PRECONDITIONED NONLINEAR ITERATIONS FOR OVERLAPPING CHEBYSHEV DISCRETIZATIONS WITH INDEPENDENT GRIDS[*]

KEVIN W. AITON[†] AND TOBIN A. DRISCOLL[†]

Abstract. The additive Schwarz method is usually presented as a preconditioner for a PDE linearization based on overlapping subsets of nodes from a global discretization. It has previously been shown how to apply Schwarz preconditioning to a nonlinear problem. By first replacing the original global PDE with the Schwarz overlapping problem, the global discretization becomes a simple union of subdomain discretizations, and unknowns do not need to be shared. In this way, restrictive-type updates can be avoided, and subdomains need to communicate only via interface interpolations. The resulting preconditioner can be applied linearly or nonlinearly. In the latter case, nonlinear subdomain problems are solved independently in parallel, and the frequency and amount of interprocess communication can be greatly reduced compared to global preconditioning of the sequence of linearized problems.

Key words. partition of unity, polynomial interpolation, domain decomposition, additive Schwarz

AMS subject classifications. 65N55, 33F05, 97N40

DOI. 10.1137/19M1242483

1. Introduction. Overlapping domain decomposition has been a valuable aid in solving PDEs since Schwarz first described his alternating method in 1870. (For technical introductions to the topic, see [8, 13]; for a more historical perspective, see [10].) Overlapping decomposition provides a way to solve a problem on a global domain by exploiting its reduction to smaller subdomains. This creates geometric flexibility and allows special effort to be focused on small parts of the domain when appropriate. Domain decomposition also affords an attractive natural parallelism.

For a linear PDE, one typically seeks to apply a preconditioner for a Krylov iteration such as GMRES in the form of solving problems on overlapping subdomains whose boundary data is in part determined by values of the solution in other subdomains. In the parallel context, this is achieved by an additive Schwarz (AS) scheme. When one partitions the unknowns of a global discretization into overlapping subsets, the best forms of AS are restricted AS (RAS) methods [5], which do not allow multiple domains to update shared unknowns independently and thus overcorrect. Typically, then, the subdomain problems are solved on overlapping sets, but the results are distributed in a nonoverlapping fashion.

For nonlinear problems, the obvious extension of AS preconditioning is to apply it as described above on the linearized equations that are determined by a quasi-Newton iteration. This process constitutes a *Newton–Krylov–Schwarz* (NKS) procedure [3], reflecting the nesting order of the different elements of linearization, linear solver, and preconditioning.

[†]Department of Mathematical Sciences, University of Delaware, Newark, DE 19716 (kaiton@udel.edu, driscoll@udel.edu).

Cai and Keyes [4] proposed instead modifying the *nonlinear* problem using the Schwarz ansatz. In addition to yielding a preconditioned linearization for the Krylov solver, the preconditioned nonlinear problem exhibited more robust convergence for the Newton iteration than did the original nonlinear problem. They called their method ASPIN, short for *additive Schwarz preconditioned inexact Newton*. As a technical matter, they did not recommend applying the true Jacobian of the system, preferring an approximation that required less effort. Subsequently, Dolean et al. [7] pointed out that Cai and Keyes did not use the RAS form of AS preconditioning, and they proposed an improved variant called RASPEN that does. We refer to this type of nonlinear preconditioning as *Schwarz–Newton–Krylov* (SNK), because the Schwarz ansatz is applied before the linearization begins.

Our interest is in applying the nonlinear preconditioning technique to spectral collocation discretizations in overlapping rectangles or cuboids, leading to globally smooth approximations constructed from a partition of unity [1]. In this context, there is not naturally a single global discretization whose degrees of freedom are partitioned into overlapping sets, because the Chebyshev (or Legendre, or other classical) nodes will not generally coincide within the overlapping regions. In principle, one could link the degrees of freedom within overlap regions by interpolating between subdomains, but this process adds complication, computational time, and, in the parallel context, communication of data within the overlap regions.

Here we present an alternative strategy that begins by replacing the original PDE problem with the Schwarz problems on the union of the subdomains. That is, rather than regarding the subdomains as solving the global PDE on a region that includes portions shared with other subdomains, each subdomain has a "private copy" of its entire region and is free to have its own solution values throughout. Of course, the new global problem is not solved until the interface values of every subdomain agree with values interpolated from other subdomains that contain the interface. As a Schwarz starting point, our technique has both NKS and SNK variants.

In this formulation, interpolations need to be done only on lower-dimensional interfaces, rather than throughout the overlap regions. Another trait is that plain AS becomes preferred to RAS, because each subdomain has to update its own values separately. We show that it is straightforward to implement exact Jacobians for SNK with nothing more than the ability to do fully local nonlinear and linearized PDE solves, plus the ability to transfer values between subdomains through interface interpolations. We also derive a two-level method to prevent convergence degradation as the number of subdomains increases. The performance of the NKS and SNK methods is validated and compared through several numerical experiments.

