# COMPOSING SCALABLE NONLINEAR ALGEBRAIC SOLVERS

PETER BRUNE*, MATTHEW G. KNEPLEY†, BARRY SMITH*, AND XUEMIN TU‡

**Abstract.** Most efficient linear solvers use composable algorithmic components, with the most common model being the combination of a Krylov accelerator and one or more preconditioners. A similar set of concepts may be used for nonlinear algebraic systems, where nonlinear composition of different nonlinear solvers may significantly improve the time to solution. We describe the basic concepts of nonlinear composite combination and preconditioning and present a number of solvers applicable to nonlinear partial differential equations. We have developed a software framework in order to easily explore the possible combinations of solvers. We show that the performance gains from using composed solvers can be substantial compared with gains from standard Newton-Krylov methods.

**1. Introduction.** Large-scale algebraic solvers for nonlinear partial differential equations (PDEs) are an essential component of modern simulations. Newton-Krylov methods [20] have well-earned dominance. They are generally robust and may be built from preexisting linear solvers and preconditioners, including fast multilevel preconditioners such as multigrid [64, 58, 7, 5, 9] and domain decomposition methods [54, 53, 46, 56]. Newton's methods start from a global linearization of the nonlinear operator. The linearization leads to a large sparse linear system where the matrix may be represented either explicitly by storing the nonzero coefficients or implicitly by various "matrix-free" approaches [11, 36]. However, Newton's method has a number of drawbacks as a stand-alone solver. The repeated construction and solution of the linearization cause memory bandwidth and communication bottlenecks to come to the fore with regard to performance. There is also a possible lack of convergence robustness when the initial guess is far from the solution. Luckily, a large design space for nonlinear solvers exists to complement, improve, or replace Newton's method. Only a small part of this space has yet been explored either experimentally or theoretically.

In this paper we consider a wide collection of solvers for nonlinear equations. In direct equivalence to the case of linear solvers, we use a small number of algorithmic building blocks to produce a vast array of solvers with different convergence and performance properties. Two related solver composition techniques, nonlinear composite combination and preconditioning, will be used to construct these solvers. The contributions of this paper are twofold: introduction of a systematic approach to combining nonlinear solvers mathematically and in software, and demonstration of the construction of efficient solvers for several problems of interest. Implementations of the solvers in this paper are available in the PETSc library and may be brought to bear on relevant applications, whether simulated on a laptop or supercomputer.

**2. Background.** We concern ourselves with the solution of nonlinear equations of the form

$$\mathbf{F}(\mathbf{x}) = \mathbf{b} \tag{2.1}$$

for a general discretized nonlinear function $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ and right hand side (RHS) $\mathbf{b}$. We define the nonlinear residual as

$$\mathbf{r}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{b}. \tag{2.2}$$

The linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

with residual

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

*prbrune@mcs.anl.gov and bsmith@mcs.anl.gov, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Ave. Argonne, IL 60439

†knepley@ci.uchicago.edu, Computations Institute, University of Chicago, Searle Chemistry Lab., 5735 S. Ellis Ave. Chicago, IL 60637

‡xtu@math.ku.edu, Department of Mathematics, University of Kansas, 1460 Jayhawk Blvd. Lawrence, Kansas 66045

is an important special case from which we can derive valuable insight into solvers for the nonlinear problem.

Stationary solvers for linear systems repeatedly apply a linear operator in order to progressively improve the solution. The application of a linear stationary solver by defect correction may be written as

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{P}^{-1}(\mathbf{A}\mathbf{x}_i - \mathbf{b}), \tag{2.3}$$

where $\mathbf{P}^{-1}$ is a linear operator, called a preconditioner, whose action approximates, in some sense, the inverse of $\mathbf{A}$. The Jacobi, Gauss-Seidel, and multigrid iterations are all examples of linear stationary solvers.

Composite combination of linear preconditioners $\mathbf{P}^{-1}$ and $\mathbf{Q}^{-1}$ may proceed in two different ways, producing two new stationary solvers. The first is the additive combination

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\alpha_{\mathbf{P}}\mathbf{P}^{-1} + \alpha_{\mathbf{Q}}\mathbf{Q}^{-1})(\mathbf{A}\mathbf{x}_i - \mathbf{b}),$$

with weights $\alpha_{\mathbf{P}}$ and $\alpha_{\mathbf{Q}}$. The second is the multiplicative combination

$$\begin{aligned}\mathbf{x}_{i+1/2} &= \mathbf{x}_i - \mathbf{P}^{-1}(\mathbf{A}\mathbf{x}_i - \mathbf{b})\\ \mathbf{x}_{i+1} &= \mathbf{x}_{i+1/2} - \mathbf{Q}^{-1}(\mathbf{A}\mathbf{x}_{i+1/2} - \mathbf{b}).\end{aligned}$$

Composite combinations consisting of $\mathbf{P}^{-1}$ and $\mathbf{Q}^{-1}$ are an effective acceleration strategy if $\mathbf{P}^{-1}$ eliminates a portion of the error space and $\mathbf{Q}^{-1}$ handles the rest. A now mature theory for these composite combinations was developed in the 1980s and 1990s in the context of domain decomposition methods [52, 56].

Linear left- and right-preconditioning, when used in conjunction with Krylov iterative methods [48], is standard practice for the parallel solution of linear systems of equations. We write the use of a linear Krylov method as $\mathrm{K}(\mathbf{A}, \mathbf{x}, \mathbf{b})$, where $\mathbf{A}$ is the matrix, $\mathbf{x}$ the initial solution, and $\mathbf{b}$ the RHS.

Linear left-preconditioning recasts the problem as

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

while right-preconditioning takes two stages,

$$\begin{aligned}\mathbf{A}\mathbf{P}^{-1}\mathbf{y} &= \mathbf{b}\\ \mathbf{P}^{-1}\mathbf{y} &= \mathbf{x},\end{aligned}$$

where one solves for the preconditioned solution $\mathbf{y}$ and then transforms $\mathbf{y}$ using $\mathbf{P}^{-1}$ to be the solution of the original problem.

**3. Nonlinear Composed Solvers.** We take the basic patterns from the previous section and apply them to the nonlinear case. We emphasized that unlike the linear case, the nonlinear case requires the solution as well as the residual to be defined both in the outer solver and in the preconditioner. With this in mind, we will show how composite combination and preconditioning may be systematically transferred to the nonlinear case.

We use the notation $\mathbf{x}_{i+1} = \mathbf{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b})$ for the action of a nonlinear stationary solver. It is also useful to consider the action of a solver that is dependent on the previous $m$ approximate solutions and the previous $m$ residuals and as $\mathbf{x}_{i+1} = \mathbf{M}(\mathbf{F}, \mathbf{x}_{i-m+1}, \cdots, \mathbf{x}_i, \mathbf{r}_{i-m+1}, \cdots, \mathbf{r}_i, \mathbf{b})$. Methods that use other state, such as previous solutions, residuals, or step directions, will have those listed in the per-iteration inputs as well.

Nonlinear composite combination consists of a sequence or series of two (or more) solution methods $\mathbf{M}$ and $\mathbf{N}$, which both provide an approximate solution to (2.1). Nonlinear preconditioning, on the other hand, may be cast as a modification of the function $\mathbf{F}$ through application of inner method $\mathbf{N}$. The modified function is then provided to an outer solver $\mathbf{M}$, which solves the preconditioned system.

An additive composite combination may be written as

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_{\mathbf{M}}(\mathbf{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i) + \alpha_{\mathbf{N}}(\mathbf{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i) \tag{3.1}$$

for weights $\alpha_{\mathbf{M}}$ and $\alpha_{\mathbf{N}}$. The multiplicative composite combination is simply

$$\mathbf{x}_{i+1} = \mathbf{M}(\mathbf{F}, \mathbf{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}), \mathbf{b}). \tag{3.2}$$

Nonlinear left-preconditioning may be directly recast from the linear stationary solver case,

$$\begin{aligned}\mathbf{P}^{-1}\mathbf{r}_i &= \mathbf{P}^{-1}(\mathbf{A}\mathbf{x}_i - \mathbf{b}) \\ &= \mathbf{x}_i + \mathbf{P}^{-1}(\mathbf{A}\mathbf{x}_i - \mathbf{b}) - \mathbf{x}_i \\ &\equiv \mathbf{x}_i - \mathbf{N}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}).\end{aligned}$$

Thus, the left-preconditioned residual is given by

$$\mathbf{r}(\mathbf{x}) = \mathbf{x} - \mathbf{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}). \tag{3.3}$$

Under the right circumstances the recast system will have better conditioning and less severe nonlinearities than the original system. The literature provides several examples of nonlinear left-preconditioning: nonlinear successive over relaxation (SOR) has been used in place of linear left-applied SOR [16], additive Schwarz-preconditioned inexact Newton (ASPIN) [13] uses an overlapping nonlinear additive Schwarz method to provide the left-preconditioned function and Jacobian to an outer Newton's method solver. Walker's fixed-point preconditioned Anderson mixing [60] uses a similar methodology. Many of these methods or variants thereof are discussed and tested later in this paper.

Nonlinear right-preconditioning involves a different recasting of the residual. This time, we treat the nonlinear solver $\mathbf{N}$ as a nonlinear transformation of the problem to one with solution $\mathbf{y}$, as $\mathbf{x} = \mathbf{P}^{-1}\mathbf{y}$ is a linear transformation of $\mathbf{x}$. Accordingly, the nonlinear system in (2.1) can be rewritten in two steps as

$$\mathbf{F}(\mathbf{N}(\mathbf{F}, \mathbf{y}, \mathbf{b})) = \mathbf{b} \tag{3.4}$$

and the solution by outer solver $\mathbf{M}$ as

$$\mathbf{y}_{i+1} = \mathbf{M}(\mathbf{F}(\mathbf{N}(\mathbf{F}, \cdot, \mathbf{b})), \mathbf{x}_i, \mathbf{b}) \tag{3.5}$$

followed by

$$\mathbf{x}_{i+1} = \mathbf{N}(\mathbf{F}, \mathbf{y}_{i+1}, \mathbf{b}).$$

Nonlinear right-preconditioning may be interpreted as $\mathbf{M}$ putting preconditioner $\mathbf{N}$ "within striking distance" of the solution. Nonlinear right-preconditioning is equivalent to right-preconditioning for a linear problem as

$$\begin{aligned}\mathbf{F}(\mathbf{N}(\mathbf{F}, \mathbf{y}, \mathbf{b})) &= \mathbf{b} \\ \mathbf{A}\mathbf{N}(\mathbf{F}, \mathbf{y}, \mathbf{b}) &= \mathbf{b} \\ \mathbf{A}\mathbf{P}^{-1}\mathbf{y} &= \mathbf{b}\end{aligned}$$

followed by

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

It's possible to solve for $\mathbf{y}$ directly, with an outer solver using function $\mathbf{F}(\mathbf{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}))$. However, the combination of inner solve and function evaluation is significantly more expensive than computing $\mathbf{F}(\mathbf{x})$ and should be avoided. We will show that in the case where $\mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})$ is Newton-Krylov, (3.5) is equivalent to (3.2). Considering them as having similar mathematical and algorithmic properties is appropriate in general.

In the special case of Newton's method, nonlinear right-preconditioning is referred to by Cai [12] and Cai and Li [14] as nonlinear elimination [39]. The idea behind nonlinear elimination is to use a local nonlinear solver to fully resolve difficult localized nonlinearities when they begin to stifle the global Newton's method; see Section 4.3. Grid sequencing [55, 35] and pseudo-transient [32] continuation methods work by a similar principle, using precursor solves to put Newton's method at an initial guesss from which it has fast convergence. Alternating a global linear or nonlinear step with local nonlinear steps has been studied as the LATIN method [38, 37, 17]. FAS-preconditioned NGMRES for recirculating flows [63] is another application of nonlinear right-preconditioning, where the FAS iteration is stabilized and accelerated by constructing a combination of several previous FAS iterates.

For solvers based on a search direction, left-preconditioning is much more natural. In fact, general line searches may be expressed as left-preconditioning by the nonlinear Richardson method. Left-preconditioning also has the property that for problems with poorly scaled residuals, the inner solver may provide a tenable search direction when one could not be found based on the original residual.

A major difficulty with the left-preconditioned option is that appropriate globalization may be much more expensive. Line searches involving the direct computation of $\mathbf{x} - \mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})$ at points along the line may be overly expensive given that the computation of the function now requires nonlinear solves. Line searches over $\mathbf{x} - \mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})$ may also miss stagnation of weak inner solvers and must be monitored. One may also base the line search on the unpreconditioned residual. The line search based on $\mathbf{x} - \mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})$ is often recommended [30] and is the "correct" one for general left-preconditioned nonlinear solvers.

Our basic notation for composite combinations and preconditioning is described in Table 3.1.

