Nonlinear Preconditioning in PETSc

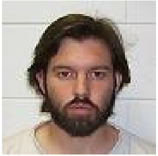
Matthew Knepley $\in$ PETSc Team

Computation Institute
University of Chicago

Department of Molecular Biology and Physiology
Rush University Medical Center

Seminar
Dept. of Mechanical and Aerospace Engineering, SUNY Buffalo
Buffalo, NY April 23, 2014

The PETSc Team


Matt Knepley


Barry Smith


Satish Balay


Jed Brown

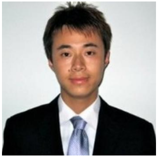

Hong Zhang

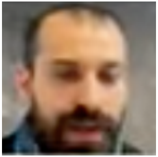

Lisandro Dalcin


Stefano Zampini


Mark Adams


Toby Issac


Hong Zhang


Pierre Jolivet


Junchao Zhang

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

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

Nonlinear Additions

Unlike the linear case, we must define

- the solution $\vec{x}$

  AND

- the residual $\vec{r}$

in both the inner and outer solvers.

Additive Combination

The linear iteration

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

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

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

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)

Nonlinear Richardson

procedure NRICH($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)

$\mathbf{d} := -\mathbf{r}(\mathbf{x}_i)$

$\quad \mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$ ▷ λ determined by line search

end procedure

return $\mathbf{x}_{i+1}$

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

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

Nonlinear Richardson

procedure NRICH($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)

$\mathbf{d} := -\mathbf{r}(\mathbf{x}_i)$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$ ▷ λ determined by line search

end procedure

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 $-_L \mathcal{N}$:

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

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.

PETSc Line Search

BT Standard cubic back-tracking

- Defaults to full step when Wolfe conditions satisfied
- No more work than necessary
- May stagnate
- Can be badly scaled apart from $\mathcal{N}$

L2 Secant minimization of residual

- Optimal damping in the residual direction
- Minimize $||\vec{r}(\vec{x} + \lambda\delta\vec{x})||_2$

CP Secant minimization of energy

- Appropriate when $\mathcal{F}$ is the gradient of an energy function
- Looks for roots of $\delta\vec{x}^T\mathcal{F}(\vec{x} + \lambda\delta\vec{x})$

Nonlinear GMRES

procedure NGMRES($\mathcal{F}, \mathbf{x}_i \cdots \mathbf{x}_{i-m+1}, \mathbf{b}$)
    $\mathbf{d}_i = -\mathbf{r}(\mathbf{x}_i)$
    $\mathbf{x}_i^M = \mathbf{x}_i + \lambda \mathbf{d}_i$
    $\mathcal{F}_i^M = \mathbf{r}(\mathbf{x}_i^M)$
    minimize $\|\mathbf{r}((1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}_i^M + \sum_{k=i-m}^{i-1} \alpha_k \mathbf{x}_k)\|_2$ over $\{\alpha_{i-m} \cdots \alpha_{i-1}\}$
    $\mathbf{x}_i^A = (1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}^M + \sum_{k=i-m}^{i-1} \alpha_k \mathbf{x}_k$
    $\mathbf{x}_{i+1} = \mathbf{x}_i^A$ or $\mathbf{x}_i^M$ if $\mathbf{x}_i^A$ is insufficient.
end procedure
return $\mathbf{x}_{i+1}$

Can emulate Anderson mixing and DIIS

Nonlinear GMRES

procedure NGMRES($\mathcal{F}, \mathbf{x}_i \cdots \mathbf{x}_{i-m+1}, \mathbf{b}$)

  $\mathbf{d}_i = -\mathbf{r}(\mathbf{x}_i)$

  $\mathbf{x}_i^M = \mathbf{x}_i + \lambda \mathbf{d}_i$

  $\mathcal{F}_i^M = \mathbf{r}(\mathbf{x}_i^M)$

  minimize $\|\mathbf{r}((1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}_i^M + \sum_{k=i-m}^{i-1} \alpha_k \mathbf{x}_k)\|_2$ over $\{\alpha_{i-m} \cdots \alpha_{i-1}\}$

  $\mathbf{x}_i^A = (1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}^M + \sum_{k=i-m}^{i-1} \alpha_k \mathbf{x}_k$