2. PDE problem and multidomain formulation. The main goal of this work is to solve the PDE

$$(2.1a) \qquad \phi(\boldsymbol{x}, u) = 0, \qquad \boldsymbol{x} \in \Omega,$$
$$(2.1b) \qquad \beta(\boldsymbol{x}, u) = 0, \qquad \boldsymbol{x} \in \partial\Omega,$$

where $u(\boldsymbol{x})$ is the unknown solution and ϕ and β are nonlinear differential operators (with ϕ being of higher order). (We can easily extend to the case where u, ϕ, and β are vector-valued, but we use scalars to calm the notation.) Many Schwarz-based algorithms for (2.1) begin with a global discretization whose solution is accelerated by an overlapping domain decomposition. In this situation, some of the numerical degrees of freedom are shared by multiple subdomains—either directly or through interpolation—and proper use of additive Schwarz (AS) calls for the restricted AS

(RAS) implementation, which essentially ensures that updates of shared values are done only once from the global perspective, not independently by the subdomains.

We take a different approach, replacing the original problem (2.1) with

$$(2.2a) \qquad \phi(\boldsymbol{x}, u_i) = 0, \qquad \boldsymbol{x} \in \Omega_i, \qquad i = 1, \ldots, N,$$

$$(2.2b) \qquad \beta(\boldsymbol{x}, u_i) = 0, \qquad \boldsymbol{x} \in \Gamma_{i0} = \partial \Omega \cap \partial \Omega_i, \qquad i = 1, \ldots, N,$$

$$(2.2c) \qquad u_i = u_j, \qquad \boldsymbol{x} \in \Gamma_{ij} = \partial \Omega_i \cap \mathbb{Z}_j, \qquad i, j = 1, \ldots, N,$$

where now $u_1, \ldots, u_N$ are unknown functions on overlapping subdomains Ω_i that cover Ω, and the Z_i are nonoverlapping zones lying within the respective subdomains. Clearly, any strong solution of (2.1) is also a solution of (2.2), and while the converse is not necessarily so in principle, we regard the possibility of finding a solution of (2.2) that is not also a solution of (2.1) as remote in practice.

The key consequence of starting from (2.2) as the global problem is that each overlapping region is covered separately by the involved subdomains; each is free to update its representation independently in order to converge to a solution. From one point of view, our discretizations of the overlap regions are redundant and somewhat wasteful. However, the fraction of redundant discrete unknowns is very modest. In return, we only need to interpolate on the interfaces, there is no need to use the RAS formulation, and the coarsening needed for a two-level variant is trivial (see section 2.3).

2.1. Discretization. We now describe a collocation discretization of (2.2) for concreteness. Each subfunction $u_i(\boldsymbol{x})$ is discretized by a vector $\boldsymbol{u}_i$ of length n_i. By $\boldsymbol{u} = [\![\boldsymbol{u}_i]\!]$ we mean a concatenation of all the discrete unknowns over subdomains $i = 1, \ldots, N$ into a single vector. Subdomain Ω_i is discretized by a node set $X_i \subset \overline{\Omega}_i$ and a boundary node set $B_i \subset \partial \Omega_i$. The total cardinality of X_i and B_i together is also n_i. The boundary nodes are subdivided into nonintersecting sets $G_{ij} = B_i \cap Z_j$ for $j \neq i$, and $G_{i0} = B_i \cap \partial \Omega$.

For each i, the vector $\boldsymbol{u}_i$ defines a function $\tilde{u}_i(\boldsymbol{x})$ on Ω_i. These can be used to evaluate ϕ and β from (2.2) anywhere in Ω_i. We define an n_i-dimensional vector function $\boldsymbol{f}_i$ as the concatenation of three vectors:

$$(2.3) \qquad \boldsymbol{f}_i(\boldsymbol{u}_i) = \begin{cases} \phi(\boldsymbol{x}, \tilde{u}_i) & \text{for all } \boldsymbol{x} \in X_i, \\ \beta(\boldsymbol{x}, \tilde{u}_i) & \text{for all } \boldsymbol{x} \in G_{i0}, \\ \tilde{u}_i(\boldsymbol{x}) & \text{for all } \boldsymbol{x} \in G_{ij}, \ j = 1, \ldots, i-1, i+1, \ldots, N. \end{cases}$$

In addition, we have the linear *transfer operator* $\boldsymbol{T}_i$ defined by

$$(2.4) \qquad \boldsymbol{T}_i \boldsymbol{u} = \begin{cases} 0 & \text{for all } \boldsymbol{x} \in X_i, \\ 0 & \text{for all } \boldsymbol{x} \in G_{i0}, \\ \tilde{u}_j(\boldsymbol{x}) & \text{for all } \boldsymbol{x} \in G_{ij}, \ j = 1, \ldots, i-1, i+1, \ldots, N. \end{cases}$$