Table 3.1: Nonlinear composite combinations and preconditioning given outer and inner solver $\mathbf{M}$ and $\mathbf{N}$.

| Composition Type | Symbol | Statement | Abbreviation |
|---|---|---|---|
| Additive Composite | $+$ | $\mathbf{x} + \alpha_{\mathbf{M}}(\mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}) - \mathbf{x}) + \alpha_{\mathbf{N}}(\mathbf{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}) - \mathbf{x})$ | $\mathbf{M} + \mathbf{N}$ |
| Multiplicative Composite | $*$ | $\mathbf{M}(\mathbf{F}, \mathbf{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{b})$ | $\mathbf{M} * \mathbf{N}$ |
| Left-Preconditioning | $-_L$ | $\mathbf{M}(\mathbf{x} - \mathbf{N}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}, \mathbf{b})$ | $\mathbf{M} -_L \mathbf{N}$ |
| Right-Preconditioning | $-_R$ | $\mathbf{M}(\mathbf{F}(\mathbf{N}(\mathbf{F}, \mathbf{x}, \mathbf{b})), \mathbf{x}, \mathbf{b})$ | $\mathbf{M} -_R \mathbf{N}$ |
| Inner Linearization Inversion | $\backslash$ | $\mathbf{y} = \mathbf{J}(\mathbf{x})^{-1}\mathbf{r}(\mathbf{x}) = \mathrm{K}(\mathbf{J}(\mathbf{x}), \mathbf{y}_0, \mathbf{b})$ | NEWT\K |

**4. Solvers.** We now introduce several algorithms, the details of their implementation and use, and an abstract notion of how they may be composed. We first describe outer solution methods and how composition is applied to them. We then move on to solvers used primarily as inner methods. The distinction is arbitrary but leads to the discussion of decomposition methods in Section 5. In addition, some of these algorithms can be treated as both a solver and a globalization strategy. We begin by commenting on two solvers, nonlinear Richardson and nonlinear GMRES, that may be directly considered as globalization strategies.

**4.1. Nonlinear Richardson (NRICH).** The most popular globalization strategy in the solution of nonlinear PDEs is the line search. Given a functional $f(\mathbf{x})$, a starting point $\mathbf{x}_i$, and a direction $\mathbf{d}$, we compute $\lambda \approx \arg\min_{\mu>0} f(\mathbf{x}_i + \mu\mathbf{d})$. The functional $f(\cdot)$ may be $\|\mathbf{F}(\cdot)\|_2^2$ or a problem-specific objective function. Many solvers which do not converge when only full or simply damped steps are used converge well when combined with a line search. Different variants of line searches are appropriate for different solvers.

The nonlinear analogue to the Richardson iteration is merely the simple application of a line search. NRICH takes a step in the negative residual direction and scales that step sufficiently to guarantee convergence. NRICH is known as steepest descent [26] in the optimization context where $\mathbf{r}$ is the gradient of a functional $f(\cdot)$ noted above. NRICH is outlined in Alg. 1.

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

Alg. 1: Nonlinear Richardson Iteration

In the numerical solution of nonlinear PDEs, we generally want to guarantee progress in the minimization of $f(\mathbf{x}) = \|\mathbf{r}\|_2^2$ at each stage. However, $\nabla\|\mathbf{r}\|_2^2$ is not $\mathbf{r}$, but instead $2\mathbf{J}^\top\mathbf{r}$. With Newton's method, the Jacobian has been freshly computed before the line search, so computing the Jacobian-vector product is tenable.

A cubic backtracking (BT) line search [21] is used in conjunction with methods based on Newton's method in this work, and the variant used here is elaborated in Alg. 2 for the sake of completeness. A major benefit of BT is that it defaults to taking the full step if that step is sufficient with respect to the Wolfe conditions [65], and does no more work unless necessary. The backtracking line search may stagnate entirely

for ill-conditioned Jacobian [59]. Even without a Jacobian, the step arising from a non-Newton's method may still be ill-scaled.

**procedure** BT($\mathbf{F}, \mathbf{y}, \lambda_0, n, \alpha = 10^{-4}$)
  $s = \mathbf{r}(\mathbf{x})^\top \mathbf{J}(\mathbf{x})\mathbf{y}$
  **if** $(\|\mathbf{r}(\mathbf{x} + \lambda_0\mathbf{y})\|_2^2 \leq \|\mathbf{r}(\mathbf{x})\|_2 + 2\alpha\lambda_0 s)$ **then**
    $\lambda = \lambda_0$
    **return**
  **end if**
  $\mu = \lambda$
  $\lambda_q = \frac{-s}{\|\mathbf{r}(\mathbf{x}+\lambda_0\mathbf{y})\|_2^2 - \|\mathbf{r}(\mathbf{x})\|_2^2 - 2\lambda s}$
  $\lambda = \begin{cases} 0.1\lambda & \text{if } \lambda_q < 0.1\lambda \\ 0.5\lambda & \text{if } \lambda_q > 0.5\lambda \\ \lambda_q & \text{otherwise} \end{cases}$
  **if** $(\|\mathbf{r}(\mathbf{x} + \lambda\mathbf{y})\|_2^2 \leq \|\mathbf{r}(\mathbf{x})\|_2 + 2\alpha\lambda\mathbf{r}(\mathbf{x})^\top s)$ **then** ▷ quadratic is sufficient
    **return**
  **end if**
  **for** $i = 0$ **do** $n - 1$
    $t_1 = 0.5(\|\mathbf{r}(\mathbf{x} + \mu\mathbf{y})\|_2^2 - \|\mathbf{r}(\mathbf{x})\|_2^2) - \lambda s$, $t_2 = 0.5(\|\mathbf{r}(\mathbf{x} + \lambda\mathbf{y})\|_2^2 - \|\mathbf{r}(\mathbf{x})\|_2^2) - \lambda s$
    $a = \frac{t_1/\lambda^2 - t_2/\mu^2}{\lambda - \mu}$, $b = \frac{\lambda t_2/\mu^2 - \mu t_1/\lambda^2}{\lambda - \mu}$
    $d = \begin{cases} b^2 - 3as & d > 0 \\ 0 & \text{otherwise} \end{cases}$
    $\lambda_c = \begin{cases} -s/2b & \text{if } a = 0 \\ (\sqrt{d} - b)/3a & \text{otherwise} \end{cases}$
    $\lambda = \begin{cases} 0.1\lambda & \text{if } \lambda_c < 0.1\lambda \\ 0.5\lambda & \text{if } \lambda_c > 0.5\lambda \\ \lambda_c & \text{otherwise} \end{cases}$
    **if** $(\|\mathbf{r}(\mathbf{x} + \lambda\mathbf{y})\|_2^2 \leq \|\mathbf{r}(\mathbf{x})\|_2 + 2\alpha\lambda s)$ **then** ▷ cubic is sufficient
      **return**
    **end if**
    $\mu = \lambda$
  **end for**
**end procedure**

Alg. 2: BT Line Search

An iterative secant search for a minimum value of $\|\mathbf{r}(\mathbf{x} + \lambda\mathbf{y}\|_2^2$ is defined in Alg. 3.

**procedure** L2($\mathbf{F}, \mathbf{y}, \lambda_0, n$)
  $\lambda_{-1} = 0$
  **for** $i = 0$ **do** $n - 1$
    $\nabla_\mathbf{y}\|\mathbf{r}(\mathbf{x} + \lambda_i\mathbf{y})\|_2^2 = \frac{3\|\mathbf{r}(\mathbf{x}+\lambda_i\mathbf{y})\|_2^2 - 4\|\mathbf{r}(\mathbf{x}+\frac{1}{2}(\lambda_i+\lambda_{i-1})\mathbf{y})\|_2^2 + \|\mathbf{r}(\mathbf{x}+\lambda_{i-1}\mathbf{y}))\|_2^2}{(\lambda_i - \lambda_{i-1})}$
    $\nabla_\mathbf{y}\|\mathbf{r}(\mathbf{x} + \lambda_{i-1}\mathbf{y})\|_2^2 = \frac{\|\mathbf{r}(\mathbf{x}+\lambda_i\mathbf{y})\|_2^2 - 4\|\mathbf{r}(\mathbf{x}+\frac{1}{2}(\lambda_i+\lambda_{i-1})\mathbf{y})\|_2^2 + 3\|\mathbf{r}(\mathbf{x}+\lambda_{i-1}\mathbf{y}))\|_2^2}{(\lambda_i - \lambda_{i-1})}$
    $\lambda_{i+1} = \lambda_i - \frac{\nabla_\mathbf{y}\|\mathbf{r}(\mathbf{x}+\lambda_i\mathbf{y})\|_2^2(\lambda_i - \lambda_{i-1})}{\nabla_\mathbf{y}\|\mathbf{r}(\mathbf{x}+\lambda_i\mathbf{y})\|_2^2 - \nabla_\mathbf{y}\|\mathbf{r}(\mathbf{x}+\lambda_{i-1}\mathbf{y})\|_2^2}$
  **end for**
**end procedure**

Alg. 3: L2 Line Search

When converged, L2 is equivalent to an optimal damping in the direction of the residual and will forestall

divergence. $\nabla_{\mathbf{y}}$ is calculated by polynomial approximation, requiring two additional function evaluations per application. In practice and in our numerical experiments the number of inner iterations $n = 1$.

NRICH is often slow to converge for general problems and stagnates quickly. However, different step directions than $\mathbf{r}$ may be generated by nonlinear preconditioning and can improve convergence dramatically.

1: **procedure** NRICH($\mathbf{x} - \mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b}), \mathbf{x}_i, 0$)
2: $\quad \mathbf{d} = \mathbf{M}(\mathbf{F}, \mathbf{x}_i, \mathbf{b}) - \mathbf{x}_i$
3: $\quad \mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{d}$ ▷ $\lambda$ determined by line search
4: **end procedure**

Alg. 4: Nonlinear Richardson Iteration: Left-Preconditioning

As shown in Alg. 4, we replace the original residual equation $\mathbf{r}(\mathbf{x})$ with $\mathbf{x} - \mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})$ and apply NRICH to the new problem. There are two choices for $f(\cdot)$ in the line search. The first is based on minimizing the original residual, as in the unpreconditioned case; the second minimizes the norm of the preconditioned residual instead. Minimizing the unpreconditioned residual with a preconditioned step is more likely to stagnate, as there is no guarantee that the preconditioned step is a descent direction with respect to the gradient of $\|\mathbf{r}(\mathbf{x})\|_2$.

**4.2. Nonlinear GMRES (NGMRES).** We may also consider more complex globalization than the line search. For instance, NGMRES constructs the optimal combination of several previous steps. A number of related methods fall under the broad category of NGMRES-style nonlinear Krylov methods. These include Anderson mixing [2], NGMRES [62], direct inversion in the iterative subspace (DIIS) [45], and other similar methods. NGMRES is outlined in Alg. 5.

1: **procedure** NGMRES($\mathbf{F}, \mathbf{x}_i \cdots \mathbf{x}_{i-m+1}, \mathbf{b}$ )
2: $\quad \mathbf{d}_i = -\mathbf{r}(x_i)$
3: $\quad \mathbf{x}_i^M = \mathbf{x}_i + \lambda \mathbf{d}_i$
4: $\quad \mathbf{F}_i^M = \mathbf{r}(\mathbf{x}_i^M)$
5: $\quad$ **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 $\{\alpha_{i-m} \cdots \alpha_{i-1}\}$
6: $\quad \mathbf{x}_i^A = (1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}^M + \sum_{k=i-m}^{i-1} \alpha_k \mathbf{x}_k$
7: $\quad \mathbf{x}_{i+1} = \mathbf{x}_i^A$ or $\mathbf{x}_i^M$ if $\mathbf{x}_i^A$ is insufficient.
8: **end procedure**

Alg. 5: Nonlinear GMRES

The $\alpha_i$ are computed by solving the minimization problem

$$f(\alpha) = \|\mathbf{r}((1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}^M + \sum_{k=i-m}^{i-1} \alpha_i \mathbf{x}_i)\|_2^2. \tag{4.1}$$

Under the assumption that

$$\frac{\partial \mathbf{r}((1 - \sum_{k=i-m}^{i-1} \alpha_i)\mathbf{x}^M + \sum_{k=i-m}^{i-1} \alpha_i \mathbf{x}_i)}{\partial \alpha_i} \approx [\mathbf{r}(\mathbf{x}_i) - \mathbf{r}(\mathbf{x}^M)] \tag{4.2}$$

the problem has approximate gradient

$$\nabla f(\alpha)_i = 2\mathbf{r}((1 - \sum_{k=i-m}^{i-1} \alpha_k)\mathbf{x}^M + \sum_{k=i-m}^{i-1} \alpha_k \mathbf{x}_k)^\top [\mathbf{r}(\mathbf{x}_i) - \mathbf{r}(\mathbf{x}^M)] \tag{4.3}$$

and Hessian

$$[\nabla^2 f(\alpha)]_{ij} = 2[\mathbf{r}(\mathbf{x}_j) - \mathbf{r}(\mathbf{x}^M)]^\top [\mathbf{r}(\mathbf{x}_i) - \mathbf{r}(\mathbf{x}^M)]. \tag{4.4}$$

Assuming a starting point of $\mathbf{x}^M$ corresponding to $\alpha_i = 0$ for all $i$ and a limit of one Newton's method step for the minimization, the calculation simplifies to a linear solve for $\vec{\alpha}$ as

$$\nabla^2 f(0)\vec{\alpha} = \nabla f(0). \tag{4.5}$$

Equation (4.5) is solved with a dense least-squares solver once every iteration. It's also possible to state the minimization as a linear program [57] if one chooses a different norm. Stagnation of the method is detected by monitoring how close the subsequent solutions are to one another and making sure that a sufficient decrease in the residual norm is achieved. If stagnation is detected for two iterations in succession, $\mathbf{x}^M$ is taken as the solution, and the minimization problem is cleared.

