(x^2 - 2x\right) \circ x^2 = 0.$$

We solve the first equation, to get

$$x_{00} = 7 \qquad x_{01} = 8,$$

Polynomial solution through decomposition

Let us solve

$$x^8 - 4x^6 - 11x^4 + 30x^2 + 56 = 0$$

which can be decomposed

$$\left(x^4 - 4x^3 - 11x^2 + 30x + 56\right) \circ x^2 = 0$$
$$\left(x^2 - 15x + 56\right) \circ \left(x^2 - 2x\right) \circ x^2 = 0.$$

and then solve

$$x^2 - 2x = 7 \text{ or } 8,$$

to get

$$x_{10} = 1 + 2\sqrt{2} \qquad x_{11} = 1 - 2\sqrt{2} \qquad x_{12} = 4 \qquad x_{13} = -2.$$

Polynomial solution through decomposition

Let us solve

$$x^8 - 4x^6 - 11x^4 + 30x^2 + 56 = 0$$

which can be decomposed

$$\begin{aligned}\left(x^4 - 4x^3 - 11x^2 + 30x + 56\right) \circ x^2 &= 0\\ \left(x^2 - 15x + 56\right) \circ \left(x^2 - 2x\right) \circ x^2 &= 0.\end{aligned}$$

At the end, we have $x^2 = x_{1j}$, so that

$$\begin{aligned}x_{0,1,2,3} &= \pm\sqrt{1 \pm 2\sqrt{2}}\\ x_{5,6} &= \pm 2\\ x_{7,8} &= \pm i\sqrt{2}.\end{aligned}$$

There is an $\mathcal{O}(d \ln d)$ algorithm for finding the unique decomposition.