Note that while $\boldsymbol{f}_i$ is purely local to subdomain i, the transfer operator $\boldsymbol{T}_i$ operates on the complete discretization $\boldsymbol{u}$, as it interpolates from "foreign" subdomains onto the parts of B_i lying inside Ω. Finally, we are able to express the complete discretization of (2.2) through concatenations over the subdomains. Let $\boldsymbol{u} = [\![\boldsymbol{u}_i]\!]$, $\boldsymbol{f}(\boldsymbol{u}) = [\![\boldsymbol{f}_i(\boldsymbol{u}_i)]\!]$, and $\boldsymbol{T}\boldsymbol{u} = [\![\boldsymbol{T}_i \boldsymbol{u}]\!]$. Then the discrete form of (2.2) is the nonlinear equation

$$(2.5) \qquad\qquad \boldsymbol{f}(\boldsymbol{u}) - \boldsymbol{T}\boldsymbol{u} = \boldsymbol{0}.$$

For a square discretization, the goal is to solve (2.5), while in the least-squares case, the goal is to minimize $\boldsymbol{f}(\boldsymbol{u}) - \boldsymbol{T}\boldsymbol{u}$ in the (possibly weighted) 2-norm.

2.2. Newton–Krylov–Schwarz. The standard approach to (2.5) for a large discretization is to apply an inexact Newton iteration with a Krylov subspace solver for finding correcting steps from the linearization. Within the Krylov solver we have a natural setting for applying an AS preconditioner. Specifically, if we have a proposed approximate solution $\boldsymbol{u}$, we evaluate the nonlinear residual $\boldsymbol{r} = \boldsymbol{f}(\boldsymbol{u}) - \boldsymbol{T}\boldsymbol{u}$. We then (inexactly, perhaps) solve the linearization $\big[\boldsymbol{f}'(\boldsymbol{u}) - \boldsymbol{T}\boldsymbol{u}\big]\boldsymbol{s} = -\boldsymbol{r}$ for the Newton correction $\boldsymbol{s}$ using a Krylov solver such as GMRES. These iterations are preconditioned by the block diagonal matrix $\boldsymbol{f}'(\boldsymbol{u})$, which is simply the block diagonal of the subdomain Jacobians $\boldsymbol{f}'_i(\boldsymbol{u}_i)$. We refer to this method as *Newton–Krylov–Schwarz*, or NKS.

Implementation of NKS requires three major elements: the evaluations of $\boldsymbol{f}(\boldsymbol{u})$ and $\boldsymbol{T}\boldsymbol{u}$ for given $\boldsymbol{u}$, the application of the Jacobian $\boldsymbol{f}'(\boldsymbol{u})$ to a given vector $\boldsymbol{v}$, and the inversion of $\boldsymbol{f}'(\boldsymbol{u})$ for given data. All of the processes involving $\boldsymbol{f}$ are embarrassingly parallel and correspond to standard steps in solving the PDE on the local subdomains. Each application of the transfer operator $\boldsymbol{T}$, however, requires a communication from each subdomain to its overlapping neighbors, as outlined in Algorithm 2.1. This step occurs once in evaluating the nonlinear residual and in every GMRES iteration to apply the Jacobian. In a parallel code, the communication steps could be expected to be a major factor in the performance of the method.

Algorithm 2.1. Apply transfer operator, $\boldsymbol{T}\boldsymbol{u}$.

Interpret input $\boldsymbol{u}$ as concatenated $[\![\boldsymbol{u}_i]\!]$.
for $j = 1, \ldots, N$ (in parallel) **do**
  for all neighboring subdomains i **do**
    Evaluate $\tilde{u}_j$ at $\boldsymbol{x} \in G_{ij}$.
  end for
end for

2.3. Two-level scheme. As is well known [8], AS schemes should incorporate a coarse solution step in order to maintain convergence rates as the number of subdomains increases. The methods described above depend on the subdomain discretization sizes n_i of the collocation nodes and solution representation, respectively. Now suppose we decrease the discretization sizes to $\hat{n}_i$ and denote the corresponding discretizations of (2.5) by $\hat{\boldsymbol{f}}(\hat{\boldsymbol{u}}) - \hat{\boldsymbol{T}}\hat{\boldsymbol{u}} = \boldsymbol{0}$. We can define a restriction operator $\boldsymbol{R}$ that maps fine-scale vectors to their coarse counterparts. This operator is block diagonal; i.e., it can be applied independently within the subdomains. We can also construct a block diagonal prolongation operator $\boldsymbol{P}$ for mapping the solution representation from coarse to fine scales.