In practice, the calculation of $\mathbf{x}^M$ can also include a line search to ensure that $\mathbf{d}$ is a reasonable search direction. In the Anderson mixing method the extra function evaluation required by the line search would be wasted. Despite the more heavyweight iteration requiring two function evaluations per iteration, NGMRES often performs better than Anderson mixing because of its ability to overcome stagnation.

Both right- and left-preconditioning can be applied to NGMRES or Anderson mixing. With right-preconditioning the computation of $\mathbf{x}^M = \mathbf{x} + \lambda\mathbf{d}$ is replaced with $\mathbf{x}^M = \mathbf{M}(\mathbf{F}, \mathbf{x}, \mathbf{b})$. Left-preconditioning can be applied by replacing $\mathbf{F}$, but the multiple computations of $\mathbf{r}$ makes right-preconditioning less costly. Right-preconditioned NGMRES has been applied for recirculating flows [63] using FAS and in stabilizing lagged Newton's methods [15, 49]. Simple preconditioning of NGMRES has also been proposed in the optimization context [19]. Preconditioned Anderson mixing has been leveraged as an outer accelerator for the Picard iteration [61, 40].

We can use the same formulation expressed in (4.5) to determine $\alpha_{\mathbf{M}}$ and $\alpha_{\mathbf{N}}$ (or any number of weights) in (3.1), using the solutions and final residuals from a series of inner nonlinear solvers instead of the sequence of previous solutions. This sort of residual-minimizing globalization is generally applicable in additive composite combinations of solvers. All instances of additive composite combination in the experiments use this formulation.

**4.3. Newton-Krylov Methods (NEWT\K).** In Newton-Krylov methods, the search direction is determined by inexact iterative inversion of the Jacobian $\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{F}(\mathbf{x})}{\partial \mathbf{x}}$ applied to the residual using a preconditioned Krylov method.

```
1: procedure NEWT\K(F, x_i, b)
2:         d = J(x_i)^{-1} r(x_i, b)        ▷ approximate inversion by Krylov method
3:     x_{i+1} = x_i + λd                   ▷ λ determined by line search
4: end procedure
```

Alg. 6: Newton-Krylov Method

NEWT\K cannot be guaranteed to converge far away from the solution, and it is routinely augmented by a line search. In our formalism, NEWT\K with a line search can be expressed as NRICH left-preconditioned by NEWT\K coming from Alg. 4. However, we will consider line-search globalized NEWT\K as the standard, and will omit the outer NRICH when using it, leading to Alg. 6. Other globalizations, such as trust-region methods, are also often used in the context of optimization but will not be covered here.

NEWT\K is often the workhorse of simulations requiring solution of nonlinear equations. A multitude of implementations and variants exist, both in the literature and as software. The general organization of the components of NEWT\K is shown in Fig. 4.1. Note that the vast majority of potential customization occurs at the level of the linear preconditioner and that access to fast solvers is limited to that step. We will now describe techniques for nonlinear preconditioning of NEWT\K.

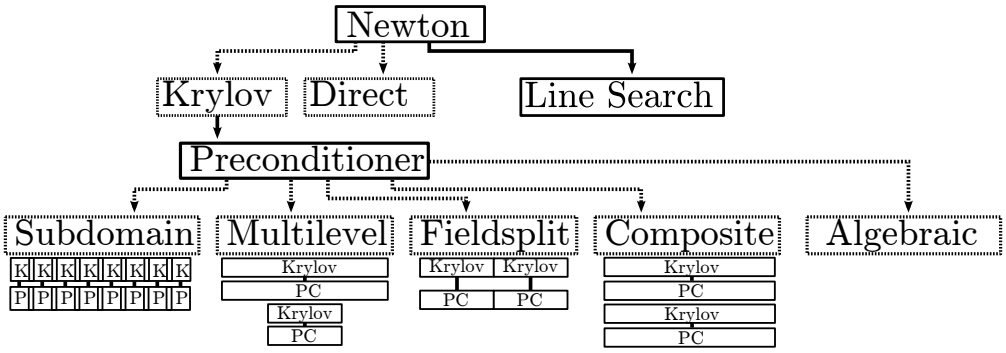


Fig. 4.1: Organization of a Newton-Krylov solver with potential points of customization. Choices for method or type of method are indicated with dashed outlines. Although one can alter the type, parameters, or tolerances of Newton's method, the Krylov solver, or the line search, we see that the majority of the potential customization and composition occurs at the level of the preconditioner. Possibilities include the use of algebraic preconditioners, such as direct solvers or sparse factorizations, or domain decomposition, multigrid, or composite solvers, which all have subcomponent solvers and preconditioners.

Nonlinear right-preconditioning of the Newton's method may take two forms. We describe our choice, and comment on the alternative. At first glance, right-preconditioning of the Newton's method requires construction or application of the right-preconditioned Jacobian

$$\frac{\partial \mathbf{F}(\mathbf{M}(\mathbf{F},\mathbf{y},\mathbf{b}))}{\partial \mathbf{y}} = \mathbf{J}(\mathbf{M}(\mathbf{F},\mathbf{y},\mathbf{b}))\frac{\partial \mathbf{M}(\mathbf{F},\mathbf{y},\mathbf{b})}{\partial \mathbf{y}}, \tag{4.6}$$

and we note that the computation of $\frac{\partial \mathbf{M}(\mathbf{F},\mathbf{y}_i,\mathbf{b})}{\partial \mathbf{y}_i}$ is generally impractical. Consider the transformation

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

up to first order. Therefore, it is reasonable to work entirely with the original solution rather than the transformed solution. NEWT\K$(\mathbf{F}(\mathbf{M}(\mathbf{F},\mathbf{x},\mathbf{b})),\mathbf{x},\mathbf{b})$ is easily implemented, as shown in Alg. 7.

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

Alg. 7: Right-Preconditioned Newton-Krylov Method

Note that NEWT\K$-_R\mathbf{M}$ is directly equivalent to $\mathbf{M}*$NEWT\K in this form. The connection between composition and preconditioning in Newton's method provides an inkling as to the advantages afforded to NEWT\K as an inner composite solver combined with other methods. An alternative approach applies an

approximation to (4.6) directly [4]. The approximation is

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

which is solved for $\mathbf{d}$. Right-preconditioning by the approximation requires an application of the nonlinear preconditioner for every inner Krylov iterate and limits our choices of Krylov solver to those tolerant of nonlinearity, such as flexible GMRES [47].

1: **procedure** NEWT\K($\mathbf{x}-\mathbf{M}(\mathbf{F},\mathbf{x},\mathbf{b}),\mathbf{x}_i,0$)
2: $\quad\mathbf{d} = \frac{\partial(\mathbf{x}_i-\mathbf{M}(\mathbf{F},\mathbf{x}_i,\mathbf{b}))}{\partial \mathbf{x}_i}^{-1}(\mathbf{x}_i-\mathbf{M}(\mathbf{F},\mathbf{x}_i,\mathbf{b}))$ ▷ approximate inversion by Krylov method
3: $\quad\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda\mathbf{d}$ ▷ $\lambda$ determined by line search
4: **end procedure**

Alg. 8: Left-Preconditioned Newton-Krylov Method

For nonlinear left-preconditioning, shown in Alg. 8, we replace the computation $\mathbf{r}(\mathbf{x}_i)$ with $\mathbf{x}_i-\mathbf{M}(\mathbf{F},\mathbf{x}_i,\mathbf{b})$ and take the Jacobian of the function as an approximation of

$$\frac{\partial(\mathbf{x}-\mathbf{M}(\mathbf{F},\mathbf{x}_i,\mathbf{b}))}{\mathbf{x}_i} = \mathbf{I} - \frac{\partial \mathbf{M}(\mathbf{F},\mathbf{x}_i,\mathbf{b})}{\partial \mathbf{x}_i}, \tag{4.7}$$

where now the Jacobian is a linearization of $\mathbf{M}(\mathbf{F},\mathbf{x},\mathbf{b})$ and impractical to compute in most cases. In the particular case where the preconditioner is NASM, this is known as ASPIN. In the case of NASM (Section 5.2)-preconditioning, one has local block Jacobians, so the approximation of the preconditioned Jacobian as

$$\frac{\partial(\mathbf{x}-\mathbf{M}(\mathbf{F},\mathbf{x},\mathbf{b}))}{\partial \mathbf{x}} = \frac{\partial(\mathbf{x}-(\mathbf{x}-\sum_b \mathbf{J}^b(\mathbf{x}^b)^{-1}\mathbf{F}^b(\mathbf{x}^b)))}{\partial \mathbf{x}} \approx \sum_b \mathbf{J}^b(\mathbf{x}^{b*})^{-1}\mathbf{J}(\mathbf{x}) \tag{4.8}$$

is used instead. The iteration requires only one inner nonlinear iteration and some small number of block solves, per outer nonlinear iteration. By contrast, direct differencing would require one inner iteration at each inner linear iteration for the purpose of repeated approximate Jacobian application. Note the similarity between ASPIN and the alternate form of right-preconditioning in terms of required components, with ASPIN being more convenient in the end.

**4.4. Quasi-Newton (QN).** The class of methods for nonlinear systems of equations known as limited-memory quasi-Newton methods [31] use previous changes in residual and solution or other low-rank expansions [34] to form an approximation of the inverse Jacobian by a series of low rank updates. The approximate Jacobian inverse is then used to compute the search direction. If the update is done directly, it requires storing and updating the dense matrix $\mathbf{K} \approx \mathbf{J}^{-1}$, which is impractical for large systems. One may overcome this limitation by using limited-memory variants of the method [43], which apply the approximate inverse Jacobian by a series of multiplicative low-rank updates. The basic form of QN methods is shown in Alg. 9.

1: **procedure** QN($\mathbf{F},\mathbf{x}_i,\mathbf{s}_{i-1}\cdots\mathbf{s}_{i-m},\mathbf{q}_{i-1}\cdots\mathbf{q}_{i-m},\mathbf{b}$ )
2: $\quad\mathbf{r}_i = \mathbf{r}(\mathbf{x}_i)$
3: $\quad\mathbf{x}_{i+1} = \mathbf{x}_i - \lambda\mathbf{K}_i\mathbf{r}_i$ ▷ $\lambda$ determined by line search
4: $\quad\mathbf{s}_i = \mathbf{x}_{i+1}-\mathbf{x}_i$
5: $\quad\mathbf{q}_i = \mathbf{F}(\mathbf{x}_{i+1})-\mathbf{F}(\mathbf{x}_i)$
6: **end procedure**

Alg. 9: Quasi-Newton Update

A popular variant of the method, L-BFGS, has a simple multiplicative construction,

$$\mathbf{K}_{i+1} = (\mathbf{I}-\frac{\mathbf{s}_i\mathbf{q}_i^\top}{\mathbf{s}_i^\top\mathbf{q}_i})\mathbf{K}_i(\mathbf{I}-\frac{\mathbf{q}_i\mathbf{s}_i^\top}{\mathbf{s}_i^\top\mathbf{q}_i}) + \frac{\mathbf{s}_i\mathbf{s}_i^\top}{\mathbf{q}_i^\top\mathbf{s}_i}, \tag{4.9}$$

that is efficiently applied recursively. Note that the update to the Jacobian in (4.9) is symmetric. Limited-memory rank-one updates, such as the "good" and "bad" variants of the Broyden method [23], may also be applied efficiently, but have worse convergence properties. Their overall performance may rival that of Newton's method [33] in many cases. QN methods also have deep connections with Anderson mixing [22]. Anderson mixing and the Broyden methods belong to a general class of Broyden-like methods [23].

The initial approximate inverse Jacobian $\mathbf{K}_0$ is often taken as the weighted [50] identity. However, it's also possible to take $\mathbf{K}_0 = \mathbf{J}_0^{-1}$ for lagged Jacobian $\mathbf{J}_0$. Such schemes greatly increase the robustness of lagged Newton's methods when solving nonlinear PDEs [10]. Left-preconditioning for QN is trivial, and either the preconditioned or unpreconditioned function may be used with the line search.

**4.4.1. Critical Point (CP) Line Search.** Many nonlinear PDEs have an energy function that may be not be obvious in the discrete form but may still be useful in forming a more effective line search. Suppose that $\mathbf{r}(x)$ is the gradient of some (hidden) objective functional $f(\mathbf{x})$ instead of $\|\mathbf{r}\|_2^2$. Trivially, one may minimize the hidden $f(\mathbf{x} + \lambda\mathbf{y})$ by seeking roots of

$$\mathbf{y}^\top \mathbf{F}(\mathbf{x} + \lambda\mathbf{y}) = \frac{df(\mathbf{x} + \lambda\mathbf{y})}{d\lambda},$$

by using a secant method. The resulting line search, CP, is outlined in Alg. 10.