  $\mathbf{x}_{i+1} = \mathbf{x}_i^A$ or $\mathbf{x}_i^M$ if $\mathbf{x}_i^A$ is insufficient.

end procedure

return $\mathbf{x}_{i+1}$

Can emulate Anderson mixing and DIIS

Newton-Krylov

procedure $\mathcal{N}\backslash\mathrm{K}(\mathbf{F}, \mathbf{x}_i, \mathbf{b})$

$\mathbf{d} = J(\mathbf{x}_i)^{-1}\mathbf{r}(\mathbf{x}_i, \mathbf{b})$ ▷ solve by Krylov method

$\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda\mathbf{d}$ ▷ λ determined by line search

end procedure

return $\mathbf{x}_{i+1}$

Left Preconditioned Newton-Krylov

procedure $\mathcal{N}\backslash\mathrm{K}(\mathbf{x} - \mathcal{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}_i, 0)$

$$\mathbf{d} = \frac{\partial(\mathbf{x}_i - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))}{\partial \mathbf{x}_i}^{-1} (\mathbf{x}_i - \mathcal{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}))$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$$

end procedure

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

procedure NK($\mathbf{F}(\mathcal{M}(\mathbf{F}, \cdot, \mathbf{b})), \mathbf{y}_i, \mathbf{b}$)

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

$\mathbf{d} = J(\mathbf{x})^{-1}\mathbf{r}(\mathbf{x}_i)$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda\mathbf{d}$ $\triangleright$ λ determined by line search

end procedure

return $\mathbf{x}_{i+1}$

Jacobian Computation

First-Order Approximation

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

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

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

$\mathcal{N}\backslash\mathrm{K} -_R \vec{M}$ 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

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

- 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)

Full Approximation Scheme (FAS)

Nonlinear Multigrid

procedure FAS($\vec{F}, \mathbf{x}_i, \mathbf{b}$)

$\mathbf{x}_s = \mathcal{M}_s(\mathcal{F}, \mathbf{x}_i, \mathbf{b})$

$\mathbf{x}_c = \widehat{\mathbf{R}}\mathbf{x}_s$

$\mathbf{b}_c = \mathcal{F}_c(\mathbf{x}_c) - \mathbf{R}[\mathcal{F}(\mathbf{x}_s) - \mathbf{b}]$

$\mathbf{x}_s = \mathbf{x}_s + \mathbf{P}[\mathrm{FAS}(\vec{F}_c, \mathbf{x}_c, \mathbf{b}_c) - \mathbf{x}_c]$

$\mathbf{x}_{i+1} = \mathcal{M}_s(\mathcal{F}, \mathbf{x}_s, \mathbf{b})$

end procedure

return $\mathbf{x}_{i+1}$

Other Nonlinear Solvers

- **NASM** Nonlinear Additive Schwarz
- **NGS** Nonlinear Gauss-Siedel
- **NCG** Nonlinear Conjugate Gradients
- **QN** Quasi-Newton methods

Outline

Outline

SNES ex16

3D Large Deformation Elasticity

$$\int_\Omega F \cdot S : \nabla v \, d\Omega + \int_\Omega \texttt{loading} \, \mathbf{e}_y \cdot v \, d\Omega = 0 \tag{17}$$

F Deformation gradient

S Second Piola-Kirchhoff tensor

Saint Venant-Kirchhoff model of hyperelasticity

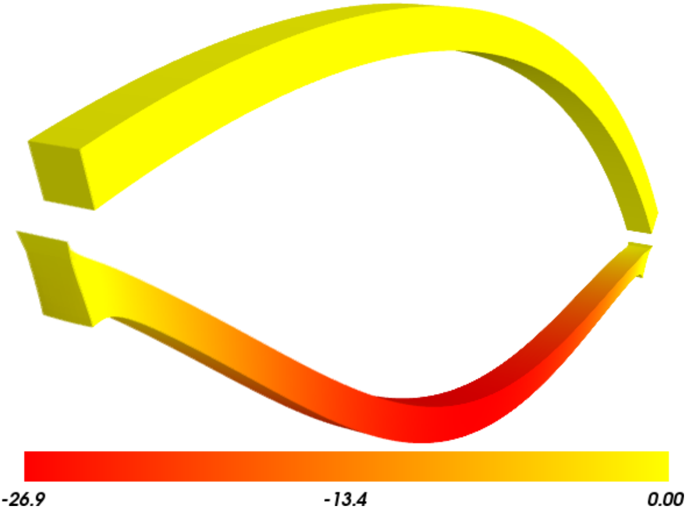
Ω `-arc` *angle* subsection of a cylindrical shell