We are then able to apply the standard full approximation scheme (FAS) using the coarsened problem [2]. Specifically, we solve the coarse problem

$$(2.6) \qquad \hat{\boldsymbol{f}}(\hat{\boldsymbol{e}} + \boldsymbol{R}\boldsymbol{u}) - \hat{\boldsymbol{T}}\hat{\boldsymbol{e}} - \hat{\boldsymbol{f}}(\boldsymbol{R}\boldsymbol{u}) + \boldsymbol{R}(\boldsymbol{f}(\boldsymbol{u}) - \boldsymbol{T}\boldsymbol{u}) = \boldsymbol{0}$$

for the coarse correction $\hat{\boldsymbol{e}}$ and define $\boldsymbol{c}(\boldsymbol{u}) = \boldsymbol{P}\hat{\boldsymbol{e}}$ as the FAS corrector at the fine level. The procedure for calculating $\boldsymbol{c}$ is outlined in Algorithm 2.2.

We also require the action of the Jacobian $\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}} = \boldsymbol{P}\frac{\partial \hat{\boldsymbol{e}}}{\partial \boldsymbol{u}}$ on a given vector $\boldsymbol{v}$. It is straightforward to derive from (2.6) that

$$(2.7) \qquad \big[\hat{\boldsymbol{f}}'(\hat{\boldsymbol{e}} + \boldsymbol{R}\boldsymbol{u}) - \hat{\boldsymbol{T}}\big]\frac{\partial \hat{\boldsymbol{e}}}{\partial \boldsymbol{u}} = -\big(\hat{\boldsymbol{f}}'(\hat{\boldsymbol{e}} + \boldsymbol{R}\boldsymbol{u}) - \hat{\boldsymbol{f}}'(\boldsymbol{R}\boldsymbol{u})\big)\boldsymbol{R} - \boldsymbol{R}\big(\boldsymbol{f}'(\boldsymbol{u}) - \boldsymbol{T}\big).$$

Algorithm 2.2. Evaluate FAS correction $c(u)$.

Apply Algorithm 2.1 to compute Tu.
Compute (in parallel) $\hat{u} = Ru$.
Compute (in parallel) $\hat{r} = R(f(u) - Tu) - \hat{f}(\hat{u})$.
Solve (2.6) for $\hat{e}$.
Compute (in parallel) the prolongation $P\hat{e}$.

Algorithm 2.3. Apply Jacobian $c'(u)$ for the FAS corrector to a vector v.

Apply Algorithm 2.2 to compute $\hat{e}$, $\hat{u}$, and the final value of $\hat{A} = \hat{f}'(\hat{e} + \hat{u})$.
Apply Algorithm 2.1 to compute Tv.
Set (in parallel) $\hat{r} = R(f'(u)v - Tv)$ and $\hat{v} = Rv$.
Set (in parallel) $\hat{b} = \hat{A}\hat{v} - \hat{f}'(\hat{u})\hat{v} + \hat{r}$.
Solve the linear system $(\hat{A} - \hat{T})\hat{y} = -\hat{b}$ for $\hat{y}$.
Compute (in parallel) $P\hat{y}$.

Note that the matrix $\hat{f}'(\hat{e} + Ru)$ should be available at no extra cost from the end of the Newton solution of (2.6). Algorithm 2.3 describes the corresponding algorithm for computing the application of $c'(u)$ to any vector v. Even though c' is of the size of the fine discretization, the computation requires only coarse-dimension dense linear algebra.

Finally, we describe how to combine coarsening with the preconditioned fine scale into a two-level algorithm. If we were to alternate coarse and fine corrections in the classical fixed-point form,

$$u^{\dagger} = u + c(u),$$
$$u^{\text{new}} = u^{\dagger} + f(u^{\dagger}) - Tu^{\dagger},$$

then we are effectively seeking a root of

$$(2.8) \qquad\qquad h(u) := c(u) + (f - T)(u + c(u)).$$

The Jacobian of the combined map is straightforwardly

$$(2.9) \qquad\qquad h'(u) = c'(u) + (f' - T)(u + c(u)) \cdot (I + c'(u)).$$

Thus, the action of h' on a vector can be calculated using the algorithms for c', f', and T.

3. Preconditioned nonlinear iterations. As shown in section 2.2, the inner Krylov iterations of the NKS method are governed by the preconditioned Jacobian $I - [f'(v)]^{-1}T$. Following the observation of Cai and Keyes [4], we next derive a method that applies Krylov iterations to the same matrix, but arising as the natural result of preconditioning the *nonlinear* problem. Specifically, we precondition (2.5) by finding a root of the nonlinear operator

$$(3.1) \qquad\qquad g(u) := u - f^{-1}(Tu).$$

Evaluation of g is feasible, because of the block diagonal (that is, fully subdomain-local) action of the nonlinear f. Since we are therefore applying the Schwarz ansatz

even before linearizing the problem, we refer to the resulting method as *Schwarz–Newton–Krylov* (SNK).