**procedure** CP($\mathbf{F}, \mathbf{y}, \lambda_0, n$)
  $\lambda_{-1} = 0$
  **for** $i = 0$ **do** $n - 1$
    $\lambda_{i+1} = \lambda_i - \frac{\mathbf{y}^\top \mathbf{r}(\mathbf{x}+\lambda_i\mathbf{y})(\lambda_i - \lambda_{i-1})}{\mathbf{y}^\top \mathbf{r}(\mathbf{x}+\lambda_i\mathbf{y}) - \mathbf{y}^\top \mathbf{r}(\mathbf{x}+\lambda_{i-1}\mathbf{y})}$
  **end for**
**end procedure**

Alg. 10: CP Line Search

CP differs from the previous line searches in that it will not minimize $\|\mathbf{r}(\mathbf{x}+\lambda\mathbf{y})\|_2$ over $\lambda$. It is also not appropriate for some problems. However, CP shows much more rapid convergence than does L2 for certain solvers and problems. Both line searches may be started from $\lambda_0$ as an arbitrary or problem-dependent damping parameter, and $\lambda_{-1} = 0$. In practice, one iteration is usually satisfactory. For highly nonlinear problems, however, overall convergence may be accelerated by a more exact line search corresponding to a small number of iterations. CP has also been suggested for nonlinear conjugate gradient methods [51].

**4.5. Nonlinear CG (NCG).** Nonlinear conjugate gradient methods [25] are a simple extension of the linear CG method but with the optimal step length in the conjugate direction determined by line search rather than the exact formula that works in the linear case. NCG is outlined in Alg. 11. NCG requires storage of one additional vector for the conjugate direction $\mathbf{c}_i$ but has significantly faster convergence than does NRICH for appropriate problems.

**procedure** NCG($\mathbf{F}, \mathbf{x}_i, \mathbf{c}_{i-1}, \mathbf{r}_{i-1}, \mathbf{b}$)
  $\mathbf{r}_i = \mathbf{r}(\mathbf{x}_i)$
  $\beta_i = \frac{\mathbf{r}_i^\top (\mathbf{r}_i - \mathbf{r}_{i-1})}{\mathbf{r}_{i-1}^\top \mathbf{r}_{i-1}}$
  $\mathbf{c}_i = -\mathbf{r}(\mathbf{x}_i) + \beta_i \mathbf{c}_{i-1}$
  $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{c}_i$ ▷ $\lambda$ determined by line search
**end procedure**

Alg. 11: Nonlinear CG

Several choices exist for construction of the parameter $\beta_i$ [25, 18, 29, 24]. We choose the Polak-Ribière-Polyak [44] variant. The application of nonlinear left-preconditioning is straightforward. Right-preconditioning is not conveniently applicable as is the case with QN in Section 4.4 and linear CG.

NCG has limited applicability because it suffers from the same issues as its linear cousin for problems with non-symmetric Jacobian. It seems to be a common practice when solving nonlinear PDEs with NCG to

rephrase the problem in terms of the normal equations, which involves finding the root of $\mathbf{F}_N(\mathbf{x}) = \mathbf{J}^\top \mathbf{r}(\mathbf{x})$. As the Jacobian must be assembled and multiplied at every iteration, the normal equation solver is not a reasonable algorithmic choice, even with drastic lagging of Jacobian assembly. The normal equations also have a much worse condition number than does the original PDE. In our experiments, we use the conjugacy-ensuring CP line search described in Section 4.4.1.

**5. Decomposition Solvers.** An extremely important class of methods for linear problems is based on domain or hierarchical decomposition and so we consider nonlinear variants of domain decomposition and multilevel algorithms. We introduce three different nonlinear solver algorithms based on point-block solves, local subdomain solves, and coarse-grid solves.

There is a trade-off in these methods between the amount of computation dedicated to solving local or coarse problems and the communication for global assembly and convergence monitoring. Undersolving the subproblems will waste the communication overhead of the outer iteration, and oversolving exacerbates issues of load imbalance and quickly has diminishing returns with respect to convergence. The solvers in this section exhibit a variety of features related to these trade-offs.

Unlike the earlier global solvers, the decomposition solvers do not guarantee convergence. While they might converge, the obvious extension is to use them in conjunction with the global solvers as nonlinear preconditioners or accelerators. As the decomposition solvers expose more possibilities for parallelism or acceleration, their effective use in the nonlinear context should provide similar benefits to the analogous solvers in the linear context. A major disadvantage of these solvers is that each of them requires additional information about the local problems, a decomposition, or hierarchy of discretizations of the domain.

**5.1. Gauss-Seidel-Newton (GSN).** Effective stationary methods may be constructed by exact solves on subproblems. Suppose that the problem easily decomposes into $n_b$ small block subproblems. If Newton's method is applied multiplicatively by blocks, the resulting algorithm is known as Gauss-Seidel-Newton and is shown in Alg. 12. Similar methods are commonly used as nonlinear smoothers [27, 28]. To construct GSN, we define the individual block Jacobians $\mathbf{J}^b(\mathbf{x}^b)$ and functions $\mathbf{F}^b(\mathbf{x}^b)$; $\mathbf{R}^b$, which restricts to a block; and $\mathbf{P}^b$, which injects the solution to that block back into the overall solution. The point-block solver runs until the norm of the block residual is less than $\epsilon^b$ or $m_b$ steps have been taken.

**procedure** GSN( $\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
  $\mathbf{x}_{i+1} = \mathbf{x}_i$
  **for** $b = 1$ **do** $n_b$
    $\mathbf{x}^b_{i,0} = \mathbf{R}^b \mathbf{x}_{i+1}$
    $\mathbf{r}^b_{i,0} = \mathbf{R}^b \mathbf{b} - \mathbf{F}^b(\mathbf{x}^b_i)$
    **while** $\|\mathbf{r}^b\|_2 < \epsilon^b$ and $j < m_b$ **do**
      $j = j + 1$
      $\mathbf{x}^b_{i,j} = \mathbf{x}^b_{i,j-1} - \mathbf{J}^b(\mathbf{x}^b_{i,j-1})^{-1} \mathbf{r}^b_{i,j-1}$ ▷ direct inversion of block Jacobian
      $\mathbf{r}^b_{i,j} = \mathbf{R}^b \mathbf{b} - \mathbf{F}^b(\mathbf{x}^b_{i,j})$
    **end while**
    $\mathbf{x}_{i+1} = \mathbf{x}_{i+1} - \mathbf{P}^b(\mathbf{x}^b_{i,0} - \mathbf{x}^b_{i,j})$
  **end for**
**end procedure**

Alg. 12: Gauss-Seidel-Newton

GSN solves small subproblems with a fair amount of computational work per degree of freedom and thus has high arithmetic intensity. It is typically greater than two times the work per degree of freedom of NRICH for scalar problems, and even more for vector problems where the local Newton's method solve is over several local degrees of freedom.

When used as a solver, GSN requires one reduction per iteration. However, stationary solvers, in both the linear and nonlinear case, do not converge robustly for large problems. Accordingly, GSN would typically be used as a preconditioner, making monitoring of global convergence unnecessary. Thus, in practice there is no synchronization; GSN is most easily implemented with multiplicative update on serial subproblems and additively in parallel.

**5.2. Nonlinear Additive Schwarz Method (NASM).** GSN can be an efficient underlying kernel for sequential nonlinear solvers. For parallel computing, however, an additive method with overlapping subproblems has desirable properties with respect to communication and arithmetic intensity. The nonlinear additive Schwarz methods allow for medium-sized subproblems to be solved with a general method, and the corrections from that method summed into a global search direction. Here we limit ourselves to decomposition by subdomain rather than splitting the problem into fields. While eliminating fields might be tempting, the construction of effective methods of this sort is significantly more involved than in the linear case, and many interesting problems do not have such a decomposition readily available. Using subdomain problems $\mathbf{F}^B$, restrictions $\mathbf{R}^B$, injections $\mathbf{P}^B$, and solvers $\mathbf{M}^B$ Alg. 13 outlines the NASM solver with $n_B$ subdomains.

1: **procedure** NASM($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
2: **for** $B = 1$ **do** $n_B$
3: $\mathbf{x}_0^B = \mathbf{R}^B \mathbf{x}_i$
4: $\mathbf{x}^B = \mathbf{M}^B(\mathbf{F}^B, \mathbf{x}_0^B, \mathbf{b}_0^B)$
5: $\mathbf{y}^B = \mathbf{x}_0^B - \mathbf{x}^B$
6: **end for**
7: $\mathbf{x}_{i+1} = \mathbf{x}_i - \sum_{B=1}^{n_B} \mathbf{P}^B \mathbf{y}^B$
8: **end procedure**

Alg. 13: Nonlinear Additive Schwarz

There are two choices for the injections $\mathbf{P}^B$ in the overlapping regions. We will use NASM to denote the variant that uses overlapping injection corresponding to the whole subproblem step. The second variant, restricted additive Schwarz (RAS), injects the step in a non-overlapping fashion. The subdomain solvers are typically Newton-Krylov, but the other methods described in this paper are also applicable.

**5.3. Full Approximation Scheme (FAS).** FAS [6] accelerates convergence by advancing the nonlinear solution on a series of coarse rediscretizations of the problem. As with standard linear multigrid, the cycle may be constructed either additively or multiplicatively, with multiplicative being more effective in terms of per-iteration convergence. Additive provides the usual advantage of allowing the coarse-grid corrections to be computed in parallel. We do not consider additive FAS in this paper.

Given the smoother $\mathbf{M}_s(\mathbf{F}, \mathbf{x}, \mathbf{b})$ at each level, as well as restriction ($\mathbf{R}$), prolongation ($\mathbf{P}$) and injection ($\hat{\mathbf{R}}$) operators, and the coarse nonlinear function $\mathbf{F}^H$, the FAS V-cycle takes the form shown in Alg. 14.

1: **procedure** FAS($\mathbf{F}, \mathbf{x}_i, \mathbf{b}$)
2: $\mathbf{x}_s = \mathbf{M}_s(\mathbf{F}, \mathbf{x}_i, \mathbf{b})$
3: $\mathbf{b}^H = \mathbf{R}[\mathbf{b} - \mathbf{F}(\mathbf{x}_s)] + \mathbf{F}^H(\hat{\mathbf{R}}\mathbf{x}_s)$
4: $\mathbf{x}_i^H = \hat{\mathbf{R}}\mathbf{x}_s$
5: $\mathbf{x}_c = \mathbf{x}_s + \mathbf{P}[\mathrm{FAS}(\mathbf{x}_i^H, \mathbf{b}^H) - \hat{\mathbf{R}}\mathbf{x}_s]$
6: $\mathbf{x}_{i+1} = \mathbf{M}_s(\mathbf{F}, \mathbf{x}_c, \mathbf{b})$
7: **end procedure**

Alg. 14: Full Approximation Scheme

The difference between FAS and linear multigrid is the construction of the coarse RHS $\mathbf{b}^H$. In FAS it is guaranteed that if an exact solution $\mathbf{x}^*$ to the fine problem is found,

$$\mathrm{FAS}(\hat{\mathbf{R}}[\mathbf{x}^*], \mathbf{b}^H) - \hat{\mathbf{R}}[\mathbf{x}^*] = 0. \tag{5.1}$$

The correction is not necessary in the case of linear multigrid. However, FAS is mathematically identical to standard multigrid when applied to a linear problem.

The stellar algorithmic performance of linear multigrid methods is well documented, but the arithmetic intensity may be somewhat low. FAS presents an interesting alternative to NEWT\K $-$ MG, as it may be configured with high arithmetic intensity operations at all levels. The smoothers are themselves nonlinear

solution methods. FAS-type methods are applicable to optimization problems as well as nonlinear PDEs [8], with optimization methods used as smoothers [42].

We have now introduced the mathematical construction of nonlinear composed solvers. We have also described two classes of nonlinear solvers, based on solving either the global problem or some partition of it. The next step is to describe a set of test problems with different limitations with respect to which methods will work for them, construct a series of instructive example solvers using composition of the previously defined solvers, and test them. We have made an effort to create flexible and robust software for composed nonlinear solvers. The organization of the software is in the spirit of PETSc and includes several interchangeable component solvers, including NEWT\K; iterative solvers such as NGMRES and QN that may contain an inner preconditioner; decomposition solvers such as NASM and FAS, which are built out of subdomain solvers; and metasolvers implementing composite combinations. A diagram enumerating the composed solver framework analogous to that of the preconditioned NEWT\K case is shown in Fig. 5.1.