`-height` *thickness*

`-rad` *inner radius*

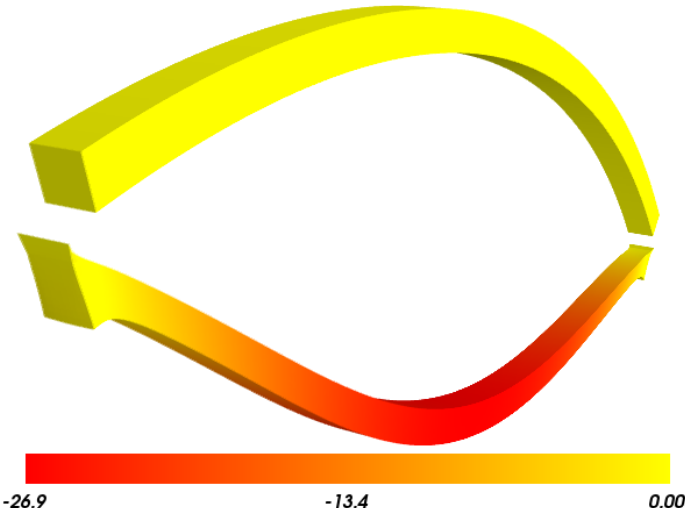
`-width` *width*

Large Deformation Elasticity



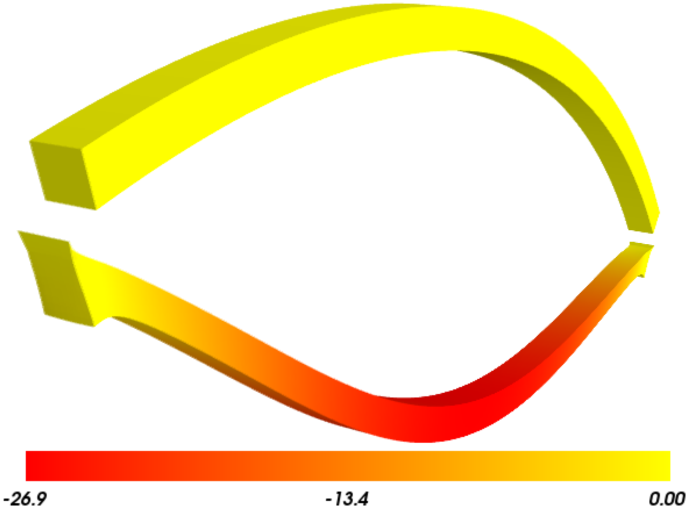
Unstressed and stressed configurations for the elasticity test problem.

Large Deformation Elasticity



Coloration indicates vertical displacement in meters.

Large Deformation Elasticity



P. Wriggers, Nonlinear Finite Element Methods, Springer, 2008.

Large Deformation Elasticity

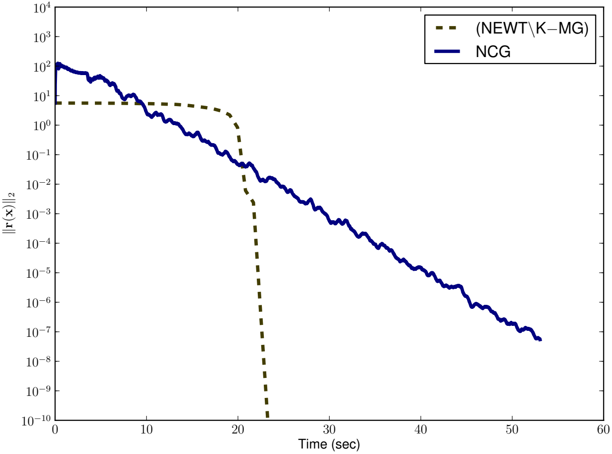
Running

SNES example 16:

```
cd src/snes/examples/tutorials
make ex16
./ex16 -da_grid_x 401 -da_grid_y 9 -da_grid_z 9
  -height 3 -width 3
  -rad 100 -young 100 -poisson 0.2
  -loading -1 -ploading 0
```