We have several motivations for a method based on (3.1). First, since the transfer operator $\boldsymbol{T}$ has nonzero rows only at interface points, the nonlinear problem is a low-rank perturbation of the identity operator (see (3.3)), which may make Newton stepping much more effective than in the original nonlinear form. Second, the inversion of $\boldsymbol{f}$ means solving independent nonlinear problems on the subdomains with no communication, which well exploits parallelism. Third, the same structure means that problems with relatively small highly active regions could isolate the need to solve a nonlinear problem to such regions, rather than having it be part of a fully coupled global nonlinear problem.

An algorithm for evaluating $\boldsymbol{g}$ is given in Algorithm 3.1. It requires one communication between subdomains to transfer interface data, followed by solving (in parallel if desired) the nonlinear subdomain problems $\boldsymbol{f}_i$ defined in (2.3). Note that the local problem in Ω_i is a discretization of the PDE with zero boundary data on the true boundary Γ_{i0} and values transferred from the foreign subdomains on the interfaces.

Algorithm 3.1. Evaluate SNK residual $\boldsymbol{g}(\boldsymbol{u})$.

Apply Algorithm 2.1 to compute $\boldsymbol{T}\boldsymbol{u}$.
for $i = 1, \ldots, N$ (in parallel) **do**
  Solve $\boldsymbol{f}_i(\boldsymbol{u}_i - \boldsymbol{z}_i) = \boldsymbol{T}_i\boldsymbol{u}$ for $\boldsymbol{z}_i$.
end for
Return $[\![\boldsymbol{z}_i]\!]$.

Using the notation of Algorithm 3.1, we have that

$$(3.2) \qquad \boldsymbol{f}_i'(\boldsymbol{u}_i - \boldsymbol{z}_i)\left[\boldsymbol{I} - \frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{u}}\right] = \boldsymbol{T}_i,$$

which implies that applying $[\partial \boldsymbol{z}_i / \partial \boldsymbol{u}]$ to a vector requires a single linear solve on a subdomain, with a matrix that is presumably already available at the end of the local Newton iteration used to compute $\boldsymbol{g}$. The process for applying $\boldsymbol{g}'(\boldsymbol{u})$ to a vector is outlined in Algorithm 3.2.

Algorithm 3.2. Apply Jacobian $\boldsymbol{g}'(\boldsymbol{u})$ to vector $\boldsymbol{v}$ for the SNK problem.

Apply Algorithm 2.1 to compute $\boldsymbol{T}\boldsymbol{v}$.
for $i = 1, \ldots, N$ (in parallel) **do**
  Solve $[\boldsymbol{f}_i'(\boldsymbol{u}_i - \boldsymbol{z}_i)]\boldsymbol{y}_i = \boldsymbol{T}_i\boldsymbol{v}$ for $\boldsymbol{y}_i$.
end for
Return $[\![\boldsymbol{v}_i - \boldsymbol{y}_i]\!]$.

Equation (3.2) can be rearranged to show that the Jacobian of the preconditioned nonlinear problem is

$$(3.3) \qquad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{u}} = \boldsymbol{I} - \left[\boldsymbol{f}'(\boldsymbol{u} - \boldsymbol{z})\right]^{-1}\boldsymbol{T}.$$

Because the transfer operator $\boldsymbol{T}$ has nonzero rows only at interface points, this confirms that the Jacobian of the preconditioned nonlinear problem is a low-rank perturbation of the identity. Moreover, if $\boldsymbol{T}$ is based on stable interpolation of values, it has

norm close to 1. Thus, Kantorovich or other standard convergence theory for Newton's method [6] may therefore suggest improved local convergence rates and larger basins of attraction. We have not yet pursued this analysis.

3.1. Two-level scheme. The SNK method can be expected to require coarse correction steps to cope with a growing number of subdomains. An obvious approach to incorporating a coarse-grid correction step is to apply FAS directly, i.e., using the analogue of (2.6) with fine g and coarse $\hat{g}$. However, doing so means inverting $\hat{g}$, which introduces another layer of iteration in the overall process.

We have found that it is simpler and successful to apply the FAS correction in the form of the original NKS method, as given in section 2.3. All we need to do is replace $f(u) - Tu$ and $f' - T$ in (2.8) and (2.9) by g and g', respectively.

4. Numerical experiments. For all experiments, we compared three methods: NKS, SNK, and the two-level SNK2, as coded by us in MATLAB. The local nonlinear problems for SNK, and the coarse global problems in SNK2, were solved using `fsolve` from the Optimization Toolbox. Each solver used an inexact Newton method as the outer iteration, continued until the residual was less than 10^{-10} relative to the initial residual. For the inner iterations we used `gmres` in MATLAB to solve for the Newton step s_k such that

$$(4.1) \qquad \|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|,$$

where $\eta_0 = 10^{-4}$ and

$$(4.2) \qquad \eta_k = 10^{-4} \left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|} \right)^2.$$

Given certain conditions on $F(x)$, if the initial solution is close enough to the true solution, then this set of tolerances will yield a sequence with near q-2 convergence [9].