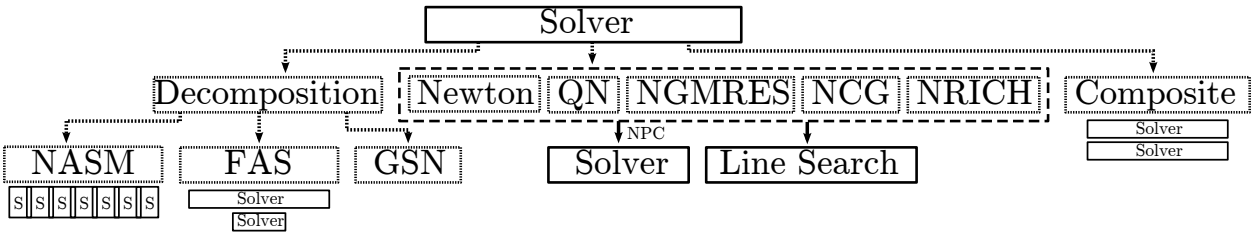


Fig. 5.1: Organization of the components of a composed nonlinear solver. We discard the difference between solvers and preconditioners and see that the nesting and potential for recursive customization lives at every level of the tree. Possibilities include iterative solvers (including Newton's method) with nonlinear preconditioning, composite solvers consisting of several subsolvers, or decomposition solvers consisting of subdomain or coarse nonlinear solvers. Choices for method or type of method are once again indicated with dashed outlines.

**6. Experiments.** We will demonstrate the efficacy of our methods by testing against a suite of nonlinear partial differential equations, that show interesting behavior in regimes with difficult nonlinearities. These problems are nonlinear elasticity, the lid-driven cavity with buoyancy, and the $p$-Laplacian.

One goal of this paper is to be instructive. We choose a subset of the above solvers for each problem and exhibit how to gain advantage over the standard solvers by using composition methods. Limitations of discretization or problem regime are noted, and we discuss how the solvers may be used under these limitations. We also explain how the reader may run the examples in this paper and experiment with the solvers both on the test examples shown here and on their own problems.

**6.1. Methods.** Our set of test algorithms is defined by the composition of a heavyweight iterative solver with a lightweight iterative or decomposition solver. In the case of the decomposition solvers, we choose one or two configurations per test problem, in order to avoid the combinatorial explosion of potential methods that already is apparent from the two forms of composition combined with the multitude of methods.

Our primary measure of performance is time to solution, which is both problem and equipment dependent. Since it depends strongly on the relative cost of function evaluation, Jacobian assembly, matrix multiplication, and subproblem or coarse solve, we also record number of nonlinear iterations, linear iterations, linear preconditioner applications, function evaluations, Jacobian evaluations, and nonlinear preconditioner applications. These measures allow us to characterize efficiency disparities in terms of major units of computational work.

We err on the side of oversolving for the inner solvers, as each method may be tweaked to greater efficiency, but considering the vast array of methods we have described here, the tuning would require too much elaboration. We make an active effort to keep solver parameters the same among methods while insuring reasonable convergence for all of the methods. These experimental results should be taken as a guide to improving solver performance, rather than definitive configurations for optimal performance. The test machine is a 64-core AMD Opteron 6274-based [1] machine with 128 GB of memory. The problems are run with one MPI process per core.

**6.2. Nonlinear Elasticity.** A Galerkin formulation for nonlinear elasticity may be stated as

$$\int_\Omega F \cdot S : \nabla v \, d\Omega + \int_\Omega b \cdot v \, d\Omega = 0 \tag{6.1}$$

for all test functions $v \in \mathcal{V}$; $F = \nabla \mathbf{u} + I$ is the deformation gradient. We use the Saint Venant-Kirchhoff model of hyperelasticity with second Piola-Kirchhoff stress tensor $S = \lambda \mathrm{tr}(E) I + 2\mu E$ for Lagrangian Green strain $E = F^\top F - I$ and Lamé parameters $\lambda$ and $\mu$, which may be derived from a given Young's modulus and Poisson ratio. We solve for displacements $\mathbf{u} \in \mathcal{V}$, given a constant load vector $b$ imposed on the structure.

The domain $\Omega$ is a 60° cylindrical arch of inner radius 100 m. Homogeneous Dirichlet boundary conditions are imposed on the outer edge of the ends of the arch. The goal is to "snap through" the arch, causing it to sag under the load rather than merely compressing it. To cause the sag, we set $b = -\mathbf{e}_y$ and set the Lamé constants consistent with a Young's modulus of 100 and a Poisson ratio of 0.2. The nonlinearity is highly activated during the process of snapping through and may be tuned to present a great deal of difficulty to traditional nonlinear solvers. The problem is discretized by using hexahedral $\mathbf{Q}_1$ finite elements on a logically structured grid, deformed to form the arch. The problem is a three-dimensional extension of a problem put forth by Wriggers [66]. The unstressed and converged solutions are shown in Fig. 6.1.

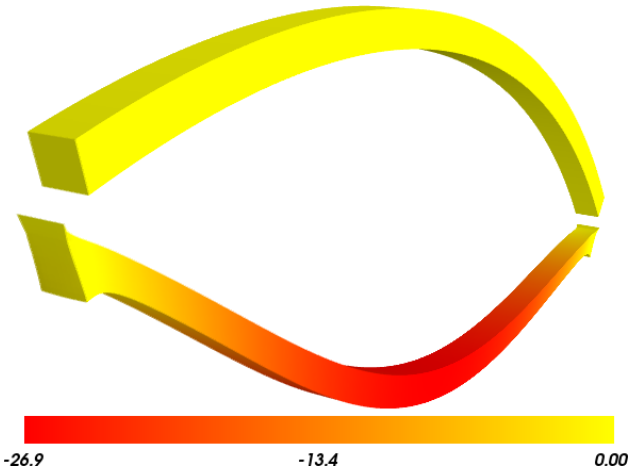


Fig. 6.1: Unstressed and stressed configurations for the elasticity test problem. Coloration indicates vertical displacement in meters.

For 3D hexahedral FEM discretizations, GSN and related algorithms are inefficient because for each degree of freedom each element in the support of the degree of freedom must be visited, resulting in eight visits to an interior element per sweep. We focus on algorithms requiring only a single visit to each cell, restricting us to function and Jacobian evaluations. Such approaches are also available in the general unstructured FEM case, and these experiments may guide users in regimes where no decomposition solvers are available.

For the experiments, we emphasize the role that nonlinear composition and alternative forms of globalization may play in the acceleration of nonlinear solvers. The problem is amenable to NCG NGMRES and NEWT\K with an algebraic multigrid preconditioner. We test a number of combinations of these solvers. Even though we have a logically structured grid, we approach the problem as if no reasonable grid hierarchy or domain decomposition were available. The solver combinations we use are listed in Table 6.1. In all the following tables, Solver denotes outer solver, LPC denotes the linear PC when applicable, NPC denotes the nonlinear PC when applicable, Side denotes the type of preconditioning, Smooth denotes the level smoothers used in the multilevel method (MG/FAS), and LS denotes line search.

Table 6.1: Series of solvers for the nonlinear elasticity test problem.

| Name | Solver | LPC | NPC | Side | Smooth | LS |
|---|---|---|---|---|---|---|
| NCG | NCG | | | | | CP |
| NEWT\K − MG | NEWT\K | MG | – | – | SOR | BT |
| NCG $-_L$ (NEWT\K − MG) | NCG | – | NEWT\K − MG | L | SOR | CP |
| NGMRES $-_R$ (NEWT\K − MG) | NGMRES | – | NEWT\K − MG | R | SOR | CP |
| NCG(10) + (NEWT\K − MG) | NCG,NEWT\K | MG | – | – | SOR | CP/BT |
| NCG(10) ∗ (NEWT\K − MG) | NCG,NEWT\K | MG | – | – | SOR | CP/BT |

For all the NEWT\K methods, we precondition the inner GMRES solve with a smoothed aggregation algebraic multigrid method provided by the GAMG package within PETSc. For all instances of the NCG solver, the CP line search initial guess for $\lambda$ is the final value from the previous nonlinear iteration. In the case of NGMRES $-_R$ (NEWT\K − MG) the inner line search is L2. NGMRES stores up to 30 previous solutions and residuals for all experiments. The outer line search for NCG $-_L$ (NEWT\K − MG) is a second-order secant approximation rather than first-order as depicted in Alg. 10. For the composite examples, 10 iterations of NCG are used as one of the subsolvers, denoted NCG(10).

The example, SNES ex16.c, can be run directly by using a default PETSc installation. The command line used for these experiments is

```
./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
```

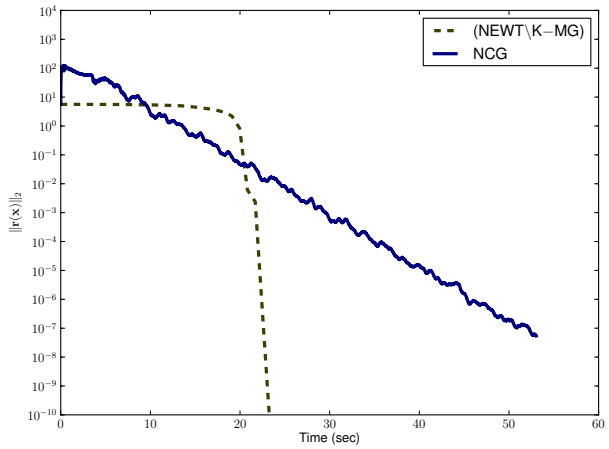


Fig. 6.2: (NEWT\K − MG) and NCG convergence.

Table 6.2: (NEWT\K − MG) and NCG results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| (NEWT\K − MG) | 23.43 | 27 | 1556 | 91 | 27 | 1618 | – |
| NCG | 53.05 | 4495 | 0 | 8991 | – | – | – |

The solvers shown in Table 6.2 converge slowly. NEWT\K − MG takes 27 nonlinear iterations and 1,618 multigrid V-cycles to reach convergence. NCG requires 8,991 function evaluations, and as it is unpreconditioned, scales unacceptably with problem size with respect to number of iterations. Note in Fig. 6.2 that while NCG takes an initial jump and then decreases at a constant rate, NEWT\K − MG is near stagnation

until the end, when it suddenly drops into the basin of attraction. If we were able to combine the advantages of NCG with those of algebraic multigrid, it would create an ideal solver for the black-box case.

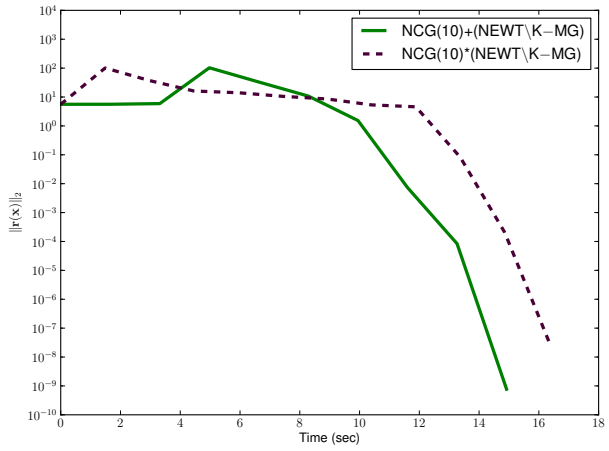


Fig. 6.3: NCG(10) + (NEWT\K − MG) and NCG(10) ∗ (NEWT\K − MG) convergence.

Table 6.3: NCG(10) + (NEWT\K − MG) and NCG(10) ∗ (NEWT\K − MG) results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| NCG(10) + (NEWT\K − MG) | 14.92 | 9 | 459 | 218 | 9 | 479 | – |
| NCG(10) ∗ (NEWT\K − MG) | 16.34 | 11 | 458 | 251 | 11 | 477 | – |

Results for composite combination are listed in Table 6.3. Additive and multiplicative composite combination provides roughly similar speedups for the problem, with more total outer iterations in the additive case. As shown in Fig. 6.3, the additive and multiplicative cases both do not stagnate. After the same initial jump and convergence at the pace of NCG, the basin of attraction is reached, and the entire iteration converges quadratically as should be expected with NEWT\K.

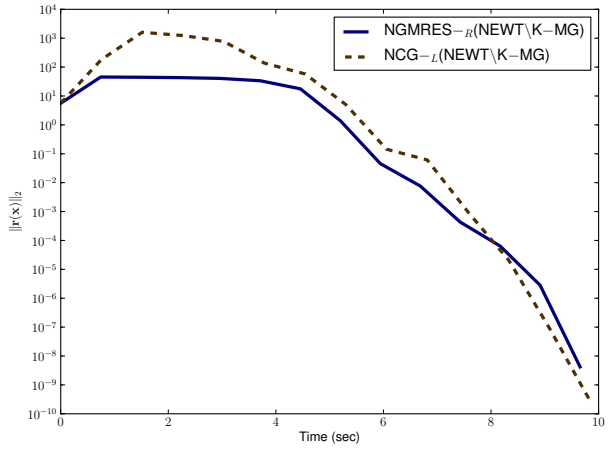


Fig. 6.4: NGMRES $-_R$ (NEWT\K − MG) and NCG $-_L$ (NEWT\K − MG) convergence.