Plain SNES Convergence

$(\mathcal{N}\backslash K - MG)$ and NCG

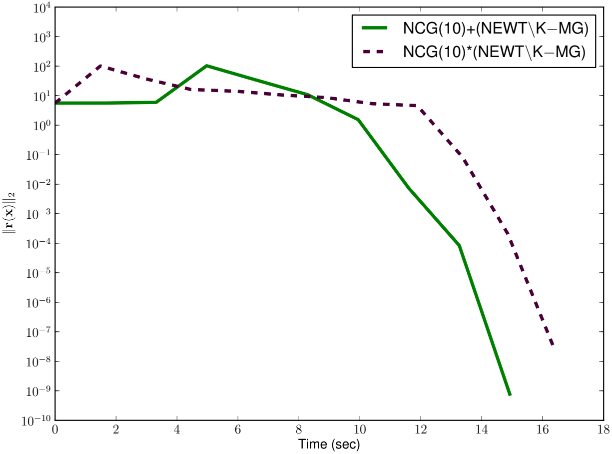


Plain SNES Convergence

Solver	T	N. It	L. It	Func	Jac	PC	NPC
NCG	53.05	4495	0	8991	–	–	–
$(\mathcal{N}\backslash \mathrm{K}-\mathrm{MG})$	23.43	27	1556	91	27	1618	–

Composed SNES Convergence

NCG(10) + ($\mathcal{N}\backslash$K − MG) and NCG(10) ∗ ($\mathcal{N}\backslash$K − MG)

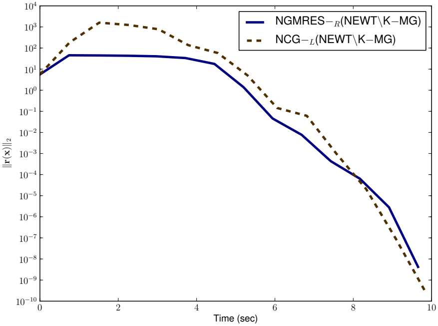


Composed SNES Convergence

Solver	T	N. It	L. It	Func	Jac	PC	NPC
NCG	53.05	4495	0	8991	–	–	–
$(\mathcal{N}\backslash\mathrm{K}-\mathrm{MG})$	23.43	27	1556	91	27	1618	–
NCG(10) $+(\mathcal{N}\backslash\mathrm{K}-\mathrm{MG})$	14.92	9	459	218	9	479	–
NCG(10) $*(\mathcal{N}\backslash\mathrm{K}-\mathrm{MG})$	16.34	11	458	251	11	477	–

Peconditioned SNES Convergence

NGMRES $-_R$ $(\mathcal{N}\backslash K - MG)$ and NCG $-_L$ $(\mathcal{N}\backslash K - MG)$



Peconditioned SNES Convergence

Solver	T	N. It	L. It	Func	Jac	PC	NPC
NCG	53.05	4495	0	8991	–	–	–
$(\mathcal{N}\backslash K - MG)$	23.43	27	1556	91	27	1618	–
NCG(10) $+(\mathcal{N}\backslash K - MG)$	14.92	9	459	218	9	479	–
NCG(10) $*(\mathcal{N}\backslash K - MG)$	16.34	11	458	251	11	477	–
NGMRES $-_R(\mathcal{N}\backslash K - MG)$	9.65	13	523	53	13	548	13
NCG $-_L(\mathcal{N}\backslash K - MG)$	9.84	13	529	53	13	554	13

Outline

2 Experiments
- Composition
- Multilevel

SNES ex19

Driven Cavity Flow

$$-\Delta \pitchfork \nabla \times \Omega = 0$$

$$-\Delta\Omega + \nabla \cdot (\dot{\Lambda}) - GR\nabla_x T = 0$$

$$-\Delta T + PR\nabla \cdot (\overline{\Lambda}) = 0$$

SNES ex19

Driven Cavity Flow

$$-\Delta \uparrow \nabla \times \Omega = 0$$

$$\Delta\Omega + \nabla \cdot (\Lambda) - GR \nabla_x T = 0$$

$$-\Delta T + PR \nabla \cdot (\uparrow) = 0$$

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
 ...
```

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