4.1. Regularized driven cavity flow. The first example is a regularized form of the lid-driven cavity flow problem [11], where we replace the boundary conditions with infinitely smooth ones. Using the velocity-vorticity formulation, in terms of the velocity u, v and vorticity ω on $\Omega = [0,1]^2$ we have the nondimensionalized equations

$$(4.3) \qquad \begin{aligned} -\Delta u - \frac{\partial \omega}{\partial y} &= 0, \\ -\Delta v + \frac{\partial \omega}{\partial x} &= 0, \\ -\frac{1}{\mathrm{Re}} \Delta \omega + u \frac{\partial \omega}{\partial x} + v \frac{\partial \omega}{\partial y} &= 0, \end{aligned}$$

where Re is the Reynolds number. On the boundary $\partial \Omega$ we apply

$$(4.4) \qquad \begin{aligned} u &= \begin{cases} \exp \left(\frac{-\left(\frac{y-1}{0.1} \right)^2}{1 - \left(\frac{y-1}{0.1} \right)^2} \right), & y > 0.9, \\ 0, & y \leq 0.9, \end{cases} \\ v &= 0, \\ \omega &= -\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \end{aligned}$$
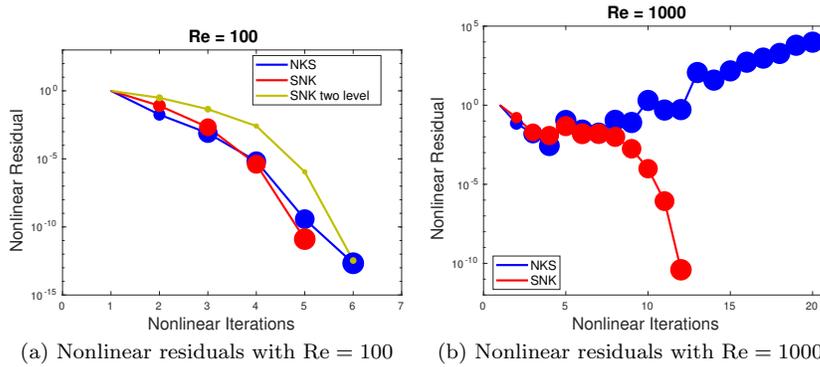
(a) Nonlinear residuals with Re = 100   (b) Nonlinear residuals with Re = 1000

FIG. 1. *Nonlinear residuals, normalized by the residual of the initial guess, of the NKS, SNK, and SNK2 solvers on the regularized cavity flow problem (4.3)–(4.4). The area of each marker is proportional to the number of inner GMRES iterations taken to meet the inexact Newton criterion.*

being similar to the boundary conditions in [12].

We divided Ω into 16 overlapping patches of equal size (i.e., a 4 by 4 patch structure). Each subdomain was discretized by a second-kind Chebyshev grid of length 33 in each dimension. For the initial guess to the outer solver iterations, we extended the boundary conditions (4.4) to all values of x in the square Ω.

The convergence of the three solvers is shown for Re = 100, and SNK and NKS for Re = 1000 in Figure 1. All three methods converge for Re = 100. The number of GMRES iterations per nonlinear iteration are similar for the NKS and SNK methods; this is to be expected, since the linear system used to solve the Newton step is similar in both methods. We do, however, see a dramatic reduction in the number of GMRES iterations with the two-level SNK method.

It is well worth noting that the computational time for each outer iteration varies greatly between solvers. The NKS residual requires only evaluating the discretized PDE and is thus is a good deal faster per iteration than the SNK solvers, which require solving the local nonlinear problems. In addition, SNK2 must solve a global coarse problem in each outer iteration, but this added relatively little computing time.

For the higher Reynolds number Re = 1000, we find that while SNK still converges, NKS does not, similar to what was reported in [4]. We also found that the coarse-level solver in SNK2 had trouble converging.

4.2. Burgers's equation. The second test problem is Burgers's equation,

$$(4.5) \qquad \nu \Delta u - u \cdot \nabla u = 0,$$

on $\Omega = [-1, 1]^2$, with Dirichlet boundary condition

$$(4.6) \qquad u = \arctan\left(\cos\left(\frac{3\pi}{16}\right) x + \sin\left(\frac{3\pi}{16}\right) y\right).$$