Table 6.4: NGMRES $-_R$ (NEWT\K $-$ MG) and NCG $-_L$ (NEWT\K $-$ MG) results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| NGMRES $-_R$ (NEWT\K $-$ MG) | 9.65 | 13 | 523 | 53 | 13 | 548 | 13 |
| NCG $-_L$ (NEWT\K $-$ MG) | 9.84 | 13 | 529 | 53 | 13 | 554 | 13 |

Left-preconditioning of NCG and NGMRES with NEWT\K$-$MG provides even greater benefits, as shown in Fig. 6.4 and Table 6.4. The number of iterations is nearly halved from the unpreconditioned case, and the number of linear preconditioner applications is only slightly increased compared with the composite combination cases. In Fig. 6.4, we see that the Newton's method-preconditioned nonlinear Krylov solvers never dive as NEWT\K did but maintain rapid, mostly-constant convergence instead. This constant convergence is a significantly more efficient path to solution than the NEWT\K $-$ MG solver alone.

**6.3. Driven Cavity.** The lid-driven cavity formulation used for the next set of experiments is stated as

$$-\Delta \mathbf{u} - \nabla \times \Omega = 0 \tag{6.2}$$

$$-\Delta \Omega + \nabla \cdot (\mathbf{u}\Omega) - \mathrm{Gr}\nabla_x T = 0 \tag{6.3}$$

$$-\Delta T + \mathrm{Pr}\nabla \cdot \mathbf{u} = 0. \tag{6.4}$$

No-slip, rigid-wall Dirichlet conditions are imposed for $\mathbf{u}$. Dirichlet conditions are used for $\Omega$, based on the definition of vorticity, $\Omega = \nabla \times \mathbf{u}$, where along each constant coordinate boundary, the tangential derivative is zero. Dirichlet conditions are used for $T$ on the left and right walls, and insulating homogeneous Neumann conditions are used on the top and bottom walls. A finite-difference approximation with the usual 5-point stencil is used to discretize the boundary value problem to obtain a nonlinear system of equations. Upwinding is used for the divergence (convective) terms and central for the gradient (source) terms.

In these experiments, we emphasize the use of NGMRES, MG or FAS, and GSN. The solver combinations are listed in Table 6.5. The multilevel methods use a simple series of structured grids, with the smallest being 5x5 and the largest being 257x257. In NEWT\K $-$ MG, the Jacobian is rediscretized on each level. GSN is used as the smoother for FAS and solves the four-component block problem corresponding to (6.2) per grid point in a simple sweep through the processor local part of the domain.

The example, SNES ex19.c, can be run directly by using a default PETSc installation. The command line used for these experiments is

```
./ex19 -da_refine 6 -da_grid_x 5 -da_grid_y 5 -grashof 2e4 -lidvelocity 100 -prandtl 1.0
```

and the converged solution using these options is shown in Fig. 6.5.

Table 6.5: Solvers for the lid driven cavity problem.

| Name | Solver | LPC | NPC | Side | MG/FASSmooth. | LS |
|---|---|---|---|---|---|---|
| (NEWT\K $-$ MG) | NEWT\K | MG | – | – | SOR | BT |
| NGMRES $-_R$ (NEWT\K $-$ MG) | NGMRES | MG | NEWT\K | R | SOR | – |
| FAS | FAS | – | – | – | GSN | – |
| NRICH $-_L$ FAS | NRICH | – | FAS | L | GSN | L2 |
| NGMRES $-_R$ FAS | NGMRES | – | FAS | R | GSN | – |
| FAS $*$ (NEWT\K $-$ MG) | FAS/NEWT\K | MG | – | – | SOR/GSN | BT |
| FAS $+$ (NEWT\K $-$ MG) | FAS/NEWT\K | MG | – | – | SOR/GSN | BT |

All linearized problems in the NEWT\K iteration are solved by using GMRES with geometric multigrid preconditioning. There are five levels, with Chebychev$-$SOR smoothers on each level. For FAS, the smoother is GSN(5). The coarse-level smoother is 5 iterations of NEWT\K-LU; chosen for robustness. In the case of NGMRES $-_R$ (NEWT\K $-$ MG) and FAS $+$ (NEWT\K $-$ MG), the NEWT\K $-$ MG line search is a simple damping of $\lambda = 0.5$ and is a demonstration of the alternative globalization strategies.

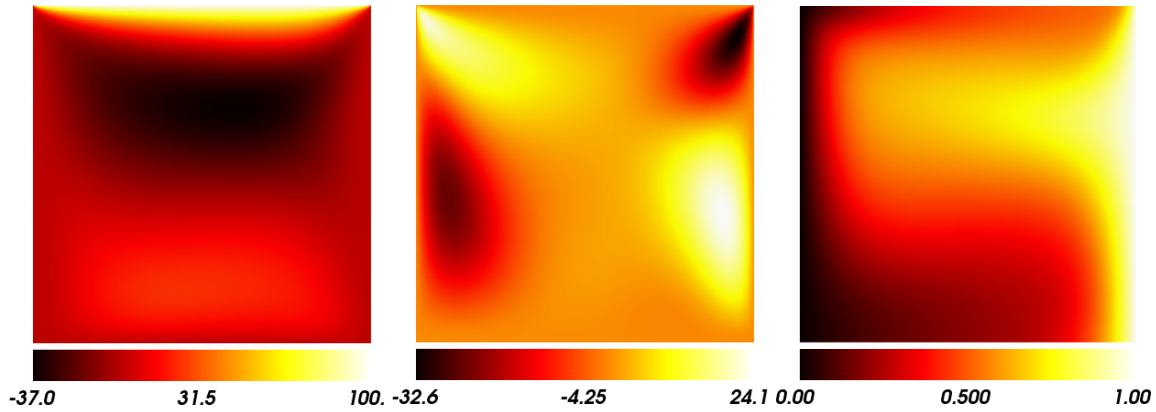


Fig. 6.5: $\mathbf{u}_x$, $\mathbf{u}_y$, and Temperature profiles for the lid drive cavity solution.

In these experiments we emphasize the total number of V-cycles, linear and nonlinear, since they are dominate the total run-time and contain the lion's share of the communication and FLOPs.

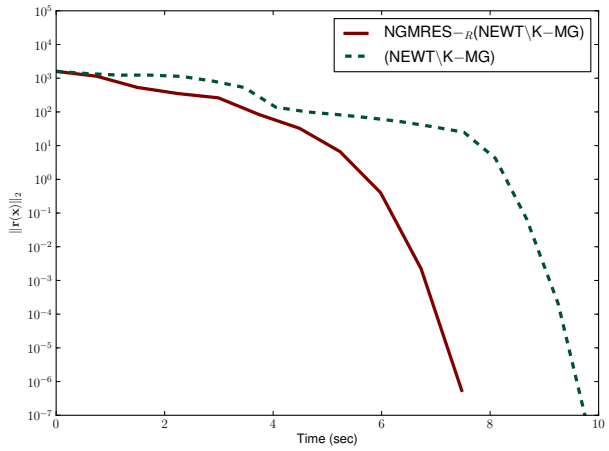


Fig. 6.6: NGMRES $-_R$ (NEWT\K $-$ MG) and (NEWT\K $-$ MG) convergence.

Table 6.6: NGMRES $-_R$ (NEWT\K $-$ MG) and (NEWT\K $-$ MG) results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| NGMRES $-_R$ (NEWT\K $-$ MG) | 7.48 | 10 | 220 | 21 | 50 | 231 | 10 |
| (NEWT\K $-$ MG) | 9.83 | 17 | 352 | 34 | 85 | 370 | – |

In Table 6.6, we see that Newton's method converges in 17 iterations, with 370 total V-cycles. Using NGMRES instead of a line search provides some benefit, as NGMRES $-$ (NEWT\K $-$ MG) takes 231 V-cycles and 10 iterations total.

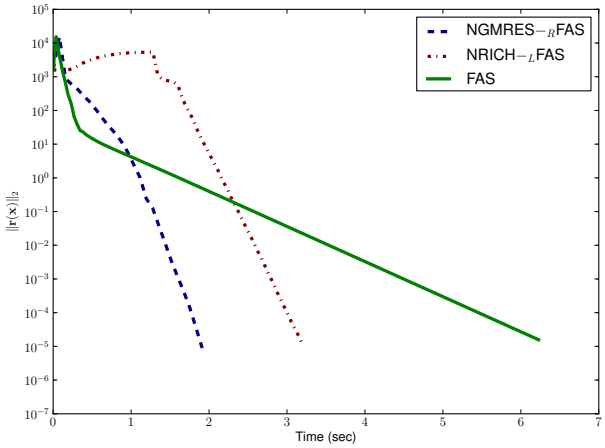


Fig. 6.7: NGMRES $-_R$ FAS, NRICH $-_L$ FAS, and FAS convergence.

Table 6.7: NGMRES $-_R$ FAS, NRICH $-_L$ FAS, and FAS results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| NGMRES $-_R$ FAS | 1.91 | 24 | 0 | 447 | 83 | 166 | 24 |
| NRICH $-_L$ FAS | 3.20 | 50 | 0 | 1180 | 192 | 384 | 50 |
| FAS | 6.23 | 162 | 0 | 2382 | 377 | 754 | – |

The additive composite combination has the added benefit that the line search in NEWT\K − MG may be turned off, reducing the number of function evaluations needed. Line search globalization as used by NRICH $-_L$ FAS takes 50 V-cycles, at the expense of three more fine-level function evaluations per iteration. NGMRES $-_R$ FAS reducing the number of V-cycles to 24 at the expense of more communication.

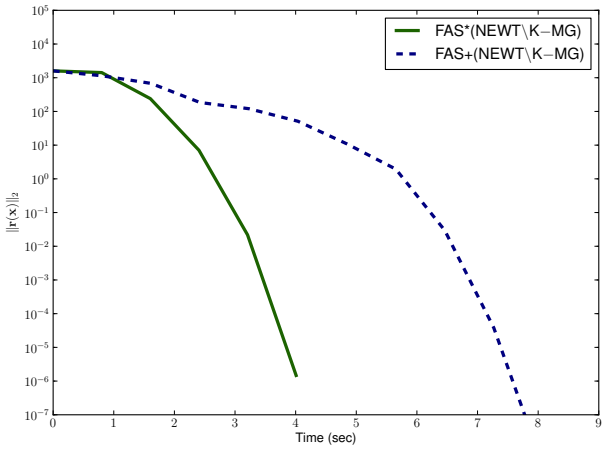


Fig. 6.8: FAS $*$ (NEWT\K − MG) and FAS + (NEWT\K − MG) convergence.

Table 6.8: FAS $*$ (NEWT\K $-$ MG) and FAS $+$ (NEWT\K $-$ MG) results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| FAS $*$ (NEWT\K $-$ MG) | 4.01 | 5 | 80 | 103 | 45 | 125 | – |
| FAS $+$ (NEWT\K $-$ MG) | 8.07 | 10 | 197 | 232 | 90 | 288 | – |

Composed nonlinear methods are shown in Table 6.8. Multiplicative and additive composite combinations consisting of FAS and NEWT\K$-$MG reduce the total number of V-cycles to 130 and 298, respectively. Note in Fig. 6.8 that both the additive and multiplicative solvers show that combining FAS and NEWT\K-MG may effectively speed solution, with multiplicative combination being significantly more effective.

The driven cavity problem may be difficult to converge with a standard Newton's method. However, it can be efficiently solved by using an accelerated nonlinear multilevel methods, or by a composite combination of nonlinear and linear multilevel methods. We note that, much like the elasticity problem in Section 6.2, other composite combinations may also be used here.

**6.4. $p$-Laplacian.** The regularized $p$-Laplacian formulation used for these experiments is

$$-\nabla \cdot ((\epsilon^2 + \frac{1}{2}|\nabla u|^2)^{(p-2)/2}\nabla u) = c,$$

where $\epsilon = 10^{-5}$ is the regularization parameter and $p$ the exponent of the Laplacian. When $p = 2$ the $p$-Laplacian reduces to the Poisson equation. We consider the case where $p = 5$ and $c = 0.1$. The domain is $[-1, 1] \times [-1, 1]$ and the initial guess is $u_0(x, y) = xy(1 - x^2)(1 - y^2)$. The initial and converged solutions are shown in Fig. 6.9.

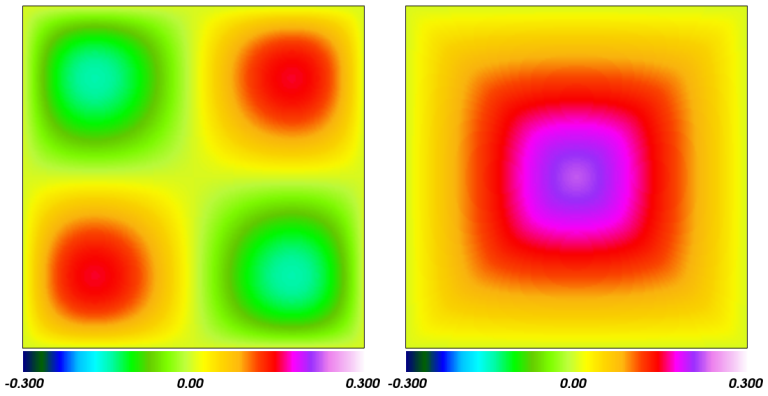


Fig. 6.9: Initial and converged solutions to the $p = 5$ $p$-Laplacian.