This PDE was solved using a subdomain structure adapted to the function

$$\exp\left(\frac{1}{1 - x^{-20}} + \frac{1}{1 - y^{-20}}\right)$$

using the methods in [1] in order to help capture the boundary layers, as shown in Figure 2. For the initial guess of the outer iterations, the boundary condition (4.6) was extended throughout Ω.
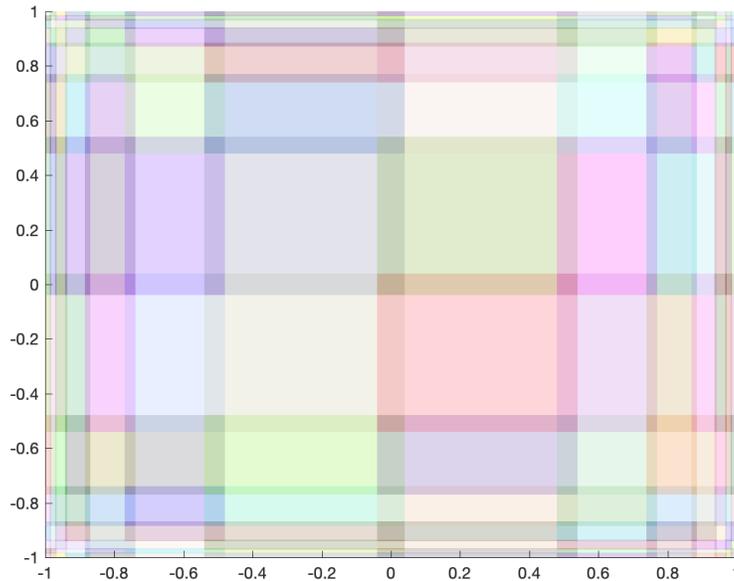
FIG. 2. *Subdomains for the Burgers experiments, found by adapting to the function* $\exp(-x^{20}/(1-x^{20}))\exp(-y^{20}/(1-y^{20}))$ *in order to increase resolution in the boundary layer.*

Convergence histories for (4.5)–(4.6) for $1/\nu = 400, 800, 1000, 1500$ are given in Figure 3. We observe again that the SNK and SNK2 solvers seem quite insensitive to the diffusion strength, while the number of outer iterations in NKS increases mildly as diffusion wanes. Furthermore, SNK2 converges in about half as many outer iterations as SNK.

5. Discussion. We have described a framework for overlapping domain decomposition in which overlap regions are discretized independently by the local subdomains, even in the formulation of the global problem. Communication between subdomains occurs only via interpolation of values to interface points. This formulation makes it straightforward to apply high-order or spectral discretization methods in the subdomains and to adaptively refine them.

The technique may be applied to precondition a linearized PDE, but it may also be used to precondition the nonlinear problem before linearization to get what we call the Schwarz–Newton–Krylov (SNK) technique. In doing so, one gets the same benefit of faster Krylov inner convergence, but the resulting nonlinear problem is demonstrably easier to solve in terms of outer iterations. In some cases, the convergence becomes more robust with respect to problem parameters, though there is no reason to believe that this will always be true.

We have demonstrated that the SNK method can easily be part of a two-level FAS in order to keep iteration counts from growing as the number of subdomains grows. The coarse level is simply a coarsening on each subdomain, so that restriction and prolongation steps can be done simply and in parallel. Indeed, the situation should make a fully multilevel implementation straightforward, as the multilevel coarsenings and refinements can all be done within subdomains.

The most time-consuming part of the SNK algorithm is expected to be typically
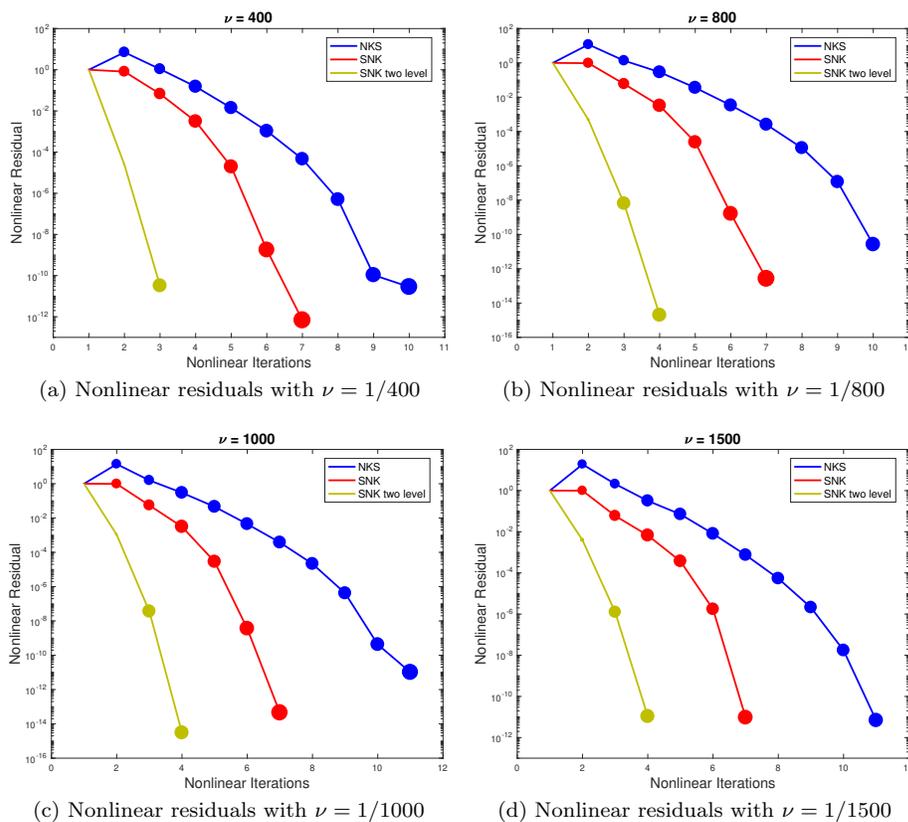
(a) Nonlinear residuals with $\nu = 1/400$

(b) Nonlinear residuals with $\nu = 1/800$

(c) Nonlinear residuals with $\nu = 1/1000$

(d) Nonlinear residuals with $\nu = 1/1500$

FIG. 3. *Nonlinear residuals, normalized by the residual of the initial guess, of the NKS, SNK, and SNK2 methods to solve* (4.5)–(4.6). *The area of each marker is proportional to the number of inner GMRES iterations taken to meet the inexact Newton criterion.*

in the solution of nonlinear PDE problems within each subdomain using given boundary data. These compute-intensive tasks require no communication and are therefore efficient to parallelize. By contrast, each inner Krylov iteration (i.e., Jacobian application) of both SNK and linearly preconditioned NKS requires a communication of interface data between overlapping subdomains, which appears to generate a more communication-bound form of parallelism. An additional feature of the SNK approach, mentioned also in [4], is that subdomains of low solution activity can be expected to be found relatively quickly. We observed this to be the case in the cavity flow problem of section 4.1, where local solutions in regions of low activity were sometimes three to four times faster than those in regions with steep solution gradients. This presents a natural way to limit the spatial scope of difficult nonlinear problems, though it also raises questions for load balancing in a parallel environment.

Finally, we remark that an important extension in [1] is to use least-squares approximation rather than interpolation to incorporate nonrectangular (sub)domains. We have been able to write a least-squares (as opposed to collocation) generalization of SNK and test it in one dimension. We hope to make it the subject of future work.

Codes for the steady PDE examples of the paper are available online from https: //github.com/tobydriscoll/schwarz-newton-krylov. The time-dependent example is

not included, since it relies on a light revision of the proprietary `ode15s` code in MATLAB.

REFERENCES

[1] K. W. AITON AND T. A. DRISCOLL, *An adaptive partition of unity method for multivariate Chebyshev polynomial approximations*, SIAM J. Sci. Comput., 41 (2019), pp. A3230–A3245, https://doi.org/10.1137/18M1184904.

[2] A. BRANDT AND O. E. LIVNE, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, revised ed., SIAM, Philadelphia, 2011, https://doi.org/10.1137/1.9781611970753.

[3] X.-C. CAI, W. D. GROPP, D. E. KEYES, AND M. D. TIDRIRI, *Newton-Krylov-Schwarz methods in CFD*, in Numerical Methods for the Navier-Stokes Equations, Notes Numer. Fluid Mech., F.-K. Hebeker, R. Rannacher, and G. Wittum, eds., Vieweg+Teubner Verlag, Wiesbaden, 1994, pp. 17–30, https://doi.org/10.1007/978-3-663-14007-8_3.

[4] X.-C. CAI AND D. E. KEYES, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200, https://doi.org/10.1137/s106482750037620X.

[5] X.-C. CAI AND M. SARKIS, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797, https://doi.org/10.1137/S106482759732678X.

[6] J. E. DENNIS, JR., AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996, https://doi.org/10.1137/1.9781611971200.

[7] V. DOLEAN, M. J. GANDER, W. KHERIJI, F. KWOK, AND R. MASSON, *Nonlinear preconditioning: How to use a nonlinear Schwarz method to precondition Newton's method*, SIAM J. Sci. Comput., 38 (2016), pp. A3357–A3380, https://doi.org/10.1137/15M102887X.

[8] V. DOLEAN, P. JOLIVET, AND F. NATAF, *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, SIAM, Philadelphia, 2015, https://doi.org/10.1137/1.9781611974065.

[9] S. C. EISENSTAT AND H. F. WALKER, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32, https://doi.org/10.1137/0917003.

[10] M. J. GANDER, *Schwarz methods over the course of time*, Electron. Trans. Numer. Anal., 31 (2008), pp. 228–255.

[11] C. HIRSCH, *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*, Elsevier, Amsterdam, 2007.

[12] J. SHEN, *Numerical simulation of the regularized driven cavity flows at high Reynolds numbers*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 273–280.

[13] B. SMITH, P. BJØRSTAD, AND W. GROPP, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 2004.