The example, SNES ex15.c, can be run directly by using a default PETSc installation. The command line used for these experiments is

```
./ex15 -da_refine 7 -da_overlap 6 -p 5.0 -lambda 0.0 -source 0.1
```

Table 6.9: Solvers for the $p$-Laplacian problem. Solver denotes outer solver, LPC denotes the linear PC when applicable, NPC denotes the nonlinear PC when applicable, Side denotes the type of preconditioning, SubPC denotes the linear solver at the block level, and LS denotes outer line search.

| Name | Solver | LPC | NPC | Side | SubPC | LS |
|---|---|---|---|---|---|---|
| NEWT\K − ASM | NEWT\K | ASM | – | – | LU | BT |
| QN | QN | – | – | – | | CP |
| RAS | RAS | – | – | – | LU | CP |
| RAS + (NEWT\K − ASM) | RAS/NEWT\K | ASM | – | – | LU | BT |
| RAS ∗ (NEWT\K − ASM) | RAS/NEWT\K | ASM | – | – | LU | BT |
| ASPIN | NEWT\K | – | NASM | L | LU | BT |
| NRICH $-_L$ (RAS) | NRICH | – | RAS | L | LU | CP |
| QN $-_L$ (RAS) | QN | – | RAS | L | LU | CP |

We concentrate on additive Schwarz and QN methods, as listed in Table 6.9. We will show that it is a combination with distinct advantages. This problem has difficult local nonlinearity that may stifle global solvers, so the local solvers should be an efficient remedy. We also consider a composite combination of Newton's method with RAS and nonlinear preconditioning of Newton's method in the form of ASPIN. NASM, RAS, and linear ASM all have single subdomains on each processor that overlap each other by 6 grid points.

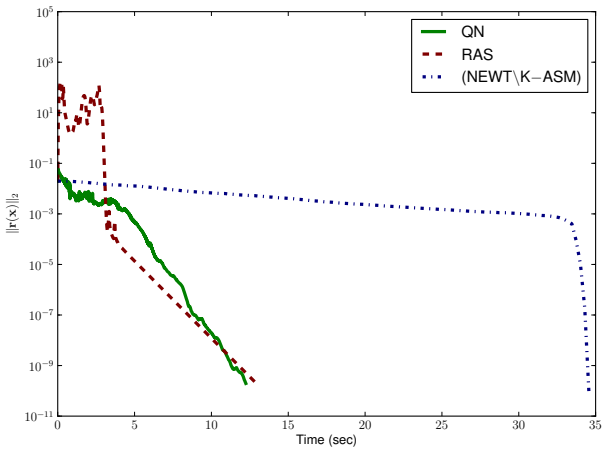


Fig. 6.10: QN, RAS, and (NEWT\K − ASM) convergence.

Table 6.10: QN, RAS, and (NEWT\K − ASM) results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| QN | 12.24 | 2960 | 0 | 5921 | – | – | – |
| RAS | 12.94 | 352 | 0 | 1090 | 352 | 352 | – |
| (NEWT\K − ASM) | 34.57 | 124 | 3447 | 423 | 124 | 3574 | – |

Table 6.10 contains results for the uncomposed solvers. The default solver, NEWT\K − ASM, takes a large number of outer Newton's method iterations and inner GMRES − ASM iterations. Unpreconditioned QN is able to converge to the solution efficiently and will be hard to beat. QN stores up to 10 previous solutions and residuals. RAS set to do one local Newton's method iteration per subdomain per outer iteration proves to be efficient on its own after some initial problems, as shown in Fig. 6.10.

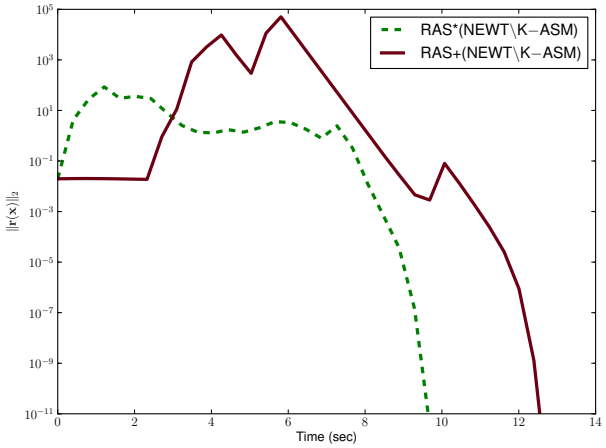


Fig. 6.11: RAS $*$ (NEWT\K $-$ ASM) and RAS $+$ (NEWT\K $-$ ASM) convergence.

Table 6.11: RAS $*$ (NEWT\K $-$ ASM) and RAS $+$ (NEWT\K $-$ ASM) results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| RAS $*$ (NEWT\K $-$ ASM) | 9.69 | 24 | 750 | 142 | 48 | 811 | – |
| RAS $+$ (NEWT\K $-$ ASM) | 12.78 | 33 | 951 | 232 | 66 | 1023 | – |

In Table 6.11 and Fig. 6.11, we show how Newton's method may be dramatically improved by composite combination with RAS and NASM. The additive composite combination reduces the number of outer iterations substantially. The multiplicative composite combination is even more effective, reducing the number of outer iterations by a factor of 5 and decreasing run-time by a factor of around 4.

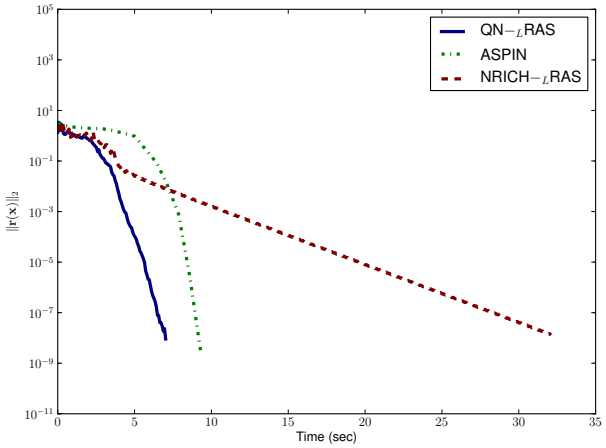


Fig. 6.12: QN $-_L$ RAS , ASPIN, and NRICH $-_L$ RAS convergence.

Table 6.12: QN $-_L$ RAS , ASPIN, and NRICH $-_L$ RAS results.

| Solver | T | N. It | L. It | Func | Jac | PC | NPC |
|---|---|---|---|---|---|---|---|
| QN $-_L$ RAS | 7.02 | 92 | 0 | 410 | 185 | 185 | 185 |
| ASPIN | 9.30 | 13 | 332 | 307 | 179 | 837 | 19 |
| NRICH $-_L$ RAS | 32.10 | 308 | 0 | 1889 | 924 | 924 | 924 |

The results for left-preconditioned methods are shown in Table 6.12 and Fig. 6.12. ASPIN is competitive, taking 13 iterations and having similar performance characteristics to the multiplicative composite combination solver listed in Table 6.11. The subdomain solvers for ASPIN are set to converge to a relative tolerance of $10^{-3}$ or 20 inner iterations. Underresolving the local problems provides an inadequate search direction and causes ASPIN to stagnate. The linear GMRES iteration also is converged to $10^{-3}$.

The most impressive improvements can be achieved by weakening RAS, as demonstrated above, and using it as a nonlinear preconditioner. Simple NRICH acceleration does not provide much benefit compared with raw RAS with respect to outer iterations, and the preconditioner applications in the line search cause significant overhead. However, QN using weakened RAS as the left-preconditioned function proves to be the most efficient solver for this problem, taking 185 Newton's method steps per subdomain. Both NRICH and QN stagnate if the original function is used in the line search instead of the preconditioned one. Subdomain QN methods [41] have been proposed before but built by using block approximate Jacobian inverses instead of working on the preconditioned system like ASPIN. As implemented here, QN $-_L$ RAS and ASPIN both construct left-preconditioned residuals and approximate preconditioned Jacobian constructions; both end up being very efficient.

**7. Conclusion.** The combination of solvers using nonlinear composition, when applied carefully, may greatly improve the convergence properties of nonlinear solvers. Hierarchical and multilevel inner solvers allow for high arithmetic intensity and low communication algorithms, making nonlinear composition a good option for extreme-scale nonlinear solvers.

Our experimentation, both in this document and others, has shown that what works best when using nonlinear composition varies from problem to problem. Nonlinear composition introduces a slew of additional solver parameters at multiple levels of the hierarchy that may be tuned for optimal performance and robustness. A particular problem may be amenable to a simple solver, such as NCG, or to a combination of multilevel solvers, such as FAS $*$ (NEWT\K $-$ MG). The implementation in PETSc [3], as described in Appendix A, allows for a great deal of flexibility in user choice of solver compositions.

Admittedly, we have been fairly conservative in the scope of our solver combinations. Our almost-complete restriction to combinations of a globalized method and a decomposition method is somewhat artificial in the nonlinear case. Without this restriction, however, the combinatorial explosion of potential methods would quickly make the scope of this paper untenable. Users may experiment, for their particular problem, with combinations of some number of iterations of arbitrary combinations of nonlinear solvers, with nesting much deeper than we explore here.

The use of nonlinear preconditioning allows for solvers that are more robust to difficult nonlinear problems. While effective linear preconditioners applied to the Jacobian inversion problem may speed the convergence of the inner solves, lack of the convergence of the outer Newton's method iteration may doom the solve. With nonlinear preconditioning, the inner and outer treatment of the nonlinear problem allows for very rapid solution. Nonlinear preconditioning and composite combination solvers allow for both efficiency and robustness gains, and the widespread adoption of these techniques would reap major benefits for computational science.

**Acknowledgments.** The authors were supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. We thank Jed Brown for many meaningful discussions and suggestions.

**Appendix A. Implementation in PETSc.** The solvers in this work are available in the current release of PETSc. Like all the other solvers in PETSc, the composable solvers can be customized from the command line. Any nonlinear solver may be composed with another, including smoothers for FAS and subproblem solvers for NASM. In contrast to linear preconditioning, all the solvers are of the nonlinear

solver type `SNES` rather than differentiating between solvers (`KSP`) and preconditioners (`PC`) as on the linear side. As with the linear solvers, subsolvers may be set up recursively, with option prefixes inherited down the chain of solvers as they are built. The options in Table A.1 allow for the basic construction of nonlinear solvers from the PETSc command line.

Table A.1: Options for the composable solvers available in PETSc

| Command Line Option | | Description |
|---|---|---|
| `-snes_type` | `newtonls` | Outer nonlinear solver type |
| | `composite` | |
| | `ngmres` | |
| | `ncg` | |
| | `qn` | |
| | `fas` | |
| | `nasm` | |
| | `nrichardson` | |
| | `aspin` | |
| `-snes_npc_side` | `right` | Nonlinear left or right-preconditioning |
| | `left` | |
| `-npc_snes_type` | see `-snes_type` | Type of nonlinear preconditioner |
| `-snes_function_type` | `preconditioned` | Preconditioned function for convergence test |
| | `unpreconditioned` | Original function for convergence test |
| `-snes_linesearch_type` | `bt` | Type of line search |
| | `l2` | |
| | `cp` | |
| | `basic` | Simple damping |
| `-snes_linesearch_max_it` | | Max. number of line search iterations |
| `-snes_linesearch_order` | | Polynomial order of the line search where appropriate |

The relevant options for controlling `-snes_type fas`, `-snes_type nasm`, and `-snes_type composite` are listed in Table A.2.

Table A.2: Options for the `composite` solver

| Command Line Option | | Description |
|---|---|---|
| `-snes_fas_levels` | | Number of levels used for FAS |
| `-snes_fas_type` | `multiplicative` | Multiplicative FAS |
| | `additive` | Additive FAS |
| `-snes_fas_cycles` | 1 for V, 2 for W, etc. | Number of cycles |
| `-fas_levels_...` | | Control of level SNES, KSP, PC, etc. |
| `-fas_coarse_...` | | Control of coarse SNES, KSP, PC, etc. |
| `-snes_nasm_type` | `basic` | NASM update with full overlap |
| | `restrict` | RAS update |
| `-snes_nasm_damping` | | Damping for the subdomain updates |
| `-sub_...` | | Control of subdomain SNES, KSP, PC, etc. |
| `-snes_composite_type` | `additive` | Additive composite combination of the solvers |
| | `multiplicative` | Multiplicative composite combination of the solvers |
| | `additiveoptimal` | Residual-minimizing additive composite combination |
| `-snes_composite_sneses` | any `snes` types | Comma-separated list of nonlinear solvers |
| `-sub_`$n$`_...` | | Control over subsolver $n$'s SNES, KSP, PC, etc. |

Not all the methods may be used in a black-box fashion, some requiring additional user input. To use nonlinear Gauss-Seidel, one must provide a mechanism for solving the local block problem, the efficiency of

which depends heavily on the choice of spatial discretization. However, a general user interface is provided for writing such routines. FAS requires a hierarchy of domains and NASM a set of subdomains. These may be user-provided but can be created automatically when using the PETSc structured grid class `DMDA`.

As an example of the type of tuning process required for using these solvers effectively, we return to the problem introduced in Section 6.3. For Grashof number $\mathrm{Gr} < 10^4$ and Prandtl number $\mathrm{Pr} = 1.0$, Newton's method converges well:

```
./ex19 -lidvelocity 100 -grashof 1e4 -da_refine 4 -pc_type lu -snes_monitor_short -snes_converged_reason

lid velocity = 100, prandtl # = 1, grashof # = 10000
  0 SNES Function norm 715.271
  1 SNES Function norm 623.41
  2 SNES Function norm 510.225
  3 SNES Function norm 382.172
  4 SNES Function norm 375.414
  5 SNES Function norm 10.634
  6 SNES Function norm 0.269178
  7 SNES Function norm 0.00110921
  8 SNES Function norm 1.09139e-09
Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 8
```

For higher Grashof number, Newton's method stagnates

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

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
            .
            .
 28 SNES Function norm 585.142
 29 SNES Function norm 585.142
 30 SNES Function norm 585.142
```

It also fails with the standard continuation strategy from coarser meshes, obtained by using `-snes_grid_sequence`. We next try nonlinear multigrid, in the hope that multiple updates from a coarse solution are sufficient, using GS as the nonlinear smoother,

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
  -snes_type fas -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
Number of SNES iterations = 4
```

But inner iteration fails, likely on the coarse grid. We can turn on monitoring of the coarse solve.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type fas -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 iterations 12
  1 SNES Function norm 574.793
         Nonlinear solve did not converge due to DIVERGED_MAX_IT iterations 50
  2 SNES Function norm 513.02
         Nonlinear solve did not converge due to DIVERGED_MAX_IT iterations 50
  3 SNES Function norm 216.721
         Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 22
  4 SNES Function norm 85.949
         Nonlinear solve did not converge due to DIVERGED_LINE_SEARCH iterations 42
Nonlinear solve did not converge due to DIVERGED_INNER iterations 4
Number of SNES iterations = 4
```

We discover that the line search is the culprit. Turning off the line search, we again see stagnation of the

FAS iteration.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type fas -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 iterations 6
  1 SNES Function norm 574.793
          Nonlinear solve did not converge due to DIVERGED_MAX_IT iterations 50
  2 SNES Function norm 1310.72
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 12
  3 SNES Function norm 1279.59
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 18
  .
  .
  .
 40 SNES Function norm 72.966
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 6
 41 SNES Function norm 78.7599
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 5
 42 SNES Function norm 73.0625
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 6
 43 SNES Function norm 78.8552
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 5
 44 SNES Function norm 73.158
          Nonlinear solve converged due to CONVERGED_FNORM_RELATIVE iterations 6
```

A nonlinear accelerator, NRICH, can be added to the FAS in the same way that linear multigrid is accelerated with a Krylov method.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type nrichardson -npc_snes_max_it 1 -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
```

We finally see convergence. The convergence is more rapid using NGMRES.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type ngmres -npc_snes_max_it 1 -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
```

And can be restored to a few iterates by increasing the power of the nonlinear smoothers in FAS.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type ngmres -npc_snes_max_it 1 -npc_snes_type fas
   -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_coarse_snes_linesearch_type basic

lid velocity = 100, prandtl # = 1, grashof # = 50000
  0 SNES Function norm 1228.95
  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
Number of SNES iterations = 5
```

One may also choose composite combination instead of preconditioning. An additive combination of single iterations of FAS and Newton's method may be used to effectively separate the regimes in which FAS converges well from the basin of convergence and choose the optimal combination of the two iteration per iteration.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type composite -snes_composite_type additiveoptimal -snes_composite_sneses fas,newtonls
   -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
Number of SNES iterations = 7
```

The composite combination may also be done multiplicatively, in which the weaker FAS iteration nudges the Newton's method close to the basin of convergence.

```
./ex19 -lidvelocity 100 -grashof 5e4 -da_refine 4 -snes_monitor_short -snes_converged_reason
 -snes_type composite -snes_composite_type multiplicative -snes_composite_sneses fas,newtonls
   -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
Number of SNES iterations = 5
```

This kind of optimization can be carried out for difficult nonlinear systems, in much the same the way we approach linear systems, albeit with fewer analytical tools.

## REFERENCES


[1] Advanced Micro Devices, *AMD Opteron 6200 series quick reference guide*, 2012.

[2] D. G. Anderson, *Iterative procedures for nonlinear integral equations*, Journal of the ACM, 12 (1965), pp. 547–560.

[3] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 3.4, Argonne National Laboratory,

2013.
[4] P. Birken and A. Jameson, *On nonlinear preconditioners in Newton-Krylov methods for unsteady flows*, International Journal for Numerical Methods in Fluids, 62 (2010), pp. 565–573.
[5] J. H. Bramble, *Multigrid Methods*, Longman Scientific and Technical, Essex, England, 1993.
[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Mathematics of computation, 31 (1977), pp. 333–390.
[7] A. Brandt, *Multigrid techniques: 1984 guide with applications for fluid dynamics*, Tech. Rep. GMD-Studien Nr. 85, Gesellschaft fur Mathematik und Dataenverarbeitung, 1984.
[8] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization and VLSI-CAD, Kluwer Academic Publishers, 2003, pp. 1–69.
[9] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial (2nd ed.)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.
[10] J. Brown and P. Brune, *Low-rank quasi-Newton updates for robust Jacobian lagging in Newton-type methods*, in International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, 2013, pp. 2554–2565.
[11] P. Brown and A. Hindmarsh, *Matrix-free methods for stiff systems of ODEs*, SIAM Journal on Numerical Analysis, 23 (1986), pp. 610–638.
[12] X.-C. Cai, *Nonlinear overlapping domain decomposition methods*, Lecture Notes in Computational Science, 70 (2009), pp. 217–224.
[13] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.
[14] X.-C. Cai and X. Li, *Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity*, SIAM Journal on Scientific Computing, 33 (2011), pp. 746–762.
[15] N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code I: In one dimension*, SIAM Journal on Scientific Computing, 19 (1998), pp. 728–765.
[16] T. Chan and K. Jackson, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM Journal on Scientific and Statistical Computing, 5 (1984), pp. 533–542.
[17] P. Cresta, O. Allix, C. Rey, and S. Guinard, *Nonlinear localization strategies for domain decomposition methods: Application to post-buckling analyses*, Computer Methods in Applied Mechanics and Engineering, 196 (2007), pp. 1436–1446. Domain Decomposition Methods: recent advances and new challenges in engineering.
[18] Y. H. Dai and Y. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM Journal on Optimization, 10 (1999), pp. 177–182.
[19] H. De Sterck, *Steepest descent preconditioning for nonlinear GMRES optimization*, Numerical Linear Algebra with Applications, 20 (2013), pp. 453–471.
[20] R. Dembo, S. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.
[21] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[22] V. Eyert, *A comparative study on methods for convergence acceleration of iterative vector sequences*, J. Comput. Phys., 124 (1996), pp. 271–285.
[23] H.-r. Fang and Y. Saad, *Two classes of multisecant methods for nonlinear acceleration*, Numer. Linear Algebra Appl., 16 (2009), pp. 197–221.
[24] R. Fletcher, *Practical Methods of Optimization, Volume 1*, Wiley, 1987.
[25] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, Computer Journal, 7 (1964), pp. 149–154.
[26] A. A. Goldstein, *On steepest descent*, Journal of the Society for Industrial & Applied Mathematics, Series A: Control, 3 (1965), pp. 147–151.
[27] W. Hackbusch, *Comparison of different multi-grid variants for nonlinear equations*, ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift fr Angewandte Mathematik und Mechanik, 72 (1992), pp. 148–151.
[28] V. E. Henson, *Multigrid methods for nonlinear problems: an overview*, Proc. SPIE, 5016 (2003), pp. 36–48.
[29] M. R. Hestenes and E. Steifel, *Methods of conjugate gradients for solving linear systems*, J. Research of the National Bureau of Standards, 49 (1952), pp. 409–436.
[30] F.-N. Hwang and X.-C. Cai, *A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations*, J. Comput. Phys., 204 (2005), pp. 666–691.
[31] J. E. D. Jr. and J. J. More, *Quasi-Newton methods, motivation and theory*, SIAM Review, 19 (1977), pp. 46–89.
[32] C. Kelley and D. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Numerical Analysis, 35 (1998), pp. 508–523.
[33] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
[34] H. Klie and M. F. Wheeler, *Nonlinear Krylov-secant solvers*, tech. rep., Dept. of Mathematics and inst. of Computational Engineering and Sciences, University of Texas at Austin, 2006.
[35] D. Knoll and P. McHugh, *Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow*, SIAM Journal on Scientific Computing, 19 (1998), pp. 291–301.
[36] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comp. Phys., 193 (2004), pp. 357–397.
[37] P. Ladevèze, J.-C. Passieux, and D. Nron, *The LATIN multiscale computational method and the proper generalized decomposition*, Computer Methods in Applied Mechanics and Engineering, 199 (2010), pp. 1287 – 1296. Multiscale Models and Mathematical Aspects in Solid and Fluid Mechanics.
[38] P. Ladevèze and J. Simmonds, *Nonlinear computational structural mechanics: New approaches and non-incremental*

*methods of calculation*, Mechanical Engineering Series, Springer, 1999.
[39] P. J. Lanzkron, D. J. Rose, and J. T. Wilkes, *An analysis of approximate nonlinear elimination*, SIAM Journal on Scientific Computing, 17 (1996), pp. 538–559.
[40] P. Lott, H. Walker, C. Woodward, and U. Yang, *An accelerated Picard method for nonlinear systems related to variably saturated flow*, Advances in Water Resources, 38 (2012), pp. 92–101.
[41] J. M. Martínez, *SOR-secant methods*, SIAM J. Numer. Anal., 31 (1994), pp. 217–226.
[42] S. G. Nash, *A multigrid approach to discretized optimization problems*, Optimization Methods and Software, 14 (2000), pp. 99–116.
[43] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Mathematics of Computation, 35 (1980), pp. 773–782.
[44] E. Polak and G. Ribiere, *Note sur la convergence de méthodes de directions conjuguées*, Revue franaise dinformatique et de recherche opérationnelle, série rouge, 3 (1969), pp. 35–43.
[45] P. Pulay, *Convergence acceleration of iterative sequences: The case of SCF iteration*, Chemical Physics Letters, 73 (1980), pp. 393–398.
[46] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford Science Publications, Oxford, 1999.
[47] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific Computing, 14 (1993), pp. 461–469.
[48] Y. Saad, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelpha, PA, 2003.
[49] M. H. Scott and G. L. Fenves, *A Krylov subspace accelerated Newton algorithm*, in Proc., 2003 ASCE Structures Congress, 2003.
[50] D. F. Shanno, *Conditioning of quasi-Newton methods for function minimization*, Mathematics of Computation, 24 (1970), pp. pp. 647–656.
[51] J. R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, tech. rep., Carnegie Mellon University, Pittsburgh, PA, USA, 1994.
[52] B. Smith, *Domain decomposition methods for partial differential equations*, ICASE LARC Interdisciplinary Series in Science and Engineering, 4 (1997), pp. 225–244.
[53] B. F. Smith, P. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.
[54] B. F. Smith and X. Tu, *Encyclopedia of Applied and Computational Mathematics*, Springer, 2013, ch. Domain Decomposition.
[55] M. Smooke and R. Mattheij, *On the solution of nonlinear two-point boundary value problems on successively refined grids*, Applied Numerical Mathematics, 1 (1985), pp. 463 – 487.
[56] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, vol. 34, Springer, 2005.
[57] A. Toth and C. T. Kelley, *Convergence analysis for anderson acceleration*, Submitted, (2013).
[58] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2001.
[59] R. S. Tuminaro, H. F. Walker, and J. N. Shadid, *On backtracking failure in Newton–GMRES methods with a demonstration for the Navier–Stokes equations*, Journal of Computational Physics, 180 (2002), pp. 549–558.
[60] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM J. Numer. Anal., 49 (2011), pp. 1715–1735.
[61] H. F. Walker, C. S. Woodward, and U. M. Yang, *An accelerated fixed-point iteration for solution of variably saturated flow*, in Proc. XVIII International Conference on Water Resources, Barcelona, 2010.
[62] T. Washio and C. Oosterlee, *Krylov subspace acceleration for nonlinear multigrid schemes*, Electronic Transactions on Numerical Analysis, 6 (1997), pp. 271–290.
[63] T. Washio and C. W. Oosterlee, *Krylov subspace acceleration for nonlinear multigrid schemes with application to recirculating flow*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1670–1690.
[64] P. Wesseling, *An Introduction to Multigrid Methods*, R. T. Edwards, 2004.
[65] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Review, 11 (1969), pp. 226–235.
[66] P. Wriggers, *Nonlinear Finite Element Methods*, Springer, 2008.